TOWARD HIGH-PERFORMANCE SIMPLE MODELS OF LEGGED LOCOMOTION

Yu-Ming Chen

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2023

Supervisor of Dissertation

Michael Posa, Assistant Professor of Mechanical Engineering and Applied Mechanics

Graduate Group Chairperson

Troy Olsson, Associate Professor of Electrical and Systems Engineering

Dissertation Committee

Nikolai Matni, Assistant Professor of Electrical and Systems Engineering
Michael Posa, Assistant Professor of Mechanical Engineering and Applied Mechanics
Daniel Koditschek, Professor of Electrical and Systems Engineering
Jerry Pratt, Chief Technology Officer, Figure

TOWARD HIGH-PERFORMANCE SIMPLE MODELS OF LEGGED LOCOMOTION

COPYRIGHT

2023

Yu-Ming Chen

ACKNOWLEDGEMENT

Toyota Research Institute provided funds to support the work in this thesis. The work about the Integrable Whole-body Orientation was also supported by DAC Cooperative Agreement W911NF2120241, ONR Grant No. N00014-22-1-2593, and ONR Contract No. N00014-19-1-2023.

This thesis would not have been possible without the people around me during my PhD study. I am sincerely grateful for every single of you.

I thank my advisor Michael Posa who does not only encouraged me but also guided me to be a researcher and an engineer. I have learned so much from him.

I thank my thesis committee members, Nikolai Matni, Daniel Koditschek and Jerry Pratt, who took their time among their busy schedules to listen to my talks, ask great questions and provide constructive feedback.

I thank my collaborators Gabriel Nelson, Robert Griffin, Hien Bui and Jianshu Hu. Special thank to Gabe who taught me the Integrable Whole-body Orientation and let me work on the project that he started when I joined Boardwalk Robotics.

I thank many members of Boardwalk Robotics and IHMC. Thank you all for making my summer in Pensacola so awesome, so fun yet so research-productive. Thank you Jerry for giving me the opportunity to work at Boardwalk Robotics and for showing me what it looks like to start a robotic company. Thank you Robert for always being available for a discussion whenever I needed one. Thank you Nomi Yu, Sylvain Bertrand, Brandon Shrewsbury, James Foster and Stephen McCrory for teaching me how to use IHMC's software. Thank you Stefan Fasano and Joseph Godwin for helping with the experiment during the crunch time. Thank you the hardware team for the amazing humanoid, Nadia, and your instant support whenever the robot is broken.

I thank every DAIR Lab member for engaging in conversations with me on various topics and for going through parts of my PhD journey with me. Thank you Will and Brian for your help in Cassie

experiments and for many great utility functions you wrote. Thank you Matt for your advice in job hunting. Thank you Hien for answering many of my questions about Reinforcement Learning. Thank you Alp for finishing the final stretch with me. Thank you everyone for all the critical feedback.

I thank the Michigan Robotics community for the very supportive environment. Particularly, I want to thank Brian and Brandt for being a friend and for supporting me during my PhD application. I miss the fun nights, basketball games and pseudoscience lunch that we had at Ann Arbor.

I thank all my friends that I made during this journey for their company, encouragement and advice in life. Your support meant a lot to me and was what motivated me to take an extra step towards completing the PhD degree when I was tired.

Lastly, I thank my parents for always being supportive of my decisions, even though it meant I had to leave Taiwan for studying on the other side of the Earth. And I thank my wife for her cares and love. The past five years would not have been as memorable, warm and happy without her.

ABSTRACT

TOWARD HIGH-PERFORMANCE SIMPLE MODELS OF LEGGED LOCOMOTION

Yu-Ming Chen

Michael Posa

This thesis addresses the challenges of model-based planning and control in legged locomotion, particularly the trade-off between computational speed and robot performance presented by different levels of model complexities. Full-order models, while rich in detail, are often too computationally demanding for real time planning, whereas conventional reduced-order models (ROMs) tend to oversimplify the dynamics, limiting overall performance potential. Our research focuses on a novel approach – the direct optimization of ROMs. This study seeks to enhance the performance of legged robots by automatically discovering the optimal ROMs that simultaneously deliver high robot performance while maintaining the necessary low dimensionality for real time planning applications.

In this work, we formulate problems, provide algorithmic solutions, and deploy optimized ROMs on real robots. In the beginning of the thesis, we focus on a special case where we aim to find whole-body orientation coordinates (WBO) for legged robots that minimize angular momentum errors. This optimal WBO, while being a simple forward kinematic function, serves as a proxy of the real angular momentum and can be applied to complex tasks such as humanoid natural walking. In the second part of the thesis, we formulate a bilevel optimization problem to find optimal ROMs agnostic to controller choices, driven by user-defined objectives and task distributions. The results show substantial improvements in walking speed, ground slope adaptability and torque efficiency on a bipedal robot Cassie. Lastly, we cast the ROM optimization problem as a model-based reinforcement learning (RL) problem to further improve the model performance. This does not only show better performance improvements in experiment but also provide an easier way to implement model optimization and to realize the model performance on the robot.

TABLE OF CONTENTS

ACKNOWLEDGEMENT iii

ABSTRACT v

LIST OF TABLES viii

LIST OF ILLUSTRATIONS ix

CHAPTER 1 : INTRODUCTION 1

1.1 Outline and Contributions 2

CHAPTER 2 : BACKGROUND 4

2.1 Full-order Models of Cassie 4

2.2 Reduced-order Models (ROMs) of Legged Robots 6

2.3 Model Predictive Control (MPC) with ROMs 8

2.4 Definition of ROMs 9

2.5 Definition of Mirrored ROMs 10

2.6 Trajectory Optimization 12

2.7 Heuristics in Trajectory Optimization 13

2.8 Bilevel Optimization 15

CHAPTER 3 : RELATED WORKS 17

3.1 Whole-body Orientation for Legged Robots 17

3.2 Model Order Reduction and ROM Optimization for Legged Robots 18

3.3 Model-based Reinforcement Learning for Legged Robots 19

CHAPTER 4 : INTEGRABLE WHOLE-BODY ORIENTATION FOR LEGGED ROBOTS 21

4.1 Whole-body Orientation of Simple Systems 23

4.2 Whole-body Orientation of Complex Robots 29

4.3 Walking Example 34
4.4 Running Example 38
4.5 Conclusion and Future Work 39

CHAPTER 5 : OPTIMAL REDUCED-ORDER MODELING OF LEGGED LOCOMOTION 41
5.1 ROM Optimization 43
5.2 MPC for a Special Class of ROMs 53
5.3 Performance Evaluation and Comparison 59
5.4 MPC for General ROMs 69
5.5 Discussion 73
5.6 Conclusion 77

CHAPTER 6 : REINFORCEMENT LEARNING FOR REDUCED-ORDER MODELS OF LEGGED LOCOMOTION 78
6.1 Problem Statement 80
6.2 ROM Optimization via Reinforcement Learning 81
6.3 Experimental Result 84
6.4 Conclusion and Future Work 87

CHAPTER 7 : CONCLUSION 89
7.1 Future work 90

BIBLIOGRAPHY 91

LIST OF TABLES

TABLE 4.1 Notation definitions and correspondences 29

TABLE 5.1 Examples of model optimization. This table includes the task space used to train models (uniform task distribution), the highest order of the monomials of basis functions, the dominant term of the cost function J_γ, and the cost reduction percentage (relative to the cost of the initial model). 50

TABLE 5.2 Trajectories and gains in the Operational Space Control (OSC) 55

TABLE 5.3 Experiments conducted in Section 5.3 (marked with x) 61

TABLE 5.4 Criteria to determine periodic walking gaits 61

TABLE 5.5 Completion time for some segments of the course. 64

TABLE 6.1 Hyperparameters for the model learning 87

LIST OF ILLUSTRATIONS

FIGURE 2.1 Cassie and its model 5

FIGURE 2.2 The linear inverted pendulum (LIP) model. It is a point mass model of which height is restricted in a plane. The point mass and the origin of this model correspond to the center of mass and the stance foot of the robot, respectively. In the examples of Chapters 5 and 6, we initialize the reduced-order model to the LIP model during model optimization. 7

FIGURE 2.3 An outline of a model predictive control. The planner outputs desired trajectories of the reduced-order model. These trajectories are then tracked by a controller based on inverse dynamics (ID) or inverse kinematics (IK) which uses the full model of the robot. 8

FIGURE 2.4 Relationship of the full-order and reduced-order models. The generalized positions q and y satisfy the embedding function r for all time, and the evolution of the velocities $\dot{q}$ and $\dot{y}$ respects the dynamics f and g, respectively. 10

FIGURE 2.5 The mirror function M mirrors the robot configuration about the sagittal plane. This function is necessary in planning and control when the embedding function (Eq. (2.5a)) of the reduced-order model only represents one side of the robot. For example, the embedding function of the LIP model was chosen to be the CoM relative to the left foot (and not the right foot). 11

FIGURE 4.1 Left: The humanoid Nadia walking with arm swing and spine yaw rotation induced by tracking WBO. Right: The simulated biped Cassie running and turning with the whole-body orientation (WBO) coordinate visualized at the CoM. 22

FIGURE 4.2 While linear momentum is integrable, angular momentum is *generally* not (Nakamura and Mukherjee, 1990; Saccon et al., 2017). 22

FIGURE 4.3 Bar-and-Flywheel model (an integrable system). The motor rotates the flywheel counter-clockwise. 25

FIGURE 4.4 Bar-and-Flywheel model with an actuated prismatic joint (a non-integrable system). This system can reorient itself without external torques. For example, the entire system can rotate counter-clockwise if the joints (d, ϕ) repeat the following cyclic motion: $(0, 0) \to (l/2, 0) \to (l/2, \pi/2) \to (0, \pi/2) \to (0, 0)$, where l is the length of the bar. 27

FIGURE 4.5 The humanoid Nadia (left) and frames of interest (right). W, B and Wbo are the world frame, base frame and the WBO frame, respectively. $Q_{W,B}$ is the base orientation in the world, and $Q_{B,Wbo}$ is the orientation of the WBO frame relative to the base. Translationally, we locate the WBO frame at the CoM for convenience. 29

FIGURE 4.6 Comparisons between real and approximated quantities. The data is from a simulation where Nadia walked in a straight line at 0.6 m/s. 33
FIGURE 4.7 Controller diagrams. Each diagram is composed of a high-level planner, low-level feedback controllers and an inverse-dynamics quadratic program (QP). Blue color highlights the difference between the two controllers. Red color is used for indicating the task priorities. Tier 0 is implemented as a constraint in the QP, while other Tiers are implemented via cost functions in the QP. Additionally, Tier n has higher priority than Tier $n+1$ for $n > 0$. We use the null-space projection technique to prioritize tasks (Hutter et al., 2013). The zeros and nominal joint configuration in the planner are constant trajectory sources. 36
FIGURE 4.8 The CAM about the z axis when Nadia walked in a straight line. We note that there were issues with Nadia's leg actuator at the time of hardware experiment and the update rate of the control loop was not fast. These issues partially caused the non-smoothness in the hardware plot. 37
FIGURE 4.9 The CAM about the z axis when Cassie runs at 2.7 m/s and follows a desired yaw trajectory that goes from 0 to $\pi/2$ rad in 10 seconds. 38

FIGURE 5.1 An outline of the synthesis and deployment of optimal reduced-order models (ROM). Offline, given a full-order model and a distribution of tasks, we optimize a new model that is effective over the task space (Section 5.1). Online, we generate new plans for the reduced-order model and track these trajectories on the true, full-order system (Section 5.2). This diagram also shows the bipedal robot Cassie (in the rightmost box) and its full model. Cassie has five motors on each leg – three located at the hip, one at the knee and one at the toe. Additionally, there are 2 leaf springs in each leg, and the spring joints are visualized by q_{16} to q_{19} in the figure. The springs are a part of the closed-loop linkages of the legs. We model these linkages with distance constraints, so there are no rods visualized in the model. 42
FIGURE 5.2 The averaged cost of the sampled tasks of each model optimization iteration in Examples 1 to 5. Costs are normalized by the cost associated with the full-order model (i.e. the cost of full model trajectory optimization without any reduced-order model embedding). Therefore, the costs cannot go below 1. The costs at iteration 1 represent the averaged costs for the robots with the embedded initial reduced-order models, LIP. Note that the empirical average does not strictly decrease, as tasks are randomly sampled and are of varying difficulty. 52

FIGURE 5.3 The diagram of the MPC introduced in Section 5.2. The MPC is composed of the controller process and the planner process, and it contains a time-based finite state machine which outputs either left or right support state. This finite state machine determines the contact sequence of the high-level planner and the contact mode of the low-level model-based controller. The high-level planner solves for the desired reduced-order model trajectories and swing foot stepping locations, given tasks (commands) and the finite state. For reduced-order models without body orientation (e.g. CoM model without moment of inertia), we send the turning rate command to the controller process instead of planner process. Inside the controller process, the regularization trajectories are used to fill out the joint redundancy of the robot. These regularization trajectories are derived from simple heuristics such as maintaining a horizontal attitude of the pelvis body, having the swing foot parallel to the contact surface, and aligning the hip yaw angle with the desired heading angle. All desired trajectories are sent to the Operational Space Controller (OSC) which is a quadratic-programming based inverse-dynamics controller (Sentis and Khatib, 2005; Wensing and Orin, 2013). 54
FIGURE 5.4 Cost comparison between the initial model (R2) and the optimal model (R3). Each plot shows the ratio of the optimal model's cost to the initial model's cost. For these examples, the ROM reduces the cost across the entire task space. The color scheme red-to-blue illustrates the degree to which the ROM shows improvement, with red corresponding to a minimal improvement and blue to a 30% reduction. The scales of the axes are the same between the trajectory optimization (C1) and the simulation (C2) for ease of comparisons. 59
FIGURE 5.5 A track designed to showcase the performance difference between the LIP and the optimal ROM in simulation. The video of Cassie finishing the track can be found in the supplementary materials. 64
FIGURE 5.6 Cost comparison between the initial model (R2) and the optimal model (R3). The cost $\tilde{h}_\gamma$ is the cost function in Eq. (TO). For trajectory optimization and simulation, we densely sampled the tasks and interpolated the costs. For the hardware experiment, we plot the costs of collected data points directly in the figure without interpolation. To collect the data on hardware, we used a remote control to walk Cassie around, and we applied a moving window of 4 foot steps to extract periodic gaits according to Table 5.4. We note that there was about 2 cm of height variation in the hardware experiment, but we normalized every extracted data point to the same height using the height-cost relationship from the trajectory optimization to make fair comparisons. . . . 65

