Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Mads Erlend Bøe Lysø

Exponentially Stable Dynamic Walking in a Sprawling Quadruped

Transfer of Methods and Comparison

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Ytterstad Pettersen
Co-supervisor: Esten Ingar Grøtli
June 2022

NTNU
Norwegian University of Science and Technology

Mads Erlend Bøe Lysø

Exponentially Stable Dynamic Walking in a Sprawling Quadruped

Transfer of Methods and Comparison

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Ytterstad Pettersen
Co-supervisor: Esten Ingar Grøtli
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Problem description

The following thesis will look at the problem of motion planning and control of dynamic walking in a sprawling quadrupedal robot. It will do so in the following manner:

- First, a range of existing methods previously used on mammalian quadrupedal robots will be reviewed.
- One of the reviewed methods will be selected as a basis for implementation on the sprawling quadrupedal robot ASTRo.
- The method will be evaluated in simulations on the ASTRo robot as well as on a mammalian robot (vision60), and results will be discussed and compared.

Abstract

Legged robots have been a topic of research of increasing academic and public interest for the last half-century. The potential uses for robust autonomous locomotion through environments which prove difficult for wheeled vehicles are manifold both in the military and commercial sectors, as well as in the realm of humanitarian work.

The topic is also interesting for strictly academic reasons: The problem of controlling such robots in ways that are both secure, robust and flexible enough to handle varied environments, while simultaneously being energy-efficient, has proven to be a tremendous challenge. While especially the last decade has seen impressive progress, the problem of robust dynamic legged locomotion is not a solved problem by any stretch of the imagination.

In the realm of quadrupedal locomotion, most research so far has focused on robots with a so-called mammalian leg configuration. However, there are other choices of leg configurations which are found in the natural world, and which may warrant further investigation.

One such example, found among other places in arachnids and certain reptiles, is the sprawling leg configuration. In this thesis we wish to draw upon the rich body of existing research on locomotion in mammalian quadrupeds, in order to investigate whether methods found there are easily adapted and transferred for use on sprawling quadrupeds. We first survey the existing literature on control for mammalian

quadrupedal locomotion, and select a method.

The selected method, which is based on an extension of the concept of zero-dynamics to Hybrid Dynamical Systems (HDS), uses trajectory optimization to directly synthesize a closed-loop IO-feedback-linearization controller which drives the system dynamics to a zero-dynamics manifold. A method of post-processing is employed to ensure that the controlled dynamics are exponentially orbitally stable on the manifold, resulting in an exponentially orbitally stabilizing controller for the system.

A modification was proposed to the control structure suggested in the original method, which represents a more principled approach to overconstrained systems, in which the number of actuators is greater than the number of actuated degrees of freedom. It is also shown that the modified approach results in the same system behavior, while using equal or smaller amounts of torque.

The method is implemented both on the sprawling quadruped ASTRo and the mammalian quadruped vision60 from Ghost Robotics, and both are tested in simulation. The results show that the resulting controller exponentially stabilizes both robots to period gait behaviors, demonstrating that the selected method is well-suited for use on sprawling quadrupeds as well. The resulting cost of transport also indicates that mammalian quadrupeds may have an energy-efficiency advantage over sprawling quadrupeds, while results obtained for unmodeled ground height changes indicate that sprawling quadrupeds may be less sensitive to such changes.

Results comparing the original controller structure to the proposed modification, suggest that there are in practice concrete and potentially significant energy-efficiency gains to be made from the modification.

Sammendrag

Roboter med ben har vært et forskningsområde av økende akademisk og offentlig interesse de siste 50 årene. Roboter som evner å bevege seg robust og autonomt gjennom områder som byr på utfordringer for konvensjonelle kjøretøy med hjul vil ha utstrakte bruksområder, både innenfor militære og kommersielle sektorer, men også innenfor humanitært arbeid.

Området er også interessant i rent akademisk øyemed: Problemet med å styre slike roboter på en måte som er sikker, robust og fleksibel nok til å håndtere varierte mijøer – og som samtidig er energieffektiv – har vist seg å være en formidabel utfordring. Selv om man særlig det siste tiåret har vært vitne til imponerende fremgang innen feltet, er robust dynamisk gange fortsatt ikke i noen forstand et løst problem.

Innenfor firbeint gange har brorparten av forskning så langt fokusert på roboter med en såkalt pattedyrliknende benkonfigurasjon. Det finnes likevel andre benkonfigurasjoner, noen eksemplifisert i dyreriket, som fortjener en grundigere behandling enn så langt gitt.

Et eksempel på dette, som vi blant annet finner hos araknider og enkelte reptiler, er en edderkoppliknende benkonfigurasjon. I denne oppgaven vil vi benytte den store mengden eksisterende forskning på gange hos pattedyrliknende roboter, for å undersøke hvorvidt metoder herfra enkelt kan tilpasses og overføres til edderkoppliknende roboter. Vi begynner med en undersøkelse av eksisterende litteratur for regulering av

pattedyrliknende roboter og velger en metode.

Den valgte metoden, som baserer seg på en utvidelse av konseptet null-dynamikk (zero dynamics) til hybride dynamiske systemer, bruker baneoptimering til direkte å produsere en lukket-sløyfe IO-feedback-lineariseringsregulator som driver system-dynamikken til en nulldynamikkmangfoldighet. En metode benyttes deretter til å postprosessere regulatorparametrene, for å sikre at den styrte dynamikken er eksponensielt orbitalt stabil på mangfoldigheten. Dette resulterer igjen i en eksponensielt orbitalt stabiliserende regulator for systemet.

En endring ble foreslått til regulatorstrukturen som var foreslått i den originale metoden. Denne endringen representerer en mer prinsipiell tilnærming til å håndtere overbegrensede systemer, hvor antall pådrag er større enn antall frihetsgrader pådragene kan påvirke. Vi viser også at denne endrede tilnærmingen resulterer i lik systemoppførsel, og bruker like mye eller mindre pådrag.

Denne metoden er implementert både på den edderkoppliknende roboten ASTRo, og på den pattedyrliknende roboten vision60 utviklet av Ghost Robotics. Begge roboter testes i simulering. Resultatene viser at den resulterende regulatoren eksponensielt stabiliserer begge roboter til periodiske ganglag, og demonstrerer dermed at den valgte metoden er velegnet til bruk på edderkoppliknende roboter. Den resulterende transportkostnaden peker også i retning av at pattedyrliknende roboter kan ha et fortrinn over edderkoppliknende med hensyn til energieffektivitet. Resultater fra simulering på terreng med umodellerte høydeendringer, tyder dog på at edderkoppliknende roboter kan være mindre sensitive til slike endringer.

Resultater som sammenlikner regulatoren med foreslåtte endringer med den opprinnelige regulatorstrukturen, tyder på at det er konkrete og til tider betydelige fordeler med hensyn til energieffektivitet ved å benytte den endrede regulatoren.

Contents

Problem description i

Abstract ii

Sammendrag iv

Preface xiv

Acronyms xvi

1 Introduction 1

1.1 Assumptions 5

1.2 Contributions 5

1.3 Outline 6

2 Literature Review 7

2.1 Selected methods 7

2.2 Zero Moment Point 8

2.3 Vertical Impulse Scaling 9

2.4 Contact Time Modulation 10

2.5 Reduced-order Convex Model Predictive Control 11
2.6 Nonlinear Model Predictive Control with soft contact model 12
2.7 Model predictive control with adaptive Control Lyapunov Function . 14
2.8 Data-driven methods using privileged learning 15
2.9 Choice of method 17

3 System Modeling **19**
3.1 System kinematics 19
3.1.1 Ground contact constraints 24
3.2 System Dynamics 25

4 Methods **29**
4.1 Hybrid Zero Dynamics 30
4.1.1 Hybrid Dynamical Systems 31
4.1.2 Continuous dynamics 32
4.1.3 Discrete dynamics 32
4.1.4 Continuous domain constraints 33
4.1.5 Guards 35
4.1.6 Zero-dynamics 36
4.1.7 IO-linearization for overconstrained systems 39
4.2 Direct collocation 45
4.3 Closed-loop Trajectory Optimization 47
4.4 Controller post-processing for orbital stability 50
4.4.1 Poincaré return maps 50
4.4.2 Floquet multipliers and Orbital stability 51
4.4.3 Reduced-dimension stability analysis 52
4.4.4 Extension to Hybrid Dynamical Systems 54
4.4.5 Linear/Bilinear matrix inequalities 56
4.4.6 Sensitivity analysis and post-processing 58
4.4.7 Managing and reducing computational complexity 62
4.5 Implementation details 64
4.5.1 Robot models 64

4.5.2 Closed-loop trajectory optimization . . . 64
4.5.3 Controller parameter post-processing . . . 66
4.5.4 Simulation . . . 66

5 Results and Discussion 67
5.1 Optimal gaits . . . 68
5.2 Gait stabilization . . . 73
5.3 Simulation results . . . 73
5.3.1 vision60 . . . 74
5.3.2 ASTRo . . . 75
5.3.3 Deviations between initial state and fixed point for the state on the Poincaré section . . . 81
5.4 Response to unmodeled changes in ground height . . . 89
5.4.1 vision60 . . . 89
5.4.2 ASTRo . . . 89
5.5 Torque and energy expenditure . . . 94
5.5.1 Cost of Transport . . . 95
5.5.2 vision60 . . . 95
5.5.3 ASTRo . . . 98
5.5.4 Performance comparison across robots . . . 98
5.5.5 Performance comparison between 11 and 12 actuators . . . 101
5.6 Possible limitations . . . 102

6 Conclusion and further work 103
6.1 Further work . . . 104

References 107

List of Tables

3.1 List of Denavit-Hartenberg (DH) parameters for the transformation from each leg frame to its respective foot. Visualization made using RVIZ (ROS visualization tool) 23
3.2 Translation to link Centers of Mass (CoMs) and feet from frames in which they are fixed 24
3.3 Masses and inertias of robot links 25

5.1 Spectral radii of the projection of the Poincaré map Jacobian for different gaits, before and after post-processing the controller parameters for exponential stability 73
5.2 Performance metrics for ASTRo and vision60 for respectively 11 (non-overconstrained) and 12 (overconstrained) actuators 102

List of Figures

2.1 Illustration of the relation between Center of Gravity (CoG) projection and Zero Moment Point (ZMP) 9

3.1 Illustration of rotational axes for a sprawling joint configuration. Blue lines indicate rotational axes and direction 20
3.2 Illustration of robot links and coordinate frames 22

4.1 Illustration of an axis $\vec{k}$ around which the available contact forces can generate no torque/moment 42
4.2 Geometric illustration of change in impact point and time as a function of initial perturbation 57
4.3 Norm of the sensitivity of $\tilde{\mathbf{A}}_0$ with respect to each parameter in $\mathbf{H}$ . . 65

5.1 Diagram for one cycle of an ambling gait. Grey stretches indicate stance phases while blue stretches indicate swing phases. 68
5.2 Phase portraits for optimized stationary ambling gait on the ASTRo robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted. . 69

5.3 Phase portraits for optimized forward ambling gait on the ASTRo robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted. . 70

5.4 Phase portraits for optimized stationary ambling gait on the vision60 robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted. . 71

5.5 Phase portraits for optimized forward ambling gait on the vision60 robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted. . 72

5.6 Phase portraits for stationary gait on the vision60 robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized. 75

5.7 Trajectories of base pose states during 250 cycles of stationary gait for the vision60 robot during simulation 76

5.8 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a stationary gait for the vision60 robot during simulation 77

5.9 Phase portraits for forward (0.2 m/s) gait on the vision60 robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized. 78

5.10 Trajectories of base pose states during 250 cycles of forward gait on flat ground for the vision60 robot during simulation 79

5.11 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on flat ground for the vision60 robot during simulation 80

5.12 Phase portraits for stationary gait on the ASTRo robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized. 82

5.13 Trajectories of base pose states during 250 cycles of stationary gait for the ASTRo robot during simulation 83

5.14 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a stationary gait for the ASTRo robot during simulation 84

5.15 Phase portraits for forward (0.2 m/s) gait on the ASTRo robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized. 85

5.16 Trajectories of base pose states during 250 cycles of forward gait on flat ground for the ASTRo robot during simulation 86

5.17 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on flat ground for the ASTRo robot during simulation 87

5.18 Trajectories of base pose states during 250 cycles of forward gait on ground with a ramp of 0.3% elevation for the vision60 robot during simulation 90

5.19 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on ground with a ramp of 0.3% elevation for the vision60 robot during simulation. Note that for this plot, the entire base position is omitted as the z-position is not periodic when the ground slopes. . . 91
5.20 Trajectories of base pose states during 250 cycles of forward gait on ground with a ramp of 2% elevation for the ASTRo robot during simulation 92
5.21 Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on ground with a ramp of 2% for the ASTRo robot during simulation. Note that for this plot, the entire base position is omitted as the z-position is not periodic when the ground slopes. 93
5.22 Overview of torques for vision60 robot during forward gait on flat ground using 11 actuators 96
5.23 Overview of torques for vision60 robot during forward gait on flat ground using 12 actuators 97
5.24 Overview of torques for ASTRo robot during forward gait on flat ground using 11 actuators 99
5.25 Overview of torques for ASTRo robot during forward gait on flat ground using 12 actuators 100

Preface

The following thesis is submitted as a part of the requiremeents for the master's degree at the Department of Enginering Cybernetics at the Norwegian University of Science and Technology, and is written in collaboration with the independent research organization SINTEF. The thesis dives into the field of robotic walking, and is simultaneously inspired by both the apparent simplicity and true difficulty of the control problem.

This master's thesis is a continuation of a specialization project I conducted during the autumn of 2021. As is customary, the specialization project is not published. This means that important background theory and methods from the project report will be restated in full throughout this report to provide the best reading experience. Below, a complete list of the material included from the specialization project [1] is listed.

- chapter 2 (specifically sections 2.2 to 2.4)
- chapter 4 (specifically sections 4.1 and 4.3 with some adaptations overall and larger extensions to section 4.1.6.)

Unless otherwise stated, all figures and illustrations have been created by the author.

I would like to express my gratitude to my academic supervisors, Prof. Kristin

Ytterstad Pettersen and Dr. Esten Ingar Grøtli for their time and invaluable guidance, expertise and feedback.

I would like to thank my partner, Silje-Marie, for her support through the ups and downs which constitute the day-to-day experience of writing such a thesis. She has listened mostly patiently to a great deal of frustrations about the searching for and finding of long-hidden computer bugs introduced by yours truly, for which I am deeply grateful. In addition, she read and reviewed the thesis. I would also like to thank my friends. In particular, I want to thank Karl Hole Totland; not only have we enjoyed (at the very least, tolerated) each others' company for 21 years, he also read and reviewed this thesis not only once but twice. Finally, I would like to thank my family; Gro Ellen, Nils Are, Joar, Ingeborg, Margrethe, Severin, Ingrid and Martin for their continued support to reach this point.

Acronyms

AD Automatic Differentiation 13

BMI Bilinear Matrix Inequality 29, 56, 58, 61, 62

CLF Control Lyapunov Function 14, 15

CoG Center of Gravity x, 9

CoM Center of Mass ix, 24, 25

CoT Cost of Transport 4, 94, 98, 101, 102, 104

DH Denavit-Hartenberg ix, 22, 23

DOF Degree of Freedom 19, 20, 41, 43

DRL Deep Reinforcement Learning 15

HDS Hybrid Dynamical System iii, 31, 35, 54

HZD Hybrid Zero-Dynamics 29, 30

LFC Lyapunov Function Candidate 60

LICQ Linear Independence Constraint Qualification 13

LMI Linear Matrix Inequality 56, 58, 61

LQ Linear-Quadratic 13

LTV Linear Time-Varying 11

MPC Model Predictive Control 12, 14, 15

MPP Moore-Penrose Pseudoinverse 44, 45

NLOC Nonlinear Optimal Control 13

NLP Nonlinear Programming 13

NMPC Nonlinear Model Predictive Control 14

ODE Ordinary Differential Equation 32, 36

PE Persistently Exciting 14

PHZD Partial Hybrid Zero Dynamics 39

RMS Root Mean Square 4, 94, 102, 104

SRBD Single Rigid-Body Dynamics 11

URDF Unified Robot Description Format 64

ZMP Zero Moment Point x, 8, 9, 11

1

Introduction

The process of automating work for increases in productivity has been a hallmark of most technological revolutions throughout history, illustrated by such examples as the printing press and the industrial revolution. The next such large-scale disruption in the conditions under which production occurs - coined as "The fourth industrial revolution" by economist Klaus Schwab [2] - is expected to rely heavily on the increased assistance and replacement of human workers by robots. This shift is expected to occur to a greater or lesser extent across various domains, from industrial and logistics settings, to construction of buildings and infrastructure, to transportation and goods-delivery, to search-and-rescue operations, etc.

Each of these domains will require robots with various degrees of autonomy, situational awareness and ability to engage with unstructured or unexpected environments. While a stationary robot will often be appropriate in an industrial setting and a wheeled robot might be sufficient for a perfectly flat storage floor, any application that requires navigation in the outside world where the ground may be neither level, flat nor free of obstacles - or that requires traversing a flight of stairs - will need greater mobility than is offered by wheels or belts. While airborne robots might fill parts of this niche, their load-bearing capacities are fairly modest compared to land-borne alternatives,

and their high power consumption puts hard constraints on their longevity between charges (assuming a battery power source).

The alternative then, which is in principle both significantly more mobile than a wheeled robot while outperforming flying alternatives with regards to longevity and load-bearing, would be legged robots. The transportation mode of legged walking has been extensively tested for these purposes with some success in the animal kingdom.

However, the problem of robotic walking introduces distinct difficulties and is a significantly more complex task than wheeled locomotion. If the robot has revolute joints - which is the typical case - its kinematics and dynamics are highly nonlinear, and the linearization of such dynamics would not be valid for a very large portion of the robot's configuration space.

The robot is also by necessity underactuated: A walking robot moves itself by the reaction forces from pushing on the ground beneath itself. Firstly, this bars the robot from moving downwards at any greater rate than what is dictated by gravitational forces. Secondly, these reaction forces must lie within a cone which is determined by the friction coefficient of the specific ground material, so as to not slip. The robot may be in configurations where it is locally fully actuated, in so far as it may change its configuration in any direction with some bounded acceleration. It may also perform entire gait patterns where this is the case. However, these gaits, known as static gaits, are known to be quite energy-inefficient. Thus, in order to achieve energy-efficient dynamic walking, the robot must exhibit behavior in which it is also locally underactuated (in that it may only change its acceleration in certain directions of the configuration space) for parts of or the entirety of its gait.

Walking robots are also examples of hybrid systems, that is, systems where the state undergoes both continuous dynamics, and discrete jumps in state space and dynamics. Such jumps will occur at any point when the robot either lifts a foot from the ground or puts it down. All of these factors contribute to making legged locomotion a continually challenging control problem, which may require as well as motivate research into novel control methods.

In the last few decades, research on walking in both bipedal and quadrupedal robots have made significant progress. Both academic endeavors and emerging commercial

efforts have contributed to advances that has taken walking robots from the realm of science fiction to what seems to be close to industry-ready products. On the commercial side, Boston Dynamics and ANYbotics have impressed with robots such as Spot and Atlas, and ANYmal, respectively, and both offer quadrupedal robots for sale to be used for work such as inspection of industrial sites. In the academic realm, notable contributors are MIT - lately behind robots such as the series of Cheetah-robots as well as being the cradle of Boston Dynamics - Caltech's AMBER lab - known for work on robots such as Cassie and DURUS - and ETH, behind StarlETH, the predecessor of the ANYmal robot.

However, most research on quadrupedal robots has been focused on robots with a leg configuration resembling that of mammalian animals such as dogs, cats, or cheetahs, where the hip abductor/adductor joint has a rotational axis which is aligned with the length of the animal's body. A different choice, also represented in nature in both reptilian and arachnid animals, would be the sprawling leg configuration. In such a configuration, the hip abductor/adductor joint axis is not aligned with the body lengthwise, but is rather orthogonal to the flat ground and the frontal plane of the animal. While the mammalian configuration might exhibit certain advantages with respect to energy efficiency, it is plausible that sprawling quadruped robots might be more robust to disturbances or uneven terrain: They have the ability to widen their stance. This again increases the area of their support polygon and with it, their safety margin with respect to falling over. This is likely to translate into a greater robustness against falling when faced with rough terrains or unforeseen obstacles.

It would thus be of interest to draw on and build upon the rich already existing literature on walking in mammalian quadrupedal robots and investigate whether it is straightforward to transfer such methods and adapt them for sprawling quadrupeds. In selecting such a method, we are interested in the following properties:

- Does the method allow for mathematical proofs of orbital stability or disturbance rejection? (Control-theoretical soundness)
- Can the resulting gait be exhibited over long periods of time without the robot losing its balance? (Empirical demonstration of stability)

- Is the energy expenditure and maximal torque output reasonably low? (Energy efficiency)
- Can the method be altered with relative ease (no structural changes) in order to produce a variety of gait behaviors? (Flexibility)

In reviewing the success of the adapted method to a sprawling quadruped, we will be interested in evaluating how these properties hold for the method on a sprawling quadruped, and considering how this compares to the results from the method on a mammalian quadruped.

Furthermore, we posit two hypotheses at the outset, whose plausibility will later be reviewed in light of the presented evidence:

- The gait of the sprawling quadruped will be less sensitive to unmodeled terrain changes than that of the mammalian quadruped, due to its wider support polygon relative to the base size
- The gait of the mammalian quadruped will be more energy efficient than that of the sprawling quadruped with respect to Cost of Transport (CoT), as the wide stance of the latter may result in more torque being needed to enact a certain force on the base

In order to investigate these questions, we will first perform a literature survey, getting an overview of the existing body of research on mammalian quadrupedal dynamic locomotion. Having evaluated the apparent strengths and weaknesses of each method with respect to these measures, we will select one for further evaluation by adapting and implementing the method for our own sprawling quadruped ASTRo, designed and built in [3]. Additionally, we will implement it on the mammalian quadruped vision60 developed by Ghost Robotics for comparison purposes.

Following this, we will simulate the closed-loop systems for various gaits and evaluate their performance on measures of energy efficiency and torque use – CoT and Root Mean Square (RMS) torque as well as peak torque output – as well as adaptability to unmodeled changes in ground height. We then build on a method for full-order

model-based control previously tested on mammalian quadrupeds and adapt it to the sprawling quadruped robot ASTRo, designed and built in [3]. Further, we implement the method for both the mammalian quadruped vision60 developed by Ghost Robotics and ASTRo and compare the two on measures of energy efficiency and robustness in simulation.

The observant reader might have noticed that it might be difficult to evaluate whether a given outcome with respect to energy efficiency is best explained by differences in the suitability of the method, or by inherent differences caused by the leg configuration which would be present regardless of chosen method. This as well will be discussed in light of the observed results and the chosen method in chapter 5.

1.1 Assumptions

The assumptions made in the development and applications of the methods in this thesis are as follows:

- The robot's links and body are all perfectly rigid, and the framework of rigid body dynamics is appropriate for modeling the system.
- The robot's foot experiences no slipping or sliding while in contact with the ground, so long as the ground reaction forces are within a friction cone given by a constant friction coefficient.
- The robot's foot contact with the ground can be modeled as a point contact, where there is only translational friction along the ground plane but no rotational friction along its normal axis.
- The establishing or breaking of contact between the robot foot and the ground are both instantaneous, and are well modeled as discrete dynamical events.

1.2 Contributions

The contributions of this thesis are

- Evaluation of the suitability of the selected method for use with sprawling quadrupeds through implementation, stability analysis and simulation results
- Comparison of the method implemented on one mammalian and one sprawling quadruped on response to unmodeled ground height changes as well as on measures of energy and torque expenditure in simulation
- Extension and systematization of Poincaré stability analysis for hybrid systems (as described in [4]) to the case with an arbitrary number of continuous domains
- Proposed modification of the controller from the selected method to handle overconstrained systems (more actuators than actuated degrees of freedom) in a more principled manner, with proof of at-worst equivalent torque use
- Empirical demonstration of reductions in energy and torque expenditure with modified controller for both mammalian and sprawling quadrupeds in simulation

1.3 Outline

The thesis proceeds in the following manner: First, in chapter 2 a literature review is conducted in which previous research is discussed in order to give an account of the current state of the field. Following this, a description is given in chapter 3 of the model of the system for which the methods are developed. Then, the theory and methods are presented and explained in chapter 4. The results are presented in chapter 5, alongside a discussion of their implications as well as possible limitations. Finally, a conclusion is given in chapter 6 along with a discussion of possible directions for further work.

2

Literature Review

Several different approaches have been taken to the problem of robotic walking, and to quadrupedal robot locomotion in particular. In what follows, a selection of such efforts is presented in order to give an overview of the history and current state of the field.

Sections 2.2 to 2.4 are from the specialization project report [1] and are restated here for the benefit of the reader.

2.1 Selected methods

We first give a short overview of the methods we selected. In [5], Hereid et.al. introduces a formulation for trajectory optimization for humanoid robots. The formulation differs from standard offline trajectory optimization-approaches in its inclusion of an IO-feedback-linearization controller in the optimization problem. Thus, what is synthesized is not an open-loop set of torques, but a closed-loop control policy. This method is extended to a quadrupedal mammalian robot in [6] by Ma et.al.

Furthermore, [4] introduces a systematic method to tune controllers that can be parameterized by a finite-dimensional parameter vector to guarantee exponential

orbital stability. For certain classes of controllers, e.g. IO-feedback-linearization controllers for underactuated systems, finding a choice of parameters that renders the full state asymptotically stable is far from trivial, making such a systematic approach to post-processing quite useful.

Both of these methods will be given a thorough treatment in chapter 4.

2.2 Zero Moment Point

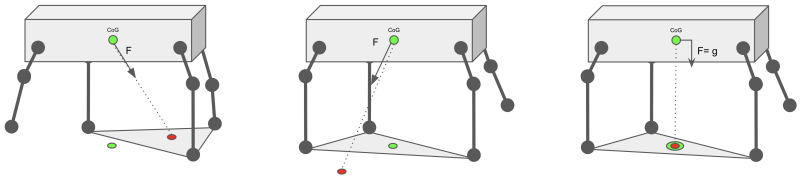
The Zero Moment Point (ZMP) concept is one of the older, more acclaimed concepts in planning and controlling dynamic gait in both bipedal and quadrupedal robots. In [7], Vukobratovic and Borovac give an account of the concept and its role in the history of legged locomotion.

In a sense, the task of achieving any sort of walking in a robot is the task of achieving some motion which propels the robot forward while ensuring that it doesn't fall over. Controlling the robot's motion is done indirectly, through the reaction forces from the ground on the robot.

The support polygon is the convex hull spanned by all the contact points (in the case of point feet). It is, in other words, the part of the ground in which ground reaction forces act on the robot. If the robot does not already have angular momentum so as to result in a tilt, then a lack of applied moment will ensure that it continues to not tilt. There is one axis along which the sum of all moments and forces acting on the robot can be replaced by a single force.

Consider the point at which this axis intersects the ground. If this point – known as the ZMP – lies within the support polygon, then the sum of ground reaction forces can be considered also as one single resultant force acting on that point. Under assumptions of no sliding, this resultant reaction force balances out all other forces acting on the mechanism.

However, if the intersection of this axis with the ground falls outside of the support polygon, this resultant force – applied at one point in the ground plane – and the ground reaction force – applied at another point – will not balance out but rather generate moments, potentially tilting the robot over. With this in mind, the goal then



(a) CoG outside of support polygon but ZMP inside

(b) CoG inside of support polygon but ZMP outside

(c) CoG corresponds with ZMP (stationary case)

Figure 2.1: Illustration of the relation between CoG projection and ZMP

becomes to plan a combination of footholds and trajectories which leads the ZMP to be within the support polygon of the robot at all times, what is known as the ZMP condition.

It may be interesting to note that, in the stationary case where the only non-ground force affecting the robot is gravity, the ZMP coincides with the projection of the Center of Gravity (CoG) onto the ground. Please see fig. 2.1 for an illustration of the relation between the projection of CoG and the ZMP.

While the ZMP concept has been widely used in the existing literature, it is somewhat limited. The concept rests on a balance of moments, and does not consider angular momentum or velocity. Especially for gaits involving higher angular velocities, i.e. more dynamic gaits, this can lead the robot to tip over even while fulfilling the ZMP condition ([8]).

2.3 Vertical Impulse Scaling

In [9], a bioinspired controller is introduced based on a limit cycle approach.

The initial observation is that in animals, swing phases remain mostly constant while stance phases may change to alter gait period. As the gait frequency increases and stance phase decreases, the magnitude of the ground reaction force profiles increases to keep making up for the vertical momentum lost to gravity.

Initially, a nonlinear optimization problem over the reaction force profiles and the

timing and duration of stances is solved offline for one desired forward velocity in order to obtain a limit cycle. Having obtained this limit cycle and the corresponding ground reaction forces and timings, the designed gait is converted to state-feedback control of ground reaction forces, using the angle between the respective stance leg and the ground as phase variable.

On top of this feed-forward signal a virtual spring and damper is added in the body height and pitch, in order to stabilize the limit cycle. Lastly, additional horizontal force is added to control speed.

Having calculated the desired ground reaction force, it is transformed to a signal in torques using a simplified version of the inverse dynamics, ignoring leg inertia as well as Coriolis effects. In order to achieve higher speed gaits while respecting friction cone considerations, the stance phase is reduced and the vertical impulse is scaled commensurately, so that the area under the reaction force curve is constant. This allows for greater horizontal force to be applied to the ground without slipping, resulting in higher speed gaits.

Though the method seems to allow for highly dynamic locomotion – galloping – both the simulation and the experiments carried out has the robot constrained to a plane. It is thus not given that the method will hold – that is, produce feasible stable locomotion – for the fully 3D case.