FIGURE 5.7 The vector fields of the ROM dynamics g over the CoM x and z position. In this example, the dynamics is the acceleration of the CoM, which is a function of the CoM position and velocity defined in Eq. (2.5b). The first plot is the initial model's dynamics, while the latter two are that of the optimal model at two different slices of CoM y position. In all plots, the CoM velocity is 0. We note that the size of the vectors only reflects the relative magnitude. The absolute magnitude of the vectors for the first two plots are shown in Fig. 5.8, although the scales of the x axis are different. 66

FIGURE 5.8 The magnitude of ROM dynamics g over the CoM x and z position. The settings of Fig. 5.8a and 5.8b are the same as Fig. 5.7a and 5.7b, respectively. The magnitude plots show that the optimal model has smaller CoM accelerations, which implies smaller ground reaction forces, particularly in the x axis (implied by the vector fields in Fig. 5.7). We observed in the experiments that these force vectors align more closely with the normal direction of the ground. 67

FIGURE 5.9 An example of the real time planner in Eq. (5.22). Given a task of covering two meters in four steps starting from a standing pose, we rapidly plan a trajectory for the reduced-order model. The high-dimensional model is used to capture the hybrid event at stepping, as illustrated in the diagram. 72

FIGURE 6.1 The diagram of our reinforcement learning framework for reduced-order models (ROMs) of legged locomotion. We learn a ROM in simulation where the robot is controlled via a real time MPC presented in Section 5.2. The MPC follows a high-level command (task) and operates based on a time-based finite state machine (FSM), governing footstep timing in left-support, right-support, or double-support state. The MPC contains two trajectory generators – a reduced-order model planner and a regularization trajectory generator to fill out the joint redundancy of the robot. All desired trajectories are converted to desired acceleration command via PD feedback control before being sent to the Operational Space Controller (OSC), a quadratic-programming-based inverse-dynamics controller (Sentis and Khatib, 2005; Wensing and Orin, 2013). In this learning framework, the ROM planner is the policy, while everything else in the closed-loop system, such as the simulation and OSC, constitutes the environment. We parameterize the policy using ROM parameters, specifically the parameters of ROM dynamics. The optimizer collects data from simulation rollouts and updates the model (policy) parameters according to a user-specified reward function. To ensure that the optimizer collects data at a consistent rate, we maintain a fixed update rate of 20Hz for the ROM planner, and we downsample the environment state (which operates at 1000Hz) to match this 20Hz rate. 79

FIGURE 6.2 Cost landscape comparisons. Fig. 6.2a compares the optimal model to the initial model. Fig. 6.2b compares the optimal models between this chapter and Chapter 5. For each model comparison, we create landscapes in two sets of experiments – one involving turning rate and stride length (with a constant ground incline of 0 rad), and the other involving ground incline and stride length (with a constant turning rate of 0 rad/s). Each plot shows the ratio of the optimal model's cost to the compared model's cost. The color scheme red-to-blue illustrates the degree to which the optimal ROM shows improvement. Ratio below 1 means the optimal ROM performs better than the compared model, and vice versa. Green color shows the task regions gained by the optimal model, and orange color shows the task regions lost by the optimal model. 85

CHAPTER 1

INTRODUCTION

State-of-the-art approaches for model-based planning and control of legged locomotion can be categorized into two types (Wensing et al., 2022). One leverages the full-order model of a robot, while the other relies on a reduced-order model (ROM). Using the full model allows us to utilize a comprehensive understanding of the robot's dynamics, leading to high performance (Westervelt et al., 2003; Zhao et al., 2017; Reher et al., 2016). However, this comes at the cost of substantial computational resources and poses challenges in formal analysis, particularly given the complexity of modern legged robots with numerous degrees of freedom (Ackerman, 2017; Agility Robotics; Robotics; Boston Dynamics). To manage this complexity, the community of legged robotics has embraced the use of reduced-order models.

Most reduced-order models adopt constraints (assumptions) on the full model dynamics while capturing the task-relevant part of the full-order dynamics. For example, the linear inverted pendulum (LIP) model (Kajita and Tani, 1991; Kajita et al., 2001) assumes that the robot is a point mass that stays in a plane, which greatly reduces energy efficiency and limits the speed and stride length of the robot. The spring loaded inverted pendulum (SLIP) model (Blickhan, 1989) is a point mass model with spring-mass dynamics, which implies zero centroidal angular momentum rate and zero ground impacts at foot touchdown event. Therefore, when we plan for motions only in the reduced-space, we unavoidably impose limitations on the full dynamics. This restricts a complex robot's motion to that of the low-dimensional model and necessarily sacrifices performance of the robot.

Recognizing the above limitations of the ROMs, researchers have explored various extensions to reduced-order models, often relying on human intuition and incorporating mechanical components (e.g., springs, dampers, rigid bodies, second legs) (Xiong and Ames, 2020; Xiong et al., 2021; Kasaei et al., 2020; Takenaka et al., 2009; Sato et al., 2010; Shimmyo et al., 2012; Kasaei et al., 2018; Faraji and Ijspeert, 2017). Several successful model extensions have enabled high-performance real-time planning on hardware. For example, Chignoli et al. (Chignoli et al., 2021) utilized a

single rigid body model with small body pitch and roll angles to formulate a convex planning problem. Xiong et al. (Xiong and Ames, 2020) extended LIP with a double-support phase while maintaining the zero ground impact assumption, so the model is still linear and conducive to a LQR controller (Shaiju and Petersen, 2008; Garcia et al., 1989; Borrelli et al., 2017). Gibson et al. (Gong and Grizzle, 2022; Gibson et al., 2022) introduced the angular-momentum-based LIP, offering improved prediction accuracy. Dai et al. and Herzog et al. (Dai et al., 2014; Herzog et al., 2016) combined the centroidal momentum model with full robot configurations, and Boston Dynamics demonstrated its application in parkour on Atlas humanoid(Marion and the team).

While some of the above extensions have improved the robot performance, it remains unclear which extension provides more performance improvement than the others, and we do not have a metric to improve the model performance with. Moreover, it has been shown that not all model extensions can significantly improve the performance of robots. For example, allowing the center of mass height to vary provides limited aid in the task of balancing (Posa et al., 2017a; Koolen et al., 2016b).

In this thesis, we aim to algorithmically discover the low-dimensional representation of a high-dimensional robot which maximizes the performance with respect to an objective function and task distribution.

1.1. Outline and Contributions

We start by presenting the necessary technical background and introducing the definition of a reduced-order model in Chapter 2. We then cover the related works of ROM optimization in Chapter 3.

Our main contributions start with Chapter 4. It investigates a special case of model optimization where we search for a whole-body orientation (WBO) with a specific objective function – minimizing the angular momentum error between the ROM and the full model. We present a simple example to clarify the concept behind the WBO, with clear definitions of how the problem and solution are structured. We provide a concise algorithm that finds an WBO representation for a general multibody robot in 3D. Lastly, we demonstrate the use of the WBO on hardware (the humanoid

robot Nadia) and in simulation (the biped robot Cassie), showing improvements in reducing angular momentum oscillation and foot yaw moment.

Chapter 5 generalizes Chapter 4 by allowing optimizing a ROM given any user-specified objective function and task distribution. We propose a bilevel optimization algorithm to automatically synthesize new reduced-order models, embedding high-performance capabilities within low-dimensional representations. We show several examples of model optimization, with different sizes of task space and basis functions. We then design a real time model predictive control for the optimal ROM, and demonstrate in simulation that the optimal model reduces the cost of joint torques by up to 23% and increases its walking speed by up to 54%. We also show hardware result that the real robot walks on flat ground with 10% lower torque cost. We analyze the source of the performance gain and discuss the lessons learned in transferring the model performance from an open-loop system to a closed-loop system. To the best of our knowledge, this is the first work that directly optimizes a reduced-order model with respect to an objective function.

Chapter 6 improves upon Chapter 5 by considering the feedback controller for the ROM during the model optimization, which closes the gap between the open-loop and closed-loop performance. We formulate a model-based reinforcement learning (RL) problem to learn the ROM. Compared to the baseline LIP model, the optimal ROM shows a 49% improvement in viable task region size for inclined walking and up to 21% improvement in joint torque cost. Compared to the approach in Chapter 5, the optimal ROM shows an up to 28% improvements in viable task region size with a mild improvement in the torque cost. Lastly, we demonstrate the flexibility of our model-based RL approach in task space parameterization in the post-training phase, in comparison to a model-free RL approach.

CHAPTER 2

BACKGROUND

In this chapter, we introduce necessary backgrounds to understand the technical chapters of this thesis (Chapters 4 to 6). Section 2.1 introduces the bipedal robot Cassie along with its models used through out the thesis. Section 2.2 provides an introduction to ROMs for legged robots and a common ROM, Linear Inverted Pendulum (LIP), which we use as a baseline model for our methods in Chapters 5 and 6. Section 2.3 outlines the model predictive control that we use for all controllers in this thesis. Sections 2.4 and 2.5 present our mathematical definition of a reduced-order model. Lastly, Sections 2.6 to 2.8 introduce trajectory optimization and bilevel optimization needed for Chapter 5.

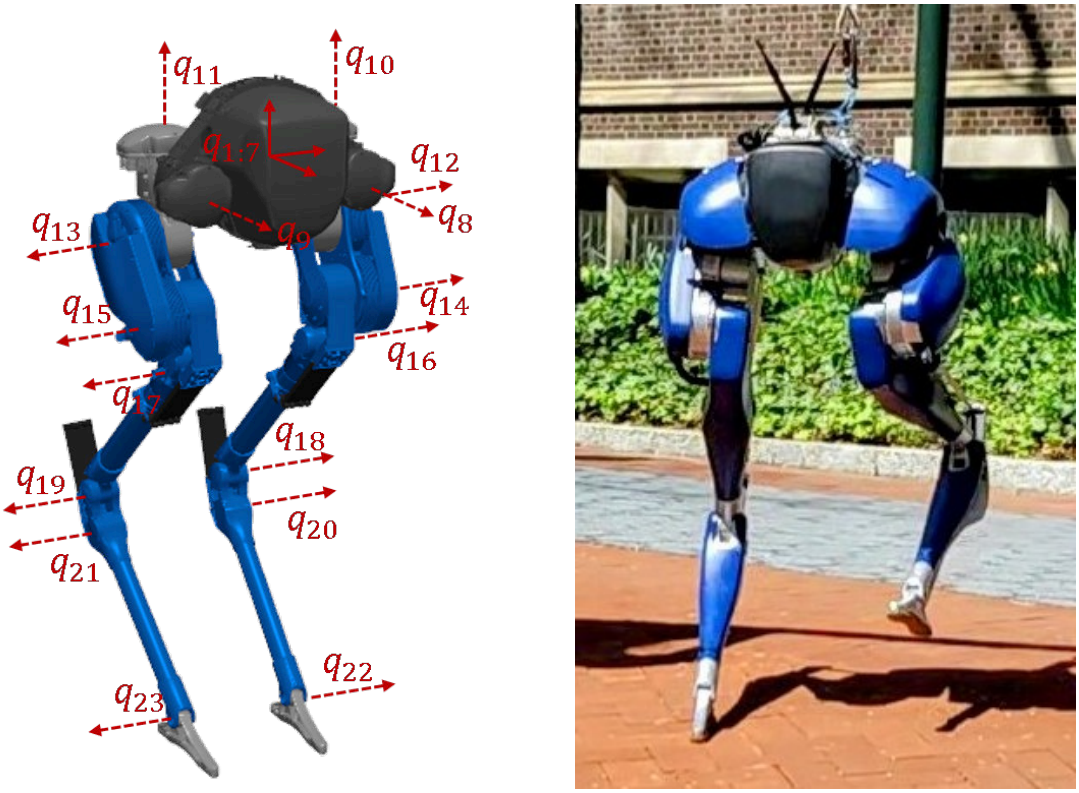
2.1. Full-order Models of Cassie

The bipedal robot Cassie (Fig. 2.1) is the platform we used to test our model optimization algorithm. Here we briefly introduce its model. Let the state of Cassie be $x = [q, v] \in \mathbb{R}^{45}$ where $q \in \mathbb{R}^{23}$ and $v \in \mathbb{R}^{22}$ are generalized position and velocity, respectively. We note that q and v have different dimensions, because the floating base orientation is expressed via quaternion. The conversion between $\dot{q}$ and v depends only on q (Wie and Barba, 1985).

The standard equations of motion are

$$M(q)\dot{v} = f_{cg}(q, v) + Bu + J_h(q)^T\lambda + \tau_{app}(q, v) \tag{2.1}$$

where M is the mass matrix which includes the reflected inertia of motors, f_{cg} contains the velocity product terms and the gravitational term, B is the actuation selection matrix, u is the actuator input, J_h is the Jacobian of holonomic constraints associated with the constraint forces λ, and τ_{app} includes the other generalized forces applied on the system such as joint damping forces. The forces λ contain ground contact forces and constraint forces internal to the four-bar linkages of Cassie. In simulation, the ground forces are calculated by solving an optimization problem based



(a) Full model of Cassie.

(b) Cassie robot.

Figure 2.1: Cassie and its model

on the simulator's contact model (Tedrake, 2019). In trajectory optimization, the forces λ are solved simultaneously with x and u while satisfying the dynamics, holonomic and friction cone constraints. Furthermore, we assume the swing foot collision with the ground during walking is perfectly inelastic in the trajectory optimization, so the robot dynamics is hybrid. Combining the discrete impact dynamics (from foot collision) with Eq. (2.1), we derive the hybrid equations of motion

$$\begin{cases} \dot{x} = f(x, u, \lambda), & x^- \notin S \\ x^+ = \Delta(x^-, \Lambda), & x^- \in S \end{cases} \tag{2.2}$$

where x^- and x^+ are pre- and post-impact state, Λ is the impulse of swing foot collision, f is the continuous-time dynamics, Δ is the discrete mapping at the touchdown event, and S is the surface in the state space where the event must occur (Hurmuzlu and Marghitu, 1994; Grizzle et al., 2014).

Cassie's legs contain four four-bar linkages – two around the shin links and the other two around the tarsus links. We simplify the model by lumping the mass of the rods of the tarsus four-bar linkages into the toe bodies, while the shin linkages are modeled with fixed-distance constraints. To simplify the model further, we assume Cassie's springs are infinitely stiff (or equivalently no springs), in which case $q \in \mathbb{R}^{19}$ and $v \in \mathbb{R}^{18}$. This assumption has been successfully deployed by other researchers (Hereid et al., 2018), and it is necessary for the coarse integration steps in the trajectory optimization[1] in Section 2.6. We also use this assumption in Section 5.3 when comparing the ROM performances between the trajectory optimization and simulation.