2.4 Contact Time Modulation

[8] introduces the Contact Time Modulation method for stabilization of various quadrupedal gaits. The method builds on observations about typical properties of walking in animals. Firstly, the duration of the swing phases of a gait tend to be constant regardless of the gait period. Thus, any variation in gait period is accounted for by a shrinking or growing of the stance phase of each leg.

Further, the ground force profile of each leg in animals has been shown to be well-approximated by a quadratic function. The impact applied to the ground by each foot is also assumed to be identical. Lastly, while one cannot control the exact time at which a foot makes contact with the ground, one can control when a foot lifts off.

Thus, one can control the stance time of each foot (within limits).

Using conservation of momentum and assuming constant height across gait cycles, one can thus calculate the nominal stance foot force profile.

A closed form approximation of the pitch after one gait period is derived as a function of the length of the stance phase of each foot. By imposing that the robot should attain zero pitch after a full cycle has passed, one can calculate the desirable stance phase – and thus liftoff time – of each foot.

The controller was proven to be stable in the sense of Lyapunov. As the closed form expression for pitch after a period was an approximation, a disturbance term was included in the analysis to account for modeling errors. By keeping the stance duration within a given disturbance-dependent interval, the pitch of the closed-loop system was shown to be stable.

Furthermore, a nonlinear disturbance observer was employed to estimate the disturbance and modeling error. The method was shown to outperform the vertical impulse scaling method as well as the ZMP method across a range of metrics, for instance robustness to disturbances and different ground stiffness.

The critique that motion is only attempted controlled in the plane applies to this method as well. Extension to roll stabilization is listed as further work.

2.5 Reduced-order Convex Model Predictive Control

In [10], Di Carlo et.al. utilizes a simplified linear kinematic and dynamic model to optimize over ground reaction forces from footsteps. The model used is a Single Rigid-Body Dynamics (SRBD) model, where only the base of the robot is considered in the dynamics. This is an approximation that works fairly well when the mass of the legs is negligible compared to the mass of the robot base. In fact, the kinematics of the legs are not considered either in the optimization problem, which rather uses the ground reaction forces as decision variables. The paper linearizes the rotational dynamics in the roll and pitch dimensions around an upright orientation, and, lastly, predetermines the footstep positions of the robot for the entire prediction horizon.

This renders model dynamics which are Linear Time-Varying (LTV), which again

allows for formulating a linear optimal control problem. The method does this in order to arrive at an optimal control problem which can be solved efficiently enough to run in a receding-horizon fashion in real time, rendering a linear Model Predictive Control (MPC) algorithm.

Having generated desired ground reaction forces, these are translated to desired joint torques through the Jacobian of the leg joint positions. The swing legs, on the other hand, are position-controlled through an inverse dynamics controller. Placement of new footsteps in the ground plane is done through the Raibert heuristic: $\mathbf{p}_{des} = \mathbf{p}_{ref} + \mathbf{v}_{com}\frac{\Delta t}{2}$ where $\mathbf{p}_{ref}$ is the position of the respective hip, $\mathbf{v}_{com}$ is the projection of the center of mass velocity onto the ground plane, while Δt is the time the foot will spend on the ground. It is this heuristic, along with knowledge of the desired robot velocity, which allows for predetermining placement of the footsteps.

The controller is tested on the MIT Cheetah 3 quadruped in experiments. The paper demonstrates a variety of behaviors, including trotting, walking up stairs without any environmental knowledge, bounding, galloping and pronking (a gait consisting of a sequence of short jumps). The controller is demonstrated to be fairly robust to disturbances, and reached a maximum galloping speed of $3\frac{m}{s}$ without constraining the robot in its sagittal plane.

2.6 Nonlinear Model Predictive Control with soft contact model

In [11], Neunert et.al. introduces a method for MPC using a nonlinear model. The method employs a full-order nonlinear model of the robotic system to pose a receding horizon optimal control problem which is then solved in a recurrent fashion.

One of the advantages of this method, is the avoidance of pre-specifying gait sequences, instead letting this be a part of the optimization problem. This is intuitively desirable: By avoiding reliance on hand-crafted gait sequences, the method is likely to be more flexible and adaptive to unexpected situations as they arise.

This also excludes use of the most typical contact modeling, which utilizes explicit

positional constraints on contact feet. One way of modeling contact which does not necessitate pre-specifying which feet are in contact at what time, is the use of complementarity constraints, $p_z \cdot \mathbf{f}_{contact} = 0$, where p_z is the height above ground of a foot while $\mathbf{f}_{contact}$ is the contact force associated with each foot. While this constraint does encode the relationship between foot position and contact forces, they do not satisfy the Linear Independence Constraint Qualifications (LICQs) which the authors argue are assumed by most or all available Nonlinear Optimal Control (NLOC) or Nonlinear Programming (NLP) solvers.

Instead, this method formulates the contact force of each foot as an explicit function of the robot state. This contact force is modeled as a combination of multiple linear springs and dampers, where some are parallel to the surface (friction) and some are perpendicular (ground normal force). These forces are then multiplied by an exponential function so as to diminish quickly when the foot leaves the surface. While this is not representative of how contact forces work, the authors argue that the non-disappearance of the contact force (and thus also its gradient) helps aid the optimization algorithm in finding new footholds. It is further argued that no detrimental effects from this "slight nonphysicality" are observed during experiments. The explicit formulation of ground contact forces allows the problem to be posed as an unconstrained optimal control problem, and to then recurrently use a first-order method where this is approximated as an Linear-Quadratic (LQ) optimal control problem.

The method utilizes packages for Automatic Differentiation (AD) for quick and accurate calculation of derivatives. It also focuses on the importance of lower-level steps to enhance computational efficiency, such as ensuring a high extent of vectorization in code and utilizing the ability of parallelizing forward-simulation when employing a multiple-shooting method. The impact of different choices of instruction sets is also briefly discussed.

The method is tested experimentally on two quadrupedal platforms, namely the ANYbotics' ANYmal and the IIT's hydraulic quadruped HyQ. The method tests different behaviors such as trotting and a squat jump. Crucially, as the method does not require nor allow for pre-specifying contact patterns, such behaviors are induced by modifying

the cost function to encourage e.g. a specific upwards base velocity to ensure a lift-off. The flexibility of the controller also results in a compliant controller that adapts to disturbances in e.g. ground height by adjusting its gait instead of reacting aggressively to such disturbances.

2.7 Model predictive control with adaptive Control Lyapunov Function

In [12], Minniti et.al. proposes a provably stable adaptive Nonlinear Model Predictive Control (NMPC) scheme. The method uses a somewhat simplified kino-dynamic model where dynamic constraints must be satisfied only on the main base, while kinematic constraints are considered for both the robot base and its legs. As mentioned earlier, this is a decent approximation in cases where the robot leg masses are negligible in comparison with the base mass.

The stability guarantee is given in terms of a Control Lyapunov Function (CLF) - i.e. a function which, if kept negative, gives a theoretical guarantee of the stability of the system - which is integrated into the MPC scheme as an inequality constraint. The CLF is defined on a subset of the system state, thus driving the state to some lower-dimensional manifold (such as an optimal trajectory).

In addition, an adaptive control scheme is integrated into the MPC formulation which accounts for an unknown disturbance torque (on a form which allows for both errors in e.g. gravitational forces as well as modeling errors in the generalized mass) in the system dynamics. The parameters of this disturbance is then estimated online based on errors between predicted and observed system behavior. For linear systems, there are guarantees to be made that the model estimate will converge to the correct model for Persistently Exciting (PE) inputs.

While such guarantees are not to be found for nonlinear systems in general (i.e. the parameters of the unmodeled disturbance may not converge to its ground truth), the resulting dynamics of the model will still converge. The result is the ability of the MPC to adjust online to apparent changes in base mass, e.g. from the robot

carrying a payload, in a way which substantially outperforms a non-adaptive version in comparisons. Also in presence of such uncertainties, the inclusion of the CLF is shown to lead to convergence of the system state onto a lower-dimensional manifold.

Compared with the MPC approaches discussed above, the method discussed here shows some clear advantages. None of the previously discussed papers consider theoretical stability guarantees, and consequently no stability properties are shown for the controlled system. Furthermore, none of them account for possible changes in base mass and inertia. In addition to possibly adjusting for modeling errors in the robot itself, this is more importantly crucial for using the robot for carrying of payload in a flexible, versatile manner.

However, the approach still uses a somewhat simplified model, not accounting for the dynamics of the legs and how they affect the dynamics of the base. This is a smaller issue for less dynamic motion, but becomes a bigger shortcoming if one wants the robot to perform running gaits, jumps, or other motion where the dynamic effects of swing legs may be used to balance the body.

2.8 Data-driven methods using privileged learning

In recent years, parts of the field has taken a turn towards the use of statistical learning methods, mostly in the form of so-called Deep Reinforcement Learning (DRL). An example of this is a recent paper out of the Robotic Systems Lab at ETH. In [13], Miki et.al. introduce a method using learning in simulation to overcome the usual problem of training data shortage often encountered in machine learning.

The main contribution of the paper is to synthesize a controller which trades off between the robustness of proprioceptive locomotion and the efficiency of exteroceptive locomotion in a principled way, without the use of hand-crafted heuristics to weight exteroceptive input. The method operates instead with an "integrated belief state" about its state and the environment, which is informed in part by exteroception and proprioception respectively.

The paper uses a privileged learning approach: At first, a controller is trained to act optimally given full ground-truth knowledge of the environment. Then, a

new controller is trained to predict the privileged controller's behavior given its own incomplete or flawed knowledge of the environment, which stands in for the rich range of ways in which exteroceptive sensors may fail.

By not modeling specific sensor failure modes or environment uncertainty explicitly, the paper argues, the controller learns the most efficient trade-off between exteroception and proprioception for flawed sensor input, including sensors being disabled completely. The results are obtained without any fine-tuning of the resulting policy, instead transferring it directly to hardware. The paper boasts impressive results, demonstrating robust and fast locomotion in a range of real-world scenarios. The longest and most challenging of these is traversing the hiking route Etzel Kulm in Switzerland, in total 2.2 km with a variety of rough terrain, reaching the summit slightly faster than the route sign's estimated time of a human hiker. This hike was completed without a single failure, only stopping to swap batteries.

In this way, the paper may credibly claim to have achieved human-level performance in traversing rough terrain. However, even with its impressive results during empirical evaluation, the controller can boast no claims of robustness or even stability on theoretical grounds. While this may sound like a somewhat pedantic objection, this is a well-known trait of learning based methods which makes one unable to guarantee that the network functions as expected even under conditions that are seemingly well within its demonstrated capabilities. Indeed, convolutional neural networks trained for image classification have been known to be prone to "one pixel attacks" in which modifying the value of a single pixel of an image can take it from being accurately identified to grossly mis-classified (exemplified in [14]).

While there are ways of ameliorating such shortfalls (such as regularization), they are still a demonstration that methods whose workings are hard to explain are prone to fail in ways which are equally hard to anticipate. For operations where robustness and safety are paramount, such shortcomings pose real issues.

2.9 Choice of method

We have presented a selection of methods from the existing literature on quadrupedal locomotion. The methods, along with their apparent strengths and weaknesses have been discussed. In choosing methods for further elaboration, implementation and evaluation through simulation, we have used the criteria we outlined in chapter 1. After evaluating the apparent merits of each method, we believe that a combination of the approaches presented in [6] and [4] presents the most promising starting point for the following reasons:

- In treating the control problem as an optimization problem, it is relatively straightforward to define objectives so as to reach solutions that are (at least locally) optimal with respect to e.g. torque use
- The use of high-order models and full system dynamics increases the likelihood that such solutions are also close to optimal for the physical system
- Defining gait patterns through optimization constraints should allow for the synthesis of a wide variety of gaits without modification to the overarching method
- The resulting controller in combination with the selected post-processing method should yield controllers that can be shown to be exponentially stable in analysis

3

System Modeling

The robotic system to be modeled is the quadrupedal robot ASTRo, short for Articulated Sprawling Tetrapod Robot. This robot is (under reasonable assumptions on velocities and applied forces/torques) well-approximated as a floating-base rigid multibody system. The robot has one floating base which is not directly actuated, as well as four legs, each having 3 actuated joints. The joint configuration of each leg is similar: The first joint in the hip/shoulder which rotational axis is normal to the dorsal plane of the robot (parallel to the yaw axis when the robot is in an upright orientation). This is followed by a joint between the hip and upper leg whose axis is in the dorsal plane (normal to the length of the leg), and a knee joint with an axis parallel to the second axis. See fig. 3.1 for an illustration of this.

3.1 System kinematics

The robot base, by virtue of being a floating base, has 6 Degrees of Freedom (DOFs) - 3 positional and 3 rotational. For the orientation of the robot we choose an intrinsic XYZ Tait-Bryan angle parametrization. While such a parametrization has a singularity at $\theta = \frac{\pi}{2}$ (and is poorly conditioned nearby), this configuration is not supposed to be

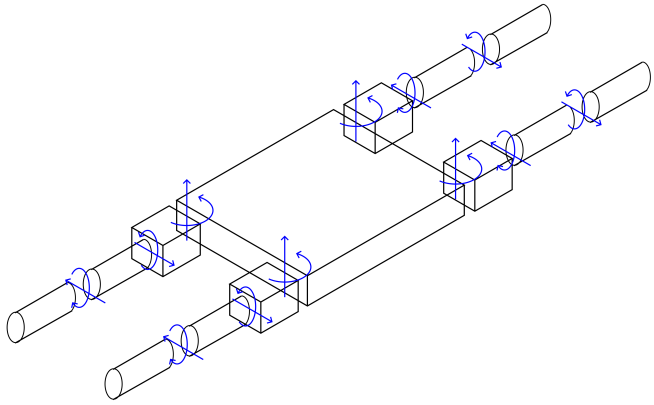


Figure 3.1: Illustration of rotational axes for a sprawling joint configuration. Blue lines indicate rotational axes and direction

reached at any point during a regular gait. Each joint has only a single orientational DOF, around its rotational axis. These DOFs are ordered by leg in the order front left - rear left - front right - rear right, and within each leg by their distance from the base.

The full set of generalized coordinates then becomes

$$\mathbf{q} = \begin{pmatrix} \mathbf{p}_b \\ \boldsymbol{\phi}_b \\ \boldsymbol{\theta}_{\text{fl}} \\ \boldsymbol{\theta}_{\text{rl}} \\ \boldsymbol{\theta}_{\text{fr}} \\ \boldsymbol{\theta}_{\text{rr}} \end{pmatrix} \tag{3.1}$$

where

$$\mathbf{p}_b = (x_b, y_b, z_b)^\top \in \mathbb{R}^3,$$
$$\boldsymbol{\phi}_b = (\psi_b, \theta_b, \phi_b)^\top \in \mathbb{R}^3,$$
$$\boldsymbol{\theta}_i = \left(\theta_{\text{hip1},i}, \theta_{\text{hip2},i}, \theta_{\text{knee},i}\right)^\top \in \mathbb{R}^3, i \in \mathcal{I}_{\text{feet}}$$

where $\mathcal{I}_{\text{feet}} = \{\text{fl}, \text{rl}, \text{fr}, \text{rr}\}$ is the index set for the robot's feet (front left, rear left, front right, rear right).

The position of the robot base in the world frame is given as $\mathbf{t}^w_{w\to b} = \mathbf{p}_b$, and the rotation matrix encoding its orientation is given as $\mathbf{R}^w_b(\boldsymbol{\phi}_b) = \mathbf{R}_x(\psi_b) \cdot \mathbf{R}_y(\theta_b) \cdot \mathbf{R}_z(\phi_b)$.

Let l_x and l_y be the unsigned distances to each of the hip abduction joints, in the x and y directions respectively, from the base origin in the base coordinate system. Further, let l_1 be the distance from the hip abduction joint to the hip joint along the x-axis of the hip coordinate system, let l_2 be the distance from the hip abduction joint to the knee joint along the x-axis of the upper leg coordinate system, and let l_3 be the length from the knee joint to the end of the foot along the x-axis of the lower leg coordinate system. We place the origins of each leg coordinate system at

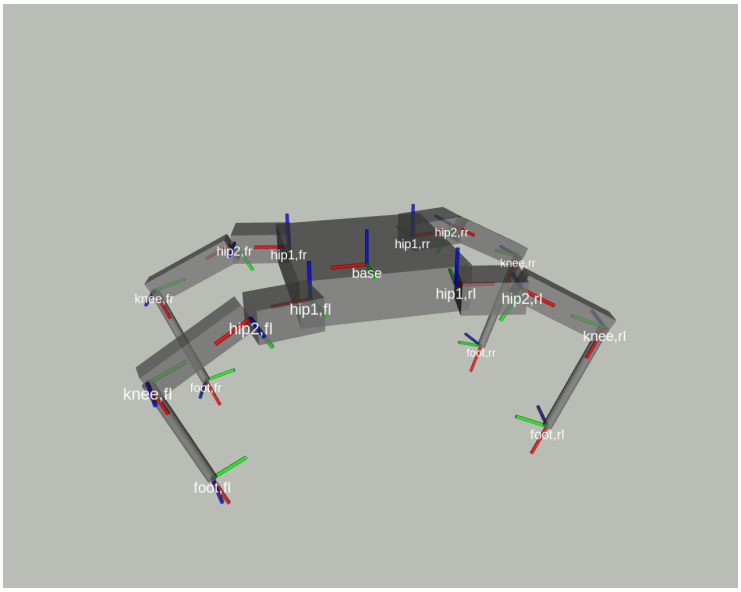


Figure 3.2: Illustration of robot links and coordinate frames

$$
\begin{aligned}
\mathbf{t}^b_{b\to fl} &= \left(l_x, l_y, 0\right)^\top \\
\mathbf{t}^b_{b\to rl} &= \left(-l_x, l_y, 0\right)^\top \\
\mathbf{t}^b_{b\to fr} &= \left(l_x, -l_y, 0\right)^\top \\
\mathbf{t}^b_{b\to rr} &= \left(-l_x, -l_y, 0\right)^\top
\end{aligned}
\tag{3.2}
$$

and, using the Denavit-Hartenberg (DH) convention, we sum up the transformation from each leg coordinate system to the coordinate system at the end of its foot in table 3.1. See fig. 3.2 for an illustration of the robot kinematics with links and corresponding coordinate frames. Frames 1, 3, 5 and 6 (as well as the base frame) from table 3.1 are shown.

The base center of mass is located at the origin of the base coordinate system. For

Leg	Frame	d	θ	a	α
fl	1 (frame at hip joint 1)	0	$\theta_{hip1,fl}$	0	0
	2	0	0	l_1	$-\frac{\pi}{2}$
	3 (frame at hip joint 2)	0	$\theta_{hip2,fl}$	0	0
	4	0	0	l_2	0
	5 (frame at knee joint)	0	$\pi - \theta_{knee,fl}$	0	0
	6 (end of foot)	0	0	l_3	0
rl	1 (frame at hip joint 1)	0	$\theta_{hip1,rl} + \pi$	0	0
	2	0	0	l_1	$-\frac{\pi}{2}$
	3 (frame at hip joint 2)	0	$\theta_{hip2,rl}$	0	0
	4	0	0	l_2	0
	5 (frame at knee joint)	0	$\pi - \theta_{knee,rl}$	0	0
	6 (end of foot)	0	0	l_3	0
fr	1 (frame at hip joint 1)	0	$\theta_{hip1,rl}$	0	0
	2	0	0	l_1	$-\frac{\pi}{2}$
	3 (frame at hip joint 2)	0	$\theta_{hip2,rl}$	0	0
	4	0	0	l_2	0
	5 (frame at knee joint)	0	$\pi - \theta_{knee,rl}$	0	0
	6 (end of foot)	0	0	l_3	0
rr	1 (frame at hip joint 1)	0	$\theta_{hip1,rl} - \pi$	0	0
	2	0	0	l_1	$-\frac{\pi}{2}$
	3 (frame at hip joint 2)	0	$\theta_{hip2,rl}$	0	0
	4	0	0	l_2	0
	5 (frame at knee joint)	0	$\pi - \theta_{knee,rl}$	0	0
	6 (end of foot)	0	0	l_3	0

Table 3.1: List of DH parameters for the transformation from each leg frame to its respective foot. Visualization made using RVIZ (ROS visualization tool)

Legs	Frame	Link	Translation
fl/rr	1	hip	$(l_{c1}, d_{c1}, 0)^\top$
	4	upper	$(l_{c2}, 0, -d_{c2})^\top$
	5	lower	$(l_{c3}, 0, 0)^\top$
	5	foot	$(l_3, 0, 0)^\top$
rl/fr	1	hip	$(l_{c1}, -d_{c1}, 0)^\top$
	4	upper	$(l_{c2}, 0, d_{c2})^\top$
	5	lower	$(l_{c3}, 0, 0)^\top$
	5	foot	$(l_3, 0, 0)^\top$

Table 3.2: Translation to link Centers of Mass (CoMs) and feet from frames in which they are fixed

completeness, we also give the translation to the center of mass of each link in the frame where it is fixed, in table 3.2.

With this, we may write the positions and orientations of the link CoMs, as well as the of the feet, as

$$\mathbf{p}_{j,i}(\mathbf{q}) = \mathbf{p}_b + \mathbf{R}_b^w(\boldsymbol{\phi}_\mathbf{b})\left(\mathbf{t}_{b\to i}^b + \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 1}\end{bmatrix}\mathbf{T}_{i\to j,i}^b(\boldsymbol{\theta}_i)\begin{pmatrix}\mathbf{t}_{j,i\to j_{\text{cm}},i}^{j,i}\\ 1\end{pmatrix}\right) \tag{3.3a}$$

$$\mathbf{R}_{j,i}^w(\mathbf{q}) = \mathbf{R}_b^w(\boldsymbol{\phi}_b)\begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 1}\end{bmatrix}\mathbf{T}_{i\to j,i}^b(\boldsymbol{\theta}_i)\begin{bmatrix}\mathbf{I}_3\\ \mathbf{0}_{1\times 3}\end{bmatrix} \tag{3.3b}$$

where $j \in \{\text{hip}, \text{upper}, \text{lower}, \text{foot}\},\ i \in \mathcal{I}_{\text{feet}}$

$\mathbf{T}_{i\to j,i}^b$ are constructed by utilizing the DH parameters found in table 3.1, while $\mathbf{t}_{j,i\to j_{\text{cm}},i}^{j,i}$ are found in table 3.2.

3.1.1 Ground contact constraints

For any foot that is in contact with the ground, we impose a kinematic constraint on the position of the foot until lift-off. Note that the constraint is only on position, which

Link	Mass [kg]	I_{xx} [kg m^2]	I_{yy} [kg m^2]	I_{zz} [kg m^2]
Base	10.10	$141 \cdot 10^{-3}$	$143 \cdot 10^{-3}$	$254 \cdot 10^{-3}$
Hip	1.18	$29.9 \cdot 10^{-3}$	$33.3 \cdot 10^{-3}$	$4.6 \cdot 10^{-3}$
Upper leg	1.39	$1.7 \cdot 10^{-3}$	$6.6 \cdot 10^{-3}$	$6.4 \cdot 10^{-3}$
Lower leg	0.28	$66.5 \cdot 10^{-6}$	$3.2 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$

Table 3.3: Masses and inertias of robot links

means we assume 0 rotational friction around the normal axis to the ground.

3.2 System Dynamics

The system base has a mass of $m_b$kg with an inertia matrix $\mathbf{J}_b$kg m^2 around the base CoM. Each leg has identical masses and inertias, which are, respectively, m_{hip}, m_{upper}, m_{lower}, and $\mathbf{I}_{\text{hip}}$, $\mathbf{I}_{\text{upper}}$, $\mathbf{I}_{\text{lower}}$, with inertias being around each link's CoM, in a coordinate system fixed to the link. These values, reported in [3], are recited in table 3.3 for completeness.

The robot has 12 actuators – one for each of its 12 joints – so that $\mathbf{u} \in \mathbb{R}^{12}$. We assume here that we control the torque applied to each motor directly, as opposed to controlling e.g. the current and subsequently modeling motor dynamics. With this assumption, the transfer from each input to the corresponding generalized input force is simply $\tau_i = u_i$ for the joint coordinates, while the pose of the robot base is not directly affected by the input. Thus, $\mathbf{B} \in \mathbb{R}^{18\times12}$:

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{6\times12} \\ \mathbf{I}_{12} \end{bmatrix}$$

Using the kinematics and inertial parameters given above, we may calculate the potential energy $\mathcal{P}$ and kinetic energy $\mathcal{K}$ of the robot as a function of $\mathbf{q}, \dot{\mathbf{q}}$. From this, we define the Lagrangian as $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{P}(\mathbf{q})$. For $\mathbf{g}_c(\mathbf{q})$ being the stack of $\left\{\mathbf{p}_{f_i}(\mathbf{q})\right\}_{i\in\mathcal{I}_{\text{sf}}}$ – where $\mathbf{p}_{f_i}(\mathbf{q})$ is the position of foot i in the inertial world frame and $\mathcal{I}_{\text{sf}}$

is the index set containing indices of stance feet – we may then use the constrained forced Euler-Lagrange-equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial}{\partial \dot{\mathbf{q}}}\mathcal{L}^{\top}\right) - \frac{\partial}{\partial \mathbf{q}}\mathcal{L}^{\top} = \boldsymbol{\tau} + \frac{\partial}{\partial \mathbf{q}}\mathbf{g}_c^{\top}(\mathbf{q})\boldsymbol{\lambda} \tag{3.4a}$$

$$\mathbf{g}_c(\mathbf{q}) = \mathbf{0} \tag{3.4b}$$

to get the full constrained system dynamics:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\mathbf{u} + \mathbf{J}_c^{\top}\boldsymbol{\lambda} \tag{3.5a}$$

$$\mathbf{J}_c\ddot{\mathbf{q}} + \frac{\partial}{\partial \mathbf{q}}\left(\mathbf{J}_c\dot{\mathbf{q}}\right) = 0 \tag{3.5b}$$

where $\mathbf{D}(\mathbf{q})$ is the generalized mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix, $\mathbf{G}(\mathbf{q})$ represents potential (incl. gravitational) terms:

$$\mathbf{D}(\mathbf{q}) = \sum_{i \in \mathcal{I}_{\text{links}}} \left(m_i \frac{\partial}{\partial \mathbf{q}}\mathbf{p}_i^{\top}(\mathbf{q}) \frac{\partial}{\partial \mathbf{q}}\mathbf{p}_i(\mathbf{q}) + \mathbf{J}_{R,i}^{i\,\top}(\mathbf{q})\, \mathbf{I}_i^i \mathbf{J}_{R,i}^i(\mathbf{q}) \right) \tag{3.6a}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{i \in \mathcal{I}_{\text{links}}} \left(m_i \frac{\partial}{\partial \mathbf{q}}\mathbf{p}_i^{\top}(\mathbf{q}) \frac{\partial}{\partial \mathbf{q}}\dot{\mathbf{p}}_i(\mathbf{q}) + \mathbf{J}_{R,i}^{i\,\top}(\mathbf{q}) \left(\mathbf{I}_i^i \dot{\mathbf{J}}_{R,i}(\mathbf{q}) + \left(\mathbf{J}_{R,i}^i(\mathbf{q})\dot{\mathbf{q}} \right) \times \mathbf{I}_i^i \mathbf{J}_{R,i}^i(\mathbf{q}) \right) \right) \tag{3.6b}$$

$$\mathbf{G}(\mathbf{q}) = \sum_{i \in \mathcal{I}_{\text{links}}} \left(-m_i g \frac{\partial}{\partial \mathbf{q}}\mathbf{p}_i^{\top}(\mathbf{q})\mathbf{e}_3 \right) \tag{3.6c}$$

where

$$\mathcal{I}_{\text{links}} = \{b\} \cup (\mathcal{I}_{\text{feet}} \times \{\text{hip}, \text{upper}, \text{lower}\})$$

$$\mathbf{J}_{R,i}^i(\mathbf{q}) = \sum_{k=1}^{3} \begin{bmatrix} r_{i_{k,3}}^w(\mathbf{q}) \frac{\partial}{\partial \mathbf{q}} r_{i_{k,2}}^w(\mathbf{q}) \\ r_{i_{k,1}}^w(\mathbf{q}) \frac{\partial}{\partial \mathbf{q}} r_{i_{k,3}}^w(\mathbf{q}) \\ r_{i_{k,2}}^w(\mathbf{q}) \frac{\partial}{\partial \mathbf{q}} r_{i_{k,1}}^w(\mathbf{q}) \end{bmatrix}$$

Here, $r_{i_{k,l}}^w(\mathbf{q})$ is entry k, l of the rotation matrix $\mathbf{R}_i^w(\mathbf{q})$ so that $\mathbf{J}_{R,i}^i(\mathbf{q})\dot{\mathbf{q}} = \boldsymbol{\omega}_{w \to i}^i$.

$\mathbf{e}_3 = (0, 0, 1)^\top$, while $g = 9.81$.

$\mathbf{J}_c = \frac{\partial}{\partial \mathbf{q}} \mathbf{g}_c(\mathbf{q})$. $\mathbf{B}$ is the input matrix, and $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers relating to the constraint forces. The constraint forces can be solved for explicitly. As $\mathbf{D}$ is assumed invertible for well-posed models of mechanical systems (all generalized coordinates are associated with non-zero generalized mass), the system can be posed on an explicit control-affine form if desired.

4

Methods

In this chapter we describe the methods used and give an account of some necessary theoretical background material. We choose to base our controller on a Hybrid Zero-Dynamics (HZD) approach, in which an IO-feedback-linearization controller is synthesized through a method of closed-loop trajectory optimization. The method was first presented in [5], where it was applied to the bipedal robot DURUS, and was extended to the mammalian quadruped vision60 in [6].