2.2. Reduced-order Models (ROMs) of Legged Robots

Modern legged robots like the Agility Robotics Cassie have many degrees of freedom and may incorporate passive dynamic elements such as springs and dampers. To manage this complexity and simplify the design of planning and control, reduced-order models have been widely adopted in the research community.

One observation, common to many approaches, lies in the relationship between foot placement, ground reaction forces, and the center of mass (CoM) (Full and Koditschek, 1999). While focusing on the CoM neglects the individual robot limbs, controlling the CoM position has proven to be an excellent proxy for the stability of a walking robot. CoM-based simple models include the LIP (Kajita and Tani, 1991; Kajita et al., 2001), SLIP (Blickhan, 1989), hopping models (Raibert et al., 1984), inverted pendulums (Garcia et al., 1998; Schwab and Wisse, 2001), and others. Since these models are universally low-dimensional, they have enabled a variety of control synthesis and analysis techniques that would not otherwise be computationally tractable. For example, numerical methods have been successful at finding robust gaits and control designs (Byl and Tedrake, 2009; Oguz Saglam and Byl, 2014; Kelly and Ruina, 2015; Koolen et al., 2012a), and assessing stability (Pratt et al., 2006).

Many of the aforementioned reduced-order models feature massless legs, eliminating any foot-ground

[1]Reher et al. (Reher et al., 2019) showed 7 times increase in solve time when using the full Cassie model (with spring dynamics) in trajectory optimization. Additionally, Cassie's spring properties can change over time and are hard to identify accurately, which discourages researchers from using the dynamic model of the springs on Cassie.

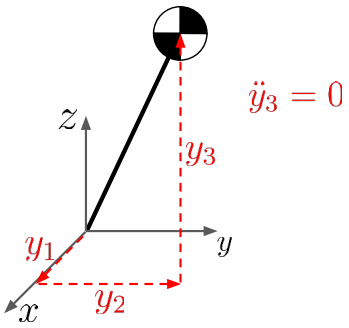


Figure 2.2: The linear inverted pendulum (LIP) model. It is a point mass model of which height is restricted in a plane. The point mass and the origin of this model correspond to the center of mass and the stance foot of the robot, respectively. In the examples of Chapters 5 and 6, we initialize the reduced-order model to the LIP model during model optimization.

impact during the swing foot touchdown event. When dealing with a robot or a model incorporating a foot of non-negligible mass, zero impacts necessitate zero swing foot velocity at touchdown. This constraint ensures the velocity continuity before and after the touchdown event.

2.2.1. Linear Inverted Pendulum (LIP)

In this thesis, we frequently use the LIP model as a benchmark to compare our optimal models against, so we briefly introduce it here.

The model is a point mass model (representing the center of mass of the robot) with a assumption that the vertical acceleration of this point mass is 0. The outcome of this model is that the point mass is restricted in a plane. The equations of motion of the 3D LIP model are

$$\ddot{y} = \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} c_g \cdot y_1 / y_3 \\ c_g \cdot y_2 / y_3 \\ 0 \end{bmatrix}, \tag{2.3}$$

where c_g is the gravitational acceleration constant. We note that, due to the point mass and point foot assumption (where the ground force has to come from), the angular momentum about the center of mass has to be zero when we embed LIP into the robot.

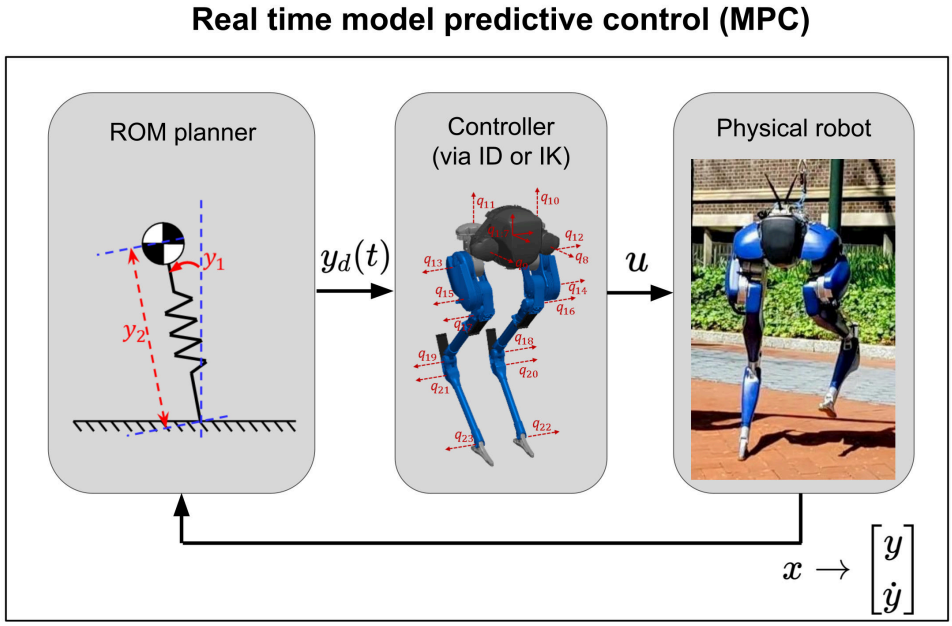


Figure 2.3: An outline of a model predictive control. The planner outputs desired trajectories of the reduced-order model. These trajectories are then tracked by a controller based on inverse dynamics (ID) or inverse kinematics (IK) which uses the full model of the robot.

2.3. Model Predictive Control (MPC) with ROMs

In thesis, we use model predictive control (MPC) in all experiments to control the robot. Here we briefly outline the structure of MPC, while the detailed implementation can be found in individual chapter (Sections 5.2 and 4.3.1).

MPC is a control technique that uses a robot model to plan for desired input trajectories in receding horizons. Its replanning (feedback) mechanism inherently rejects disturbances to the system. For legged locomotion, the horizon is typically around 0.7 to 1 seconds, so it is necessary to use a simplified robot model in order to plan in real time. Commonly, given an initial and a goal position of the robot, the MPC plans for both the footstep locations and the ROM trajectories, and these trajectories are then tracked using a low-level controller. Fig. 2.3 outlines the MPC.

It consists of two main components – a high-level planner that generates the ROM trajectories and footstep locations, and a low-level controller that tracks these desired targets. The low-level

controller are commonly implemented with Inverse Kinematics (IK) or Inverse Dynamics (ID).

2.4. Definition of ROMs

Let q and u be the generalized position and input of the full-order model, and let y and τ be the generalized position and input of the reduced-order model. We define a reduced-order model μ of dimension n_y by two functions – an embedding function $r : q \mapsto y$ and the second-order dynamics of the reduced-order model $g(y, \dot{y}, \tau)$. That is,

$$\mu \triangleq (r, g), \tag{2.4}$$

with

$$y = r(q), \tag{2.5a}$$

$$\ddot{y} = g(y, \dot{y}, \tau), \tag{2.5b}$$

where $\dim y < \dim q$ and $\dim \tau \leq \dim u$. As an example, to represent SLIP, r is the center of mass position relative to the foot, g is the spring-mass dynamics, and $\dim \tau = 0$ as SLIP is passive. Additionally, we note that the choices of r and g are independent of each other. For example, LIP and SLIP share the same r, but they have different dynamics function g (one has zero vertical acceleration and the other is the spring-mass dynamics).

The embedding function r can explicitly include the left or right leg of the robot (e.g. choosing left leg as support leg instead of right leg), in which case there will be two reduced-order models. In this thesis, we assume to parameterize over left-right symmetric reduced-order models. As such, we explicitly optimize over a model corresponding to left-support, which will then be mirrored to cover both left and right-support phases. The details of this mirroring operation can be found in the following section (Section 2.5).

Fig. 2.4 shows the relationship between the full-order and the reduced-order models. If we integrate the two models forward in time with their own dynamics, the resultant trajectories will still satisfy

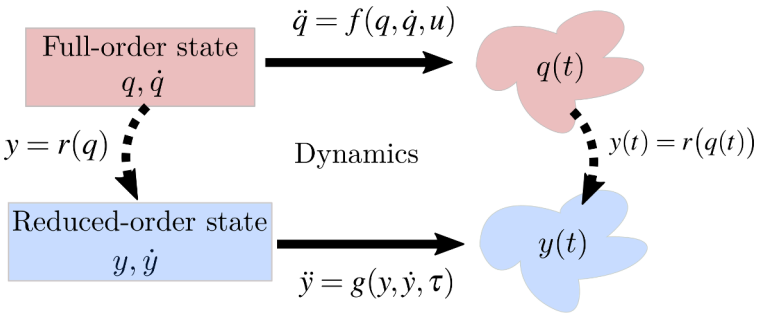


Figure 2.4: Relationship of the full-order and reduced-order models. The generalized positions q and y satisfy the embedding function r for all time, and the evolution of the velocities $\dot{q}$ and $\dot{y}$ respects the dynamics f and g, respectively.

the embedding function r at any time in the future.

2.5. Definition of Mirrored ROMs

2.5.1. Definition

The model representation in Eq. (2.5) could be dependent on the side of the robot. For example, when using the LIP model as the ROM, we might choose the generalized position of the model y to be the CoM position relative to the left foot (instead of the right foot). In this case, we need to find the reduced-order model for the right support phase of the robot. Fortunately, we can derive this ROM by mirroring the robot configuration q about its sagittal plane (Fig. 2.5) and reusing the ROM of the left support phase. We refer to this new ROM as the *mirrored reduced-order model*.

Let q_m and v_m be the generalized position and velocity of the "mirrored robot", shown in Fig. 2.5. Mathematically, the mirrored ROM μ_m is

$$\mu_m \triangleq (r_m, g), \tag{2.6}$$

with

$$y_m = r_m(q) = r(q_m) = r(M(q)), \tag{2.7a}$$

$$\ddot{y}_m = g(y_m, \dot{y}_m, \tau), \tag{2.7b}$$

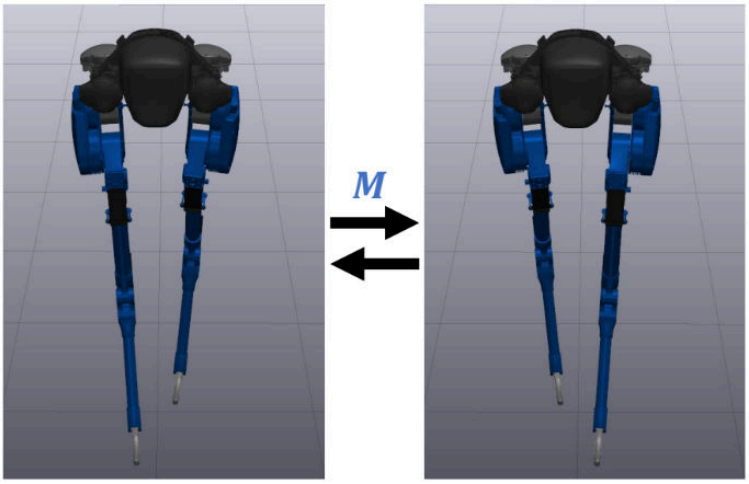


Figure 2.5: The mirror function M mirrors the robot configuration about the sagittal plane. This function is necessary in planning and control when the embedding function (Eq. (2.5a)) of the reduced-order model only represents one side of the robot. For example, the embedding function of the LIP model was chosen to be the CoM relative to the left foot (and not the right foot).

where r_m is the embedding function of the mirror model, and M is the mirror function such that $q_m = M(q)$ and $q = M(q_m)$. We note that the two models, in Eq. (2.4) and (2.6), share the same dynamics function g.

2.5.2. Time derivatives of the embedding function

Feedback control around a desired trajectory often requires the first and the second time derivatives information. Here, we derive these quantities for the mirrored ROM in terms of the original embedding function r in Eq. (2.5a) and its derivatives.

$\dot{y}_m$:

Let J_m be the Jacobian of the mirrored model embedding r_m with respect to the robot configuration q, such that

$$\dot{y}_m = J_m \dot{q}. \tag{2.8}$$

The time derivatives of y_m is

$$\dot{y}_m = \frac{\partial r(M(q))}{\partial M(q)} \frac{\partial M(q)}{\partial q} \dot{q} = J(q_m) \frac{\partial M(q)}{\partial q} \dot{q} = J(q_m) \dot{q}_m \tag{2.9}$$

where $J(q_m)$ is the Jacobian of the original model embedding r evaluated with q_m. From Eq. (2.8) and (2.9), we derive

$$J_m = J(q_m)\frac{\partial M(q)}{\partial q} \tag{2.10}$$

where $\frac{\partial M(q)}{\partial q}$ is a matrix which contains only 0, 1, and -1.

$\ddot{y}_m$:

The i-th element of $\ddot{y}_m$ is

$$\begin{aligned}
\ddot{y}_{m,i} &= \frac{d}{dt}\left(\frac{\partial r_i(q_m)}{\partial q_m}\dot{q}_m\right) = \frac{d}{dt}\sum_j \frac{\partial r_i(q_m)}{\partial q_{m,j}}\dot{q}_{m,j} \\
&= \sum_j \frac{d}{dt}\left(\frac{\partial r_i(q_m)}{\partial q_{m,j}}\right)\dot{q}_{m,j} + \sum_j \frac{\partial r_i(q_m)}{\partial q_{m,j}}\frac{d}{dt}\left(\dot{q}_{m,j}\right) \\
&= \sum_{jk} \frac{\partial^2 r_i(q_m)}{\partial q_{m,j}\partial q_{m,k}}\dot{q}_{m,j}\dot{q}_{m,k} + \frac{\partial r_i(q_m)}{\partial q_m}\frac{d}{dt}\left(\dot{q}_m\right).
\end{aligned}$$

where r_i is the i-th element of the embedding function. The above equation can be expressed in the vector-matrix form

$$\begin{aligned}
\ddot{y}_m &= \dot{q}_m^T \nabla^2 r(q_m)\dot{q}_m + J(q_m)\ddot{q}_m \\
&= \dot{J}(q_m, v_m)\dot{q}_m + J(q_m)\ddot{q}_m \\
&= \dot{J}(q_m, v_m)\dot{q}_m + J_m\dot{v} \quad (\because \text{Eq. (2.10)})
\end{aligned} \tag{2.11}$$

where $\dot{J}(q_m, v_m)$ is the time derivatives of the J (of the original model) evaluated with the mirrored position q_m and velocity v_m.

2.6. Trajectory Optimization

Chapter 5 will heavily leverage trajectory optimization within the inner loop of a bilevel optimization problem. We briefly review it here, but the reader is encouraged to see (Betts, 2001) for a more complete description. Generally speaking, trajectory optimization is a process of finding state $x(t)$ and input $u(t)$ that minimize some measure of cost h while satisfying a set of constraints C. Following the approach taken in prior work (Posa et al., 2013, 2016), we explicitly optimize over state, input, and constraint (contact) forces $\lambda(t)$,

$$\begin{aligned} \min_{x(t),u(t),\lambda(t)} \quad & \int_{t_0}^{t_f} h(x(t),u(t))dt \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t),u(t),\lambda(t)), \\ & C(x(t),u(t),\lambda(t)) \leq 0, \end{aligned} \tag{2.12}$$

where f is the dynamics of the system, λ are the forces required to satisfy holonomic constraints (inside $C \leq 0$), and t_0 and t_f are the initial and the final time respectively. Standard approaches discretize in time, formulating (2.12) as a finite-dimensional nonlinear programming problem. For the purposes of this chapter, any such method would be appropriate, while we use DIRCON (Posa et al., 2016) to address the closed kinematic chains of the Cassie robot. DIRCON transcribes the infinite dimensional problem in Eq. (2.12) into a finite dimensional nonlinear problem

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{n-1} \frac{1}{2}\big(h(x_i,u_i) + h(x_{i+1},u_{i+1})\big)\delta_i \\ \text{s.t.} \quad & f_c(x_i,x_{i+1},u_i,u_{i+1},\lambda_i,\lambda_{i+1},\delta_i,\alpha_i) = 0, \\ & \qquad\qquad\qquad\qquad\qquad\qquad i = 1,...,n-1 \\ & C(x_i,u_i,\lambda_i) \leq 0, \qquad\qquad i = 1,...,n \end{aligned} \tag{2.13}$$

where n is the number of knot points, f_c is the collocation constraint for dynamics, $C \leq 0$ contains all the other constraints such as the four-bar-linkage kinematic constraints, δ_i is a constant time interval between knot point i and $i+1$, and the decision variables are

$$w = [x_1,...,x_n,u_1,...,u_n,\lambda_1,...,\lambda_n,\alpha_1,...,\alpha_{n-1}] \in \mathbb{R}^{n_w},$$

where $\alpha_1,...,\alpha_{n-1}$ are slack variables specific to DIRCON. Eq. (2.13) uses the trapezoidal rule to approximate the integration of the running cost in Eq. (2.12), simplifying the selection of decision variables at knot points for evaluating the function h.

2.7. Heuristics in Trajectory Optimization

Solving the trajectory optimization problem in Eq. (2.13) for a high-dimensional robot is hard, since the problem is nonlinear and of large scale. Even although there are off-the-shelf solvers such

as IPOPT (Wächter and Biegler, 2006) and SNOPT (Gill et al., 2005) designed to solve large-scale nonlinear optimization problem, it is often impossible to get a good optimal solution without any heuristics, since there are many local optima. In this section, we will talk from our experience about the heuristics that might help to solve the problem faster and also find a solution with a lower cost and closer to the global optimum. That said, we have no objective manner in which to assess proximity to global optimality, and thus this is a purely observational criterion.

Let the nonlinear problem be

$$\begin{aligned} \min_{w} \quad & \tilde{h}(w) \\ \text{s.t.} \quad & \tilde{f}(w) \leq 0 \end{aligned} \tag{2.14}$$

where w contains all decision variables, $\tilde{h}$ is the cost function, and $\tilde{f}$ is the constraint function. It turned out that scaling either w, $\tilde{h}$ or $\tilde{f}$ could help to improve the condition of the problem.

- w: Sometimes the decision variables are in different units and can take values of different orders. For example, joint angles of Cassie are roughly less than 1 (rad), while its contact forces are usually larger than 100 (N). In this case, we can scale w by some factor s, such that the decision variables of the new problem are $w_{scaled,i} = s_i w_i$ for $i = 1, 2, ..., n_w$. After the problem is solved, we scale the optimal solution of the new problem back by $w_i^* = \frac{1}{s_i} w_{scaled,i}^*$.

- $\tilde{f}$: The constraints $\tilde{f}$ can take various units just like w. Similarly, we can scale each constraint individually. Note that scaling constraints affects how well the original constraints are satisfied, so one should make sure that the constraint tolerance is still meaningful.

- $\tilde{h}$: In theory, scaling the cost does not affect the optimal solution. However, it does matter in the solver's algorithm. It is desirable to scale the cost so that it is not larger than 1 around the area of interest.

For more detail about scaling, we refer the readers to Chapter 8.4 and Chapter 8.7 of (Gill et al., 1981). In addition to scaling the problem, the following heuristics could also be helpful:

- Provide the solver with a good initial guess.

- Add small randomness to the initial guess: This helps to avoid singularities.

- Add regularization terms to the cost function: This could remove local minima in the cost landscape and can also speed up the solve time. Adding regularization terms is similar to the traditional reward shaping of Reinforcement Learning (Hu et al., 2020) and the policy-regularized MPC (Bledt et al., 2017).

- Add intermediate variables (also called slack variables (Hereid and Ames, 2017)): This can sometimes improve the condition number of the constraint gradients with respect to decision variables. One example of this is reformulating the trajectory optimization problem based on the single shooting method into that based on the multiple shooting method by introducing state variables (Betts, 1998).

- Use solver's internal scaling option: In the case of SNOPT (Philip et al., 2015), we found setting *Scale option* to 2 helps to find an optimal solution of better quality. Note that this option increases the solve time and demands a good initial guess to the problem.

2.8. Bilevel Optimization

Our formulation in Chapter 5 can be broadly categorized as bilevel optimization (Bracken and McGill, 1973). We briefly review its basics here. The basic structure of our bilevel optimization problem is written as

$$\min_{\theta} \left[\sum_{i} \min_{w} \Psi_i(w, \theta) \right] \tag{2.15}$$

The goal is to minimize the outer-level objective function $\sum_i \Psi_i(w_i^*(\theta), \theta)$ with respect to θ, where $w_i^*(\theta)$ is obtained by minimizing the inner-level objective $\Psi_i(w, \theta)$ parameterized by θ. Bilevel optimization has recently been used in various applications such as meta-learning (Franceschi et al., 2018), reinforcement learning (Rajeswaran et al., 2020), robotics (Jin et al., 2022; Pfrommer et al., 2021), etc.

Solving a bilevel program is generally NP-hard (Sinha et al., 2017). There are two types of methods to approach bilevel optimization. The first type is constraint-based (Hatz et al., 2012; Shi et al.,

2005), where the key idea is to replace the inner level optimization with its optimality condition (such as the KKT conditions (Kuhn and Tucker, 1951)), and finally solve a "single-level" constrained optimization. However, those methods are difficult to apply to the problem of our work, because our inner-level is a trajectory optimization, and replacing it with its optimality condition will additionally introduce a large number of dual variables and co-states, dramatically increasing the size of the single-level optimization. The second type is gradient-based (Jin et al., 2020; Domke, 2012). The idea is to maintain and solve the inner-level optimization, and then update the outer-level decision variable by differentiating through the inner-level solution using graph-unrolling approximation (Domke, 2012; Das et al., 2021) or implicit function theorem (Krantz and Parks, 2002). Compared to constraint-based methods, gradient-based methods maintain the bilevel structure and make bilevel optimization more tractable and efficient to solve.

In this work, we use the Envelope Theorem (Afriat, 1971; Takayama and Akira, 1985) and exploit the fact that our problem uses the same objective functions in the outer level and the inner level. This structure enables us to develop a more efficient gradient-based method (the second type) to solve our problem. Specifically, the gradient of the outer-level objective does not require differentiating the solution of the inner-level optimization with respect to the parameters. This leads to two numerical advantages of our method over existing gradient-based methods. First, our method bypasses the computationally intensive implicit theorem, which requires the inverse of Hessian of the inner-level optimization. Second, our method leverages the inner-loop solver's understanding of active and inactive constraints, avoiding implementing the algorithm ourselves and avoiding tuning parameters such as the active set tolerance.

CHAPTER 3

RELATED WORKS

In this chapter, we provide the existing works pertaining to our main contributions of this thesis (in Chapters 4, 5 and 6).

3.1. Whole-body Orientation for Legged Robots

For complex tasks, such as human-like walking and running, back-flips or aggressive turning, we typically need to consider the orientation of the robot beside the center of mass (CoM). In regulating orientation, many researchers have used models based on centroidal angular momentum (Orin and Goswami, 2008; Dai et al., 2014). Centroidal angular momentum models are usually based on velocity level constraints, and tend not to readily produce an (unique) absolute orientation coordinate for the entire system. Other researchers use a single rigid body (SRB) model (Bledt et al., 2018), where a single SO(3) coordinate represents the entire robot's orientation. The choice for this single body may stem from the morphology and mass distribution of a specific robot. For example, for robots with a heavy torso and light limbs, such as the MIT Mini Cheetah (Katz et al., 2019), the torso orientation can act as a good proxy for total system orientation. However, for robots with relatively heavy and/or long limbs, where appreciable mass is distributed throughout the system and far from the CoM (e.g IHMC Nadia (Nad) and Agility Robotics Cassie (Batke et al., 2022)), the coordinate choice for system orientation is much less clear. Proposing a useful whole-body orientation coordinate for these types of systems, that is not attached to any specific link on the robot, is the focus of this chapter.

Unlike the CoM, a weighted averaging of each link's orientation does not produce a consistently meaningful whole-body orientation. To derive this orientation, some have used angular excursion, which is an integral of an angular velocity about the CoM derived from centroidal angular momentum (Zordan et al., 2014). However, there is not a unique angular excursion value for a defined joint configuration, as the integral is path dependent. Another approach uses the principal axes of the whole-body inertia tensor to derive a whole-body orientation (Du et al., 2021). This approach

also relies on the history of the axes in order to produce a meaningful continuous orientation.

Researchers in geometric mechanics have, for about a decade, devised ways of finding history-independent optimal coordinates, called the *minimum perturbation coordinates* (Travers et al., 2013; Hatton and Choset, 2011, 2015). However, to our knowledge, no literature has yet shown the application of these coordinates to a complex dynamic robot (except for Boston Dynamics' patent (Khripin and Rizzi, 2016)), and they remain less prevalent than other approaches.

3.2. Model Order Reduction and ROM Optimization for Legged Robots

Classical approaches to reduced-order modeling often seek to find low-dimensional models which approximate some collected data (e.g. in approximating the solutions to fluid dynamics (Peherstorfer and Willcox, 2015)). For a controlled system, such as a bipedal robot, the data is necessarily a product of the control policy, and thus the data and the ROM-induced controller are invariably intertwined. Nonetheless, recent work in locomotion has attempted this classical approach via model-based RL (Yang et al., 2020), or via robust control (treating the gap between a ROM and the full model as a disturbance (Pandala et al., 2022)).

In contrast, we jointly optimize over the ROM and the induced control policy. Robot actuators are partially utilized to ensure that the system behaves like the reduced-order model, up to the limits of control performance. With this perspective, a good ROM is not one that matches some existing set of robot behaviors. Rather, we find task-optimal ROMs that maximize robot performance when deployed in conjuction with modern model-based planning and control architectures. Specifically, both Chapter 5 and Chapter 6 evaluate this performance via a user-specified objective function.

Several researchers have sought to enhance the performance of the reduced-order model by mixing it with a full model in the planning horizon of MPC. Li et al. (Li et al., 2021) divided the horizon into two segments, utilizing a full model for the immediate part and a reduced-order model for the distant part. Subsequently, Khazoom et al. (Khazoom et al., 2023) systematically determined the optimal scheduling of these two models. Norby et al. (Norby et al., 2022) blended the full and reduced-order models while adaptively switching between the two. Our work is different from

these existing works in that we directly optimize a reduced-order model instead of reasoning about scheduling two models to improve the overall model performance.

3.3. Model-based Reinforcement Learning for Legged Robots

Model-based planning and control have shown significant success for many years in navigating legged robots(Kuindersma et al., 2016; Ackerman, 2012; Marion and the team; Gibson et al., 2022; Xiong and Ames, 2022; Wensing et al., 2022). For example, Boston Dynamics's bipedal robots performed parkour and natural walking (Marion and the team; PET), while SRI's humanoid DURUS achieved high energy efficiency in walking (Reher et al., 2016; Ackerman, 2015). Moreover, having robot models makes it possible to analyze the system stability and provide stability or safety guarantees via techniques such as capturability (Koolen et al., 2012b), sums of squares (Posa et al., 2017b), hybrid zero dynamics (Westervelt et al., 2003) or control barrier functions (Ames et al., 2019; Nguyen et al., 2016; Khazoom et al., 2022). However, as these methods can be computationally costly, reduced-order models (ROMs) are frequently deployed to achieve real time planning and control, albeit at the cost of reduced performance (Wensing et al., 2022; Chen et al., 2023a).

On the other hand, model-free reinforcement learning (RL) has emerged as a powerful tool for automatically synthesizing high-performance control policies (Siekmann et al., 2021; Crowley et al., 2023; Dao et al., 2022; Ma et al., 2023; Miki et al., 2022). Siekmann et al. highlighted the robustness achieved by a neural network (NN) policy in blind stair-walking (Siekmann et al., 2021), while Ma et al. developed a policy utilizing a robot's arm for fall damage mitigation and recovery (Ma et al., 2023). These works demonstrate that neural networks, as universal approximators, enable robots to excel in specific tasks. However, model-free polices lack interpretability, rendering the existing model-based stability and safety techniques unsuitable. Additionally, the model-free methods struggle with generalizing policies to new task parameters without policy retraining, as task space parameterization is determined during the training phase. For example, the policy described in (Ma et al., 2023) effectively recovers a fallen robot but does not consider walking velocity commands. To enable the robot to walk, it is necessary to incorporate these velocity commands as additional inputs to the NN and undergo a fresh policy training process. Moreover, robots may

encounter a wide array of potential tasks, such as footstep timing, adaptive limb adjustment during hardware failure, head movement for item detection, whole-body interactions between arms and legs, or collaborative furniture carrying with humans. Thus, it is impractical to enumerate all possible tasks to avoid future policy retraining.