In order to ensure the exponential stability of the full system state, we have chosen a method that post-processes the controller parameters to ensure exponential orbital stability. The method, first introduced in [4] and elaborated upon in [15][16], formulates the problem of exponential orbital stability as an optimization problem using Bilinear Matrix Inequalities (BMIs). In order to improve computational tractability, the controller is linearized around its current set of parameters. A perturbation of parameters is found that makes the linearized problem exponentially stable, while minimizing the norm of the perturbation to reduce the gap in behavior between the linearized approximation and the actual system. If the true system is not yet exponentially stable for the new set of parameters, this approach can be applied iteratively until an exponentially stabilizing set of parameters is reached.

The trajectory optimization method was chosen due to the generality of the proposed framework; as the method involves few simplifications to the system the resulting gaits should comply with system dynamics also for more dynamic gaits, where higher-order effects such as Coriolis forces become more pronounced. Even for more conservative gaits, it is likely that this consideration of system dynamics results in gaits which are close to being optimal for the physical system as well. The post-processing method was chosen for its ability to synthesize demonstrably stabilizing controllers, and to solve this highly nontrivial problem in a systematic manner.

Sections 4.1 and 4.3 are based on sections from the specialization project report [1] – with some organizational changes done for clarity as well as some bigger extensions to section 4.1.6.

4.1 Hybrid Zero Dynamics

The framework of HZD is a method in which a controller considering the full state and full-order dynamics of the robot is realized. This sets it apart from most other control efforts for quadrupeds, in which clever model simplification and heuristics have been more prominent. This method has been applied successfully in several bipedal locomotion applications, and was extended for use on quadrupedal robots in [6].

This is an input-output feedback linearization approach in which the dynamics of a higher-dimensional system is forced to evolve on a lower-dimensional manifold which is defined by a set of virtual constraints, or outputs. The dynamics as they evolve once constrained to the lower-dimensional manifold is known as the zero-dynamics of the system ([17]).

The framework of HZD extends the notion of zero-dynamics framework to hybrid systems, i.e. systems which exhibit both continuous and discrete dynamics. An example of such a system is a legged robot, in which foot lift-off and impact are phenomena which are well-modeled as discrete jumps in state and transitions between different continuous domains.

4.1.1 Hybrid Dynamical Systems

Adhering to the formulation in [18], the hybrid system of quadruped locomotion can be given as the tuple $\mathcal{H} = (\Lambda, \mathcal{X}, \mathcal{U}, \mathcal{S}, \mathcal{D}, \Delta, FG)$.

Here, Λ represents a directed graph $\Lambda = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices, while $\mathcal{E}$ is the set of edges. Each vertex here represents a continuous dynamical subsystem, while each edge represents a discrete instantaneous transition between two such dynamical systems. We define $\mu : \mathcal{V} \longrightarrow \mathcal{V}$ to be the function mapping each vertex to its succeeding vertex. In this way, we also have that $\mathcal{E} = \{(v \rightarrow \mu(v))\}_{v \in \mathcal{V}}$.

$\mathcal{X}$ is the set of state manifolds for the vertices, i.e. $\mathcal{X} = \{\mathcal{X}_v\}_{v \in \mathcal{V}}$. Likewise, the set of admissible control inputs is $\mathcal{U} = \{\mathcal{U}_v\}_{v \in \mathcal{V}}$. $\mathcal{S} = \{S_{v \rightarrow \mu(v)}\}_{v \in \mathcal{V}}$ is the set of guards so that the instantaneous transition from one domain to the next occurs when the state and control input (x, u) intersects the guard $\mathcal{S}_{v \rightarrow \mu(v)}$. $\mathcal{D} = \{\mathcal{D}_v\}_{v \in \mathcal{V}}$ is the set of domains of admissibility for each continuous dynamical system, so that $\mathcal{D}_v \subseteq \mathcal{X}_v \times \mathcal{U}_v$. $\Delta = \{\Delta_{v \rightarrow \mu(v)}\}_{v \in \mathcal{V}}$ is then the set of reset laws which relate the end state in one domain to the initial state in the next, so that $x^+ = \Delta_{v \rightarrow \mu(v)}(x^-)$ where x^- is the state the instant before impact, while x^+ is the state the instant after impact. $FG = \{(f_v, g_v)\}_{v \in \mathcal{V}}$ is the set of control systems for each domain, according to which the state evolves on vertex v. We assume (f_v, g_v) to be control affine $\forall v \in \mathcal{V}$ so that $\dot{x}_v = f_v(x) + g_v(x)u$ for $(x, u) \in \mathcal{D}_v$. This involves, among others, all systems whose dynamics can be derived from the Euler-Lagrange equations.

The discrete transitions in the case of a quadrupedal robot represent changes in the number and placement of stance legs, a change which is modeled as instantaneous. Thus, each lifting or landing of a foot is represented by such a transition between two vertices. It should be noted that while each vertex $v \in \mathcal{V}$ must be part of at least one directed cycle, the framework in general allows for multiple cycles to exist in Λ. In this case each vertex v will be associated with multiple guards, and which guard is intersected determines which vertex will be given by the successor function.

As the dynamics related to each vertex for this Hybrid Dynamical System (HDS) differs only because of holonomic constraints, it is characterized by which feet are currently supporting the robot and which are swinging. Consider the index set $\mathcal{I}_{\text{feet}}$

as introduced in section 3.1. We now introduce $\mathcal{I}_{\text{sf},v}$ and $\mathcal{I}_{\text{nsf},v}$ to be the index sets containing indices for respectively stance feet and swing feet (non-stance feet) for vertex v, where $\mathcal{I}_{\text{sf},v} \cup \mathcal{I}_{\text{nsf},v} = \mathcal{I}_{\text{feet}}$, $\mathcal{I}_{\text{sf},v} \cap \mathcal{I}_{\text{nsf},v} = \emptyset$.

4.1.2 Continuous dynamics

In the continuous-time domain $\mathcal{D}_v$, the evolution of the state $\mathbf{x} = [\mathbf{q}^\top, \dot{\mathbf{q}}^\top]^\top$ is given by the control system (f_v, g_v), subject to holonomic constraints $\boldsymbol{\eta}_v(\mathbf{q}) = \mathbf{0}$ resulting from static friction on the stance feet. The derivative of holonomic constraints is then $\mathbf{J}_v(\mathbf{q})\dot{\mathbf{q}}$. The system dynamics relating to domain $\mathcal{D}_v$ can then be expressed as an implicit second-order Ordinary Differential Equation (ODE) resulting from the Euler-Lagrange equations as described in the general case in eq. (3.5), given here for vertex v:

$$\begin{aligned} \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) &= \mathbf{B}\mathbf{u} + \mathbf{J}_v^\top(\mathbf{q})\boldsymbol{\lambda} \\ \mathbf{J}_v(\mathbf{q})\ddot{\mathbf{q}} + \frac{\partial}{\partial \mathbf{q}}\left(\mathbf{J}_v\dot{\mathbf{q}}\right)\dot{\mathbf{q}} &= 0 \end{aligned} \tag{4.1}$$

4.1.3 Discrete dynamics

The hybrid system transitions between continuous domains as dictated by its discrete dynamics, Δ. As mentioned in section 4.1.1, the transition between vertices for the system in question can be divided into cases where a stance foot lifts off, and cases where a swing foot hits the ground. The related discrete dynamics are distinctly different for each of the two cases. In the case of lift-off, we assume no instantaneous change in state. Thus, the reset-law $\Delta_{v\to\mu(v)}$ simply becomes the identity map.

In the case of landing, we assume a perfectly plastic impact. We also assume no discontinuous changes in the generalized coordinates $\mathbf{q}$, although the generalized velocities $\dot{\mathbf{q}}$ may have discontinuous jumps during impact. From conservation of generalized momentum, we get

$$\begin{aligned} \mathbf{D}\dot{\mathbf{q}}^+ - \mathbf{D}\dot{\mathbf{q}}^- &= \mathbf{J}_{\mu(v)}\boldsymbol{\lambda}_{\text{impulse}} \\ \implies \dot{\mathbf{q}}^+ &= \dot{\mathbf{q}}^- + \mathbf{D}^{-1}\mathbf{J}_{\mu(v)}\boldsymbol{\lambda} \end{aligned} \tag{4.2}$$

where $\dot{\mathbf{q}}^-$ is the generalized velocity right before impact, while $\dot{\mathbf{q}}^+$ is the generalized velocity right after impact. $\boldsymbol{\lambda}_{\text{impulse}}$ is the intensity of the impulsive contact forces occurring upon impact. These are determined by the generalized momentum of the system before impact, along with the holonomic constraints on the stance feet after impact (velocity of stance feet must be identically zero).

Alongside the assumptions of no abrupt changes to generalized coordinates, i.e. $\mathbf{q}^+ = \mathbf{q}^-$, this determines the reset map $\Delta_{v\to\mu(v)}$ in the cases of impact.

4.1.4 Continuous domain constraints

For each vertex v, there exist an admissible domain $\mathcal{D}_v \subseteq \mathbb{R}^n$. The admissible domain can be constrained by a set of equalities or inequalities imposed on variables. Here, we consider two types of inequalities which define the boundaries of the admissible domain:

First, there are constraints imposed on the contact wrenches (or in the case of point feet, as in this instance, contact forces) of the stance feet, denoted as $\mathbf{v}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v)\boldsymbol{\lambda}_v \geq \mathbf{0}$. These amount to 1) requiring the normal force from the ground on the robot to be positive and 2) requiring the tangential forces to be within the friction cone – or a linearization of it – of the foot, as the model assumes no slipping of the feet:

$$\lambda_{i,v}^{f_z} \geq 0 \tag{4.3a}$$

$$\|\lambda_{i,v}^{f_x}\|_1 \leq \mu\lambda_{i,v}^{f_z} \tag{4.3b}$$

$$\|\lambda_{i,v}^{f_y}\|_1 \leq \mu\lambda_{i,v}^{f_z} \tag{4.3c}$$

where $\lambda_{i,v}^{f_z}$ are the contact forces in the z-direction while $\lambda_{i,v}^{f_x}, \lambda_{i,v}^{f_y}$ are the contact forces in the x- and y-directions respectively, working on foot $i \in \mathcal{I}_{\text{sf},v}$.

With this, we may write $\mathbf{v}(\mathbf{q}_v, \dot{\mathbf{q}}_v)_v$ as

$$\begin{aligned} \mathbf{v}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v) &= \text{blkdiag}\left(\{\mathbf{v}_{i,v}(\mathbf{q}_v, \dot{\mathbf{q}}_v)\}_{i \in \mathcal{I}_{\text{sf},v}}\right) \\ &\text{where} \\ \mathbf{v}_{i,v}(\mathbf{q}_v, \dot{\mathbf{q}}_v) &= \begin{bmatrix} -1 & 0 & \mu \\ 1 & 0 & \mu \\ 0 & -1 & \mu \\ 0 & 1 & \mu \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.4}$$

where blkdiag $(\{\cdot\})$ signifies the blockdiagonal matrix constructed from the elements of the set. We here assume that $\boldsymbol{\lambda}_v$ is a vertical concatenation of $\left\{(\lambda_{i,v}^{f_x}, \lambda_{i,v}^{f_y}, \lambda_{i,v}^{f_z})^\top\right\}_{i \in \mathcal{I}_{\text{sf},v}}$, stacked in the same order as $\mathbf{v}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v)$.

It is here assumed that the robot is walking on flat ground, so that the normal force is identical to the force in the z-direction. For the general case, the constraints in eq. (4.3) should be modified so that λ^{f_n} – the force along the normal direction of the contact surface – is substituted for λ^{f_z}, while $\lambda^{f_{T1}}, \lambda^{f_{T2}}$ – contact forces along two vectors spanning the tangent plane of the contact surface – should be substituted for $\lambda^{f_x}, \lambda^{f_y}$.

In the case of unmodeled uneven terrain, it would thus not be the unexpected height changes per se, but the differences in slope which could cause problems. For a certain unexpected slope, a proportion of the force in the z-direction proportional to the angle would not actually work along the normal vector, but in the tangent plane. This could either increase or decrease the net tangential force depending on alignment with the already existing tangent force. Likewise, a component of the force in the tangent plane would work not in the tangent plane but along the normal vector. This as well could both increase and decrease the net normal force. However, if normal force is decreased below 0 this would result in violation of **??** regardless of what the tangential force is.

Physically, the violation of eq. (4.3a) would result in breaking contact with the

ground and exiting the current continuous domain. violating eq. (4.3b) or eq. (4.3c) would result in slipping, which would likely impact controller performance less seriously. For small unmodeled slopes, however, these effects should be negligible – especially so if a conservative estimate for the friction constant μ is used.

Secondly, there are constraints relating to the state of the system other than the contact forces, denoted $\mathbf{h}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v) \geq \mathbf{0}$. Here, we require that the swing feet are above the ground:

$$\mathbf{h}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v) = \left(\{ z_{\text{foot},i}(\mathbf{q}_v) \}_{i \in \mathcal{I}_{nsf,v}} \right) \tag{4.5}$$

where $(\{\cdot\})$ signifies a column vector resulting from vertically stacking the elements of the set. For the system in question, $z_{\text{foot},i}(\mathbf{q})$ is the third element of $\mathbf{p}_{\text{foot},i}(\mathbf{q})$ as described in eq. (3.3a).

We summarize these constraints as

$$\mathbf{A}_v = \begin{bmatrix} \mathbf{v}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v)\boldsymbol{\lambda}_v(\mathbf{q}_v) \\ \mathbf{h}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v) \end{bmatrix} \geq \mathbf{0} \tag{4.6}$$

4.1.5 Guards

Guards are associated with the transition of the HDS from one continuous domain to another along an edge of the graph. A guard $\mathcal{S}_{v \to \mu(v)}$ is defined as a proper subset of the boundary of the domain, that is, $\mathcal{S}_{v \to \mu(v)} \subset \partial \mathcal{D}_v$.

For a discrete transition to occur, the state must be about to exit $\mathcal{D}_v$ through a guard. Thus, for some chosen element $H_{v \to \mu(v)}$ of eq. (4.6) which determines part of $\partial \mathcal{D}_v$ we can define a corresponding guard as $\mathcal{S}_{v \to \mu(v)} = \{(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}) | H_{v \to \mu(v)} = 0, \dot{H}_{v \to \mu(v)} < 0\}$. In this paper no-slip contact between the stance feet and the ground are always assumed. Thus, there are two types of guard conditions.

Firstly, if a swing foot hits the ground, part of the boundary associated with $\mathbf{h}_v$ is reached, and the system undergoes a discrete jump to a continuous system in which the previous swing foot is now a stance foot. Secondly, if the normal force of any stance foot becomes zero with a negative derivative, this indicates that the associated

foot is about to lift from the ground and become a swing foot.

As each domain differs by which feet are currently supporting the robot and which ones are swinging, each guard condition may be associated with a foot. Within the framework as presented, as only one scalar guard condition may be associated with a guard and thus with a transition, edges ($v \rightarrow \mu(v)$) will only occur between a two vertices that differ only by the addition or removal of exactly one stance foot. For edges where no new stance feet are added, some swing foot is added which determines the guard of the edge. For edges where no new swing feet are added, some stance foot is added which determines the guard of the edge. We may now state the guard conditions as

$$H_{v \rightarrow \mu(v)} = \begin{cases} z_i, i = \mathcal{I}_{\text{nsf},v} \cap \mathcal{I}_{\text{sf},\mu(v)} & \text{if } \mathcal{I}_{\text{sf},v} \cap \mathcal{I}_{\text{nsf},\mu(v)} = \emptyset \\ \lambda_i^{f_z}, i = \mathcal{I}_{\text{sf},v} \cap \mathcal{I}_{\text{nsf},\mu(v)} & \text{if } \mathcal{I}_{\text{nsf},v} \cap \mathcal{I}_{\text{sf},\mu(v)} = \emptyset \end{cases} \tag{4.7}$$

4.1.6 Zero-dynamics

The system is to be controlled using an IO feedback linearization controller, in which a control signal is designed so as to drive a set of virtual holonomic constraints, or "outputs", to zero. As described in [5, 18], these outputs are typically the deviation in some given state, or combination of states, from a desired set-point or time-parametrized trajectory:

$$\mathbf{y}_v(\mathbf{q}, \dot{\mathbf{q}}, t) = \mathbf{y}_{v,a}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{y}_{v,d}(t) \tag{4.8}$$

where $\mathbf{y}_{v,a}(\mathbf{q}, \dot{\mathbf{q}})$ is referred to as the actual output, a combination of system states, while $\mathbf{y}_{v,d}(t)$ is referred to as the desired output. Here, each $\mathbf{y}_{v,d}$ is parametrized as a Bézier curve, with corresponding parameter vector $\boldsymbol{\alpha}_v$, i.e. $\mathbf{y}_{v,d} = \mathbf{y}_{v,d}(t, \boldsymbol{\alpha}_v)$. By driving the outputs $\mathbf{y}_v$ to zero, the state is driven exponentially to a lower-dimensional manifold of the state space, called the zero-dynamics manifold. Outputs can be grouped by their relative degree. An output with relative degree r_i is one for which the system input directly affects its $r_{i_{th}}$ derivative. Consequently, the dynamics of an output of relative degree r_i (when decoupled) are modeled by an $r_{i_{th}}$-order ODE. For second order

systems, we may have outputs of either relative degree 1 (outputs from generalized velocities) and relative degree 2 (outputs from generalized coordinates). We may group all outputs of the same relative degree as a vector and denote them as $\mathbf{y}_{1,v}$ and $\mathbf{y}_{2,v}$ respectively.

The zero-dynamics manifold is a manifold where all outputs are identically zero, and which has (for the closed-loop system) the property of forward invariance: That is, when the system state has reached the manifold it stays there for all future time. In addition to $\mathbf{y}_{r_i,v}$ being 0, all output derivatives not directly affected by inputs must also equal 0. We then arrive at this definition of the zero-dynamics manifold:

$$\mathcal{Z}_v = \left\{(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{D}_v^X : \mathbf{y}_{1,v} = 0, \mathbf{y}_{2,v} = 0, \dot{\mathbf{y}}_{2,v} = 0\right\} \tag{4.9}$$

where $\mathcal{D}_v^X$ is the subset of $\mathcal{X}_v$ so that $\mathcal{D}_v = \left(\mathcal{D}_v^X \times \mathcal{D}_v^U\right) \subset (\mathcal{X}_v \times \mathcal{U}_v)$.

For the case of walking, this lower-dimensional manifold should contain an orbitally stable or stabilizable periodic behavior, which is the gait. The driving of system dynamics exponentially to the zero-dynamics is realized by designing $\mathbf{u}$ so that

$$\begin{aligned} \dot{\mathbf{y}}_{1,v}(\dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) &= -\epsilon \mathbf{y}_{1,v}(\dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) \\ \ddot{\mathbf{y}}_{2,v}(\mathbf{q}, \dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) &= -2\epsilon \dot{\mathbf{y}}_{2,v}(\mathbf{q}, \dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) - \epsilon^2 \mathbf{y}_{2,v}(\mathbf{q}, t, \boldsymbol{\alpha}_v) \end{aligned} \tag{4.10}$$

where ϵ is a freely chosen tuning parameter. Roughly speaking, outputs with relative degree 2 correspond to virtual holonomic constraints (as they depend only on configuration) while outputs with relative degree 1 correspond to virtual nonholonomic constraints, and may enforce constraints on the derivatives of the configuration variables which correspond to generalized velocity.

Now, consider a set of chosen outputs $\mathbf{y}_v = [\mathbf{y}_{1,v}^\top \quad \mathbf{y}_{2,v}^\top]^\top$. It is important to choose the outputs in such a way that the map from $\mathbf{q}$ to $\mathbf{y}$ is a diffeomorphism. This is to say that the map is invertible, and that both the map and its inverse are differentiable. If and only if $rank\left(\left.\frac{\partial \mathbf{y}}{\partial \mathbf{q}}\right|_{\mathbf{q}_0}\right) = n$ (n being the dimension of the configuration space) the map is locally diffeomorphic in a neighborhood around $\mathbf{q}_0$ ([19, p. 508]). This map being diffeomorphic further implies that the matrix given in eq. (4.11) – called the

decoupling matrix – is invertible ([20]).

$$\mathcal{A}_v(\mathbf{q}, \dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) = \begin{bmatrix} L_g \mathbf{y}_{1,v}(\dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) \\ L_g L_f \mathbf{y}_{2,v}(\mathbf{q}, t, \boldsymbol{\alpha}_v) \end{bmatrix} \tag{4.11}$$

where L_f, L_g are the Lie derivatives with respect to f and g respectively, as introduced in section 4.1.1. Note that this denotation of the decoupling matrix is specific to the case with two vector outputs, $\mathbf{y}_1$ with relative degree 1 and $\mathbf{y}_2$ with relative degree 2. In this case, choosing $\mathbf{u}$ as in eq. (4.12) ensures the desired output dynamics as shown in eq. (4.10) (dependencies omitted for space reasons).

$$\mathbf{u}_v = -\mathcal{A}_v^{-1} \left(\begin{bmatrix} L_{f_v} \mathbf{y}_{1,v} \\ L_{f_v}^2 \mathbf{y}_{2,v} \end{bmatrix} + \begin{bmatrix} \epsilon \mathbf{y}_{1,v} \\ 2\epsilon \dot{\mathbf{y}}_{2,v} + \epsilon^2 \mathbf{y}_{2,v} \end{bmatrix} \right) \tag{4.12}$$

As can be seen from eq. (4.10), the output dynamics are exponentially stable and the zero-dynamics manifold is forward-invariant throughout the continuous domain. If the manifold is also invariant through discrete transitions between domains, i.e. $(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{Z}_v \implies \Delta_{v \to \mu(v)}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{Z}_{\mu(v)} \forall (\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{S}_{v \to \mu(v)} \forall v \in \mathcal{V}$, then we say that the system has a Hybrid Zero Dynamics. However, as discussed in section 4.1.3, the discrete dynamics $\Delta_{v \to \mu(v)}$ of a domain will in general preserve the configuration of the system across a domain (ignoring the potential reset map) but will not in general preserve the generalized velocities of the system.

Furthermore, the discrete dynamics of the system are not a function of the actuation, so that we will not have a means of counteracting such jumps in generalized velocities through control action. Thus it is impossible in general to ensure this invariance through certain discrete transitions, such as impacts. We may then consider the partial zero dynamics manifold, which is the manifold defined by only the position-modulating outputs:

$$\mathcal{PZ}_v = \left\{ (\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{D}_v^X : \mathbf{y}_{2,v} = 0, \dot{\mathbf{y}}_{2,v} = 0 \right\} \tag{4.13}$$

We denote that the manifold (across the entire hybrid system) $\mathcal{PZ} = \bigcup_{v \in \mathcal{V}} \mathcal{PZ}_v$ as

hybrid invariant if it is invariant over all continuous domains and $(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{PZ}_v \implies \Delta_{v\to\mu(v)}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{PZ}_{\mu(v)} \forall v \in \mathcal{V}$. If the closed-loop system has such a hybrid invariant manifold $\mathcal{PZ}$, then we say that the system has a Partial Hybrid Zero Dynamics (PHZD). This relaxation is necessary to encompass hybrid systems with impact dynamics, which are typically present in legged locomotion.

We now stack eq. (3.5) and eq. (4.10) in implicit form:

$$\mathbf{F}_v(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{u}, \boldsymbol{\lambda}, t, \boldsymbol{\alpha}_v) = \begin{pmatrix} \mathbf{D}_v(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}_v(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}_v(\mathbf{q}) - \mathbf{B}_v\mathbf{u} - \mathbf{J}_{c,v}^\top(\mathbf{q})\boldsymbol{\lambda} \\ \mathbf{J}_{c,v}(\mathbf{q})\ddot{\mathbf{q}} + \frac{\partial}{\partial \mathbf{q}}\left(\mathbf{J}_{c,v}(\mathbf{q})\dot{\mathbf{q}}\right) \\ \dot{\mathbf{y}}_{1,v}(\dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) + \epsilon \mathbf{y}_{1,v}(\dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) \\ \ddot{\mathbf{y}}_{2,v}(\mathbf{q}, \dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) + 2\epsilon\dot{\mathbf{y}}_{2,v}(\mathbf{q}, \dot{\mathbf{q}}, t, \boldsymbol{\alpha}_v) + \epsilon^2 \mathbf{y}_{2,v}(\mathbf{q}, t, \boldsymbol{\alpha}_v) \end{pmatrix} \tag{4.14}$$

so that $\mathbf{F}_v(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{u}, \boldsymbol{\lambda}, t, \boldsymbol{\alpha}_v) \equiv 0$ signifies that $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, t)$ satisfy the constrained system dynamics of vertex v, and that the chosen outputs parametrized by $\boldsymbol{\alpha}_v$ tend exponentially towards 0.

4.1.7 IO-linearization for overconstrained systems

A nonlinear n-dimensional system subject to certain assumptions about its inputs and outputs, may be transformed into an equivalent system where a subset of the states are the outputs and their derivatives up to their relative degree.

A number of unobservable states are added to ensure that the transformation between the state space of the original and the transformed system is a diffeomorphism, at least locally. For an in-depth discussion on feedback linearization and IO-linearization, we refer the reader to either [19] or [21]. Consider the system equations of the transformed system, which are reproduced for convenience from [21, pp. 160–164] in eq. (4.15):

$$\dot{\xi}_1^i = \xi_2^i \tag{4.15a}$$

$$\dot{\xi}_2^i = \xi_3^i \tag{4.15b}$$

$$\vdots$$

$$\dot{\xi}_{r_i}^i = b_i(\boldsymbol{\xi}, \boldsymbol{\eta}) + \sum_{j=1}^{m} a_{ij}(\boldsymbol{\xi}, \boldsymbol{\eta}) u_j \tag{4.15c}$$

$$\dot{\boldsymbol{\eta}} = \mathbf{q}(\xi, \eta) + \sum_{j=1}^{m} \mathbf{p_j}(\boldsymbol{\xi}, \boldsymbol{\eta}) u_j \tag{4.15d}$$

$$y_i = \xi_1^i \tag{4.15e}$$

Here, the states ξ_j^i relate to the i_{th} output of the original system, r_i is the relative degree of the i_{th} output, $\boldsymbol{\eta}$ are the internal (unobservable) dynamics of the system, and b_i can be written as a function of $\mathbf{x}$ as $b_i(\mathbf{x}) = L_f^{r_i} h_i(\mathbf{x})$. We may now define $\boldsymbol{\xi}_r \triangleq \left(\xi_{r_1}^1, \xi_{r_2}^2, \ldots, \xi_{r_{m_y}}^{m_y}\right)^\top$, with the following dynamics:

$$\dot{\boldsymbol{\xi}}_r = \mathbf{b}(\mathbf{x}) + \mathcal{A}(\mathbf{x})\mathbf{u} \tag{4.16}$$

where $\mathbf{b}(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_{m_y}(\mathbf{x})]^\top$ and $\mathcal{A}$ is the decoupling matrix, which is defined in the general case as

$$\mathcal{A}(\mathbf{x}) = \begin{bmatrix} L_{\mathbf{g}_1} L_{\mathbf{f}}^{r_1 - 1} h_1(\mathbf{x}) & \ldots & L_{\mathbf{g}_{m_u}} L_{\mathbf{f}}^{r_1 - 1} h_1(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ L_{\mathbf{g}_1} L_{\mathbf{f}}^{r_{m_y} - 1} h_{m_y}(\mathbf{x}) & \ldots & L_{\mathbf{g}_{m_u}} L_{\mathbf{f}}^{r_{m_y} - 1} h_{m_y}(\mathbf{x}) \end{bmatrix} \tag{4.17}$$

The goal of IO-linearization is to choose $\mathbf{u}$ so that $\dot{\boldsymbol{\xi}}_r = \mathbf{v}$ where $\mathbf{v}$ is chosen so that the (now linear) output dynamics are exponentially stable. To proceed, the relative degree of each output must be known and constant, and the decoupling matrix must

be invertible. In this case, the input $\mathbf{u}$ is chosen according to the control law

$$\mathbf{u}(\mathbf{x}) = \mathcal{A}^{-1}(\mathbf{x})\,(-\mathbf{b}(\mathbf{x}) + \mathbf{v}) \tag{4.18}$$

For the method of IO-linearization as described above to work, the decoupling matrix $\mathcal{A}$ must be square and full rank – as its inverse is used for calculating $\mathbf{u}$. The decoupling matrix has as many rows as there are outputs, and as many columns as there are actuators. Thus, in the method as described above, there must be as many outputs as there are actuators – and actuated degrees of freedom. However, in the case of multi-contact robotics, one must take extra care when determining how many actuated degrees of freedom are actually present.

A legged robot is often in some contact configuration which makes it underactuated – as has been mentioned already – this is what constitutes dynamic walking as opposed to static walking. However, as has been noted in [6] and more in-depth in [22, pp. 62-63], a robot may simultaneously be underactuated and overconstrained. An overconstrained system has directions along which internal forces may be enacted by different actuators in such a way as to cancel each other out; that is to say, there are several different choices of the vector $\mathbf{u}$ which lead to the same system dynamics, but result in different internal forces.

Consider the four-legged robot system eq. (3.5) in a configuration where two of its legs are in contact with the ground. The unconstrained system has 18 DOFs. Each of the holonomic foot position constraints impose 3 independent constraints, leaving the constrained system with 12 DOFs. At first glance, as the robot has 12 actuators, one might imagine that this leaves us with a fully actuated system. However, consider the line passing through each of the foot contacts (see fig. 4.1 for an illustration). As the feet are modeled as point contacts, there is no ground reaction forces or moments which may be enacted by the stance feet so as to rotate the robot body around this axis. Thus, we have one unactuated DOF. As the total DOFs equals the sum of actuated and unactuated DOFs, it follows that the system must have only 11 actuated DOFs. Having 12 actuators, the system with two contact feet must be overconstrained.

The IO-linearization method can be applied straightforwardly to underactuated

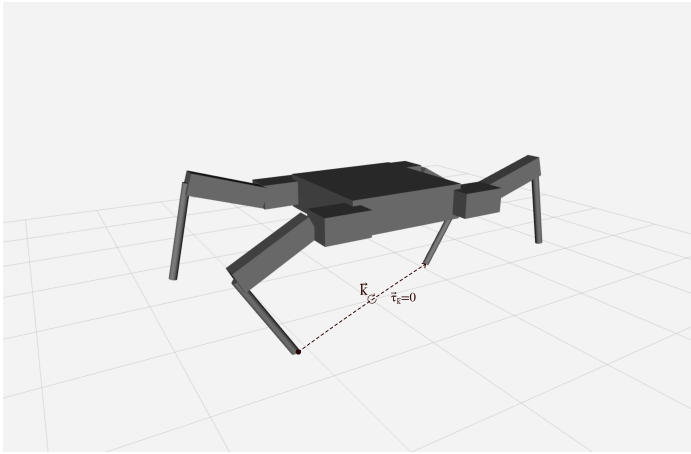


Figure 4.1: Illustration of an axis $\vec{k}$ around which the available contact forces can generate no torque/moment

systems, so long as the number of outputs match the number of actuators, and the relation between the chosen outputs and the actuators are such that $\mathcal{A}$ has full rank at all times. However, as far as the author of this thesis could find, there is not a canonical way to apply IO-linearization to overconstrained systems. Indeed, in [6] in order to circumvent this issue, the author suggests simply to remove one stance leg actuator from each continuous domain. This again would leave one of the stance leg joints unactuated, rendering the system only underactuated (not overconstrained).

There are several objections one might have to this approach. Firstly, the choice of which actuator is "turned off" is not justified or elaborated upon further. Secondly, even if such a justification were given, the removal of one available actuator hardly seems like an optimal solution in any sense. It is likely – or at least conceivable – that a "better" (with respect to minimization of e.g. actuator torques or minimization of tangential contact forces) choice of actuation will involve some combination of all of the actuators collaborating in an overconstrained system, as opposed to leaving joints unactuated to avoid overconstrainedness.

We propose a modification to IO-feedback-linearization which makes it readily

applicable to overconstrained systems. This is a modification to the approach suggested in [6] which is arguably more principled, as well as resulting in concrete benefits with respect to torque use. We will show that

1. For an overconstrained system with m_u actuators and a choice of m_y outputs for which some subset of m_y actuators can be chosen so as to successfully synthesize a conventional IO-linearization controller, the output-dynamics of our closed-loop system will exhibit the same behavior.

2. Of all inputs that result in this behavior of the output-dynamics, our controller ensures that the inputs are as small as possible, in a least-squares sense.

3. Given sufficiently smooth system dynamics and output functions, the resulting controller can be made continuous, and, furthermore, its derivative can be made well-defined and continuous.

Suppose a system with state $\mathbf{x} \in \mathbb{R}^n$, input $\mathbf{u} \in \mathbb{R}^{m_u}$ and control-affine dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$ with

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \mathbf{g}_1(\mathbf{x}) & \mathbf{g}_2(\mathbf{x}) & \dots & \mathbf{g}_{m_u}(\mathbf{x}) \end{bmatrix} \tag{4.19}$$

Suppose further that the system has m_y actuated DOFs with $m_y < m_u$ so that the system is overconstrained. Let $\mathcal{S}_g = \{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{m_u}\}$ and let $\mathcal{I}_g$ be its index set, $\{\mathcal{I}_g = 1, 2, \dots, m_u\}$.

Suppose a selection of actuators and corresponding set $\mathcal{S}_{\tilde{g}}$ given by an index set $\mathcal{I}_{\tilde{g}} \subset \mathcal{I}_g$, $|\mathcal{I}_{\tilde{g}}| = m_y$ so that $\mathcal{S}_{\tilde{g}} = \{\mathbf{g}_i\}_{i \in \mathcal{I}_{\tilde{g}}}$ with corresponding $\tilde{\mathbf{u}} \in \mathbb{R}^{m_y}$. The auxiliary system will have system dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \tilde{\mathbf{g}}(\mathbf{x})\tilde{\mathbf{u}}$, where

$$\tilde{\mathbf{g}}(\mathbf{x}) = \begin{bmatrix} \mathbf{g}_{\mathcal{I}_{\tilde{g}}\{1\}}(\mathbf{x}) & \mathbf{g}_{\mathcal{I}_{\tilde{g}}\{2\}}(\mathbf{x}) & \dots & \mathbf{g}_{\mathcal{I}_{\tilde{g}}\{m_y\}}(\mathbf{x}) \end{bmatrix} \tag{4.20}$$

As the original system is overconstrained, there is some index set $\mathcal{I}_{\tilde{g}}$ so that the number of actuated DOFs remains unchanged, but the system is no longer overconstrained. Suppose we have a set $\mathcal{S}_h = \{h_i(\mathbf{x})\}$ of m_y outputs so that, when considering the system as actuated by $\tilde{\mathbf{u}}, \tilde{\mathbf{g}}(\mathbf{x})$ the decoupling matrix $\tilde{A} \in \mathbb{R}^{m_y \times m_y}$ has full rank.

We may now construct a non-square "decoupling matrix" $\mathcal{A} \in \mathbb{R}^{m_y \times m_u}$ to the original system. As each row of the decoupling matrix corresponds to an output while each column corresponds to an input, all columns of $\tilde{\mathcal{A}}$ are also columns of $\mathcal{A}$. Thus, the rank of $\mathcal{A}$ must be at least the rank of $\tilde{\mathcal{A}}$, which is assumed to be full rank. But the rank of $\mathcal{A}$ can also not be greater than its number of rows. Thus, rank $(\mathcal{A})$ = rank $\left(\tilde{\mathcal{A}}\right)$.

We now compare the output dynamics of the system actuated by $\tilde{\mathbf{u}}$, $\mathcal{S}_{\tilde{\mathbf{g}}}$ to the ones of the system actuated by $\mathbf{u}$, $\mathcal{S}_{\mathbf{g}}$. In deriving the equations to the transformed system, the functions $\left\{\mathbf{g}_i\right\}_{i\in\mathcal{I}_{\tilde{\mathbf{g}}}}$, resp. $\left\{\mathbf{g}_i\right\}_{i\in\mathcal{I}_{\mathbf{g}}}$, show up only in the equations of $\dot{\xi}^i_{r_i}$ (through $\tilde{a}_{ij}(\boldsymbol{\xi},\boldsymbol{\eta})$ resp. $a_{ij}(\boldsymbol{\xi},\boldsymbol{\eta})$) as well as in the equations of $\dot{\boldsymbol{\eta}}$ (through $\tilde{\mathbf{p}}_j(\boldsymbol{\xi},\eta)$, resp. $\mathbf{p}_j(\boldsymbol{\xi},\eta)$).

Our goal is to render the output-dynamics of the overconstrained system equal to the output-dynamics of the dynamical system rendered by substituting $\mathcal{S}_{\tilde{g}}$ for $\mathcal{S}_g$. However, as $\mathcal{A}$ is not square, we may not simply invert it as in eq. (4.18).

The columns of $\mathcal{A}$ and $\tilde{\mathcal{A}}$ both span $\mathbb{R}^{m_y}$ (by assumption). Thus, if we want to achieve

$$\mathcal{A}(\mathbf{x})\mathbf{u} \equiv \tilde{\mathcal{A}}(\mathbf{x})\tilde{\mathbf{u}} = -\mathbf{b}(\mathbf{x}) + \mathbf{v} \tag{4.21}$$

we may choose

$$\mathbf{u}(\mathbf{x}) = \mathcal{A}^{+}(\mathbf{x})\left(-\mathbf{b}(\mathbf{x}) + \mathbf{v}\right) \tag{4.22}$$

where $\mathcal{A}^{+}$ is the Moore-Penrose Pseudoinverse (MPP) of $\mathcal{A}$.

The MPP has a number of properties which makes it desirable. Most importantly, for non-overdetermined systems (at least as many linearly independent columns as rows), the MPP solution solves the system exactly. This ensures that our controller as described in eq. (4.22) impacts the output dynamics exactly as would a conventional IO-linearization controller, see eq. (4.21). As the conventional IO-linearization controller can exponentially stabilize the output dynamics by the proper choice of $\mathbf{v}$, so will our controller by the same choice of $\mathbf{v}$. This shows the first claim.

Moreover, the solution yielded by the MPP in the underdetermined case is guaranteed to be the minimum-norm solution in the least-squares sense. This shows the second claim.

Thirdly, if a matrix $\mathbf{M}(x)$ has constant rank then its MPP is continuous, and its derivative is well-defined and continuous [23] (dependency on x omitted for space reasons):

$$\begin{aligned}&\frac{\partial}{\partial x}(\mathbf{M}^+) = \\ &\mathbf{M}^+\left(\frac{\partial}{\partial x}\mathbf{M}\right)\mathbf{M}^+ + \mathbf{M}^+\mathbf{M}^{+\top}\left(\frac{\partial}{\partial x}\mathbf{M}\right)\left(\mathbf{I}-\mathbf{M}\mathbf{M}^+\right) + \left(\mathbf{I}-\mathbf{M}^+\mathbf{M}\right)\left(\frac{\partial}{\partial x}\mathbf{M}\right)\mathbf{M}^{+\top}\mathbf{M}^+\end{aligned} \tag{4.23}$$

Under the assumptions posed, $\mathcal{A}$ will have constant rank m_y. The Lie derivatives in the controller will be smooth under the assumption of sufficiently smooth system dynamics, and $\mathbf{v}$ can be chosen smooth – a typical choice is a linear combination of system outputs and their derivatives. This shows the third claim.

The effect of this choice of $\mathbf{u}$ on the internal dynamics is not the topic of discussion here. However, the construction of conventional IO-linearization controllers give no guarantees on the stability of the internal dynamics to begin with. Thus, the stability of the internal dynamics will be a matter of separate analysis in both cases.

4.2 Direct collocation

There are several ways of transcribing a control problem into a discrete time optimization problem. While some of the simpler methods - such as single and multiple shooting methods - may be computationally cheaper for each time step, there may be a trade-off compared to more complex methods with respect to accuracy as a function of step length. One could say that a collocation based optimization scheme relates to a given implicit integration scheme (depending on which form of collocation) in the same way multiple shooting methods relates to the simple forward-Euler integration scheme. As a result, using a collocation scheme to relate the time steps - while not as straightforward as a simple forward Euler-relation - allows for longer intervals between discretization steps (nodes) while retaining solution accuracy. In the collocation framework one operates with two different types of nodes, interior and cardinal, which alternate throughout the step sequence. The first and the last nodes are both

cardinal, so that (with a 0-indexed set of nodes) even nodes are cardinal, while odd nodes are interior.

For each cardinal node, the state of the system is made a decision variable. The method rests on the notion that one can approximate the state between two cardinal nodes from the system state and slope at these nodes. The slope at each cardinal node is then given by the dynamics of the system. The set of states and slopes at two neighboring cardinal nodes may vary across 4 dimensions in total. If an approximation in between them is desired that matches the state and slope at both endpoints and is completely determined by them, it should have 4 parameters. One possible choice, which we will use in our case, is the cubic polynomial:

$$\mathbf{p}(t) = \sum_{i=0}^{3} \mathbf{a}_i t^i \tag{4.24a}$$

s.t.

$$\mathbf{p}(t^{(n_{c_i})}) = \mathbf{x}^{(n_{c_i})} \tag{4.24b}$$

$$\mathbf{p}(t^{(n_{c_{i+1}})}) = \mathbf{x}^{(n_{c_{i+1}})} \tag{4.24c}$$

$$\dot{\mathbf{p}}(t^{(n_{c_i})}) = \mathbf{f}(\mathbf{x}^{(n_{c_i})}) \tag{4.24d}$$

$$\dot{\mathbf{p}}(t^{(n_{c_{i+1}})}) = \mathbf{f}(\mathbf{x}^{(n_{c_{i+1}})}) \tag{4.24e}$$

$$\tag{4.24f}$$

Let $\Delta t^{(i)} = t^{(i+1)} - t^{(i-1)}$ for each interior node i. Then, by choosing each interior node as lying at the center between its adjacent cardinal nodes, we may approximate both the state and the slope at the interior node in terms of the two adjacent nodes as:

$$\mathbf{x}^{(i)} = \frac{1}{2}\left(\mathbf{x}^{(i+1)} + \mathbf{x}^{(i-1)}\right) + \frac{\Delta t^{(i)}}{8}\left(\mathbf{f}(\mathbf{x}^{(i-1)}) - \mathbf{f}(\mathbf{x}^{(i+1)})\right) \tag{4.25a}$$

$$\dot{\mathbf{x}}^{(i)} = \frac{3}{2\Delta t^{(i)}}\left(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i-1)}\right) - \frac{1}{4}\left(\mathbf{f}(\mathbf{x}^{(i-1)}) + \mathbf{f}(\mathbf{x}^{(i+1)})\right) \tag{4.25b}$$

Finally, the relation between the cardinal nodes $\mathbf{x}^{(i-1)}, \mathbf{x}^{(i+1)}$ is constrained by

imposing $\dot{\mathbf{x}}^{(i)} = \mathbf{f}(\mathbf{x}^{(i)})$. This form of collocation is known as Hermite-Simpson collocation.

4.3 Closed-loop Trajectory Optimization

A large difficulty in designing this type of controllers for walking is the gait design, which relates to finding well-suited specific trajectories $\mathbf{y}_{v,d}(t, \boldsymbol{\alpha}_v)$. These trajectories should result in gaits that are both feasible and stable, and that are energetically efficient.

While hand-crafting these desired trajectories is a possibility, the more common approach is to find these feasible trajectories and corresponding inputs by the use of optimization methods, a method known as optimal control. The framework of nonlinear optimization is both powerful and flexible enough to incorporate holonomic constraints, virtual holonomic and nonholonomic constraints, state inequality constraints and bounds, while searching for solutions that minimize either energy expenditure, average torque or other desired objectives.

In the approach suggested in [5] and later in [6], direct collocation methods are used. As mentioned in section 4.2, such methods allow for greater accuracy at longer time steps, decreasing the number of decision variables needed. In this formulation referred to as "modified Hermite-Simpson collocation", both the state at interior nodes and the slope at cardinal nodes, as well as all constraint forces, are introduced explicitly as defect variables as opposed to being calculated in closed form. Although this introduction of defect variables increases the size of the NLP, it is done to improve the convergence properties of the problem ([5]). It also avoids matrix inversion, which is desirable for a few reasons:

Firstly, there are concerns of numerical stability and accuracy when inverting poorly conditioned matrices. Secondly, matrix inversion is a highly costly operation with time complexity which scales poorly in the number of variables.

The collocation constraints are a set of constraints that forces the state and slope $\mathbf{x}^{(i)} = (\mathbf{q}^{(i)}, \dot{\mathbf{q}}^{(i)})$ and $\dot{\mathbf{x}}^{(i)} = (\dot{\mathbf{q}}^{(i)}, \ddot{\mathbf{q}}^{(i)})$ at the interior nodes to adhere to some interpolation of the states at adjacent cardinal nodes. In Hermite-Simpson collocation, these

constraints equal state and slope to the cubic interpolation of state and quadratic interpolation of slope calculated from the previous and following cardinal nodes. There are in total $\mathcal{N}_v$ nodes approximating the dynamics on vertex v, of which $\mathcal{N}_v^c = (\mathcal{N}_v + 1)/2$ are cardinal nodes and $\mathcal{N}_v^i = (\mathcal{N}_v - 1)/2$ are interior nodes. The collocation constraint on state is $\boldsymbol{\delta}^{(i)}$ while the constraint on slope is $\boldsymbol{\zeta}^{(i)}$ for the i_{th} node:

$$\begin{aligned} \boldsymbol{\delta}^{(i)} &= \mathbf{x}^{(i)} - \frac{1}{2}\left(\mathbf{x}^{(i+1)} + \mathbf{x}^{(i-1)}\right) - \frac{\Delta t^{(i)}}{8}\left(\dot{\mathbf{x}}^{(i-1)} - \dot{\mathbf{x}}^{(i+1)}\right) \\ \boldsymbol{\zeta}^{(i)} &= \dot{\mathbf{x}}^{(i)} - \frac{3}{2\Delta t^{(i)}}\left(\mathbf{x}^{(i+1)} - \mathbf{x}^{(i-1)}\right) + \frac{1}{4}\left(\dot{\mathbf{x}}^{(i-1)} + \dot{\mathbf{x}}^{(i+1)}\right) \end{aligned} \tag{4.26}$$

with $\Delta t^{(i)} = t^{(i+1)} - t^{(i-1)}$, for $i = 1, 3, 5, \ldots, N_v - 2$ where N_v is the total number of nodes The decision variables are, in addition to the states, slopes and input at each node, parameters for the desired output trajectories as well as the placement in time of each cardinal node, $t^{(i)}$ for $i = 0, 2, 4, \ldots, N_v - 1$

As the method includes closed-loop controller in the NLP, it synthesizes a control law for $\mathbf{u}$ directly. This is distinct from typical optimal control, where an open-loop optimal control signal which is assumed to be piecewise linear or constant, is calculated. We denote the constraint on input (dependencies on the right hand side have been omitted for space reasons) as

$$\mathbf{G}_v(\mathbf{q}^{(i)}, \dot{\mathbf{q}}^{(i)}, \mathbf{u}^{(i)}, t^{(i)}, \boldsymbol{\alpha}_v) = \mathbf{u}^{(i)} + \mathcal{A}_v^{-1}\begin{pmatrix} L_{f_v}\mathbf{y}_{1,v} + \epsilon\mathbf{y}_{1,v} \\ L_{f_v}^2\mathbf{y}_{2,v} + 2\epsilon\dot{\mathbf{y}}_{2,v} + \epsilon^2\mathbf{y}_{2,v} \end{pmatrix} \tag{4.27}$$

where $\mathcal{A}_v^{-1}$ may be replaced by $\mathcal{A}_v^{+}$ if the system is overconstrained.

While, as mentioned in section 4.1.1, the graph Λ may in general contain several cycles, for the sake of employing the trajectory optimization tool we restrict Λ to consist of a single cycle so that each vertex v has only one possible successor.

Let $\boldsymbol{\delta}_v = \left(\boldsymbol{\delta}_v^{(1)\top}, \boldsymbol{\delta}_v^{(3)\top}, \ldots, \boldsymbol{\delta}_v^{(N_v-2)\top}\right)^\top$ and $\boldsymbol{\zeta}_v = \left(\boldsymbol{\zeta}_v^{(1)\top}, \boldsymbol{\zeta}_v^{(3)\top}, \ldots, \boldsymbol{\zeta}_v^{(N_v-2)\top}\right)^\top$. Also, let $\mathbf{F}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v, \ddot{\mathbf{q}}_v, \mathbf{u}_v, \boldsymbol{\lambda}_v, \mathbf{t}_v, \boldsymbol{\alpha}_v)$ be the stacked vector of

$\{\mathbf{F}_v(\mathbf{q}^{(i)}, \dot{\mathbf{q}}^{(i)}, \ddot{\mathbf{q}}^{(i)}, \mathbf{u}^{(i)}, \boldsymbol{\lambda}^{(i)}, t^{(i)}, \boldsymbol{\alpha}_v)\}$. Finally, let $\mathbf{G}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v, \mathbf{u}_v, \mathbf{t}_v, \boldsymbol{\alpha}_v)$ be the stacked

vector of $\{\mathbf{G}_v(\mathbf{q}^{(i)}, \dot{\mathbf{q}}^{(i)}, \mathbf{u}^{(i)}, t^{(i)}, \boldsymbol{\alpha}_v)\}$. Then, the full NLP can be stated as follows:

$$
\begin{aligned}
\min_{z} \quad & \mathrm{J}(\mathbf{z}) \quad \text{s.t.} \\
\mathbf{F}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v, \ddot{\mathbf{q}}_v, \boldsymbol{\lambda}_v, \mathbf{t}_v, \boldsymbol{\alpha}_v) &= \mathbf{0} \\
\mathbf{G}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v, \mathbf{u}_v, \mathbf{t}_v, \boldsymbol{\alpha}_v) &= \mathbf{0} \\
\boldsymbol{\delta}_v &= \mathbf{0} \\
\boldsymbol{\zeta}_v &= \mathbf{0} \\
(\mathbf{q}_v^{(\mathcal{N}_v)}, \dot{\mathbf{q}}_v^{(\mathcal{N}_v)}, \mathbf{u}_v^{(\mathcal{N}_v)}) &\in \mathcal{S}_{v \to \mu(v)} \\
(\mathbf{q}_{\mu(v)}^{(0)}, \dot{\mathbf{q}}_{\mu(v)}^{(0)}, \mathbf{u}_{\mu(v)}^{(0)}) &\in \mathcal{S}_{v \to \mu(v)} \\
\mathbf{A}_v(\mathbf{q}_v, \dot{\mathbf{q}}_v, \boldsymbol{\lambda}_v) &\geq \mathbf{0} \\
\mathbf{y}_{2,v}(\mathbf{q}^{(0)}, t^{(0)}, \boldsymbol{\alpha}_v) &= 0 \\
\dot{\mathbf{y}}_{2,v}(\mathbf{q}^{(0)}, \dot{\mathbf{q}}^{(0)}, t^{(0)}, \boldsymbol{\alpha}_v) &= 0 \\
\mathbf{J}_v(\mathbf{q}^{(0)})\dot{\mathbf{q}}^{(0)} &= 0 \\
\forall v \in \mathcal{V}
\end{aligned}
\tag{4.28}
$$

Here, $\mathbf{z}$ is the entire vector of decision variables, and $\mathrm{J}(\mathbf{z})$ is an objective function of choice.

Two typical choices for $\mathrm{J}(\mathbf{z})$ might either be an integral cost over the 2-norm of expended torque:

$$
J(\mathbf{z}) = \sum_{v \in \mathcal{V}} \int_{t^{(0)}}^{t^{(\mathcal{N}_v)}} \|\mathbf{u}_v(\tau)\|_2 d\tau \tag{4.29}
$$

or the 2-norm of expended torque divided by the time interval:

$$
J(\mathbf{z}) = \sum_{v \in \mathcal{V}} \frac{1}{t^{(\mathcal{N}_v)} - t^{(0)}} \int_{t^{(0)}}^{t^{(\mathcal{N}_v)}} \|\mathbf{u}_v(\tau)\|_2 d\tau \tag{4.30}
$$

Here, the second choice might be preferable: Intuitively, one solution for a gait cycle whose integral of torque is 0 is one which lasts for 0 seconds. From this, we

might infer that the first cost function would drive the solver towards solutions which are as short as possible, which might not be the solution that minimizes torque when walking for a given duration of time (not a given number of gait cycles).

4.4 Controller post-processing for orbital stability

The closed-loop dynamics of the system subject to the controller synthesized through the optimization procedure discussed previously exhibit an orbit in system state which corresponds to the desired gait. However, the existence of such an orbit does not guarantee its stability as such. The IO-feedback linearization controller only guarantees the exponential stability of the chosen outputs, driving the system to its zero-dynamics manifold. While for simpler systems one may have some luck in choosing "intuitive" outputs which successfully accomplishes the (asymptotic) stability of the zero-dynamics, selecting the outputs in such a way is in general non-trivial.

In order to keep the system's trajectory close to the orbit over time – given initial conditions which are merely in the close vicinity of the orbit – the orbit must be stable, and in order for such a trajectory to converge to the orbit from some vicinity of it, it must be attractive as well – thus asymptotic stability of some sort is desirable.

Here, we describe a systematic post-processing of the IO-feedback-linearization controller, based on work presented in [4], so as to render the full state orbit in the closed-loop dynamics exponentially orbitally stable, thus achieving our desired goal.

4.4.1 Poincaré return maps

Consider a dynamical system $\mathbf{f}(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^n$. Define the flow

$$\begin{gathered} \varphi(\mathbf{x}_0, t) = \mathbf{x}(t) \\ \text{s.t.} \\ \mathbf{x}(0) = \mathbf{x}_0, \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \; \forall t \geq 0 \end{gathered} \tag{4.31}$$

For a given $\mathbf{x}_0$ the flow describes the system's trajectory through time from the initial condition. Let $\mathcal{S}$ be some $n-1$-dimensional surface which is transverse to $\varphi(\mathbf{x}_0, t)$,

and which intersects it in $\mathbf{x}_0$. Further, let $h_{\mathcal{S}} : \mathbb{R}^n \rightarrow \mathbb{R}$ be the function that defines $\mathcal{S}$ in the following way:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n | h_{\mathcal{S}}(\mathbf{x}) = 0\} \tag{4.32}$$

Then, let $\mathbf{y}_0 \triangleq \text{proj}_{\mathcal{S}}(\mathbf{x}_0)$ be the projection of $\mathbf{x}_0 \in \mathbb{R}^n$ onto the $n-1$-dimensional manifold $\mathcal{S}$ (i.e. a $n-1$-dimensional description of the point on $\mathcal{S}$ that intersects $\mathbf{x}_0$). Let $\text{proj}_{\mathcal{S}}^{-1}(\mathbf{y})$ denote the canonical embedding of $\mathbf{y}$ in $\mathbb{R}^n$ (i.e. the point in $\mathbb{R}^n$ that intersects $\mathcal{S}$ at $\mathbf{y}$). Then, the first Poincaré map $\mathbf{P}_{\mathcal{S}}(\mathbf{y})$ for $\mathcal{S}$ is defined as

$$\begin{aligned} \mathbf{P}_{\mathcal{S}}(\mathbf{y}_0)) &= \text{proj}_{\mathcal{S}}(\varphi(\text{proj}_{\mathcal{S}}^{-1}(\mathbf{y}_0), t_1)) \\ &s.t. \\ \varphi(\text{proj}_{\mathcal{S}}^{-1}(\mathbf{y}_0), t_1) &\in \mathcal{S}, \\ \varphi(\text{proj}_{\mathcal{S}}^{-1}(\mathbf{y}_0), t) &\notin \mathcal{S} \ \forall t \in (0, t_1) \end{aligned} \tag{4.33}$$

If $\mathbf{P}_{\mathcal{S}}(\mathbf{y}_0) = \mathbf{y}_0$, i.e. $\mathbf{y}_0 \triangleq \mathbf{y}^*$ is a fixed point of the Poincaré first return map, then the trajectory originating at $\mathbf{x}_0 \triangleq \mathbf{x}^*$ is periodic and the set $\mathcal{O} = \{\varphi(\mathbf{x}^*, t) | t \geq 0\}$ is an orbit of the system.

4.4.2 Floquet multipliers and Orbital stability

Now, consider at first $\varphi(\mathbf{x}_0, t_1)$ with $\varphi(\cdot)$ as defined in eq. (4.31), where $\mathbf{x}_0$ is the initial condition of the system and t_1 is the time of the trajectory's first return to the Poincaré section $\mathcal{S}$. Its Jacobian $\mathbf{\Phi}(\mathbf{x}_0, t_1) \triangleq \frac{\partial}{\partial \mathbf{x}_0}\varphi(\mathbf{x}_0, t_1)$, is known as the fundamental solution matrix. The set of eigenvalues of this matrix for a periodic solution, $\lambda(\mathbf{\Phi}(\mathbf{x}_0, t_1))$ is known as the Floquet multipliers of the solution, and gives information about the stability properties of the orbit. The linearization of $\varphi(\mathbf{x}, t_1)$ around the fixed point $\mathbf{x}^*$ results in a linear discrete dynamical system of the form

$$\delta\mathbf{x}_{k+1} = \mathbf{\Phi}(\mathbf{x}^*, t1)\delta\mathbf{x}_k \tag{4.34}$$

One of the Floquet multipliers of a periodic solution will necessarily be 1, with

the corresponding eigenvector equal to the direction of flow at the fixed point, i.e. $\mathbf{\Phi}(\mathbf{x}^*, t_1)\mathbf{f}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}^*)$. However, if the rest of the Floquet multipliers have magnitude strictly less than one, then the orbit is locally asymptotically – in fact, exponentially – stable [24].

A closed-form analytical expression for $\varphi(\mathbf{x}_0, t)$ is generally not available, and one may therefore not differentiate $P_{\mathcal{S}}(\mathbf{y})$ with respect to $\mathbf{y}$ in a straightforward manner. However, using the variational equation of $\mathbf{f}$ along the trajectory originating at $\mathbf{x}^*$ one might obtain $\mathbf{\Phi}(\mathbf{x}^*, t_1)$ as the solution of the LTV matrix differential equation

$$\begin{aligned} \dot{\mathbf{\Phi}} &= \left.\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\varphi(\mathbf{x}^*, t)} \mathbf{\Phi}, \\ \mathbf{\Phi}(0) &= \mathbf{\Phi}_0 \end{aligned} \tag{4.35}$$

evaluated at time t_1. Further, from observing that $\varphi(\mathbf{x}_0, 0) = \mathbf{x}_0$ and thus $\frac{\partial \varphi(\mathbf{x}_0, 0)}{\partial \mathbf{x}_0} = \mathbf{I}_n$, it follows that $\mathbf{\Phi}_0 = \mathbf{I}_n$.

4.4.3 Reduced-dimension stability analysis

Suppose we want to arrive at a linear discrete-time system that is exponentially stable when the orbit of the original system is exponentially stable. We may then simply project the system in eq. (4.34) onto the tangent space of $\mathcal{S}$ around $\mathbf{x}^*$ – as $\mathbf{f}(\mathbf{x}^*)$ is by construction transverse.

However, for certain systems and orbits such as a legged robot walking forward, we do not want orbital stability of the full state: At the very least the horizontal position of the base should in fact move forward if the system acts as desired, and so the desired behavior isn't even an orbit in the full state. Thus, it is the system state projected on some lower-dimensional subspace (or in general, manifold) $\tilde{\mathcal{X}}$ of dimension n_{ld} that has an orbit. Thus, we want to study the stability properties by looking at how this projection of the state trajectory intersects with the projection of the Poincaré section onto the same manifold.