Our work in Chapter 6 attempts to combine model-based planning and control with reinforcement learning framework to obtain the best of both worlds in the context of bipedal locomotion. It aims to retain the physical intepretabilty and the task flexibility of the model-based approach and utilizes the RL capability in maximizing the robot performance. Specifically, we use model predictive control (MPC) as the model-based control policy and learn a model within the MPC that maximizes the robot's performance via RL.

CHAPTER 4

INTEGRABLE WHOLE-BODY ORIENTATION FOR LEGGED ROBOTS

Parts of this chapter were previously published as parts of Yu-Ming Chen, Gabriel Nelson, Robert Griffin, Michael Posa, and Jerry Pratt. Integrable whole-body orientation coordinates for legged robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ©2023 IEEE. Gabe Nelson formulated the problem to find the whole-body orientation and devised an algorithm to solve the problem. The remaining included portions represent original contributions.

In this chapter, we look at a special case of ROM optimization where we aim to find an embedding function r that minimizes centroidal angular momentum errors. Specifically, we aim to create an *integrable* (history and path invariant) measure of whole-body orientation for multi-link humanoid and legged robots, where this orientation coordinate (e.g. an angle, or Euler angles, or quaternion, etc.) has dynamically analogous behavior to a CoM, but crucially in an angular sense. For example, once formulated, our desire would be that, should there be no external moments acting on the system, this orientation coordinate would remain at rest or rotate at a constant speed. We call this coordinate the integrable whole-body orientation (abbreviated WBO in this thesis), though it has been called the *minimum perturbation coordinates* for general coordinates (other than SO(3) coordinates) (Hatton and Choset, 2011; Travers et al., 2013). As such, we design the WBO to have the same mathematical form as other forward kinematics quantities such as a hand position, or the CoM. Thus, most or all tools and techniques that apply to forward kinematics can be applied to a WBO, such as inverse kinematics, task-space control and planning, etc. The accompanying video for this chapter can be found at https://www.youtube.com/watch?v=p4nRva-AcMU.

This chapter is organized as follows. Section 4.1 uses simple 2D examples to introduce our WBO. Section 4.2 extends this to complex 3D systems and demonstrates the WBO algorithm with Nadia and Cassie. Sections 4.3 and 4.4 apply the derived WBO in walking and running controllers. Section 4.5 summarizes this chapter and describes future work.

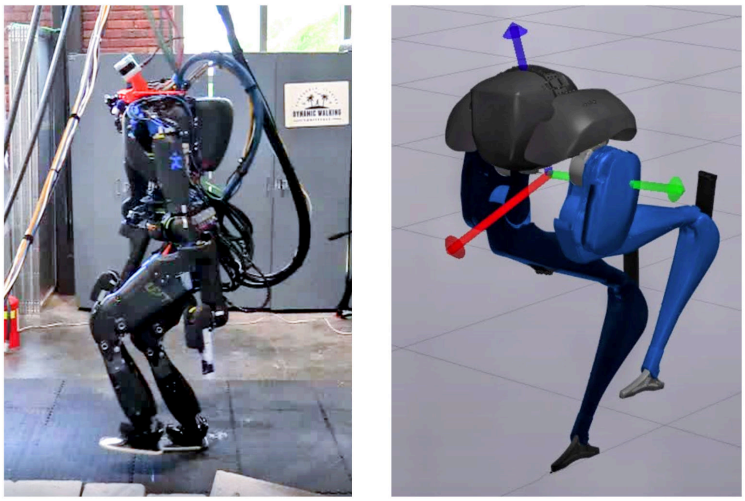


Figure 4.1: Left: The humanoid Nadia walking with arm swing and spine yaw rotation induced by tracking WBO. Right: The simulated biped Cassie running and turning with the whole-body orientation (WBO) coordinate visualized at the CoM.

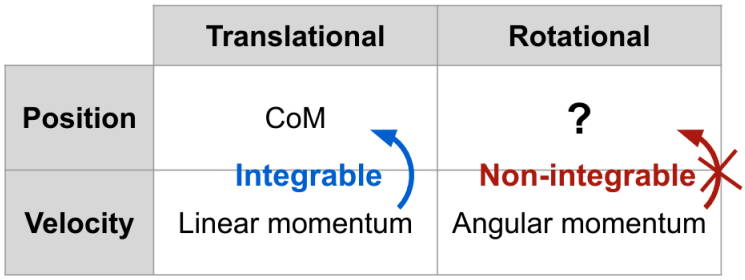


Figure 4.2: While linear momentum is integrable, angular momentum is *generally* not (Nakamura and Mukherjee, 1990; Saccon et al., 2017).

4.1. Whole-body Orientation of Simple Systems

4.1.1. Motivation and Problem Definition

Fig. 4.2 conceptually compares standard centroidal translational and rotational quantities often used in whole-body control. While the total system CoM (a position), and linear and angular momenta are concrete properties of a multibody system, there is in general no unique rotational coordinate (an orientation) corresponding to system CoM. Differentiation of system CoM (scaled by total mass) will arrive at linear momentum. Angular momentum though, for general systems, is *not* integrable (Nakamura and Mukherjee, 1990), highlighting that angular momentum does not represent differential motion along any unique history or path invariant manifold in configuration space.

Thus, as discussed above, the *WBO* is an effort to approximate an angular measurement for the upper-right quadrant in Fig. 4.2. For general systems, it will be an angle, a set of Euler angles, or a quaternion, etc. Its value is a manufactured quantity (but not without physical relevance). It is the result of a design process that can take various forms, and we present one approach in this chapter that we have found useful.

Often this topic involves a larger, typically *formal*, mathematical discussion about differential calculus and geometric mechanics, which we will consider beyond the scope of this work. Deeper treatments can be found in (Travers et al., 2013; Hatton and Choset, 2011, 2015). Our goal is rather to provide a concise WBO formulation recipe, and demonstrate its initial use on a few legged robots.

We propose the following benefits from using a WBO for legged robots: (1) As a dynamically relevant WBO for controller tracking: e.g. providing a feedback signal for a proportional term on WBO control; (2) For planning WBO motions or changes; (3) For encouraging low angular momentum behavior for steady-state walking or running (Popovic et al., 2005; Erez and Todorov, 2012; Miyata et al., 2019). We believe that a suitable WBO measure for an anthropomorphic robot can aid in producing more natural looking movement, since the whole-body orientation control can

be achieved by regulating the WBO directly, rather than controlling the orientation of some specific *base* link (often the pelvis or torso of the robot). This means the *base* link is now free to be treated as just another link on the robot, available for other user-specified objectives: e.g. smoothing system CoM motion by extending leg reach or stride length, stepping while ascending/descending terrain, whole-body reaching motions, etc.

4.1.2. The Bar-and-Flywheel Model

We propose the following simple example as an aid in understanding the WBO that we intend to find. The example has simple and complex versions, which are meant to demonstrate what the WBO does and does not represent.

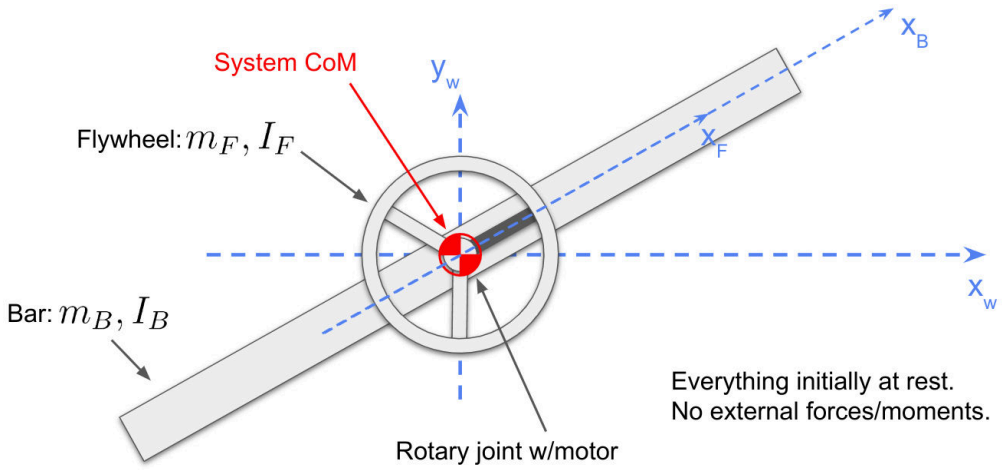
Fig. 4.3 shows a planar "Bar-and-Flywheel" model: a long solid bar attached, at its CoM, to the axis of a flywheel via a rotary joint. The bar and flywheel have mass properties as indicated, and are free floating with no gravity or external forces acting. θ is the bar orientation in an inertially fixed world-frame, and ϕ is the flywheel orientation relative to the bar. A motor actuates the joint between the bar and flywheel. Thus, the bar represents the *base* of a multibody system, and the flywheel is an outboard body connected to the base by an actuated joint. The angular momentum about the CoM, also called the *centroidal angular momentum* (CAM), is

$$H_{CoM} = (I_B + I_F)\dot{\theta} + I_F\dot{\phi}. \tag{4.1}$$

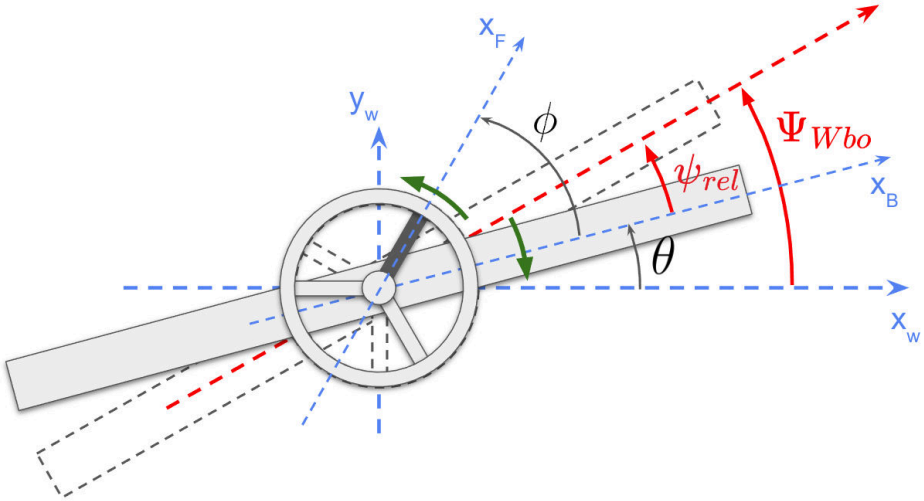
In our definitions, we will delineate *base* orientation (θ in this example; in general a SO(3) representation) from *joint* positions or *joint* configuration (ϕ in this example; in general represented by the vector q).

Let the system start at rest with $\phi = 0$ (Fig. 4.3a). If the motor drives the flywheel counterclockwise, the reaction torque will rotate the bar in the opposite direction. These rotation directions are indicated by the green arrows. Fig. 4.3b shows both the starting (dashed lines) and ending configurations. We would like to understand how *the bar* moves due to the motion of the flywheel.

An important relationship often used in these problems is the *reconstruction equation*, which includes



(a) The starting configuration ($\phi = 0$)



(b) The ending configuration ($\phi > 0$)

Figure 4.3: Bar-and-Flywheel model (an integrable system). The motor rotates the flywheel counter-clockwise.

a *local connection* (Hatton and Choset, 2011). The reconstruction equation describes what we have simulated going from Fig. 4.3a to Fig. 4.3b: It maps velocities in joint space ($\dot{\phi}$) to the velocity of the base ($\dot{\theta}$), as the base will counter-rotate due to changes in joint space. In this example ($H_{CoM} = 0$), this reconstruction equation is

$$\dot{\theta} = -\frac{I_F}{I_B + I_F}\dot{\phi}, \tag{4.2}$$

where the coefficient in front of $\dot{\phi}$ (without the negative sign) is the local connection. We can see

that Eq. (4.2) can be integrated directly and also lets us predict the change in θ. Let $\Delta\theta$ and $\Delta\phi$ be the changes in the angles. We define

$$\psi_{rel} \triangleq -\Delta\theta = \frac{I_F}{I_B + I_F}\Delta\phi, \tag{4.3}$$

and we note that $\Delta\phi = \phi$ since the starting position of ϕ is 0. In Fig. 4.3b, we label these various angles. The *initial* orientation of the bar, relative to the world, will be called Ψ_{Wbo}. In a general configuration, we can predict the final orientation of the system using Eq. (4.3):

$$\Psi_{Wbo} = \theta + \psi_{rel} = \theta + \frac{I_F}{I_B + I_F}\phi. \tag{4.4}$$

This Ψ_{Wbo} is the WBO of the system. Note that ψ_{rel} is the *relative* orientation of the WBO to the base (bar), such that the final Ψ_{Wbo} will be a base orientation relative to the world plus ψ_{rel}.

For this simple system, conservation of angular momentum dictates that Ψ_{Wbo} will never change, regardless of how we actuate the motor. Differentiating Eq. (4.4), we have (after some rearrangement)

$$(I_B + I_F)\dot{\Psi}_{Wbo} = (I_B + I_F)\dot{\theta} + I_F\dot{\phi}. \tag{4.5}$$

Note that the right hand sides of Eq. (4.1) and Eq. (4.5) are the same. Thus:

$$H_{CoM} = (I_B + I_F)\dot{\Psi}_{Wbo}. \tag{4.6}$$

Eq. (4.6) shows that the WBO has behavior analogous to a center of mass, *but in a rotational sense*: it represents an underlying orientation state for the system that can only be changed by external moments. The effective WBO mass moment of inertia (MoI) is, not surprisingly, the sum of the MoI's of the bar and flywheel.

The H_{CoM} above is an integrable differential form, meaning it can be derived from the differentiation of a manifold in configuration space, independent of the joints' history. This manifold can be expressed exactly as Eq. (4.4).