For the case at hand, where the state trajectory is an orbit in the state except

horizontal position, $\tilde{\mathcal{X}}$ is the subspace of $\mathbb{R}^n$ spanned by orthonormal unit vectors $\{\mathbf{e}_i\}_{i\in 3:n}$. We want to look at the behavior of the state projected on the intersection of this subspace and $\mathcal{S}$, which around $\mathbf{x}^*$ locally behaves as its tangent space $\mathcal{T}_{\mathcal{S}}(\mathbf{x}^*)$. This tangent space will be the orthogonal complement of $\frac{\partial}{\partial \mathbf{x}} h_{\mathcal{S}}(\mathbf{x})\big|_{\mathbf{x}^*}$ where $h_{\mathcal{S}}$ is as defined in eq. (4.32). We denote the projection matrix from $\mathbb{R}^n$ onto the intersection of $\tilde{\mathcal{X}}$ and $\mathcal{T}_{\mathcal{S}}(\mathbf{x}^*)$ as $\boldsymbol{\pi}_{\text{proj}} \in \mathbb{R}^{n_{\text{ld}}\times n}$. We also introduce the lift matrix $\boldsymbol{\pi}_{\text{lift}} \in \mathbb{R}^{n\times n_{\text{ld}}}$, which takes a point from the subspace to its canonical embedding in $\mathbb{R}^n$. This is equivalent to satifying the conditions $\boldsymbol{\pi}_{\text{proj}}\boldsymbol{\pi}_{\text{lift}} = \mathbf{I}_{n_{\text{ld}}}$, $\boldsymbol{\pi}_{\text{proj}} - \boldsymbol{\pi}_{\text{proj}}\boldsymbol{\pi}_{\text{lift}}\boldsymbol{\pi}_{\text{proj}} = \mathbf{0}_{n_{\text{ld}}}$.

The orthogonal complement of a matrix's range is the null space of its transpose. Furthermore, lifted points lie in the null space of span $\left\{ \frac{\partial}{\partial \mathbf{x}} h_{\mathcal{S}}(\mathbf{x})\big|_{\mathbf{x}^*}, \mathbf{e}_1, \mathbf{e}_2 \right\}$. Together, this gives us

$$\boldsymbol{\pi}_{\text{proj}} = \text{Null}\left(\begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} h_{\mathcal{S}}(\mathbf{x})\big|_{\mathbf{x}^*} & \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}^\top \right) \tag{4.36a}$$

$$\boldsymbol{\pi}_{\text{lift}} = \text{Null}\left(\begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} h_{\mathcal{S}}(\mathbf{x})\big|_{\mathbf{x}^*} & \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix} \right) \tag{4.36b}$$

Let $\tilde{\mathcal{S}} \triangleq \mathcal{S} \cap \tilde{\mathcal{X}}$ and denote $\mathbf{z} \triangleq \text{proj}_{\tilde{\mathcal{S}}}(\mathbf{x})$. Further, let $P_{\tilde{\mathcal{S}}} : \mathbb{R}^{n_{\text{ld}}-1} \to \mathbb{R}^{n_{\text{ld}}-1}$ be the Poincaré map of the system projected on $\tilde{\mathcal{X}}$ with respect to the surface $\tilde{\mathcal{S}}$. The linearization of the system of Poincaré iterations around $\mathbf{z}^* = \text{proj}_{\tilde{\mathcal{S}}}(\mathbf{x}^*)$ will be

$$\begin{aligned} \delta\mathbf{z}_{k+1} &= \mathbf{A}(\mathbf{z}^*)\delta\mathbf{z}_k \\ &\text{where} \\ \mathbf{A}(\mathbf{z}^*) &= \boldsymbol{\pi}_{\text{proj}}\boldsymbol{\Phi}(\boldsymbol{\pi}_{\text{lift}}\mathbf{z}^*, t_1)\boldsymbol{\pi}_{\text{lift}} \end{aligned} \tag{4.37}$$

The exponential stability of this system is then equivalent to the orbital exponential stability of the orbit in $\tilde{\mathcal{X}}$. For the rest of the chapter, any use of $P(\cdot)$ refers to $P_{\tilde{\mathcal{S}}}$.

4.4.4 Extension to Hybrid Dynamical Systems

For the case of a HDS, while the idea of the Poincaré map and the related stability analysis remains unchanged, the calculation of $\mathbf{A}(\mathbf{z}^*)$ becomes somewhat more involved. For a cycle in the graph representation of the HDS, consisting of an ordered alternating sequence of n_c continuous domains and n_c discrete transitions, let $\varphi_{v_i}(\mathbf{x}, t)$ describe the flow of the ith continuous domain and let $\Delta_{v_i}(\mathbf{x})$ be the discrete dynamics succeeding the ith continuous domain. Let $\mathbf{x}_{0_1}$ be the initial state of the system in the first continuous domain of the cycle, and assume that $\mathbf{x}_{0_1}$ is the initial condition for some solution of the system whose projection on $\tilde{\mathcal{X}}$ is periodic. Seeing as we have a good candidate for the Poincaré section in the form of the guard of the last edge, $\mathcal{S}_{v_{n_c}\to v_1}$, we want the point at which we analyze the Poincaré map to lie on this surface. Thus, let $\mathbf{z}^* \triangleq \boldsymbol{\pi}_{\text{proj}}\Delta^{-1}_{v_{n_c}\to v_1}(\mathbf{x}_{0_1})$ be the projection of the preimage of $\mathbf{x}_{0_1}$ under $\Delta_{v_{n_c}\to v_1}$.

The Poincaré map for the system around $\mathbf{z}^*$ then becomes a composition of functions alternating between the flow through each domain and the discrete dynamics function from the guard of one vertex to the domain of the successor:

$$
\begin{aligned}
P(\mathbf{z}_0) &= \boldsymbol{\pi}_{\text{proj}}(\varphi_{v_{n_c}}(t_{I_n}) \circ \Delta_{v_{n_c-1}\to v_{n_c}} \circ \ldots \circ \Delta_{v_1\to v_2} \circ \varphi_{v_1}(t_{I_1}) \circ \Delta_{v_{n_c}\to v_1})(\boldsymbol{\pi}_{\text{lift}}\mathbf{z}^*) \\
&\quad \text{where} \\
t_{I_i}(\mathbf{x}_{0_i}) &= t \ \text{ s.t. } \ \varphi_{v_i}(\mathbf{x}_{0_i}, t) \in \mathcal{S}_{v_i\to v_{i+1}}, \\
\mathbf{x}_{0_i} &= \Delta_{v_{i-1}\to v_i}(\varphi_{v_{i-1}}(\mathbf{x}_{0_{i-1}}, t_{I_{i-1}}))
\end{aligned}
\tag{4.38}
$$

Here, it is important to note that as the time of impact t_I for each domain is simply the time at which $\varphi(\mathbf{x}_0, t)$ intersects a guard $\mathcal{S}_{v_i\to v_{i+1}}$, it too is dependent on the initial condition of the domain. In order to calculate $\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}_0}\varphi(\mathbf{x_0}, t_I)$, consider within one continuous domain the solution $\mathbf{x}$ originating at $\mathbf{x}_0$ which intersects with $\mathcal{S}_v$ at t_I, and consider also the slightly perturbed solution $\mathbf{x}^p$ originating at $\mathbf{x}_0^p$ which intersects with $\mathcal{S}_v$ at t_I^p. Denote the difference between the initial conditions as $\delta\mathbf{x} = \mathbf{x}_0^p - \mathbf{x}_0$. There are two cases to be considered: Either $t_I \leq t_I^p$, or $t_I > t_I^p$.

Assume first $t_I \leq t_I^p$, so that the non-perturbed solution intersects first with $\mathcal{S}_v$ or simultaneously. In this case, both $\mathbf{x}(t_I), \mathbf{x}^p(t_I)$ are in the closure of the admissible domain. For this small perturbation, we would like to find $\mathbf{x}^p(t_I^p) - \mathbf{x}(t_I)$. We first approximate $\mathbf{x}^p(t_I)$ by a first-order Taylor expansion, $\mathbf{x}^p(t_I) \approx \mathbf{x}(t_I) + \frac{\partial}{\partial \mathbf{x}_0}\mathbf{x}(t_I) \cdot \delta\mathbf{x}$. Thus, $\delta\mathbf{x}_I := \mathbf{x}^p(t_I) - \mathbf{x}(t_I) \approx \frac{\partial}{\partial \mathbf{x}_0}\mathbf{x}(t_I) \cdot \delta\mathbf{x}$. Secondly, we need $\mathbf{x}^p(t_I^p) - \mathbf{x}^p(t_I)$. Assuming the perturbation is small, this vector is well-approximated by $\mathbf{f}(\mathbf{x}(t_I)) \cdot \delta t$ where $\delta t = t_I^p - t_I$. As $h(\mathbf{x}^p(t_I^p) = 0$ must hold, we have $0 \approx h(\mathbf{x}(t_I) + \delta\mathbf{x}_I + \mathbf{f}(\mathbf{x}(t_I)) \cdot \delta t) \approx h(\mathbf{x}(t_I)) + \frac{\partial}{\partial \mathbf{x}} h(\mathbf{x}(t_I)) \cdot (\delta\mathbf{x}_I + \mathbf{f}(\mathbf{x}(t_I) \cdot \delta t)$. As $h(\mathbf{x}(t_I)) = 0$, this gives

$$\delta t \approx -\frac{\frac{\partial}{\partial \mathbf{x}} h(\mathbf{x}(t_I)) \cdot \delta\mathbf{x}_I}{\frac{\partial}{\partial \mathbf{x}} h(\mathbf{x}(t_I)) \cdot \mathbf{f}(\mathbf{x}(t_I))}$$

or, by denoting $\frac{\frac{\partial}{\partial \mathbf{x}} h(\mathbf{x}(t_I))^\top}{\| \frac{\partial}{\partial \mathbf{x}} h(\mathbf{x}(t_I)) \|}$ as the normal vector $\mathbf{n}$ to the surface $\mathcal{S}_v$ at $\mathbf{x}(t_I)$ (and, equivalently for small perturbations, at $\mathbf{x}^p(t_I^p)$)

$$\delta t \approx -\frac{\mathbf{n}^\top \cdot \delta\mathbf{x}_I}{\mathbf{n}^\top \cdot \mathbf{f}(\mathbf{x}(t_I))}$$

In the end, we get

$$\mathbf{x}^p(t_I^p) - \mathbf{x}^p(t_I) \approx -\mathbf{f}(\mathbf{x}(t_I)) \cdot \frac{\mathbf{n}^\top \cdot \delta\mathbf{x}_I}{\mathbf{n}^\top \cdot \mathbf{f}(\mathbf{x}(t_I))}$$

and thus

$$\begin{aligned} \mathbf{x}^p(t_I^p) - \mathbf{x}(t_I) &\approx \delta\mathbf{x}_I - \mathbf{f}(\mathbf{x}(t_I)) \cdot \frac{\mathbf{n}^\top \cdot \delta\mathbf{x}_I}{\mathbf{n}^\top \cdot \mathbf{f}(\mathbf{x}(t_I))} \\ &= \left(\mathbf{I} - \frac{\mathbf{f}(\mathbf{x}(t_I)) \cdot \mathbf{n}^\top}{\mathbf{n}^\top \cdot \mathbf{f}(\mathbf{x}(t_I))} \right) \cdot \frac{\partial}{\partial \mathbf{x}_0}\mathbf{x}(t_I) \cdot \delta\mathbf{x} \end{aligned}$$

Now, consider the case when $t > t_p$, i.e. the perturbed solution intersects $\mathcal{S}_v$ before the non-perturbed solution. In this case, both $\mathbf{x}(t_I^p)$ and $\mathbf{x}^p(t_I^p)$ are in the closure of the domain. By switching the roles of $\mathbf{x}, \mathbf{x}^p$ and t_I, t_I^p, and observing that $\frac{\partial}{\partial \mathbf{x_0}}\mathbf{x}(t_I)\delta\mathbf{x} = \frac{\partial}{\partial \mathbf{x_0^p}}\mathbf{x^p}(t_I^p)\delta\mathbf{x}$ to the first-order approximation, one will for reasons of symmetry get the same expression.

The derivation here is based on one in [24], but is altered so as to apply to the case

when one solution is at the boundary, as opposed to the case when one has crossed the boundary. Please see fig. 4.2 for an illustration of the magnitudes involved.

Letting $\delta\mathbf{x}$ have the direction of each vector in the standard basis for $\mathbb{R}^n$ respectively, and dividing both sides by its norm and letting it tend towards 0, we end up with

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}_0}\varphi(\mathbf{x}_0, t_I(\mathbf{x}_0)) &= \Pi(\mathbf{x}_f)\cdot\frac{\partial}{\partial\mathbf{x}_0}\varphi(\mathbf{x}_0, t_I), \\ \Pi(\mathbf{x}_f) &= \left(\mathbf{I} - \frac{\mathbf{f}(\mathbf{x}_f)\cdot\mathbf{n}^\top}{\mathbf{n}^\top\cdot\mathbf{f}(\mathbf{x}_f)}\right), \\ \mathbf{x}_f &= \varphi(\mathbf{x}_0, t_I) \end{aligned} \tag{4.39}$$

Seeing as each discrete dynamics function $\Delta_{v_i\to v_{i+1}}$ depends solely on the pre-impact state, we get the following:

$$\begin{aligned} &\mathbf{A}(\mathbf{z}^*) = \\ &\boldsymbol{\pi}_{\text{proj}}\prod_{i=1}^{n_c}\left(\Pi_{v_i}(\mathbf{x}_{f_i})\cdot\frac{\partial}{\partial\mathbf{x}_{0_i}}\varphi_{v_i}(\mathbf{x}_{0_i}, t_{I_i})\cdot\frac{\partial}{\partial\mathbf{x}_{f_{\mu^{-1}(v_i)}}}\Delta_{\mu^{-1}(v_i)\to v_i}(\mathbf{x}_{f_{\mu^{-1}(v_i)}})\right)\boldsymbol{\pi}_{\text{lift}} \end{aligned} \tag{4.40}$$

If all of the eigenvalues of $\mathbf{A}(\mathbf{z}^*)$ are strictly less than 1 in magnitude then the trajectory originating at $\mathbf{x}_{0_1}$, projected on $\tilde{\mathcal{X}}$, is exponentially orbitally stable.

4.4.5 Linear/Bilinear matrix inequalities

In typical optimization problems, the task is to minimize some scalar objective function under a series of scalar linear or nonlinear constraints. However, there are two other classes of constraints (one a subset of the other) which are nontrivial to translate into a series of scalar inequalities: Linear Matrix Inequalities (LMIs) and BMIs.

A LMI posits the following:

$$\mathbf{A}_0 + \sum_{i=1}^{p}\left(\mathbf{A}_p x_p\right) \succ 0 \tag{4.41}$$

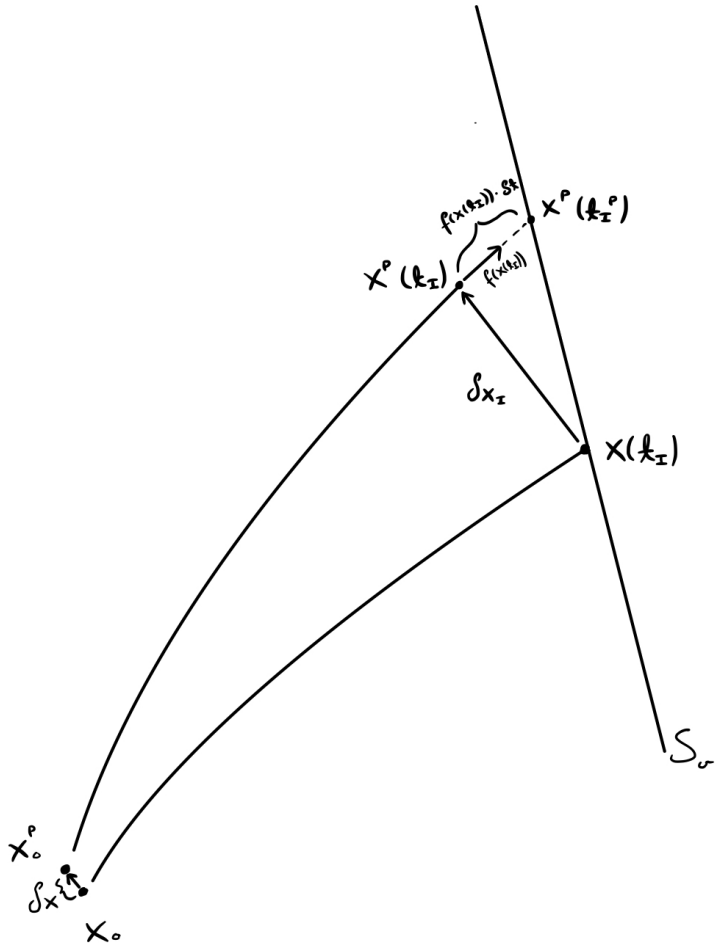


Figure 4.2: Geometric illustration of change in impact point and time as a function of initial perturbation

where $\mathbf{A}_i \in \mathbb{R}^{n\times n}$ are constant and given, while $\mathbf{x} \in \mathbb{R}^p$ is a variable to be determined. Several nonlinear constraints, including quadratic constraints, can be formulated as LMIs through certain reformulation tricks. However, there are some problems, for instance in control, which do not admit to being reformulated as an LMI. For many of these problems, they do however admit to being reformulated into a more general class of matrix constraints, called BMIs. These inequalities are on the following form:

$$\mathbf{A}_0 + \sum_{i=1}^{p} (\mathbf{A}_i x_i) + \sum_{i=1}^{q} (\mathbf{B}_i y_i) + \sum_{i=1}^{p}\sum_{j=1}^{q} \left(\mathbf{C}_{ij} x_i y_j\right) \succ 0 \tag{4.42}$$

where $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$ are square constant matrices, while $\mathbf{x}, \mathbf{y}$ are variables to be determined. Their upside is that they capture a much wider class of problems. However, their downside (as opposed to the case of LMIs) is the lack of systematic, efficient ways of solving optimization problems that contain such constraints [25].

One relation used to transform constraints which are not obviously LMIs/BMIs into such constraints is to consider the relation between a block matrix and its Schur complement, reproduced here from [26, p. 34]:

$$\begin{gathered} \text{for } \mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^\top & \mathbf{A}_{22} \end{bmatrix}, \\ \mathbf{A} \succ 0 \iff \mathbf{A}_{11} \succ 0 \text{ and } \mathbf{A}/\mathbf{A}_{11} \succ 0 \\ \text{where } \mathbf{A}/\mathbf{A}_{11} \text{ is the Schur complement of } \mathbf{A} \text{ with respect to } \mathbf{A}_{11} \end{gathered} \tag{4.43}$$

4.4.6 Sensitivity analysis and post-processing

As mentioned in section 4.4, while the synthesized controller yields a closed loop trajectory that is periodic in part of the system state, the trajectory is not necessarily orbitally stable. Our goal here is to alter the actual and desired outputs of the controller in a way which retains the same periodic trajectory, but changes its properties so as to make it orbitally stable. The methods described here are based on the paper [4], though the formulation is extended to systems which exhibit arbitrarily many domains

and left-right symmetry simultaneously.

First, note that the actual outputs chosen in the original controller synthesis is a special case of

$$\mathbf{y}_a(t) = \mathbf{H} \cdot \mathbf{q}(t)$$

Here, $\mathbf{H} \in \mathbb{R}^{n_y \times n_q}$ is a matrix of constant coefficients. Similarly, the desired outputs is a special case of

$$\mathbf{y}_d(t) = \mathbf{H} \cdot \mathbf{q}^*(t)$$

where $\mathbf{q}^*(t)$ is the periodic trajectory which we want to stabilize. By considering the synthesized controller as an instance of this more general form, we have a finite-dimensionally parametrized family of controllers which all relate to the same trajectory, in the sense that $\mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0 \ \forall \mathbf{q}(t) = \mathbf{q}^*(t) \ \forall \mathbf{H} \in \mathbb{R}^{n_y \times n_q}$ where $\mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, t)$ is as described in eq. (4.8).

Consider the discrete-time system in eq. (4.44). We now want to consider this system describing the development of $\delta\mathbf{z}$ lying on the Poincaré section $\mathcal{S}_{v_{n_c} \to v_1} \cap \tilde{\mathcal{X}}$, where $\mathbf{A}(\mathbf{z}^*)$ is given as in eq. (4.40).

We may consider the controller as parametrized by a vector $\boldsymbol{\xi}$ which is a concatenation of each column of $\mathbf{H}$ and denote the parametrization which coincides with the previously synthesized controller as $\boldsymbol{\xi}^*$. Then, the linear discrete-time system describing the evolution of $\delta\mathbf{z}$ is given by

$$\delta\mathbf{z}_{k+1} = \mathbf{A}(\mathbf{z}^*, \boldsymbol{\xi}^*)\delta\mathbf{z}_k \tag{4.44}$$

Thus, it is necessary to perturb the vector $\boldsymbol{\xi}^*$ by some $\delta\boldsymbol{\xi}$ so that the system with system matrix $\mathbf{A}(\mathbf{z}^*, \boldsymbol{\xi}^* + \delta\boldsymbol{\xi})$ is exponentially stable. Tuning $\boldsymbol{\xi}$ directly through nonlinear optimization would require recalculating $\mathbf{A}(\mathbf{z}^*, \boldsymbol{\xi})$ for each iteration, which would entail a substantial computational burden. An alternative to this is to linearize eq. (4.44) around $(\mathbf{z}^*, \boldsymbol{\xi}^*)$ wrt. $\boldsymbol{\xi}$, and to find $\delta\boldsymbol{\xi}$ which stabilizes the system

$$\begin{aligned}\delta \mathbf{z}_{k+1} &= \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \delta \mathbf{z}_k \\ &\text{where} \\ \mathbf{A}_0 &= \mathbf{A}(\mathbf{z}^*, \boldsymbol{\xi}^*), \\ \mathbf{A}_i &= \frac{\partial}{\partial \xi_i} \mathbf{A}(\mathbf{z}^*, \boldsymbol{\xi}^*)\end{aligned} \tag{4.45}$$

Now, consider the Lyapunov Function Candidate (LFC) $V_k = \delta \mathbf{z}_k^\top \mathbf{W}^{-1} \delta \mathbf{z}_k$ where $\mathbf{W} = \mathbf{W}^\top \succ 0$. We get

$$V_{k+1} = \delta \mathbf{z}_{k+1}^\top \mathbf{W}^{-1} \delta \mathbf{z}_{k+1} = \delta \mathbf{z}_k^\top \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right)^\top \mathbf{W}^{-1} \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \delta \mathbf{z}_k, \tag{4.46a}$$

$$\Delta V_k \triangleq V_{k+1} - V_k \tag{4.46b}$$

The LFC itself satisfies the criteria for showing exponential stability (positive definiteness). If one can also show $\Delta V_k < -\mu V_k \ \forall \delta \mathbf{z} \neq 0$ for some $\mu \in \mathbb{R} > 0$, then the system in eq. (4.45) is exponentially stable. This property is equivalent to

$$\begin{aligned}&\left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right)^\top \mathbf{W}^{-1} \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) - \mathbf{W}^{-1} \prec -\mu \mathbf{W}^{-1} \\ \iff &(1-\mu)\mathbf{W} - \mathbf{W}\left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right)^\top \mathbf{W}^{-1} \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \mathbf{W} \succ 0\end{aligned} \tag{4.47}$$

By observing that eq. (4.47) is the Schur complement of a symmetric block matrix

$$\mathbf{M} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix}, \ \mathbf{A} = \mathbf{W}, \ \mathbf{B} = \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \mathbf{W}, \ \mathbf{C} = (1-\mu)\mathbf{W}$$

and considering eq. (4.47), one can see that the positive definiteness of $\mathbf{W}$ and the

requirement $\Delta V_k < -\mu V_k \ \forall \delta \mathbf{z} \neq 0$ are both satisfied iff

$$\begin{bmatrix} \mathbf{W} & \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \mathbf{W} \\ \mathbf{W}^\top \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right)^\top & (1-\mu)\mathbf{W} \end{bmatrix} \succ 0 \tag{4.48}$$

However, this constraint only ensures that the linearization wrt. $\boldsymbol{\xi}$ is stable. To increase the likelihood that the linearization approximates the true system well, it is also desirable to search for solutions that are close to the linearization point. In [4] this is done by introducing a scalar decision variable γ and minimizing this while imposing the constraint $\|\delta\boldsymbol{\xi}\|^2 \leq \gamma$ (though reformulated as an LMI constraint).

However, during implementation it was found that by simply including $\|\delta\boldsymbol{\xi}\|^2 = \delta\boldsymbol{\xi}^\top\delta\boldsymbol{\xi}$ directly in the cost function - making the cost function quadratic but removing a large matrix constraint - the computations sped up by a factor of ~ 60.

Furthermore, the rate of convergence to the stable orbit increases for larger μ, thus it is also desirable while minimizing $\|\delta\boldsymbol{\xi}\|^2$ to maximize μ with a coefficient w determining the soft priority between the two. All in all, this can be formulated as a BMI constrained optimization problem:

$$\min_{\delta\boldsymbol{\xi},\mu,\mathbf{W}} \quad -w\cdot\mu + \delta\boldsymbol{\xi}^\top\delta\boldsymbol{\xi} \tag{4.49a}$$

s.t.

$$\begin{bmatrix} \mathbf{W} & \left(\mathbf{A}_0 + \sum_{i=1}^{n_\xi} \mathbf{A}_i \delta\xi_i\right) \mathbf{W} \\ \bigstar & (1-\mu)\mathbf{W} \end{bmatrix} \succ 0 \tag{4.49b}$$

$$\mu > 0 \tag{4.49c}$$

There are several both commercial and open-source solvers which are able to solve BMI constrained problems. However, some solvers require a feasible starting point, which would in this case translate into having an exponentially stable solution to begin with, and would thus defeat the purpose of solving the optimization problem. However, certain solvers rely on an Augmented Lagrangian approach (related to both

penalty and barrier methods) which does not require a feasible initial point.

4.4.7 Managing and reducing computational complexity

Even when avoiding the recalculation of $\mathbf{A}(\mathbf{z}^*, \xi)$ for each iteration of the optimization routine, the BMI constrained optimization problem is fairly computationally heavy. This computational burden scales with both the size of the matrix constraints as well as the number of variables.

In this subsection we describe some measures taken to reduce the size of the problem to curtail the computational burden somewhat. Firstly, one could argue that from a consideration of what properties the gait should have, there are certain states that it is not desirable to include in the feedback. For instance, while one may have included a constraint on velocity in the xy-plane during the synthesis of the gait, the exact xy-position at a certain time t is not really a matter of concern: If the gait keeps the robot stable with respect to its height above the ground, and its roll and pitch orientation, it would still "keep its balance" so to speak, and its exact position in the xy-plane might be better accounted for by a higher-level controller.

Additionally, attempting to ensure stability in the absolute base height may not be desirable. The height relative to the ground is already described through a combination of base orientation and stance leg configuration, and stabilizing the absolute height to some precomputed trajectory might lead to trouble if the robot encounters an unexpected slope and the absolute base height increases if the robot walks forward.

Secondly, consider gait symmetry. Intuitively, this is when the second half of a gait is a mirroring of the first half. We may write this more formally as $\mathbf{x}_{v_i}(t) = \mathrm{S}_x\mathbf{x}_{v_j}(t)\ \forall i - j = \frac{n_c}{2}$ for some matrix S_x s.t. $\mathrm{S}_x^2 = \mathbf{I}$. Assume the same is true for $\mathbf{y}_a$ for some matrix S_y, i.e. $\mathbf{H}_{v_i}\mathbf{x}_{v_i} = \mathrm{S}_y\mathbf{H}_{v_j}\mathbf{x}_{v_j}$.

Further, as we assume the above conditions hold $\forall \mathbf{x}_{0_v} \forall t \in [t_{0_v}, t_{I_v})$, it follows that $\mathbf{f}_{v_i}(\mathbf{x}_{v_i}) = \mathrm{S}_x\mathbf{f}_{v_j}(\mathbf{x}_{v_j})$, $\Delta_{v_i\to\mu(v_i)}(\mathbf{x}_{f_{v_i}}) = \mathrm{S}_x\Delta_{v_j\to\mu(v_j)}(\mathbf{x}_{f_{v_j}})$, or equivalently, that $\mathrm{S}_x\mathbf{f}_{v_i}(\mathrm{S}_x\mathbf{x}_{v_j}) = \mathbf{f}_{v_j}(\mathbf{x}_{v_j})$, and $\mathrm{S}_x\Delta_{v_i\to\mu(v_i)}(\mathrm{S}_x\mathbf{x}_{f_{v_j}}) = \Delta_{v_j\to\mu(v_j)}(\mathbf{x}_{f_{v_j}})$.

With this, we may rewrite eq. (4.38) as

$$
\begin{aligned}
P(\mathbf{z}^*) = \boldsymbol{\pi}_{\text{proj}}((\mathrm{S}_x \circ \varphi_{v_{\frac{n_c}{2}}}(t_{I_{\frac{n}{2}}}) \circ \mathrm{S}_x) \circ (\mathrm{S}_x \circ \Delta_{v_{\frac{n_c}{2}-1} \to v_{\frac{n_c}{2}}} \circ \mathrm{S}_x) \circ \dots \\
\circ (\mathrm{S}_x \circ \Delta_{v_1 \to v_2} \circ \mathrm{S}_x) \circ (\mathrm{S}_x \circ \varphi_{v_1}(t_{I_1}) \circ \mathrm{S}_x) \circ (\mathrm{S}_x \circ \Delta_{v_{n_c} \to v_1} \circ \mathrm{S}_x)) \\
\circ (\varphi_{v_{\frac{n_c}{2}}}(t_{I_{\frac{n}{2}}}) \circ \Delta_{v_{\frac{n_c}{2}-1} \to v_{\frac{n_c}{2}}} \circ \dots \circ \Delta_{v_1 \to v_2} \circ \varphi_{v_1}(t_{I_1}) \circ \Delta_{v_{n_c} \to v_1})(\boldsymbol{\pi}_{\text{lift}} \mathbf{z}^*) \\
= P_{half} \circ P_{half}(\mathbf{z}^*) \\
\text{where} \\
P_{half}(\mathbf{z}^*) = \boldsymbol{\pi}_{\text{proj}} \mathrm{S}_x \circ \varphi_{v_{\frac{n_c}{2}}}(t_{I_{\frac{n}{2}}}) \circ \Delta_{v_{\frac{n_c}{2}-1}} \circ \dots \circ \Delta_{v_1} \circ \varphi_{v_1}(t_{I_1}) \circ \Delta_{v_{n_c}}(\boldsymbol{\pi}_{\text{lift}} \mathbf{z}^*)
\end{aligned}
\tag{4.50}
$$

As the stability analysis of an orbit should not depend on which point in the orbit is chosen as the initial point, we may now equivalently analyze the orbit from $\tilde{\mathbf{z}}^* \triangleq \boldsymbol{\pi}_{\text{proj}} \mathbf{x}_{f_{v_{\frac{n_c}{2}}}}$ instead, leaving us with

$$
\begin{aligned}
\tilde{P}(\tilde{\mathbf{z}}^*) = \tilde{P}_{half} \circ \tilde{P}_{half}(\tilde{\mathbf{z}}^*), \\
\tilde{P}_{half}(\tilde{\mathbf{z}}^*) = \mathrm{S}_x \circ P_{half} \circ \mathrm{S}_x
\end{aligned}
\tag{4.51}
$$

The derivative of $\tilde{P}_{half}$ then becomes

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\tilde{\mathbf{z}}^*} \tilde{P}_{half}(\tilde{\mathbf{z}}^*) = \\
&\boldsymbol{\pi}_{\text{proj}} \left[\prod_{i=1}^{\frac{n_c}{2}} \left(\Pi_{v_i}(\mathbf{x}_{f_{v_i}}) \cdot \frac{\partial}{\partial \mathbf{x}_{0_i}} \varphi_{v_i}(\mathbf{x}_{0_i}, t_{I_i}) \cdot \frac{\partial}{\partial \mathbf{x}_{f_{\mu^{-1}(v_i)}}} \Delta_{\mu^{-1}(v_i)}(\mathbf{x}_{f_{\mu^{-1}(v_i)}}) \right) \right] \cdot \mathrm{S}_x \boldsymbol{\pi}_{\text{lift}}
\end{aligned}
\tag{4.52}
$$

We denote this last matrix $\tilde{\mathbf{A}}(\tilde{\mathbf{z}}^*)$.

As $\frac{\mathrm{d}}{\mathrm{d}\tilde{\mathbf{z}}^*} \tilde{P}(\tilde{\mathbf{z}}^*) = \left(\frac{\mathrm{d}}{\mathrm{d}\tilde{\mathbf{z}}^*} \tilde{P}_{half}(\tilde{\mathbf{z}}^*) \right)^2$ we have that $\lambda(\frac{\mathrm{d}}{\mathrm{d}\tilde{\mathbf{z}}^*} \tilde{P}(\tilde{\mathbf{z}}^*)) = \left\{ \lambda_i^2 | \lambda_i \in \lambda(\frac{\mathrm{d}}{\mathrm{d}\tilde{\mathbf{z}}^*} \tilde{P}_{half}(\tilde{\mathbf{z}}^*)) \right\}$. This implies that the spectral radius of the former is less than 1 iff the spectral radius of the latter is less than 1. We may thus conduct our analysis on half of the cycle, reducing both the number of domains to integrate over and the number of parameters

to optimize over by half.

Secondly, having conducted the initial sensitivity analysis of $\tilde{\mathbf{A}}(\tilde{\mathbf{z}}^*, \boldsymbol{\xi})$ with respect to $\boldsymbol{\xi}$, we may find that its spectral radius exhibits a much greater sensitivity to some parameters than others. Indeed, by plotting the matrix norm of each $\frac{\partial}{\partial \xi_i}\tilde{\mathbf{A}}(\tilde{\mathbf{z}}^*, \boldsymbol{\xi}^*)$ we see that three columns of the **H**-matrix for each domain account for the vast majority of the sensitivity of $\tilde{\mathbf{A}}$ with respect to $\boldsymbol{\xi}$ (see fig. 4.3).

By restricting our optimization problem to these variables, we reduce the size of $\delta\boldsymbol{\xi}$ from 616 variables to 66 variables. As the optimization method involves computations of the Hessian, which has a lower bound of $O(n^2)$ in terms of computational complexity in the general case, this is expected to speed up computations significantly.

4.5 Implementation details

4.5.1 Robot models

The model of the ASTRo robot was written and generated based on the properties found in [3] using the Xacro XML macro language to generate a model in Unified Robot Description Format (URDF). The model of the vision60 robot was obtained from GitHub: `https://github.com/KodlabPenn/kodlab_gazebo` and is free to use under the MIT license.

4.5.2 Closed-loop trajectory optimization

The closed-loop trajectory optimization problem is formulated using the FROST (Fast Robotics Optimization and Simulation Toolkit) framework [27]. FROST is an open-source framework developed in collaboration between researchers at the AMBER lab at California Institute of Technology and the BipedLab at the University of Michigan, which is designed for formulating optimal control problems for hybrid dynamical systems.

FROST utilizes a collocation based discretization, implicit formulation of dynamics pre-compilation of symbolic derivatives to allow for full-order optimal control of even high-dimensional systems such as quadrupeds while retaining tractable computation

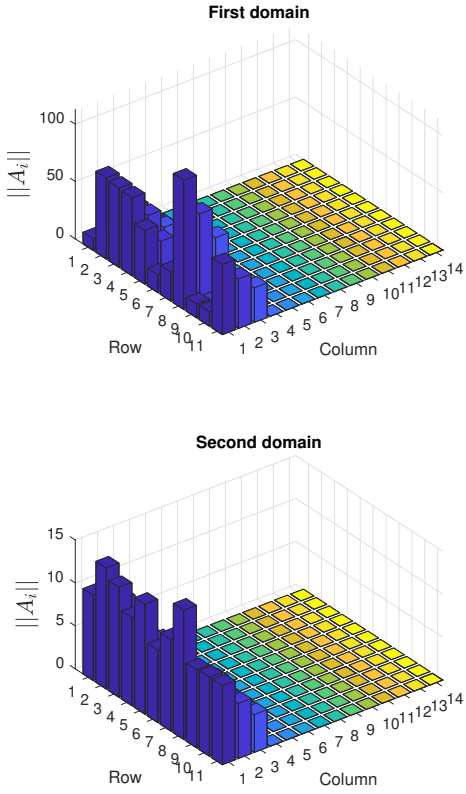


Figure 4.3: Norm of the sensitivity of $\tilde{\mathbf{A}}_0$ with respect to each parameter in $\mathbf{H}$

times. It is written primarily in MATLAB, but uses the Wolfram Mathematica symbolic engine for formulating and pre-compiling symbolic expressions to efficient C-code.

FROST is hosted at Github:`https://github.com/ayonga/frost-dev`.

4.5.3 Controller parameter post-processing

The Bilinear Matrix Inequality optimization problem for post-processing controller parameters was formulated in YALMIP [28], a MATLAB toolbox for formulating optimization problems. While YALMIP comes with some in-house solvers, it interfaces with numerous known commercial and open-source solvers. Here, the PENBMI [29] solver of PENLAB was chosen. The PENBMI solver utilizes an Augmented Lagrangian-approach to reformulate the nonlinearly constrained problem as a nonconstrained NLP approximation, which is solved iteratively. The primary challenge of some BMI solvers – such as the open-source BMIsolver MATLAB package which utilizes concave-convex decomposition [30] – is that it requires a feasible initial guess. For many BMI problems finding a feasible initial guess to feed a solver which requires feasibility is nontrivial; the problem to be solved here, for instance, is essentially a feasibility problem with some additional minimization of the solution norm to improve validity of the solution. Thus, finding an initial feasible guess is some of the point of the optimization problem. The PENBMI solver, using an Augmented Lagrangian-approach, allows for initial guesses which are infeasible.

4.5.4 Simulation

The simulation of the closed-loop systems was done in MATLAB through the FROST framework. For simulation FROST simply serves as an interface for the standard set of MATLAB solvers, and uses `ode45` as its standard solver.

The code-base for the implementation can be found at `https://github.com/Norwegian-Legged-Lab/Tetrapod-Robot` under the `hzd-control-feature` branch.

5

Results and Discussion

In this chapter we present and discuss the obtained results. Gaits were synthesized for the sprawling ASTRo robot as well as the mammalian vision60. The chapter proceeds with presentation and discussion of results as follows:

- Results of the closed loop trajectory optimization for two gaits for each robot
- Results of the controller post-processing procedure for all gaits
- Results from simulation of the closed loop systems under nominal conditions
- Results from simulation of the closed loop systems with unmodeled changes to the ground topology
- Results on energy and torque expenditure

Evaluating the method for both robots with respect to successful synthesis and stability of gaits was done in order to evaluate its suitability for a sprawling robot, and to compare it directly with replication of results previously achieved in [6].

For results regarding responses to unmodeled change in ground height, we compare across robots to evaluate whether results support the hypothesis that sprawling robots have an advantage to mammalian ones when it comes to handling such changes.

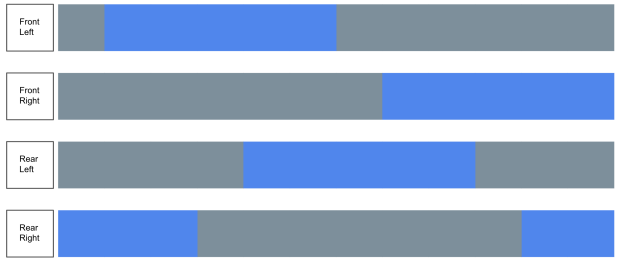


Figure 5.1: Diagram for one cycle of an ambling gait. Grey stretches indicate stance phases while blue stretches indicate swing phases.

When it comes to the results on torque expenditure, both robots were evaluated to compare inter-robot differences and consider them in light of the hypothesis that mammalian robots may be more energetically efficient. Importantly, this was also done to evaluate the effect of the modified controller on robots of both types of robot.

5.1 Optimal gaits

An ambling gait was chosen for synthesis. The ambling gait consists of 4 phases, where the two last phases constitute a left-right mirroring of the two first. In the first phase, the front right and the rear left legs are supporting legs, with the front left and rear right legs swinging. In the second phase, the front right and rear right legs are supporting legs, while the front left and rear left legs are swinging. See fig. 5.1 for an illustration.

For each robot, gait patterns were synthesized for two desired forward velocities. Firstly, an ambling-in-place gait was synthesized. Secondly, a forward ambling gait was synthesized with a desired forward velocity of $0.2m/s$. For both robots the bounds of the optimization problem (e.g. desired foot clearance, initial and terminal swing foot velocities etc.) were kept similar, in order to attempt to keep the comparison of method fair. The objective function used was the squared integral average of torque over time. At first simply the integral of squared torque was chosen, but it was found

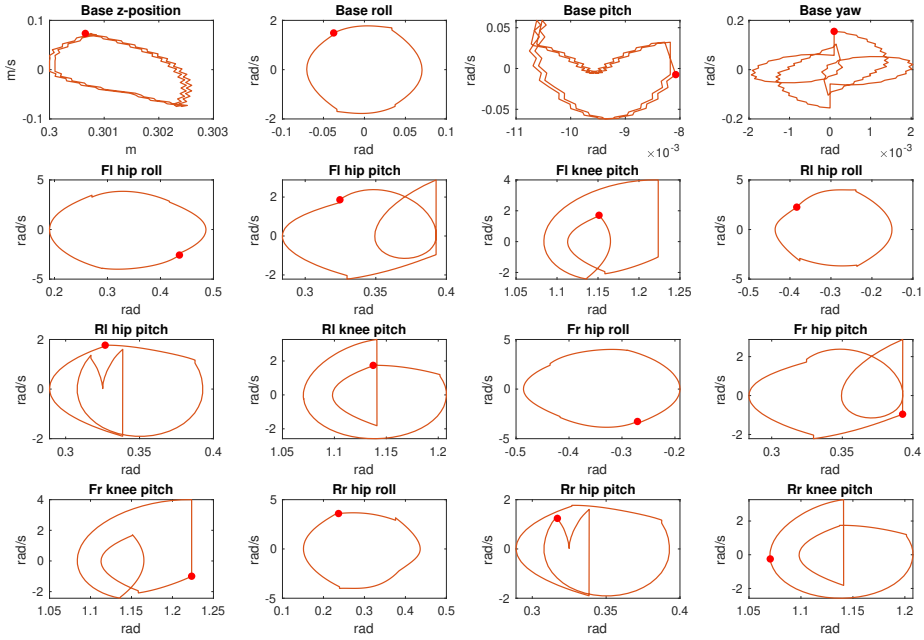


Figure 5.2: Phase portraits for optimized stationary ambling gait on the ASTRo robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted.

that this resulted in the solution always converging to the shortest allowed gait period.

In general, a moving gait on flat ground will exhibit periodic behavior in all states except for the base horizontal position. Therefore, phase portraits are shown of all system states except these. Phase portraits for respectively the stationary and forward gaits of ASTRo are shown in fig. 5.2 and fig. 5.3. Phase portraits for respectively the stationary and forward gaits of vision60 are shown in fig. 5.4 and fig. 5.5.

During optimization, no clear differences were noticed in convergence rates of the problem, neither for the presented gaits nor for any other produced gaits. This would indicate that the method presented in [6] for closed loop trajectory optimization is equally well-suited for sprawling quadrupeds as it is for mammalian ones.

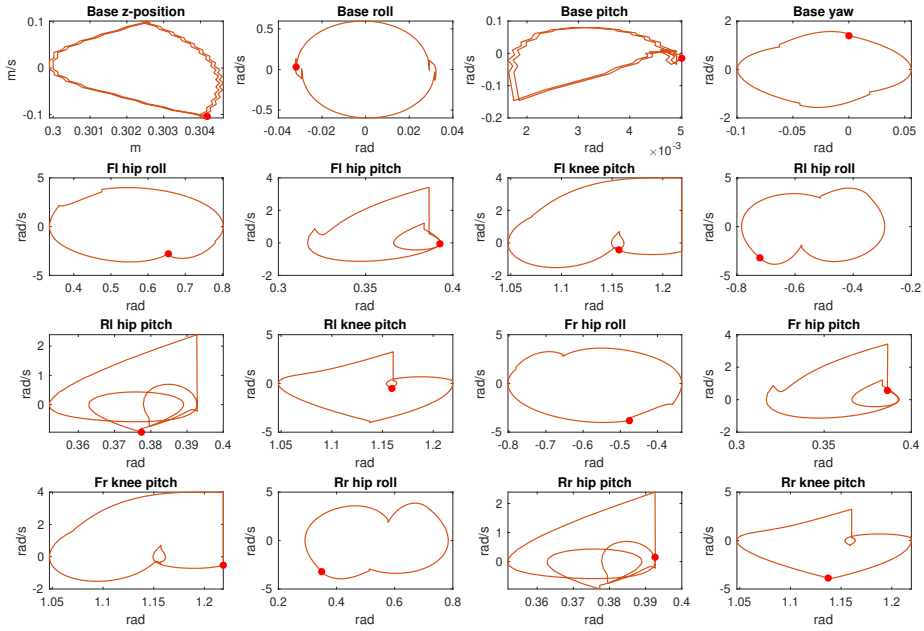


Figure 5.3: Phase portraits for optimized forward ambling gait on the ASTRo robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted.

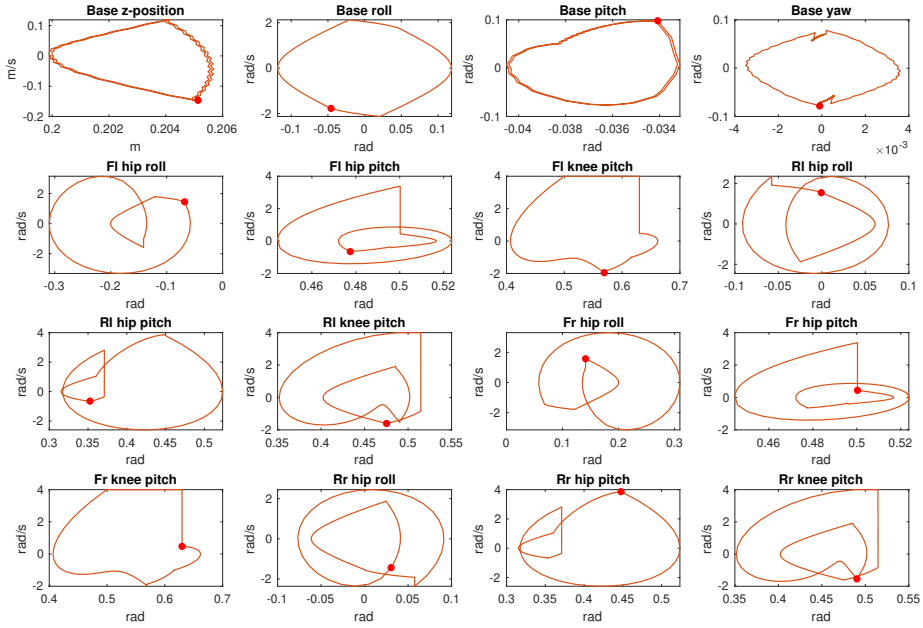


Figure 5.4: Phase portraits for optimized stationary ambling gait on the vision60 robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted.

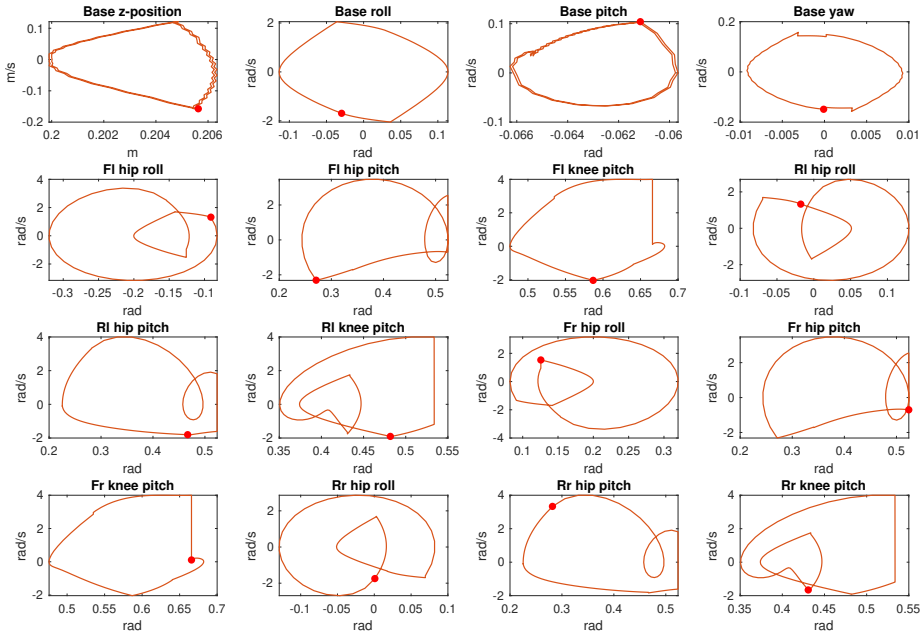


Figure 5.5: Phase portraits for optimized forward ambling gait on the vision60 robot. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots signify initial values. Phase portraits for x- and y-position do (by design) not constitute clear orbits and are omitted.

	ASTRo		vision60	
Gait type	Stationary	Forward	Stationary	Forward
SR before BMI opt.	1.0018	1.2024	0.9962	1.1296
SR after BMI opt.	0.7266	0.6267	0.5402	0.6392

Table 5.1: Spectral radii of the projection of the Poincaré map Jacobian for different gaits, before and after post-processing the controller parameters for exponential stability

5.2 Gait stabilization

Here, we present the results of the post processing-procedure to render an exponentially stabilizing controller. As discussed in section 4.4, the system state modulo x- and y-position is orbitally exponentially stable if the spectral radius of $\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}^*}P(\mathbf{x}^*)$ is strictly less than 1. In table 5.1, we present the spectral radii of the closed loop system for both robots for both gaits, before and after the postprocessing of controller parameters.

As can be seen, the spectral radius is significantly reduced for all gaits, successfully changing the corresponding closed loop system orbits from either unstable or close to unstable, to exponentially stable within a comfortable margin. While the primary goal is simply to reduce the spectral radius to less than 1, any extra reduction in spectral radius also entails a lower bound on how quickly trajectories in the orbit's vicinity converge to the orbit, and is as such desirable.

5.3 Simulation results

In what follows, we present the results from simulating the closed loop systems for both gait behaviors for both robots. In order to empirically verify and demonstrate the exponential stability of the gaits, each gait was simulated for 500 consecutive full gait cycles. In plots where fewer cycles are plotted for clarity, this is specified in the figure text.

5.3.1 vision60

By looking at the phase portraits for the stationary gait in fig. 5.6, we notice clear cyclic behavior in each state of the vision60 robot's stationary gait modulo horizontal position. This indicates an asymptotically stable limit cycle in the closed loop behavior, corresponding to an asymptotically stable gait.

From fig. 5.7, we see clear convergence to periodic behavior the entire base state except for x-position. This is hardly surprising: while the goal x-velocity for this gait is 0, there is no state feedback on x-position to the controller and as such some drift is to be expected over time.

In fig. 5.8 we have plotted the distance in states modulo base position between each successive intersection between the state and the Poincaré section previously used for stability analysis. This discrete dynamical system of "snapshots" of the system state at each intersection with the Poincaré section, is exponentially stable iff the orbit of the original system is exponentially stable.

As the discrete dynamical system converges to a fixed-point instead of a periodic behavior, it might be simpler to evaluate the orbital stability of the original system by looking at the distance between each iteration of this system. As we see, the state distance (modulo position) converges to 0 as the number of iterations increase, which does indeed indicate that the system is orbitally asymptotically stable.

In fig. 5.9 we see the phase portraits for the forward gait. Once again, convergence to periodic behavior is clear in all depicted states. This once again indicates asymptotically stable behavior of the limit cycle corresponding to the forward gait. In fig. 5.10, we see the state trajectories of the robot base. Once again, convergence to periodic behavior is evident for all states except x-position and y-position. The x-velocity of the gait is by design nonzero, and the x-position is seen here evolving linearly as expected, though the gait velocity seems to be slightly lower than $0.2m/s$. As there is currently no state feedback from the x-velocity, some discrepancy between desired and actual forward velocity might be expected. There is some very slight drift in y-position as well, which is again explained by the lack of state-feedback from y-position.

The orbital stability is also once again clearly seen from the evolution of the state

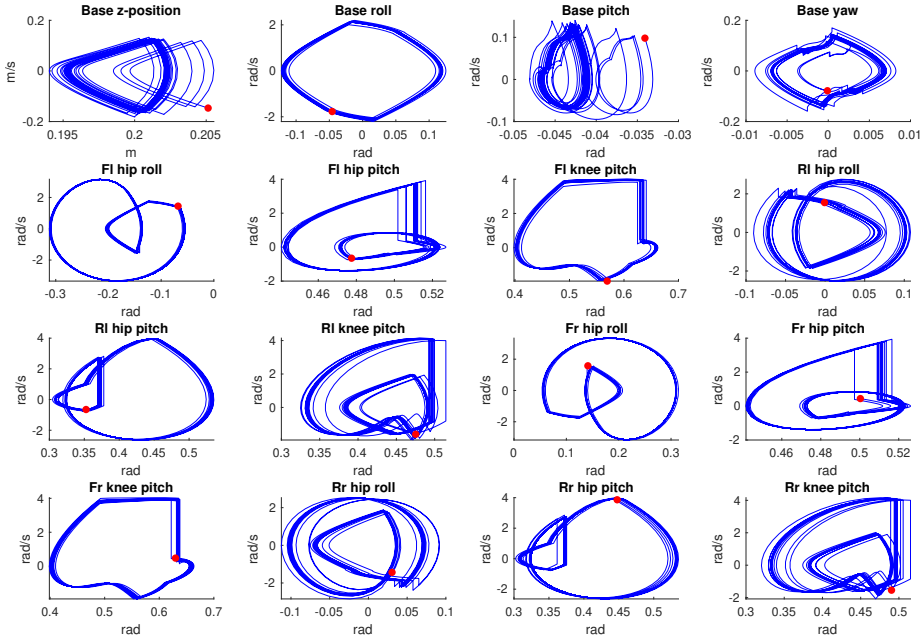


Figure 5.6: Phase portraits for stationary gait on the vision60 robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized.

(modulo horizontal position) on the Poincaré section, as seen in fig. 5.11.

5.3.2 ASTRo

For the stationary gait of the ASTRo robot, we see from the phase portraits in fig. 5.12 clear convergence to periodic behavior modulo horizontal position, though the convergence in base pitch and yaw is somewhat slower than the remaining states. This is in agreement with what can be seen from looking at the base state trajectory in fig. 5.13, where convergence to periodic behavior can be seen in all states except horizontal position. Again, while the desired x-velocity is 0, some drift is to be expected in the absence of state feedback from x-position. Here as well, the evolution of the state on

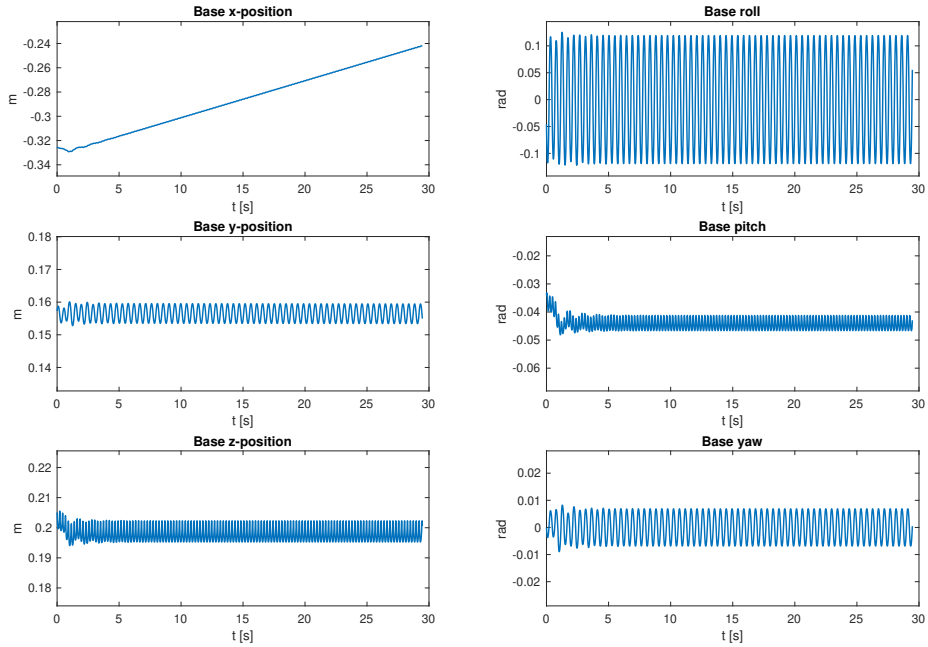


Figure 5.7: Trajectories of base pose states during 250 cycles of stationary gait for the vision60 robot during simulation

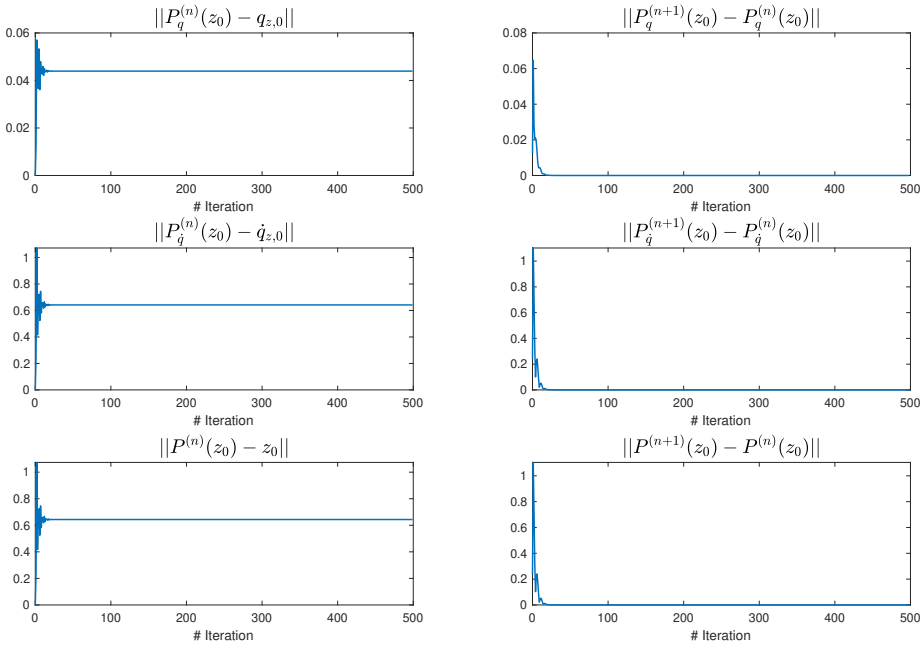


Figure 5.8: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a stationary gait for the vision60 robot during simulation

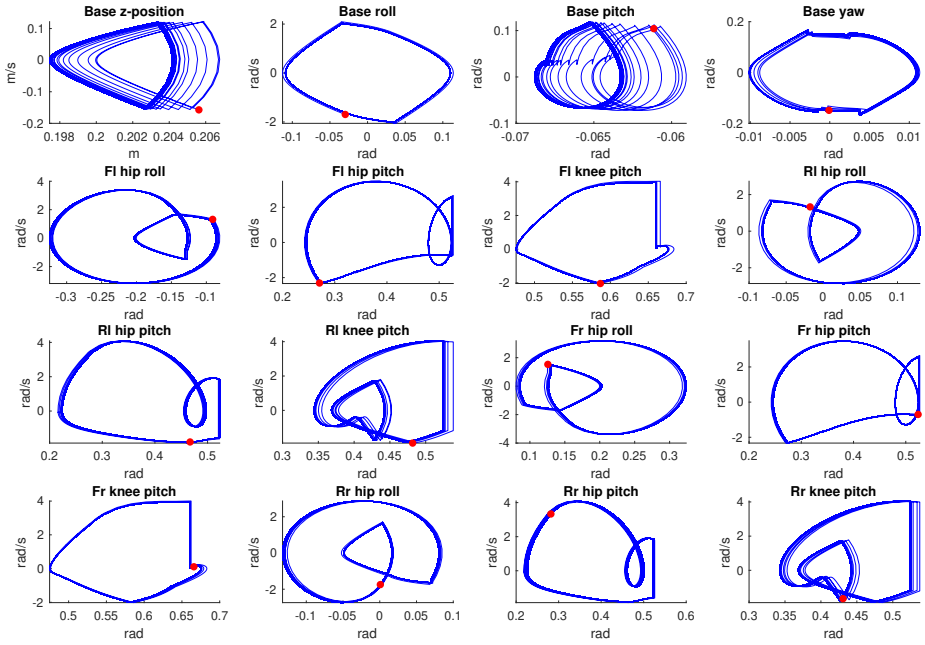


Figure 5.9: Phase portraits for forward (0.2 m/s) gait on the vision60 robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized.