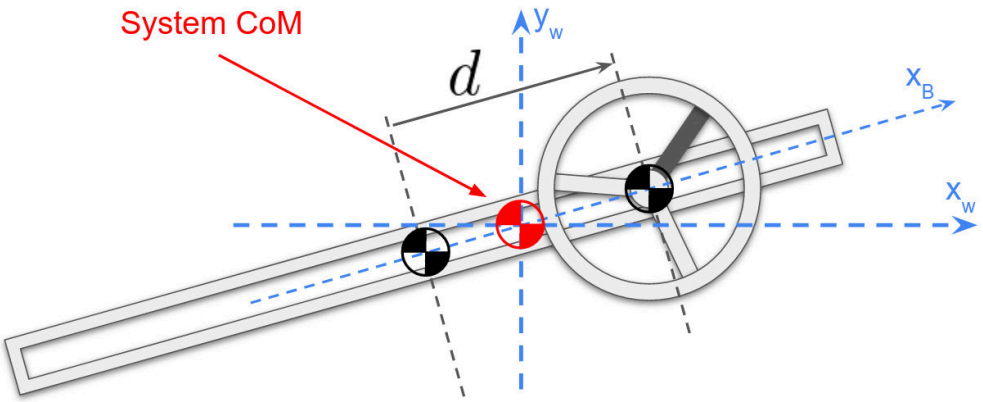


Figure 4.4: Bar-and-Flywheel model with an actuated prismatic joint (a non-integrable system). This system can reorient itself without external torques. For example, the entire system can rotate counter-clockwise if the joints (d, ϕ) repeat the following cyclic motion: $(0,0) \rightarrow (l/2,0) \rightarrow (l/2,\pi/2) \rightarrow (0,\pi/2) \rightarrow (0,0)$, where l is the length of the bar.

Also, looking at the left and right-hand sides of Eq. (4.6), we could think of a system having two representations of angular momentum: the *actual* system angular momentum about the CoM in Eq. (4.1), and an *approximated* angular momentum based on $\dot{\Psi}_{Wbo}$ in Eq. (4.6). For this integrable example system, we see that these angular momentum representations are equivalent, though for general systems (usually non-integrable) they are not.

We now extend this model by adding a long slot that allows the flywheel to slide along the length of the bar (Fig. 4.4). A linear actuator controls this movement, and d is the offset of the flywheel axis from the CoM of the bar. This change increases the size of our joint space (q) to 2 dimensions: $q = [d, \phi]^T$. The new CAM is

$$H_{CoM} = (I_B + I_F + \frac{m_B m_F}{m_B + m_F} d^2)\dot{\theta} + I_F \dot{\phi}. \tag{4.7}$$

If d is fixed, then this system is much the same as the simpler Bar-and-Flywheel model above. If, though, d is allowed to change, the angular momentum becomes non-integrable, and the goal of our WBO formulation becomes *contriving a differentiable and time independent function for ψ_{rel} that, when differentiated with respect to q, maximally approximates the local connection of the actual system over a user-defined region of joint-space.* We will call this contrived function $\tilde{\psi}_{rel}(q)$. Like

Eq. (4.2), the reconstruction equation is found by setting $H_{CoM} = 0$ in Eq. (4.7) and solving for $\dot{\theta}$:

$$\dot{\theta} = -\begin{bmatrix} 0 & \frac{I_F}{I_B + I_F + \frac{m_B m_F}{m_B + m_F} d^2} \end{bmatrix}_{1\times 2} \begin{bmatrix} \dot{d} \\ \dot{\phi} \end{bmatrix}_{2\times 1}, \tag{4.8}$$

where the 1x2 matrix is the local connection, which is a function of d. In mathematical terms, the goal (stated above) is finding a differentiable function $\tilde{\psi}_{rel}(q)$, such that

$$\begin{bmatrix} \frac{\partial \tilde{\psi}_{rel}(q)}{\partial d} & \frac{\partial \tilde{\psi}_{rel}(q)}{\partial \phi} \end{bmatrix} \approx \begin{bmatrix} 0 & \frac{I_F}{I_B + I_F + \frac{m_B m_F}{m_B + m_F} d^2} \end{bmatrix}. \tag{4.9}$$

Coming up with this function is the core design process, bearing in mind that an exact fit is impossible owing to the non-integrability of the physical system. For instance, an example function for $\tilde{\psi}_{rel}(q)$ could be $c_1\, d^2\phi + c_2\, d^2\phi^3 + c_3\, \phi + c_4\, \phi^3$, where the coefficients c_i are found numerically in order to maximize the approximation implied by Eq. (4.9) over a user-defined region in joint-space. Our final WBO now becomes (like Eq. (4.4)):

$$\tilde{\Psi}_{Wbo} \triangleq \theta + \tilde{\psi}_{rel}(q). \tag{4.10}$$

Finally we note that, mirroring Eq. (4.6), we now have *actual* and *approximated* representations for angular momentum (approximation resulting from Eq. (4.9)):

$$H_{CoM} \approx \tilde{H}_{CoM} \tag{4.11}$$

with

$$\tilde{H}_{CoM} = (I_B + I_F + \frac{m_B m_F}{m_B + m_F} d^2)\dot{\tilde{\Psi}}_{Wbo}, \tag{4.12}$$

where

$$\dot{\tilde{\Psi}}_{Wbo} = \dot{\theta} + \begin{bmatrix} \frac{\partial \tilde{\psi}_{rel}(q)}{\partial d} & \frac{\partial \tilde{\psi}_{rel}(q)}{\partial \phi} \end{bmatrix} \begin{bmatrix} \dot{d} \\ \dot{\phi} \end{bmatrix}. \tag{4.13}$$

The main compromise in our approach is approximating a non-integrable differential system with

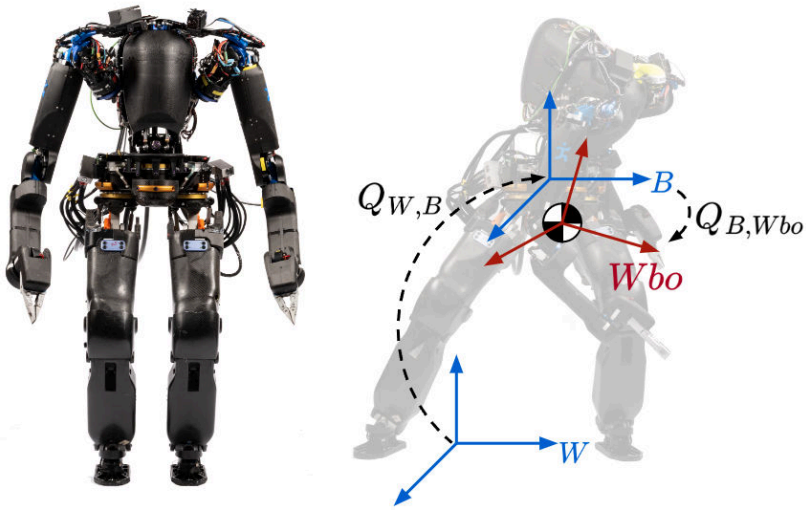


Figure 4.5: The humanoid Nadia (left) and frames of interest (right). W, B and Wbo are the world frame, base frame and the WBO frame, respectively. $Q_{W,B}$ is the base orientation in the world, and $Q_{B,Wbo}$ is the orientation of the WBO frame relative to the base. Translationally, we locate the WBO frame at the CoM for convenience.

	Bar-Flywheel	in 3D
joint configuration	$[d, \phi]$	q
base orient. r.t. world	θ	$Q_{W,B}$
WBO r.t. base	$\tilde{\psi}_{rel}$	$Q \triangleq Q_{B,Wbo}$
WBO r.t. world	$\tilde{\Psi}_{Wbo}$	$Q_{W,Wbo}$

r.t. = relative to Q = Quaternion

Table 4.1: Notation definitions and correspondences

an integrable one. This "collapses" explicit representation of the nonholonomic motion of the actual physical system (Nakamura and Mukherjee, 1990; Hatton and Choset, 2011, 2015). Our WBO will still measure nonholonomic motions, but its resulting dynamics will not correspond with fidelity to the actual externally applied moments.

4.2. Whole-body Orientation of Complex Robots

4.2.1. Extending WBO to 3D

The goal of this section is to express Eq. (4.9) in the general 3D case, while the concepts discussed using the Bar-and-Flywheel models remain the same. Table 4.1 will be important in translating

this structure into 3D. Fig. 4.5 shows the relevant frames, corresponding conceptually to Fig. 4.3b.

We begin by translating Eq. (4.11) into 3D. The 3D angular momentum is

$$H_{CoM} = M_B\omega_B + M_q\dot{q} = M_B[\omega_B + A\dot{q}], \tag{4.14}$$

where M_B and M_q are base and joint space centroidal momentum matrices (Orin and Goswami, 2008), $A \triangleq M_B^{-1}M_q$ is the local connection and a function of q, ω_B is the angular velocity of the base relative to the world expressed in the base frame, and $\dot{q}$ are the joint velocities. For the WBO, we have an approximated angular momentum

$$\tilde{H}_{CoM} = M_B[\omega_B + \tilde{A}\dot{q}], \tag{4.15}$$

where $\tilde{A}$ will be the approximated local connection as discussed in arriving at Eq. (4.9). In 3D, trying to minimize the differences between these two representations (Eqs. (4.14) and (4.15)) means making:

$$A\dot{q} \approx \tilde{A}\dot{q}. \tag{4.16}$$

We will keep $\dot{q}$ on both sides of the approximation until we sort out the 3D rotation representation for WBO in Section 4.2.2. We note that $A\dot{q}$ on the left hand side of Eq. (4.16) is the *relative angular velocity* of the system (Miyata et al., 2019). On the other hand, $\Omega_{Wbo} \triangleq \tilde{A}\dot{q}$ is the angular velocity of the WBO frame relative to the base, expressed with respect to the base frame.

Paralleling the 2D example above (see Table 4.1), we would like to find a function for $Q \triangleq Q_{B,Wbo}$. This represents the WBO frame orientation relative to the base. Thus, Ω_{Wbo} can be expressed from the quaternion rate $\dot{Q}$ using

$$\Omega_{Wbo} = 2R_Q E_Q \dot{Q} \tag{4.17}$$

where R_Q is the rotation matrix representation of Q, and the matrix $2 \cdot E_Q$ maps a quaternion rate to an angular velocity (details are omitted here for brevity; see (Wie and Barba, 1985)).

Algorithm 1 WBO optimization

Input: N random joint configurations $q_i,\ i = 1, ..., N$
Output: Θ^*
1: $\Theta \leftarrow 0$ (initialize to constant identity rotation)
2: **repeat**
3: Substitute $Q(q_i; \Theta)$ into T_Q in Eq. (4.20) for $i = 1, ..., N$
4: $\Theta \leftarrow$ Solve Eq. (4.20) with given T_Q
5: **until** convergence
6: **return** Θ

4.2.2. Parameterization and Optimization Algorithm

Noting that Q has two portions $Q = [Q_s;\ Q_{x,y,z}]$, we now parameterize $Q_{x,y,z}$ by a vector of basis functions $\lambda(q)$ with dimension n_λ:

$$Q_{x,y,z}(q; \Theta) \ = \ \Theta\lambda(q), \tag{4.18}$$

where $\Theta \in \mathbb{R}^{3\times n_\lambda}$ is a coefficient matrix. Q_s can be recovered from the unit norm constraint $\|Q\|_2^2 = 1$. Similarly, we take the time derivatives of $Q_{x,y,z}$ and recover $\dot{Q}_s$ from $\frac{d}{dt}\|Q\|_2^2 = 0$. These algebraic manipulations will lead to a final form:

$$\tilde{A}\dot{q} = T_Q \Theta J_\lambda \dot{q} \tag{4.19}$$

where T_Q and J_λ are functions of q and are respectively defined as

$$T_Q \triangleq 2R_Q E_Q \begin{bmatrix} -Q_s^{-1} Q_{x,y,z}^T \\ I_{3\times 3} \end{bmatrix} \in \mathbb{R}^{3\times 3}, \text{ and}$$
$$J_\lambda \triangleq \frac{\partial \lambda(q)}{\partial q} \in \mathbb{R}^{n_\lambda \times n_q}.$$

Given Eq. (4.16), our objective is to minimize the difference between A and $\tilde{A}$, both of which are functions of q. Since minimizing the difference over an infinite number of q in a region of joint space is often intractable, we pre-select N number of random configurations (uniformly distributed in the

robot's operating joint space) to simplify the problem:

$$\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \|A_i - T_{Q_i} \Theta J_{\lambda i}\|_F^2 \tag{4.20}$$

where $\| \cdot \|_F$ is the Frobenius norm. Given our choice in Eq. (4.18), we note that A and J_λ are independent of Θ, while T_Q is nonlinear in Θ.

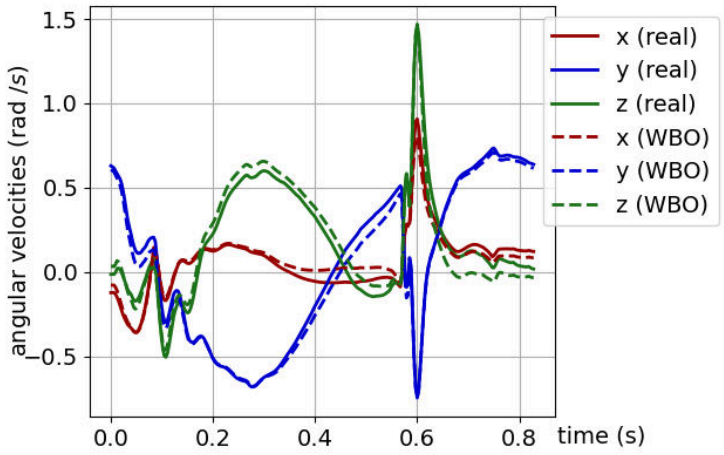
One can solve Eq. (4.20) with many nonlinear solvers. In practice, we found that our simple algorithm in Alg. 1 works. The algorithm exploits the structure of the cost function by identifying that Eq. (4.20) is a least squares problem if T_Q is given. In each iteration, we first substitute the current solution Θ into T_Q to turn (4.20) into a least squares problem[2], and then solve the problem to get a new solution Θ. We repeat the above steps until Θ converges. This algorithm is similar to the Gauss-Newton method, differing in that it avoids linearizing the objective function at the solution in each iteration.