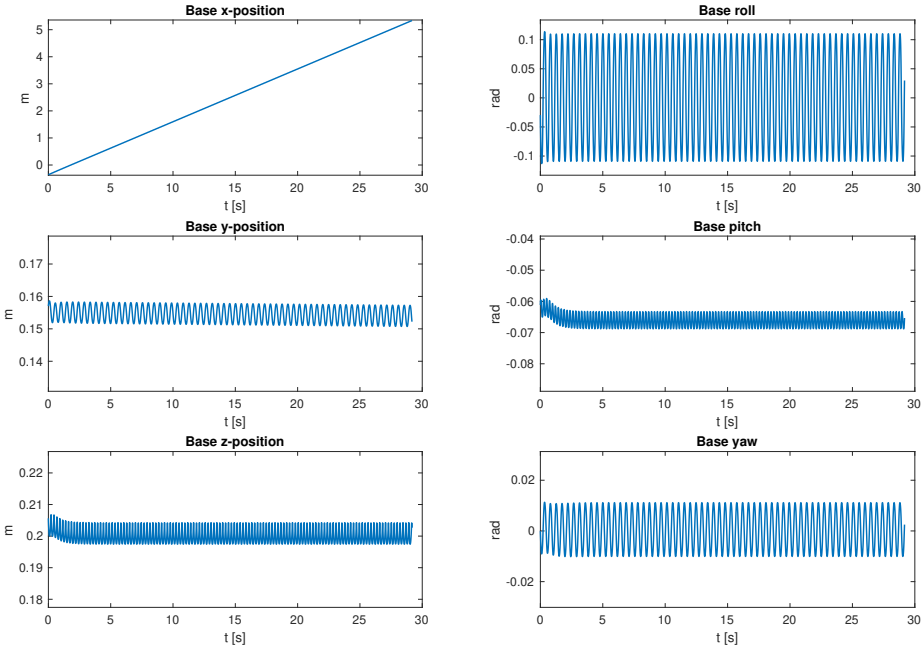


Figure 5.10: Trajectories of base pose states during 250 cycles of forward gait on flat ground for the vision60 robot during simulation

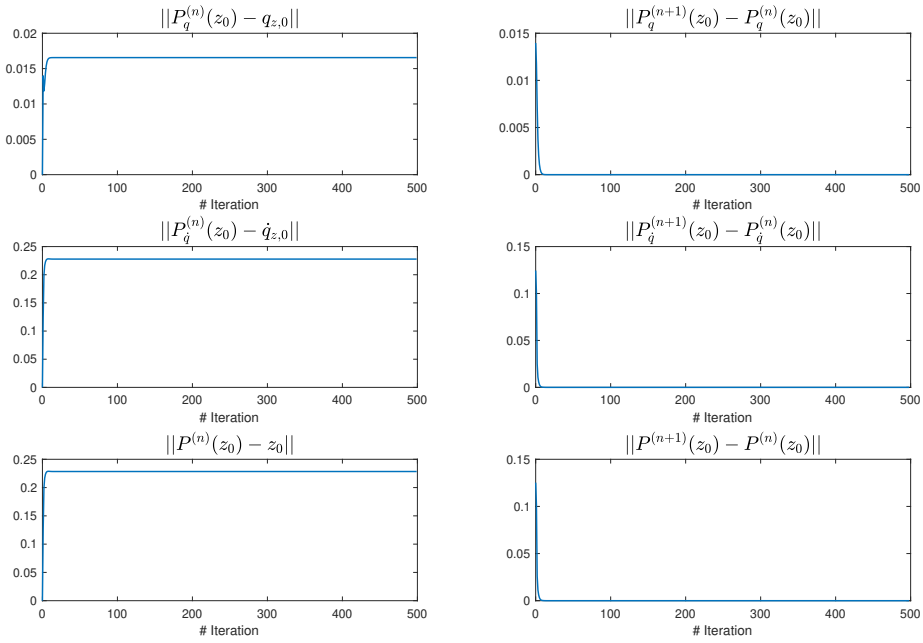


Figure 5.11: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on flat ground for the vision60 robot during simulation

the Poincaré section in fig. 5.14 shows convergence, further supporting that the system exhibits asymptotic orbital stability for the gait.

In the phase portraits of the states for the forward gait in fig. 5.15, we again see convergence to periodic behavior in the state modulo horizontal position, although this is slightly less clear in the base pitch. However, from fig. 5.16 we see that all base states modulo horizontal position seem to converge to periodic behavior. The y-position seems to drift slightly, which again is not unexpected as it is not attempted stabilized. The x-position develops linearly, which corresponds to the base maintaining a desired constant forward velocity. Again, the forward velocity is slightly smaller than specified during the gait optimization, but this is not unexpected due to the gait velocity not being explicitly stabilized by the controller.

Given that there is some more transient behavior leading to slightly less clear phase portraits here, it is especially helpful to look at the evolution of the state on the Poincaré section in fig. 5.17, which still clearly shows convergence to a fixed point and thus indicates asymptotic stability.

No clear differences were noted in the convergence rates of the BMI optimization problem to create stabilizing controllers for the two robots. Both gaits for both robots also end up with corresponding spectral radii well under 1, and their orbital exponential stability is backed up by considering the simulated trajectories as presented above. One may, however, from figures 5.8, 5.11, 5.14, 5.17 note that vision60 exhibits slightly less transient behavior before converging to a truly periodic gait than ASTRo does.

5.3.3 Deviations between initial state and fixed point for the state on the Poincaré section

The initial state for all simulations is set to be the initial state $\mathbf{x}_0$ of the respective optimized gait. This initial state is supposed to be part of a stable or unstable orbit of the system in closed loop. When looking at the discrete dynamical system of successive intersections between system state and the Poincaré section, this would be equivalent to $\mathbf{z}_0 = \boldsymbol{\pi}_{\text{proj}}\mathbf{x}_0$ being a fixed point of the discrete dynamical system. The post-processing is supposed to ensure specifically that the orbit resulting from

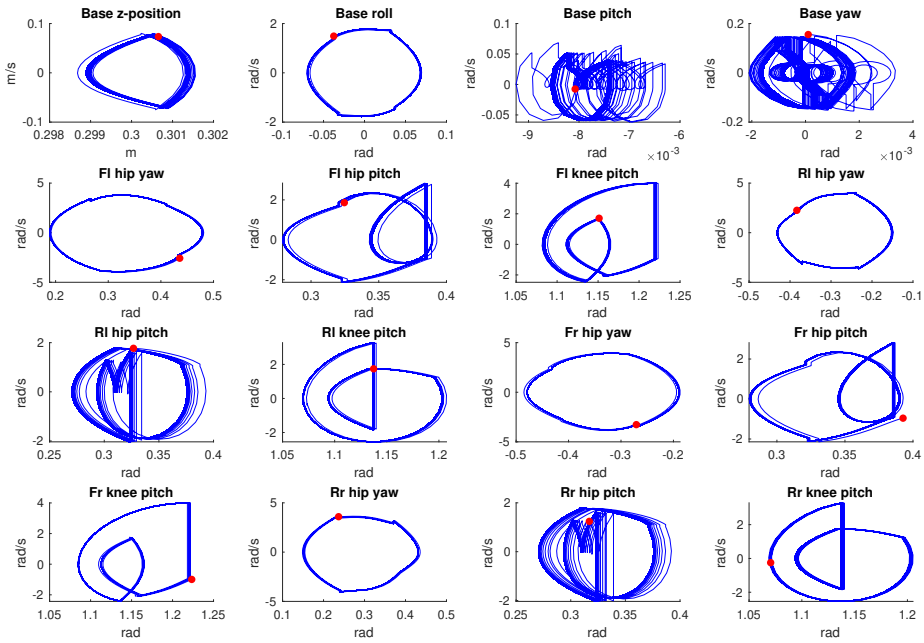


Figure 5.12: Phase portraits for stationary gait on the ASTRo robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized.

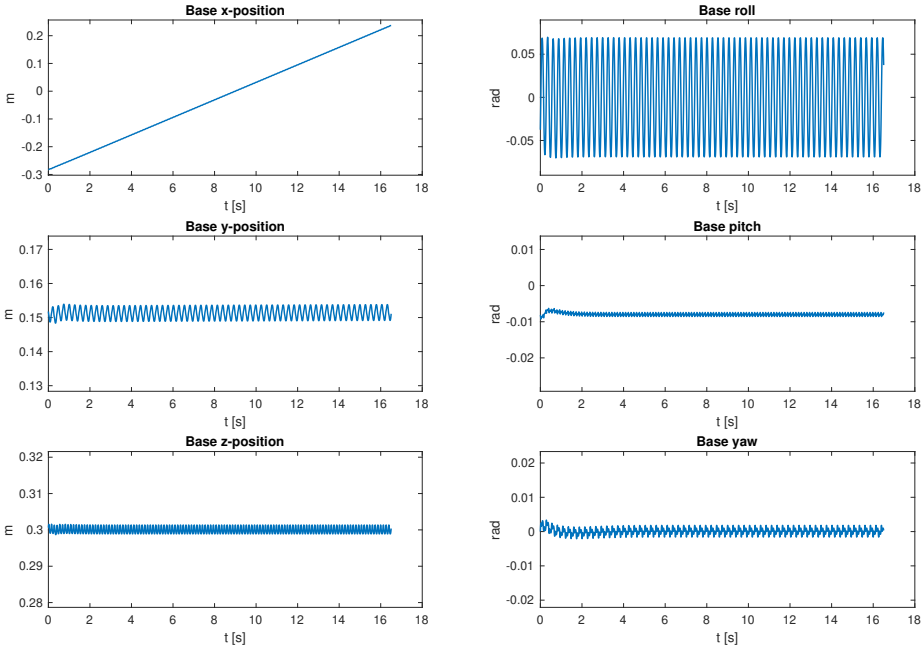


Figure 5.13: Trajectories of base pose states during 250 cycles of stationary gait for the ASTRo robot during simulation

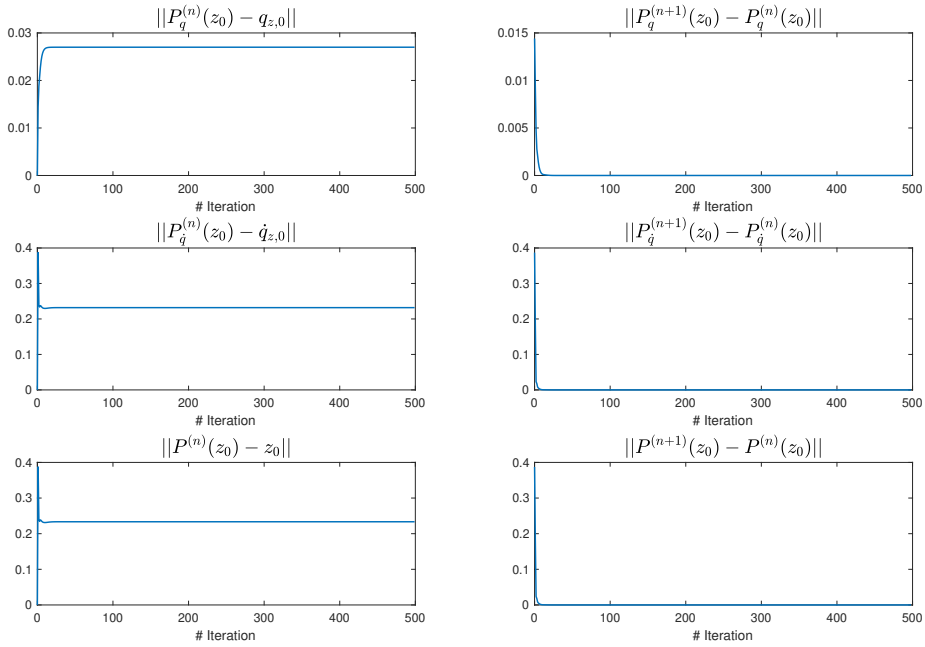


Figure 5.14: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a stationary gait for the ASTRo robot during simulation

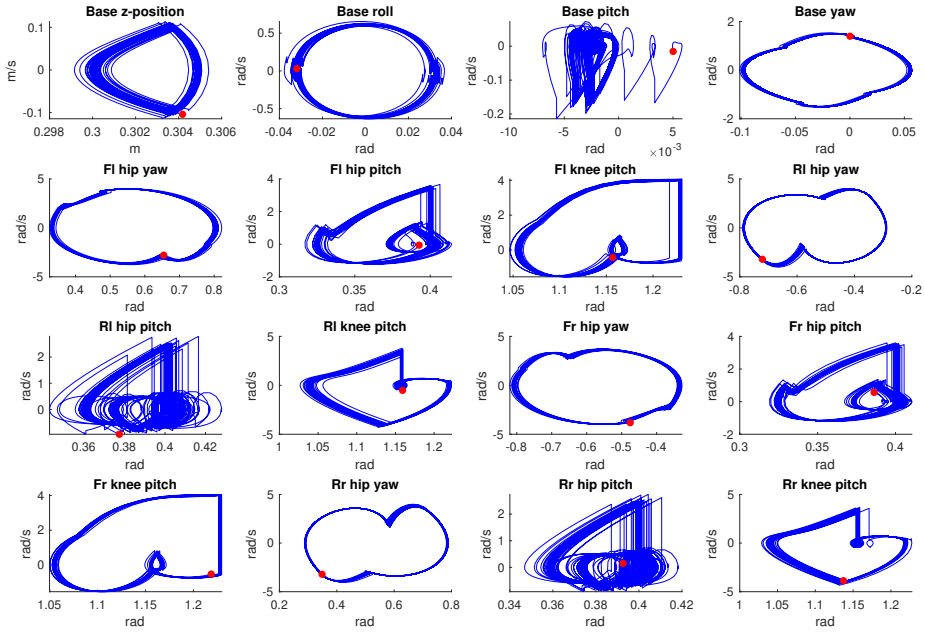


Figure 5.15: Phase portraits for forward (0.2 m/s) gait on the ASTRo robot during simulation. First letter signifies "front" or "rear" leg, last letter signifies "left" or "right" leg. Red dots indicate initial values. x- and y-positions are excluded as they are not attempted stabilized.

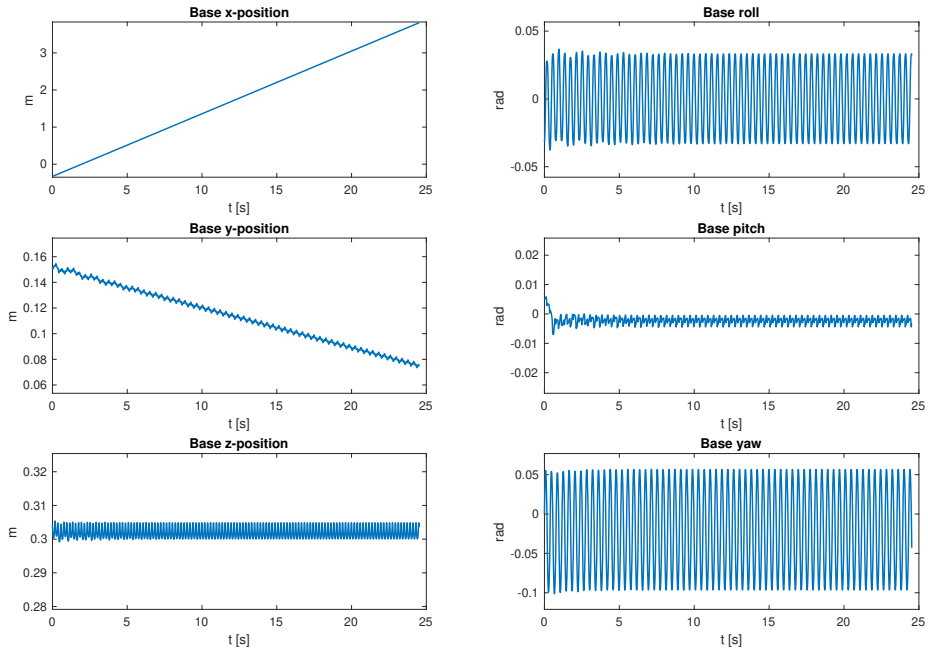


Figure 5.16: Trajectories of base pose states during 250 cycles of forward gait on flat ground for the ASTRo robot during simulation

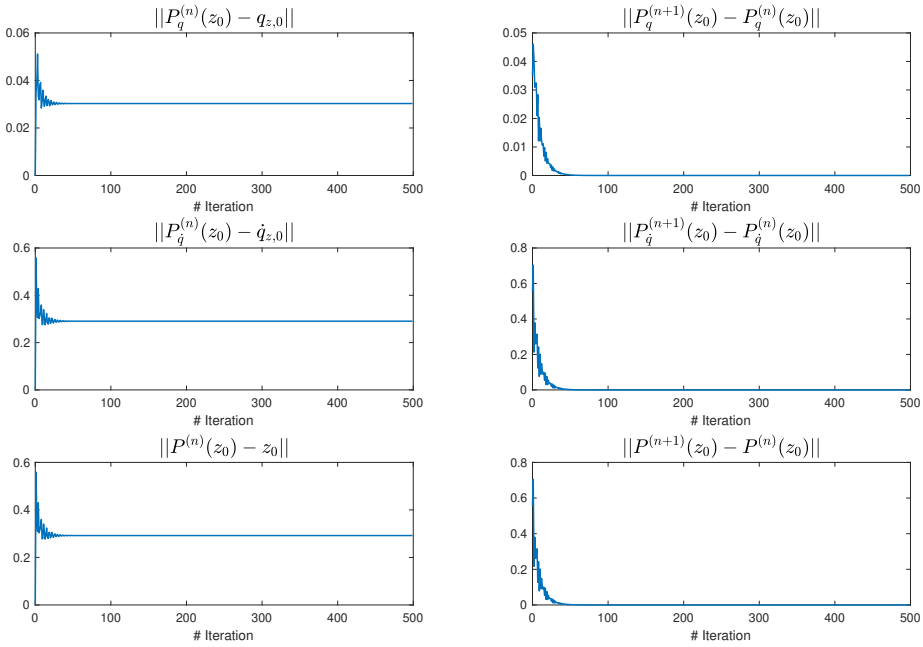


Figure 5.17: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on flat ground for the ASTRo robot during simulation

the initial state and the controller parameters, becomes exponentially orbitally stable, which would be equivalent to $\mathbf{z}_0$ being an exponentially stable fixed point of the discrete dynamical system. However, from looking at any of figures 5.8, 5.11, 5.14, 5.17, it is clear that while the state of the discrete system converges, it does not converge to $\mathbf{z}_0$ even though it begins there. This would indicate that $\mathbf{z}_0$ is not even a fixed point, and certainly not an asymptotically stable one.

The explanation for this is likely numerical errors. Firstly, assuming that the collocation scheme of the trajectory optimization results in a close-to-perfect representation of the system dynamics, the equality imposed between the initial and terminal condition of the state during trajectory optimization has some slack (in this case 10^{-4}), as has all "equality constraints" in optimization problems when solved numerically. Secondly, choosing the number of nodes for the collocation scheme is a trade-off between accuracy of the solution and computational tractability, and accuracy of the "integration" of the dynamics in an optimization problem will by necessity likely be lower than during simple forward integration during simulation of the system afterwards.

The variational equation will still characterize system behavior around whatever trajectory it is integrated across. However, for this characterization to say something about the orbital stability of trajectory, it requires both that the trajectory is a feasible trajectory given the system dynamics, and that the trajectory is an orbit of the system. That the trajectory given by the trajectory optimization problem is not truly or exactly a feasible orbit of the system, might explain why the systems do not converge to an orbit that includes $\mathbf{z}_0$. However we see from the same plots that the system state does indeed converge to a fixed point on the Poincaré section, even if that fixed point is not the initial value and expected fixed point $\mathbf{z}_0$.

While not guaranteed, it is not by complete accident that when the spectral radius of $\frac{\mathrm{d}}{\mathrm{d}\mathbf{z}}P(\mathbf{z})$ is less than 1 around $\mathbf{z}_0$ and $\|P(\mathbf{z}_0)-\mathbf{z}_0\|$ is small, then we find an attractive fixed point in its vicinity. Let $\rho(\cdot)$ denote the spectral radius. By assumption, $\rho\left(\frac{\mathrm{d}}{\mathrm{d}\mathbf{z}}P(\mathbf{z}_0)\right) < 1$. The induced matrix norm of the Jacobian of a discrete map determines whether the map is a contraction. If for a map $\mathbf{f} : \mathcal{X} \to \mathcal{X}$ we have $\|\frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x})\| \leq k < 1$ for $\mathbf{x} \in \mathcal{X}$, then the map is a contraction on $\mathcal{X}$[31]. Furthermore, the norm of a matrix is always lower bounded by its spectral radius, and for a matrix $\mathbf{M}$ we have $\lim_{n\to\infty}\|\mathbf{M}^n\|^{\frac{1}{n}} = \rho(\mathbf{M})$. The

matrix norm and the spectral radius are continuous functions, so if $\rho(\frac{\mathrm{d}}{\mathrm{d}\mathbf{z}}P(\mathbf{z}_0)) = k < 1$ then this should also hold in some vicinity around $\mathbf{z}_0$. It would therefore stand to reason that a lower spectral radius of $\frac{\mathrm{d}}{\mathrm{d}\mathbf{z}}P(\mathbf{z}_0)$ should increase the chances of $P^{(n)}(\mathbf{z}_0)$ converging to a nearby attractive fixed point (though this is by no means given).

5.4 Response to unmodeled changes in ground height

To check how both robots responded to unmodeled changes in ground height, we simulated the forward gaits (which were synthesized under an asssumption of flat ground) of the robots in a scenario where the ground, after staying flat for 2 m, started ramping upwards at a constant slope. Elevation is given in percent throughout, with 100% elevation corresponding to a 45° angle with the horizontal plane.

5.4.1 vision60

For the forward gait of the vision60 robot, the maximum elevation that still allowed the robot to continue walking for an indefinite period was 0.3% elevation. When increasing the elevation beyond this, the robot would stumble and eventually fall. In fig. 5.18, from looking at the pitch we see that the robot encounters the ramp at around $t = 10$ s, and that the trajectories seem to not change noticeably as a result (except for base height and pitch). However, we see from the fig. 5.19 that having already converged to a stable orbit prior to the ramp, the previously stable orbit no longer exists for the system with sloping ground. Thus, the system dynamics converge to a stable orbit of the new system if it is currently within its region of attraction. This can be seen in the plot as the discrete system suddenly moving away from the previously attractive fixed point and converging to a new fixed point.

5.4.2 ASTRo

For the forward gait of the ASTRo robot, the maximum elevation that could be reached while seemingly allowing the robot to keep walking indefinitely was 2% elevation. Increasing the slope beyond this would make the robot stumble and fall. We see in

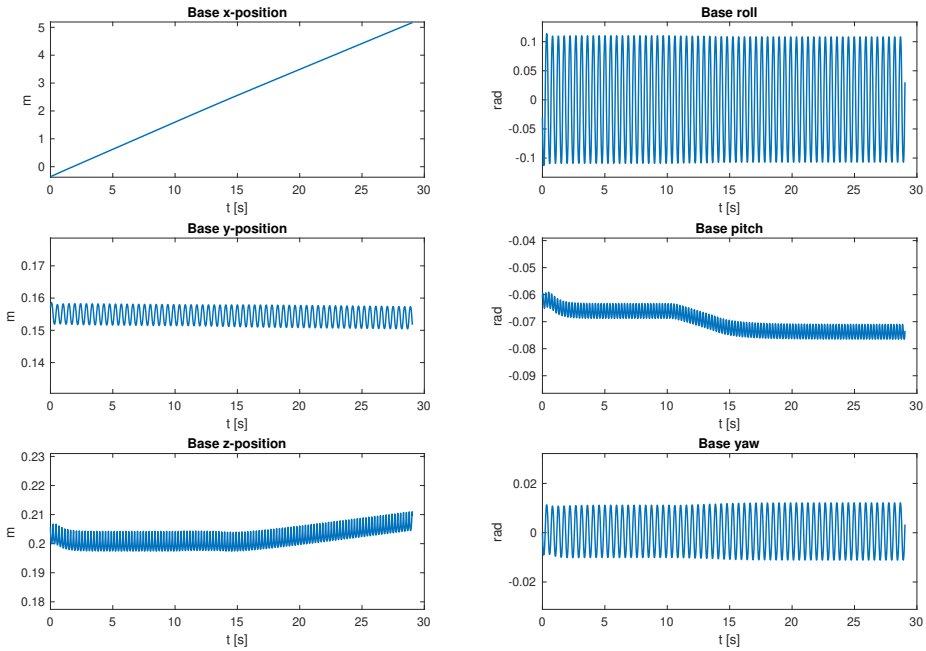


Figure 5.18: Trajectories of base pose states during 250 cycles of forward gait on ground with a ramp of 0.3% elevation for the vision60 robot during simulation

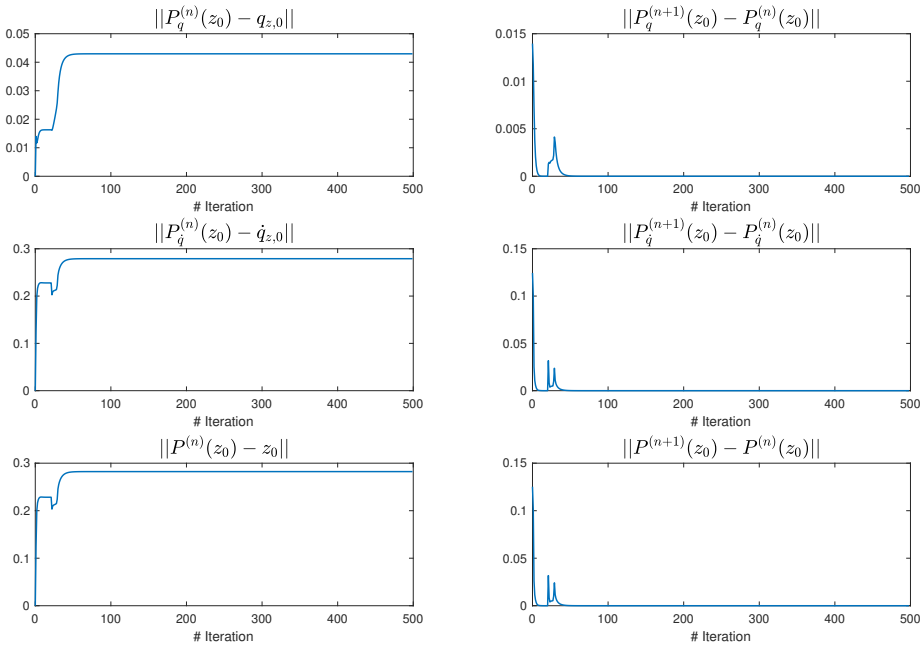


Figure 5.19: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on ground with a ramp of 0.3% elevation for the vision60 robot during simulation. Note that for this plot, the entire base position is omitted as the z-position is not periodic when the ground slopes.

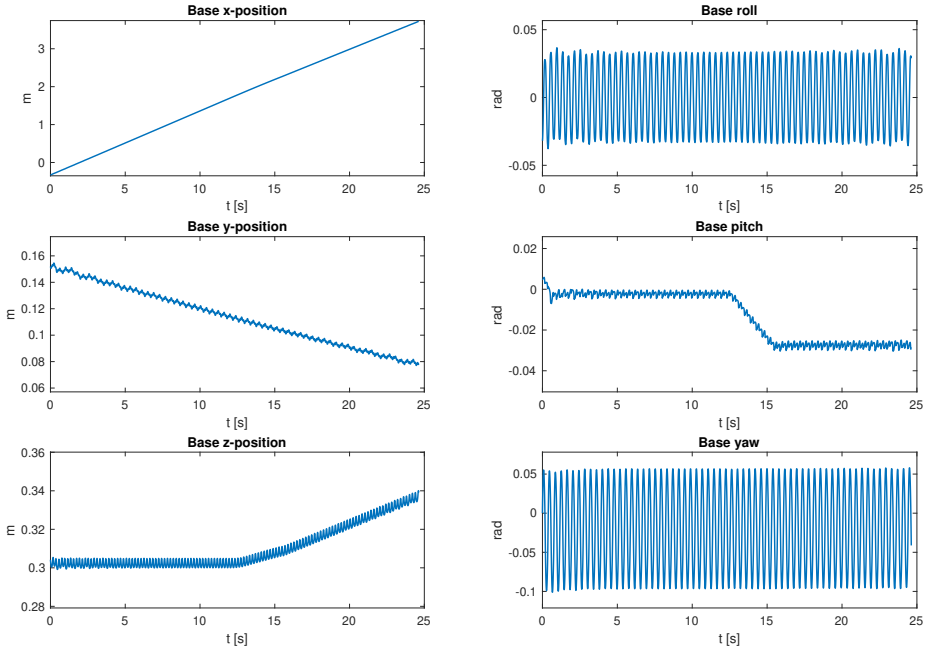


Figure 5.20: Trajectories of base pose states during 250 cycles of forward gait on ground with a ramp of 2% elevation for the ASTRo robot during simulation

fig. 5.20 from the pitch that the ramp is encountered at around $t = 12.5$ s. Here as well, only base height and pitch of the base seem to be clearly affected by the change in ground slope. Looking at fig. 5.21, the picture is not quite as clear as for the vision60 robot. After the robot reaches the slope, the discrete state does not seem to converge to a fixed point. However, the robot was tested on this slope for extended periods without failing. Also, while there is not convergence, there does seem to be some periodicity to the pattern emerging. There may be that while no fixed point is reached, some point $\mathbf{x}^*$ is reached so that $P^{(n)}(\mathbf{x}^*) = \mathbf{x}^*$ for some n. In that case, the behavior would still have converged to an orbit, albeit one that corresponds to n gait cycles as opposed to 1.