4.2.3. WBO Optimization and Result

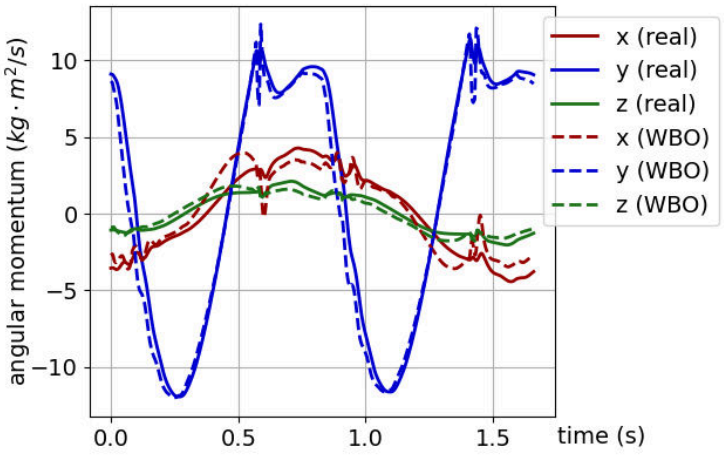
We optimized for an WBO function for Nadia (Fig. 4.5) using Alg. 1. Nadia is a humanoid robot with 31 degrees of freedom (DoF) – 6DoF legs, 7DoF arms, 1DoF grippers and a 3DoF spine. We randomly select 1000 configuration pairs mirrored about the sagittal plane (so $N = 2000$). We also keep the gripper, wrist and ankle joints at neutral positions, because their contribution to the CAM is relatively small. This reduces the configuration space (q) to 19 dimensions. The basis functions are monomials in terms q, with all possible monomials up to 3rd order being used (e.g. q_i, q_i^2, $q_i q_j$, q_i^3, $q_i^2 q_j$, ...), producing 1539 basis functions. The optimization converged smoothly in about 2 minutes or 10 iterations. After the optimization, we dropped terms with coefficients (in Θ) smaller than 1e-8.

In addition to Nadia, we also optimized a WBO function for Cassie running (Fig. 4.1). Cassie has 16 joints. We ignored the toe and ankle spring joints during optimization, reducing the configuration space to 12 dimensions. The optimization converged within 10 seconds and 7 iterations.

[2]The Kronecker product identity $vec(XYZ) = Z^T \otimes X\, vec(Y)$ is useful in vectorizing the matrix Θ in preparation for solving the least squares.



(a) Angular velocities of one step of Nadia robot walking on flat ground. The solid and dashed lines are $A\dot{q}$ and $\tilde{A}\dot{q}$ in Eq. (4.16), respectively. The spikes around 0.6 seconds are from the swing foot impact event and the feedback reaction of the walking controller.



(b) The solid lines are the real CAM H_{CoM} in Eq. (4.14), and the dashed lines are the approximated CAM by the WBO $\tilde{H}_{CoM}$ in Eq. (4.15).

Figure 4.6: Comparisons between real and approximated quantities. The data is from a simulation where Nadia walked in a straight line at 0.6 m/s.

By comparing actual (measured) and approximated quantities, we can evaluate our WBO approximation at different signal scales. For Nadia walking, Fig. 4.6a plots both sides of Eq. (4.16).

Average angular velocity errors for each axis are about $[0.034, 0.035, 0.061]$ rad/s. Fig. 4.6b plots Eqs. (4.14) and (4.15). We see that H_{CoM} and $\tilde{H}_{CoM}$, which are larger signals dominated by base motion, are relatively close: average errors for each axis are about [0.74, 0.84, 0.32] $\mathrm{kg} \cdot \mathrm{m}^2/\mathrm{s}$. Thus, WBO reflects the actual H_{CoM} in a meaningful way.

4.3. Walking Example

In this section, we design a walking controller for Nadia using the WBO derived in Section 4.2.3, and show that a WBO reference tracking can induce natural upper body motions during walking (Fig. 4.1).

4.3.1. Controller

Fig. 4.7b shows our WBO controller, while Fig. 4.7a shows the baseline controller which fixes the desired joint positions for the upper body. Each controller has a planner that generates desired trajectories. These are then converted into acceleration commands by the feedback controllers shown in the diagrams. With the acceleration commands, we use an inverse dynamics whole body controller (the rightmost block in each diagram in Fig. 4.7) to get the desired actuator commands for the robot (Koolen et al., 2016a). We can roughly separate the controller into leg and upper body parts. The leg part handles tracking the desired path and heading of the robot, while the rest of the controller handles the upper body motion.

Legs

This part of the controller is the same between the baseline and WBO controllers. We use Capture Point (CP) control for the locomotion task (Koolen et al., 2012b; Seyde et al., 2018). The footsteps are generated given desired velocity commands from a higher level controller. The planner outputs a reference Centroidal Moment Pivot trajectory that is converted into a linear momentum rate command in the feedback controller, which is sent to the inverse dynamics controller.

Upper body

Our short-term goal was getting more natural arm swing and spine yaw rotation by servoing just WBO yaw[3]. In Fig. 4.7b, we servo the WBO yaw axis relative to the world frame, while both the pelvis and the upper body tasks reside in the null space of the WBO task. In simulation, we achieved straight-line walking with this controller. When moving to hardware, we temporarily focused on demonstrating arm swing and spine yaw rotation. To do so, we servoed the pelvis orientation relative to the world and regulated the WBO yaw angle relative to the pelvis to 0. More complex motions have been left to future work, where we would like to take full advantage of our WBO.

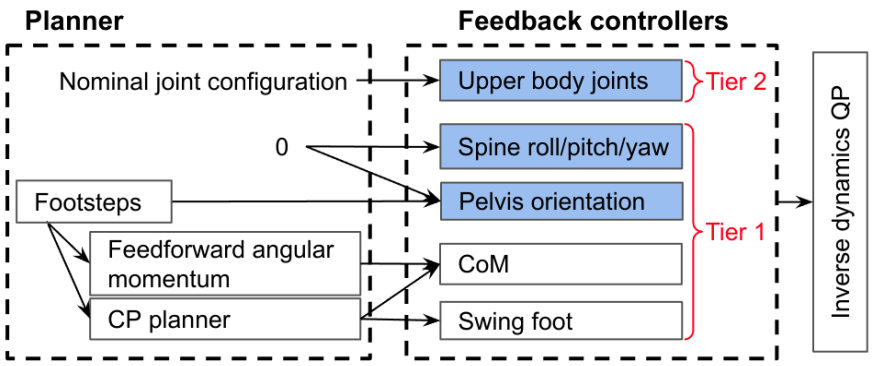
We use a task hierarchy (Hutter et al., 2013) in our whole body controller, shown in Fig. 4.7 as "Tiers". In experiments, we noticed that the inverse dynamics QP solver would trade swing foot orientation tracking performance for WBO tracking performance. This happened when the robot could not regulate WBO yaw to 0 with only the upper body. Thus, in order to prevent the WBO task from impairing the leg tasks, we set the WBO task to a lower priority than the leg tasks.

Besides the above task objectives, we also add nominal joint configuration tracking to handle the system's redundancy. This task can exist in the null space of the WBO task or at the same level as WBO. The parameters for this upper body joint controller can be used to sculpt the desired motion. For example, increasing the cost weight on the spine joint achieves more arm swing and less spine rotation.

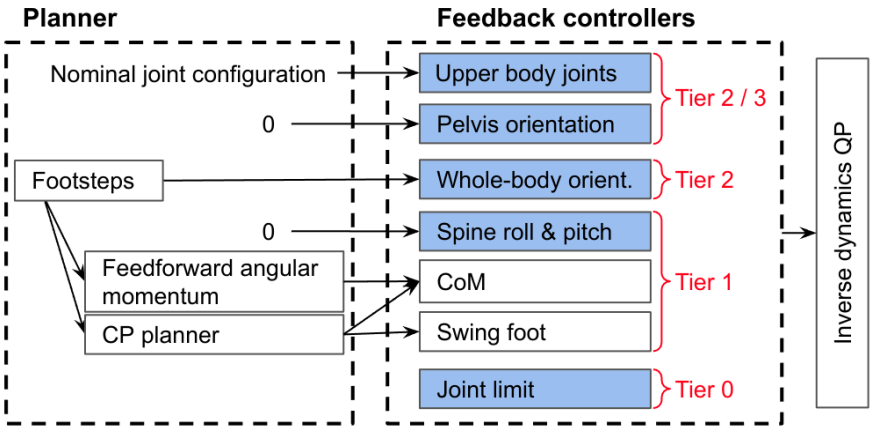
Joint limits

The joint limit controller takes current joint positions and ranges of motion, and outputs limits on joint accelerations for the whole body controller. These limits are used for self-collision avoidance and aesthetics. Because the legs on Nadia are much heavier than the arms, when regulating the WBO yaw to 0, the robot can generate excessive arm swing or spine rotation. Thus, self-collision avoidance helps contain these motions, and therefore affects WBO tracking and overall appearance.

[3]In our experiments, we found that arm swing and spine yaw rotation were mostly induced by servoing the WBO yaw angle to zero. Additionally, Miyata et al. (Miyata et al., 2019) only used the yaw part of angular momentum to generate the arm swing.



(a) Controller with constant desired joint positions of the upper body.



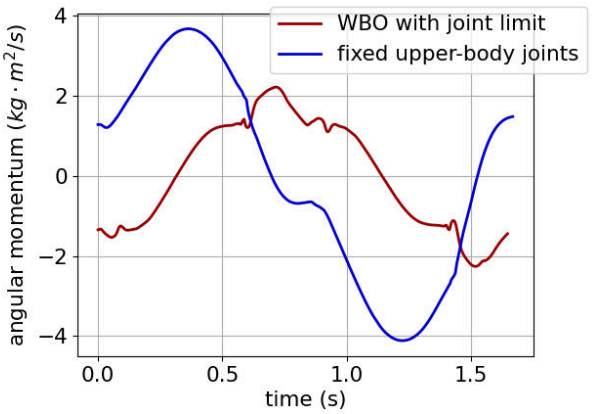
(b) Controller with WBO-induced upper body motions.

Figure 4.7: Controller diagrams. Each diagram is composed of a high-level planner, low-level feedback controllers and an inverse-dynamics quadratic program (QP). Blue color highlights the difference between the two controllers. Red color is used for indicating the task priorities. Tier 0 is implemented as a constraint in the QP, while other Tiers are implemented via cost functions in the QP. Additionally, Tier n has higher priority than Tier $n+1$ for $n > 0$. We use the null-space projection technique to prioritize tasks (Hutter et al., 2013). The zeros and nominal joint configuration in the planner are constant trajectory sources.

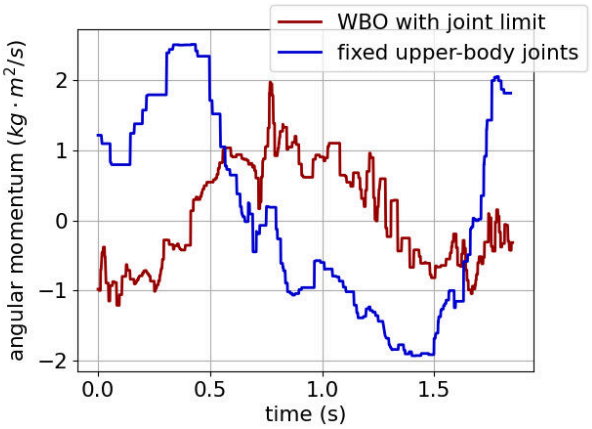
We note that conventional momentum approaches (Erez and Todorov, 2012; Miyata et al., 2019) would also exhibit this same behavior on Nadia.

4.3.2. Experiment Result

In both simulation and hardware experiments, we saw natural upper body motion induced by tracking a constant WBO. Additionally, although the controller for the upper body motion was designed for straight-line walking, we found that, in simulation, the robot was also able to walk forward, backward, sideways, and turn. The video clips can be found in the introduction of this



(a) Simulation (average walking speed ≈ 0.6 m/s)



(b) Hardware (average walking speed ≈ 0.37 m/s)

Figure 4.8: The CAM about the z axis when Nadia walked in a straight line. We note that there were issues with Nadia's leg actuator at the time of hardware experiment and the update rate of the control loop was not fast. These issues partially caused the non-smoothness in the hardware plot.

chapter.

Fig. 4.8 shows the z-component of CAM of straight-line walking for both the WBO and the baseline controller. We see that the angular momentum profiles look similar between the simulation and hardware, and that the CAM is 50% smaller when the desired WBO yaw is set to 0. Joint limit constraints prevent the CAM from tracking closer to zero. Additionally, we observed in simulation that the WBO controller reduces foot yaw moment against the ground. These are some of the advantages of using the upper body to counter moments generated by the legs during walking

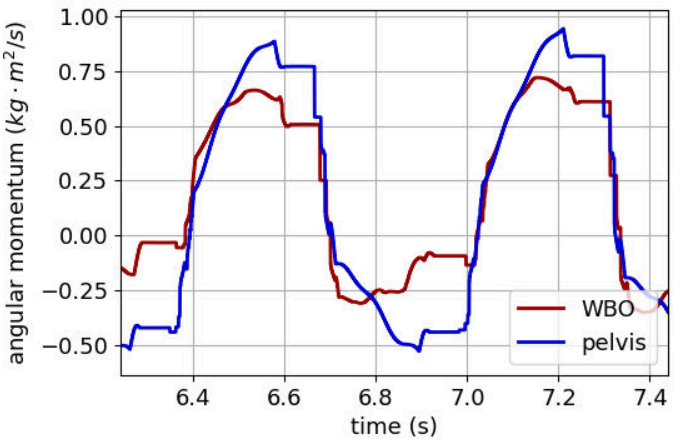


Figure 4.9: The CAM about the z axis when Cassie runs at 2.7 m/s and follows a desired yaw trajectory that goes from 0 to $\pi/2$ rad in 10 seconds.

(Miyata et al., 2019).

The conventional approach to generating natural upper body motions is directly minimizing the CAM (Erez and Todorov, 2012; Miyata et al., 2019). The downside of this approach is that the CAM controller is a feedback controller based on mutually constrained velocities rather than positions. Thus the upper body configuration could gradually drift away from a neutral target unless care is taken. To address this, a competing control objective is typically introduced that servos the robot, or some specific link on the robot, back to a desired orientation relative to the world. In contrast, a control law based on the WBO provides a single desired orientation for the entire robot, and thus need not employ competing objectives.

4.4. Running Example

Besides the walking example, we also want to test our WBO on a slightly more agile motion. For this, we implemented two running controllers on Cassie.

4.4.1. Controller

The baseline running controller uses a finite state machine with four states – left stance, left flight, right stance and right flight. The state transitions are triggered by foot touch-down and lift-off events. In the left/right stance state, the stance leg behaves like a vertical virtual spring, the pelvis pitch and roll angles are regulated to 0, and pelvis yaw follows a desired trajectory. The swing leg

uses a Raibert-style control law (Raibert, 1986), while the leg length is determined by the nominal leg length at touchdown. In the flight state, the controller continues to track the desired orientation of the pelvis and the desired positions of the leading swing foot. The desired pelvis orientation is relative to the world frame.

The second (preliminary) running controller is the same as the baseline controller, except that we replace the pelvis yaw with WBO yaw.