Unfortunately, none of the synthesized gaits managed to produce stable walking

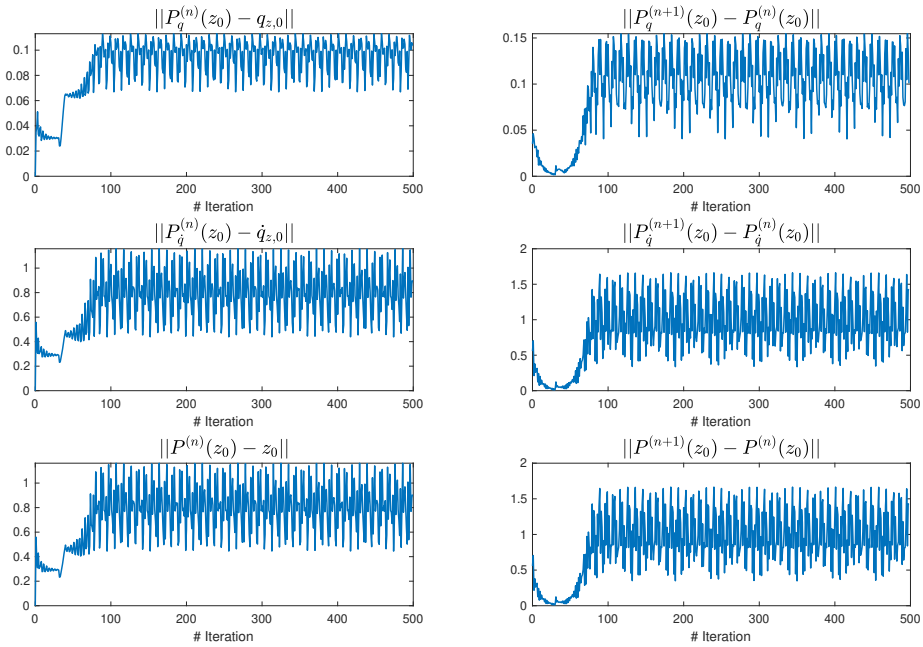


Figure 5.21: Norms of the error for iterative applications of the Poincaré map (corresponding to successive intersections between state and Poincaré section) between each other and between initial state, for a forward gait on ground with a ramp of 2% for the ASTRo robot during simulation. Note that for this plot, the entire base position is omitted as the z-position is not periodic when the ground slopes.

for ramps of elevation greater than 2%. However, the ASTRo robot managed this for a relatively significantly greater slope than the vision60 robot, somewhat supporting the hypothesis that sprawling quadrupeds may handle unmodeled terrain changes better than mammalian ones.

Here, a caveat must be given. As there is no practical way to optimize for adaptability to ground changes within the trajectory optimization problem, and as stability of the gait is also quite a different matter, it may well be that gaits could be produced for both robots which would have handled greater changes to the terrain that were presented here. As tuning the parameters for the optimization problem in order to yield such a gait is something of an art, we did unfortunately not yield any such gaits during experimentation with the gait synthesis.

5.5 Torque and energy expenditure

In what follows, the performance in terms of torque and energy expenditure are compared in two ways. Firstly, the performance is compared between the two robots which serve as examples of respectively sprawling and mammalian quadrupedal robots. Although a comparison with one sample of each is hardly comprehensive, it may serve to give an indication of what impact robot joint configuration has on performance. Secondly, we compare the performance for each robot between the scheme where 11 of 12 actuators are used in each domain – which is compatible with regular IO-feedback linearization and suggested as a solution to the problem of an over-constrained system in [6] – with our suggested modification to the conventional IO-feedback linearization controller as presented in section 4.1.7 – which allows for using all 12 actuators in an over-constrained system in a way which is, in one sense, optimal. In the interest of making a fair comparison, the same choice as in [6] is made for which actuator to "turn off" for each domain, namely, the hip pitch actuator of the rear stance leg.

The performance is summarized in the three measures CoT, peak torque and the RMS of the torques. We give a quick introduction to the CoT below.

5.5.1 Cost of Transport

Cost of Transport is a measure of the amount of energy expended by an entity (e.g. a vehicle or an animal) per mass transported and the distance it is moved. As the cost of transport is calculated per mass unit, it is a practical measure for comparing different modes of transportation or different entities against one another. The expression for calculating cost of transport is

$$\mathrm{CoT} = \frac{E}{mgd} \tag{5.1}$$

where E is the energy expended during transport, m and d is the mass transported and the distance it is transported, and g is standard gravity.

5.5.2 vision60

In fig. 5.22 we see a plot of the torque profiles of the vision60 robot while ambling forward using 11 actuators. If we compare this with the plot in fig. 5.23 where the robot is ambling forward using 12 actuators we may notice a couple of things.

Firstly, the peaks in hip roll torque is markedly reduced for all legs. For hip pitches the picture is less clear cut, with front leg torques being markedly decreased while rear torques actually increasing quite a bit. However, this should not be that surprising: The actuator which is being turned off for each domain is consistently the rear stance leg hip pitch actuator.

Assuming that each leg exerts most torque while being a supporting leg, this increase is exactly what one should expect to see. Further, it seems that the torque exertion is divided more evenly between front and rear hip pitches when all 12 actuators are used: While the front legs had peaks above 20 in the first case, the rear legs have peaks close to but below 20 N m in the second case.

For the knee pitch torques, we also notice a redistribution: While all legs had peaks above 40 N m for the 11 actuator case, only rear legs have peaks above 40 N m. However, it also seems that the peak torque expenditure among the knees are actually greater for the 12 actuator case.

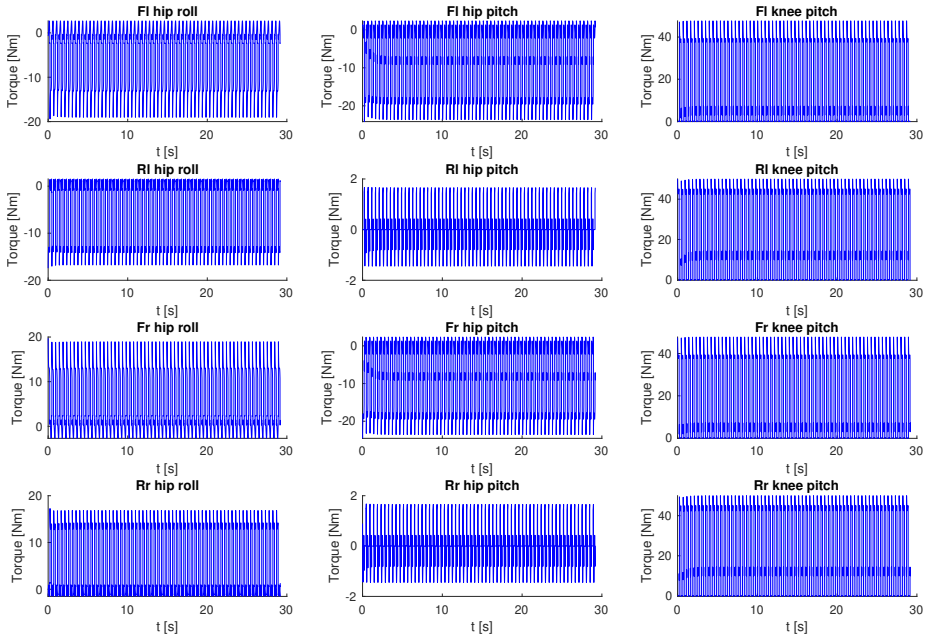


Figure 5.22: Overview of torques for vision60 robot during forward gait on flat ground using 11 actuators

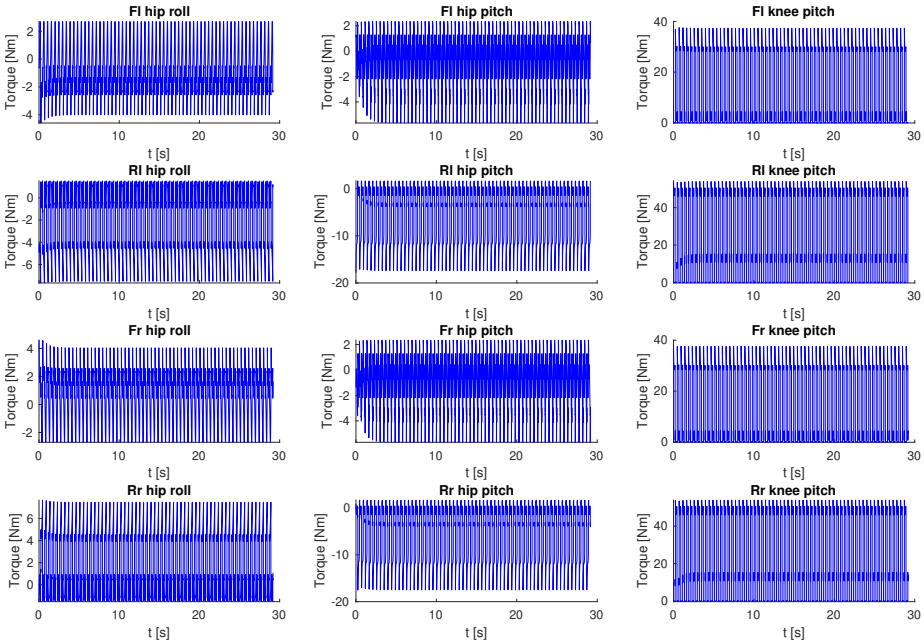


Figure 5.23: Overview of torques for vision60 robot during forward gait on flat ground using 12 actuators

5.5.3 ASTRo

In fig. 5.24, we see a plot of the torque profiles for the ASTRo robot while ambling forward using 11 actuators. We may compare this with the plot in fig. 5.25 of ASTRo ambling forward with 12 actuators.

For the hip yaw actuators, we note quite a clear reduction in the peaks for the front legs, and no clear increase (if anything, a light decrease) in the peaks for the rear legs.

For the hip pitch actuators, we notice that the torque exertion is more evenly divided for the 12 actuator case: The front legs experience a marked reduction from above 50 N m peaks (after initial transients) to below 40 N m, while the rear legs increase from below 2 N m peaks to slightly above 20 N m after transients. Again, the increase in rear leg torque peaks is to be expected as they were only active in the swing phase for the 11-actuator scenario.

For the knee pitches, the front legs seem to experience a reduction in peaks from slightly below 50 N m to slightly below 30 N m, while the rear legs increase peaks from approximately 30 N m to approximately 50 N m. In other words, there seems to be no clear net increase or decrease in the knee torques or how evenly they are distributed between legs for the knees.

5.5.4 Performance comparison across robots

We now compare vision60 and ASTRo on the performance metrics listed in table 5.2. It should be noted that the torque metrics are not directly comparable across robots, as the vision60 robot is somewhat heavier than ASTRo (26 kg vs 21.5 kg). However, in the case of the peak torques – though one would assume a heavier robot to have higher torques in general – we note that vision60 actually has markedly lower peak torques than ASTRo.

The CoT, being normalized with respect to weight, is directly comparable between robots. Here, we notice a marked difference both for the 11 and 12 actuator scenarios: For the first one, ASTRo has a CoT which is ~ 2.9 times as high as vision60. For the second scenario – as ASTRo benefits slightly more from the use of all actuators – this reduces somewhat to ~ 2.6 times, which is still quite significant.

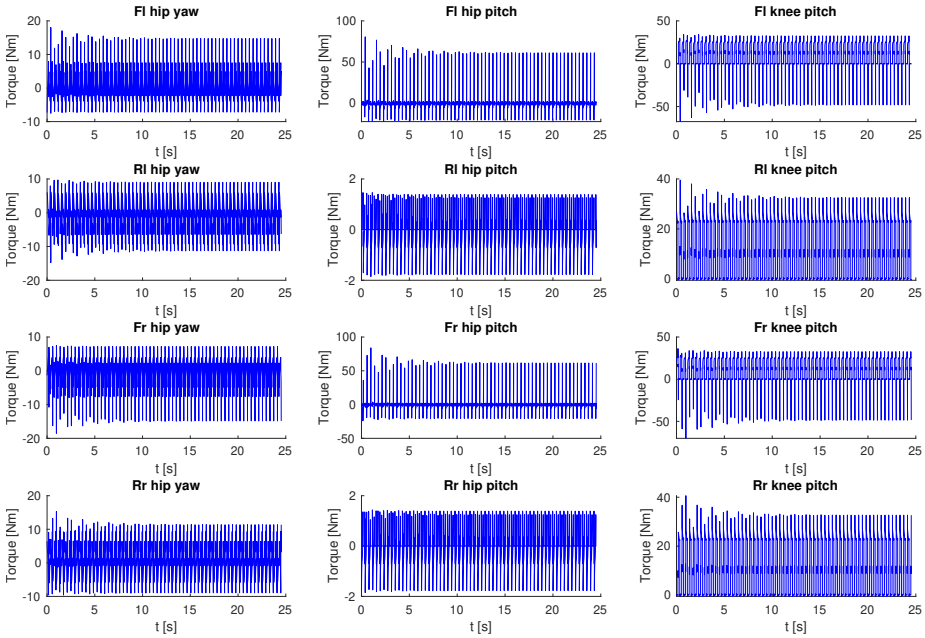


Figure 5.24: Overview of torques for ASTRo robot during forward gait on flat ground using 11 actuators

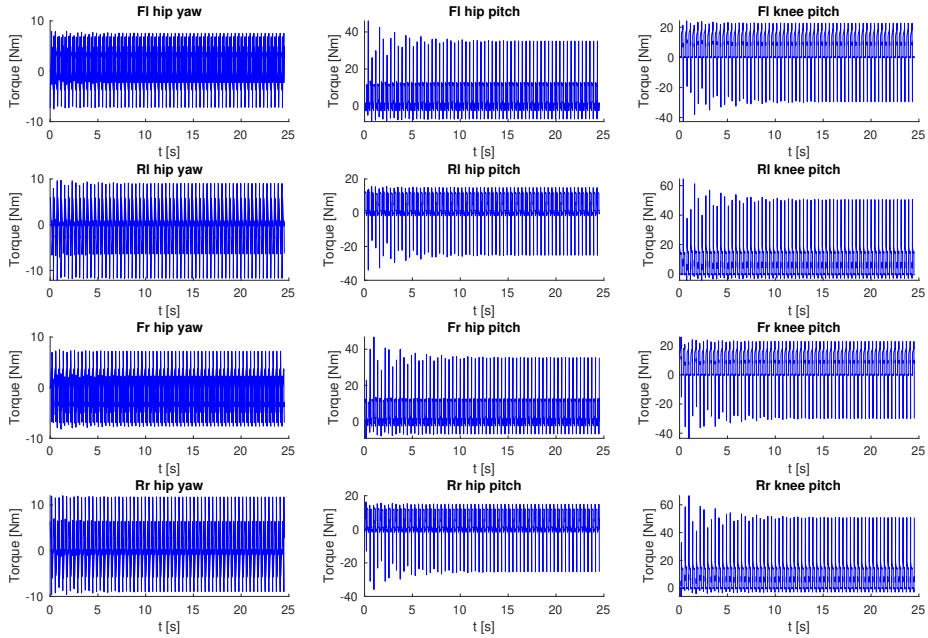


Figure 5.25: Overview of torques for ASTRo robot during forward gait on flat ground using 12 actuators

Whether this difference in energy efficiency points to the method being less suited for use on sprawling quadrupeds, or whether it points to sprawling quadrupeds simply being less energy efficient, is a matter of interpretation. However, the trajectory optimization problem converges without any clear trouble for the sprawling quadruped. This would by definition mean that the resulting gait is an – at least locally – optimal gait for ASTRo with respect to torque expenditure and for a given desired velocity. We take this to mean that it is unlikely that the difference in energy efficiency stems from some shortcoming in the method when applied to sprawling quadrupeds, in so far as it seems unlikely that a different method would yield a less torque-hungry gait.

All in all, this would indicate that sprawling robots may indeed be less energy-efficient than their mammalian counterparts.

5.5.5 Performance comparison between 11 and 12 actuators

We now compare the conventional IO-linearization controller utilizing 11 actuators – as suggested in [6] – to our modified IO-linearization controller which allows utilizing all 12 actuators at all times.

When comparing the summarizing metrics in table 5.2 for the vision60 robot, we see that the 12 actuator case does have a slightly higher peak torque than the 11 actuator case ($\sim$ 8.6% increase). While this may seem surprising on its face, the guarantee given in section 4.1.7 on torque expenditure was that the vector of all torques would be point-wise smaller in a *least-squares* sense ($\|\cdot\|_2$). This does not necessarily mean that the largest unsigned torque ($\|\cdot\|_\infty$) would be smaller. Further, we note that there is indeed a $\sim$ 9.9% decrease in RMS torque and, arguably most importantly, a $\sim$ 10.9% decrease in CoT from the 11 to the 12 actuator scenario.

For the ASTRo robot, the picture of difference in performance is quite clear. The peak torque is reduced by $\sim$ 20.7%, the RMS of the torque is reduced by $\sim$ 17.9%, and – importantly – the CoT is reduced by $\sim$ 20.3%.

All in all, the evidence put forth supports that there are concrete, practical advantages to the modified IO-linearization controller as a solution to overconstrained systems, when compared to the suggestion put forth in [6]. A $10 - 20\%$ reduction in

		ASTRo		vision60	
# actuators		11	12	11	12
CoT		4.1974	3.3449	1.4310	1.2744
Peak torque	[N m]	83.97	66.55	50.03	54.32
Torque RMS	[N m]	9.66	7.93	17.52	15.79

Table 5.2: Performance metrics for ASTRo and vision60 for respectively 11 (non-overconstrained) and 12 (overconstrained) actuators

CoT is quite significant, and we also invite the reader to consider that this is achieved without any change in the desired gait. In that sense, this decrease in CoT is achieved "for free" so to speak.

5.6 Possible limitations

The empirical evidence from simulations seems to support and confirm what has been shown mathematically – for orbital stability and the energy expenditure of the modified controller – and hypothesized at the outset – for robustness with respect to unmodeled ground changes and energy expenditure of different robots.

However, we also point out that we have synthesized only two gaits for one robot of each type. This is arguably a greater weakness for the hypotheses regarding inter-robot differences, than for the demonstration of what is already shown theoretically. Thus we point out that the results merely point in a direction of support when it comes to which type of robot is more robust to unmodeled changes and which is more energetically efficient, and should not be considered confirmatory in any way.

When it comes to the comparison in energy expenditure for the different controllers, while the efficiency increase may not be as dramatic as demonstrated here for all gaits, there should never be an increase in torque expenditure.

6

Conclusion and further work

The recent years have seen impressive progress in the fields of bipedal and quadrupedal robotic locomotion, and with it, the attempts at tackling increasingly challenging problems. The first commercial robots are already walking around industrial plants for surveillance, and the last year has seen the first quadrupedal robot complete a rough terrain hiking trail without falling, faster than the suggested average human time.

In this thesis we have surveyed various methods used to produce dynamic walking in mammalian quadrupedal robots, with the goal of evaluating if such methods are suited for sprawling quadrupedal robots as well. A method utilizing closed-loop trajectory optimization and stabilizing post-processing was chosen for adaptation and implementation on a sprawling quadrupedal robot. The chosen method utilizes a full-order model of the system, which sets it apart from most methods used for quadrupedal dynamic walking and should allow for the synthesis of a wide variety of highly dynamic gaits. Additionally, a method for post-processing the parameters of the synthesized controller is used to render controllers that are shown to be exponentially stable in analysis. Two different gaits were synthesized for one mammalian and one sprawling quadruped, and results from simulation were obtained in order to validate and evaluate the methods.

Results were evaluated on rate of convergence of the resulting trajectories, adaptability to unmodeled changes in ground height, peak and RMS torque and CoT. Results indicate that the method is well-suited for use on sprawling quadrupeds, as gaits were synthesized successfully and simulation of post-processed gaits indicated exponential orbital stability. Comparisons were made between robots on adaptability to terrain and energy-efficiency. Results indicate that mammalian robots may be more energy-efficient in terms of CoT, while sprawling robots may be less sensitive to unmodeled changes in ground height. However, a larger comparison involving several robots of each type and several gaits should be conducted to confirm these findings.

Additionally, a modification to the control structure of the chosen method was proposed to handle overconstrained systems without making an arbitrary selection of actuators, and it was shown that the modified method results in an equal or smaller torque use. Results indicate that this modified method compares favorably to the method as originally proposed on measures such as CoT and RMS torque.

6.1 Further work

The use of higher-order models for which stability can be ensured theoretically is part of a maturation in the field of quadrupedal robotic locomotion, both in order to develop and execute highly dynamic and sophisticated gaits as well as to give better guarantees on robot behavior. However, in order to flexibly react to unmodeled real-world environments, controllers must be endowed with the ability to deviate from precomputed trajectories. Consequently, the gap between high-order, computationally expensive trajectory optimization methods and flexible, fully-online locomotion methods with weaker theoretical guarantees should be bridged.

While this may take many forms, efforts to combine offline trajectory optimization with Model Predictive Control approaches – while still attempting to show stability – might be a valuable way forward. This might take the form of utilizing a bank of such trajectories as initial guesses, as in [32], or to integrate convergence to such a gait into the terminal cost of the problem, as in [33]. However, it is also clear that a combination of clever problem formulations, research within optimization and

advances in hardware has and continues to narrow the gap to fully online trajectory optimization with high-order models.

In conclusion, we strongly believe that the coming decades will see the already emerging market of legged robots come to affect a wide variety of commercial sectors. Further research into the strengths and weaknesses of sprawling and mammalian quadrupeds should be done in order to appreciate which niches are best filled by each.

References

[1] M. E. B. Lysø, "Robust dynamic walking in a sprawling quadrupedal robot," Norwegian University of Science and Technology, Tech. Rep., 2022.

[2] K. Schwab, *The fourth industrial revolution*. Geneva: World Economic Forum, 2016, tex.added-at: 2017-04-15T12:00:39.000+0200 tex.biburl: https://www.bibsonomy.org/bibtex/2ff7fa4c3e5e06e332bc41758a4f3e74b/flint63 tex.interhash: 715dea162e742a1b489ffcd54183d463 tex.intrahash: ff7fa4c3e5e06e332bc41758a4f3e74b tex.timestamp: 2018-04-16T12:09:57.000+0200 tex.username: flint63.

[3] A. Ghansah and P. Thorseth, "Design and Control of a Torque Controllable Quadrupedal Robot - A study on the development of ASTRo," Ph.D. dissertation, Norwegian University of Science and Technology, Nov. 2021.

[4] K. Akbari Hamed, B. Buss, and J. Grizzle, "Exponentially Stabilizing Continuous-Time Controllers for Periodic Orbits of Hybrid Systems: Application to Bipedal Locomotion with Ground Height Variations," *The International Journal of Robotics Research, accepted to appear*, Jun. 2015.

[5] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic Humanoid Locomotion: A Scalable Formulation for HZD Gait Optimization," *IEEE Transac-*

tions on Robotics, vol. 34, no. 2, pp. 370–387, Apr. 2018, conference Name: IEEE Transactions on Robotics.

[6] W.-L. Ma, K. A. Hamed, and A. D. Ames, "First Steps Towards Full Model Based Motion Planning and Control of Quadrupeds: A Hybrid Zero Dynamics Approach," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 5498–5503, iSSN: 2153-0866.

[7] M. Vukobratovic and B. Borovac, "Zero-Moment Point - Thirty Five Years of its Life." *I. J. Humanoid Robotics*, vol. 1, pp. 157–173, Mar. 2004.

[8] H. Yeom and J. Bae, "A dynamic gait stabilization algorithm for quadrupedal locomotion through contact time modulation," *Nonlinear Dynamics*, vol. 104, no. 3, pp. 2275–2289, May 2021. [Online]. Available: https://doi.org/10.1007/s11071-021-06376-5

[9] H.-W. Park and S. Kim, "Quadrupedal galloping control for a wide range of speed via vertical impulse scaling," *Bioinspiration & Biomimetics*, vol. 10, no. 2, p. 025003, Mar. 2015, publisher: IOP Publishing. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-3190/10/2/025003/meta

[10] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9, iSSN: 2153-0866.

[11] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018, conference Name: IEEE Robotics and Automation Letters.

[12] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive CLF-MPC With Application to Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 565–572, Jan. 2022, conference Name: IEEE Robotics and Automation Letters.

[13] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, Jan. 2022, arXiv: 2201.08117. [Online]. Available: http://arxiv.org/abs/2201.08117

[14] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, Oct. 2019, arXiv: 1710.08864. [Online]. Available: http://arxiv.org/abs/1710.08864

[15] K. A. Hamed and R. D. Gregg, "Decentralized Feedback Controllers for Robust Stabilization of Periodic Orbits of Hybrid Systems: Application to Bipedal Walking," *IEEE transactions on control systems technology: a publication of the IEEE Control Systems Society*, vol. 25, no. 4, pp. 1153–1167, Jul. 2017.

[16] K. A. Hamed, R. D. Gregg, and A. D. Ames, "Exponentially Stabilizing Controllers for Multi-Contact 3D Bipedal Locomotion," in *2018 Annual American Control Conference (ACC)*, Jun. 2018, pp. 2210–2217, iSSN: 2378-5861.

[17] A. Isidori, "The Zero Dynamics of a Nonlinear System: from the Origin to the Latest Progresses of a Long Successful Story," in *Proceedings of the 30th Chinese Control Conference*, Jul. 2011, pp. 18–25, iSSN: 2161-2927.

[18] K. A. Hamed, Wen-Loong, and A. D. Ames, "Dynamically Stable 3D Quadrupedal Walking with Multi-Domain Hybrid System Models and Virtual Constraint Controllers," *arXiv:1810.06697 [math]*, Oct. 2018, arXiv: 1810.06697. [Online]. Available: http://arxiv.org/abs/1810.06697

[19] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002, google-Books-ID: t_d1QgAACAAJ.

[20] A. D. Ames, "Human-Inspired Control of Bipedal Walking Robots," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1115–1130, May 2014, conference Name: IEEE Transactions on Automatic Control.

[21] M. A. Henson and D. E. Seborg, *Nonlinear Process Control*. Prentice Hall PTR, 1997, google-Books-ID: tZceAQAAIAAJ.

[22] M. Hutter, *StarlETH & Co - Design and Control of Legged Robots with Compliant Actuation*. ETH, 2013, google-Books-ID: KSa2zQEACAAJ.

[23] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 413–432, 1973, tex.eprint: https://doi.org/10.1137/0710036. [Online]. Available: https://doi.org/10.1137/0710036

[24] R. I. Leine and H. Nijmeijer, "Fundamental Solution Matrix," in *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*, ser. Lecture Notes in Applied and Computational Mechanics, R. I. Leine and H. Nijmeijer, Eds. Berlin, Heidelberg: Springer, 2004, pp. 101–124. [Online]. Available: https://doi.org/10.1007/978-3-540-44398-8_7

[25] "Convex Relaxation of Bilinear Matrix Inequalities Part I: Theoretical Results," Sep. 2018, number: arXiv:1809.09814 arXiv:1809.09814 [math]. [Online]. Available: http://arxiv.org/abs/1809.09814

[26] F. Zhang, "Block Matrix Techniques," in *The Schur Complement and Its Applications*, ser. Numerical Methods and Algorithms, F. Zhang, Ed. Boston, MA: Springer US, 2005, pp. 83–110. [Online]. Available: https://doi.org/10.1007/0-387-24273-2_4

[27] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 719–726, iSSN: 2153-0866.

[28] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, Sep. 2004, pp. 284–289.

[29] D. Henrion, J. Lofberg, M. Kocvara, and M. Stingl, "Solving polynomial static output feedback problems with PENBMI," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 7581–7586, iSSN: 0191-2216.

[30] Q. Tran Dinh, S. Gumussoy, W. Michiels, and M. Diehl, "Combining Convex–Concave Decompositions and Linearization Approaches for Solving BMIs, With Application to Static Output Feedback," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1377–1390, Jun. 2012, conference Name: IEEE Transactions on Automatic Control.

[31] "k-Contraction: Theory and Applications," Sep. 2021, number: arXiv:2008.10321 arXiv:2008.10321 [math]. [Online]. Available: http://arxiv.org/abs/2008.10321

[32] M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros, and M. Hutter, "Offline motion libraries and online MPC for advanced mobility skills," Jun. 2022. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/02783649221102473

[33] M. Y. Galliker, N. Csomay-Shanklin, R. Grandia, A. J. Taylor, F. Farshidian, M. Hutter, and A. D. Ames, "Bipedal Locomotion with Nonlinear Model Predictive Control: Online Gait Generation using Whole-Body Dynamics," *arXiv:2203.07429 [cs]*, Mar. 2022, arXiv: 2203.07429. [Online]. Available: http://arxiv.org/abs/2203.07429

Mads Erlend Bøe Lysø

Dynamic Walking in a Sprawling Quadruped

NTNU
Norwegian University of
Science and Technology