4.4.2. Experiment Result

In our experiments, Cassie is commanded to run at 2.7 m/s in the Drake simulator (Tedrake, 2019). We also set a desired yaw trajectory, which goes from 0 to $\pi/2$ rad linearly in 10 seconds. The baseline controller tracks this desired yaw with the pelvis, while the WBO controller tracks it with WBO. Fig. 4.9 shows the CAM of Cassie. We can see that the momentum oscillates less with the WBO controller (more than a 26% reduction). This reduction is due to the WBO representing the orientation of the entire system, and the total momentum is approximated by its time derivative via Eq. (4.15). Also, the WBO controller is able to adjust the desired pelvis orientation when the legs move. In contrast, the baseline controller considers the pelvis motion only and ignores the contributions from the legs. One could, of course, regulate the CAM while tracking the desired orientation of the pelvis, but these two objectives could conflict since the pelvis alone does not represent the entire system well. The advantage of the WBO approach here is the consistency between the orientation-tracking and momentum objectives.

4.5. Conclusion and Future Work

We introduced the integrable whole-body orientation (WBO) with simple examples and clear problem motivation, so it is more accessible to a general robotic audience. A formulation of the WBO problem was provided, including an algorithm that solves the problem quickly by exploiting its structure. WBO functions were synthesized offline for the Nadia and Cassie robots, and were then used to induce arm swing and spine yaw rotation in a walking example and to turn the robot's global orientation in a running example.

The WBO enables us to servo the orientation of the entire system. Thus, it can free up the base link (e.g. pelvis) to achieve high-level goals such as natural walking with natural pelvis motion and stepping up/down terrain. In this work, we mostly demonstrated more natural arm swing and spine rotation. Future work will utilize the WBO to achieve more complex behaviors, such as whole-body natural walking. Another area of future research involves incorporating high-level planning for the WBO trajectories (e.g. using the SRB model in planning), which could potentially enable more agile motions for the robots. Lastly, this chapter does not explore the impact of WBO on system stability, leaving it for further investigation. However, a prior study (Posa et al., 2017b) found larger regions of attraction for balance and step recovery by moving from a point-mass to SRB model. We believe substituting WBO for the SRB model could also improve stability for the above systems.

CHAPTER 5

OPTIMAL REDUCED-ORDER MODELING OF LEGGED LOCOMOTION

Parts of this chapter were previously published as parts of Yu-Ming Chen and Michael Posa. Optimal reduced-order modeling of bipedal locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8753–8760. ©2020 IEEE, and also as parts of Yu-Ming Chen, Jianshu Hu, and Michael Posa. Beyond inverted pendulums: Task-optimal simple models of legged locomotion. *arXiv preprint arXiv:2301.02075*, 2023.

This chapter generalizes Chapter 4 to optimize a model (both embedding function r and the reduced-order dynamics g) with respect to any user-specified objective function. Specifically, we propose a model optimization algorithm that automatically synthesizes reduced-order models, optimal with respect to a user-specified distribution of tasks and corresponding cost functions. All videos and code of this chapter can be found at https://sites.google.com/view/ymchen/research/optimal-rom.

This chapter is organized as follows. Section 5.1 formulates the model optimization problem, provides an algorithm that solves the problem, and finally demonstrates model optimization with a few examples. Section 5.2 introduces an MPC for a specific class of ROMs. Section 5.3 uses this MPC to embed a ROM into the robot, and compares and analyzes the performance improvement in trajectory optimization, in simulation and in hardware. Section 5.4 discusses the hybrid nature of legged robot dynamics and introduces the MPC for a general ROM. Finally, we discuss some of the lessons learned during the journey of realizing better performance on the robot in Section 5.5, and conclude the paper in Section 5.6.

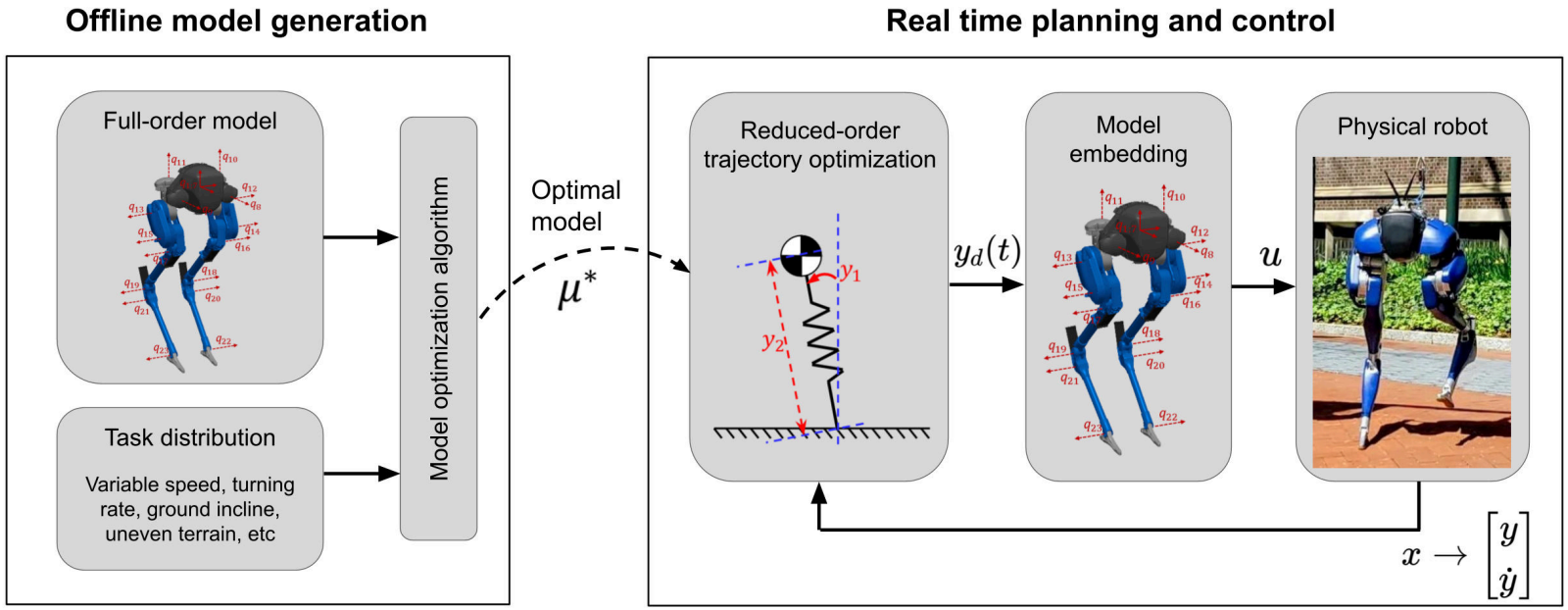


Figure 5.1: An outline of the synthesis and deployment of optimal reduced-order models (ROM). Offline, given a full-order model and a distribution of tasks, we optimize a new model that is effective over the task space (Section 5.1). Online, we generate new plans for the reduced-order model and track these trajectories on the true, full-order system (Section 5.2). This diagram also shows the bipedal robot Cassie (in the rightmost box) and its full model. Cassie has five motors on each leg – three located at the hip, one at the knee and one at the toe. Additionally, there are 2 leaf springs in each leg, and the spring joints are visualized by q_{16} to q_{19} in the figure. The springs are a part of the closed-loop linkages of the legs. We model these linkages with distance constraints, so there are no rods visualized in the model.

5.1. ROM Optimization

In this section, we introduce a notion of quality (or cost) for reduced-order models. We then introduce a bilevel optimization algorithm to optimize within our class of models.

5.1.1. Problem Statement

As shown in the left half of Fig. 5.1, the goal is to find an optimal model μ^*, given a distribution Γ over a set of tasks. The distribution could be provided *a priori* or estimated via the output of a higher-level motion planner. The tasks might include anything physically achievable by the robot, such as walking up a ramp at different speeds, turning at various rates, jumping, running with a specified amount of energy, etc. The goal, then, is to find a reduced-order model that enables low-cost motion over the space of tasks,

$$\mu^* = \underset{\mu \in \boldsymbol{M}}{\operatorname{argmin}} \, \mathbb{E}_\gamma \left[\mathcal{J}_\gamma(\mu) \right], \tag{5.1}$$

where $\boldsymbol{M}$ is the model space, $\mathbb{E}_\gamma$ takes the expected value over Γ, and $\mathcal{J}_\gamma(\mu)$ is the cost required to achieve the tasks $\gamma \sim \Gamma$ while the robot is restricted to a particular model μ.

With our model definition in Eq. (2.4), the problem in Eq. (5.1) is infinite dimensional over the space of embedding and dynamics functions, r and g. To simplify, we parametrize r and g with basis functions $\{\phi_{e,i} \mid i = 1, \ldots, n_e\}$ and $\{\phi_{d,i} \mid i = 1, \ldots, n_d\}$ with linear weights $\theta_e \in \mathbb{R}^{n_y \cdot n_e}$ and $\theta_d \in \mathbb{R}^{n_y \cdot n_d}$. Further assuming that the dynamics are affine in τ with constant multiplier, r and g are given as

$$y = r(q; \theta_e) \qquad = \Theta_e \phi_e(q), \tag{5.2a}$$

$$\ddot{y} = g(y, \dot{y}, \tau; \theta_d) = \Theta_d \phi_d(y, \dot{y}) + B_y \tau, \tag{5.2b}$$

where $\Theta_e \in \mathbb{R}^{n_y \times n_e}$ and $\Theta_d \in \mathbb{R}^{n_y \times n_d}$ are θ_e and θ_d arranged as matrices, $\phi_e = [\phi_{e,1}, \ldots, \phi_{e,n_e}]$, $\phi_d = [\phi_{d,1}, \ldots, \phi_{d,n_d}]$, and $B_y \in \mathbb{R}^{n_y \times n_\tau}$. For simplicity, we choose a constant value for B_y. Observing that physics-based rigid-body models lead to state-dependent values for B_y, one can also extend

this method by parameterizing $B_y(y, \dot{y})$. Moreover, while we choose linear parameterization here, any differentiable function approximator (e.g. a neural network) can be equivalently used.

Let the model parameters be $\theta = [\theta_e, \theta_d] \in \mathbb{R}^{n_t}$. Eq. (5.1) can be rewritten as

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \; \mathbb{E}_\gamma \left[\mathcal{J}_\gamma(\theta) \right]. \tag{O}$$

From now on, we work explicitly in θ, rather than μ. As we will see in the next section, $\mathcal{J}_\gamma(\theta)$ is an optimal cost of a trajectory optimization problem, making Eq. (O) a bilevel optimization problem. Additionally, given the parameterization in Eq. (5.2), the ROM dimension n_y is fixed during the model optimization.

5.1.2. Task Evaluation

We use trajectory optimization to evaluate the task cost $\mathcal{J}_\gamma(\theta)$. Under this setting, the tasks γ are defined by a cost function h_γ and task-specific constraints C_γ. $\mathcal{J}_\gamma(\theta)$ is the optimal cost to achieve the tasks while simultaneously respecting the embedding and dynamics given by θ. We note that the cost function h_γ is a function of the full model, although we occasionally refer to the cost evaluated by this function as the ROM performance because the ROM is embedded in the full model.

The resulting optimization problem is similar to (2.13), but contains additional constraints and decision variables for the reduced-order model embedding,

$$\begin{aligned} \mathcal{J}_\gamma(\theta) \triangleq \; \min_{w} \quad & \sum_{i=1}^{n-1} \frac{1}{2}\big(h_\gamma(x_i, u_i) + h_\gamma(x_{i+1}, u_{i+1})\big)\delta_i \\ \text{s.t.} \quad & f_c(x_i, x_{i+1}, u_i, u_{i+1}, \lambda_i, \lambda_{i+1}, \delta_i, \alpha_i) = 0, \\ & \qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, n-1 \\ & g_c\left(x_i, u_i, \lambda_i, \tau_i; \theta\right) = 0, \quad i = 1, \ldots, n \\ & C_\gamma(x_i, u_i, \lambda_i) \leq 0, \qquad\quad i = 1, \ldots, n \end{aligned} \tag{5.3}$$

where f_c and g_c are dynamics constraints for the full-order and reduced-order dynamics, respectively. The decision variables are $w = [x_1, ..., x_n,\ u_1, ..., u_n,\ \lambda_1, ..., \lambda_n,\ \tau_1, ..., \tau_n,\ \alpha_1, ..., \alpha_{n-1}]$, noting the

addition of τ_i.

The formulation of dynamics and holonomic constraints of the full-order model are described in (Posa et al., 2016), while the reduced-order constraint g_c is

$$\begin{aligned} g_c &= \ddot{y}_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0 \\ \Rightarrow g_c &= J_i \dot{v}_i + \dot{J}_i v_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0 \end{aligned} \tag{5.4}$$

where

$$y_i = r(q_i; \theta_e), \quad \dot{y}_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i} \dot{q}_i, \quad J_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i}, \quad \text{and}$$

$$\dot{v}_i = M(q_i)^{-1} \left(f_{cg}(q_i, v_i) + B u_i + J_h(q_i)^T \lambda_i + \tau_{app}(q_i, v_i) \right).$$

The constraint $g_c = 0$ not only explicitly describes the dynamics of the reduced-order model but also implicitly imposes the embedding constraint r via the variables y and $\dot{y}$. Therefore, the problem (5.3) is equivalent to simultaneous optimization of full-order and reduced-order trajectories that must also be consistent with the embedding r.

For readability, we rewrite Eq. (5.3) as

$$\begin{aligned} \mathcal{J}_\gamma(\theta) = \min_{w} \quad & \tilde{h}_\gamma(w) \\ \text{s.t.} \quad & \tilde{f}_\gamma(w, \theta) \leq 0, \end{aligned} \tag{TO}$$

where $\tilde{h}_\gamma$ is the cost function of Eq. (5.3) and $\tilde{f}_\gamma \leq 0$ encapsulates all the constraints in Eq. (5.3). In Section 5.3, we will use $\tilde{h}_\gamma$ to evaluate both the open-loop and closed-loop performance.

Remark 1. *Model optimization can change the physical meaning of a ROM. Regardless, if $J_i M^{-1} B$ (which maps a full model input u to a reduced-order acceleration $\ddot{y}_i$) has full row rank, the ROM can be exactly-embedded into the full model.*

5.1.3. Bilevel Optimization Algorithm

Since there might be a large or infinite number of tasks $\gamma \sim \Gamma$ in Eq. (O), solving for the exact solution is often intractable. Therefore, we use stochastic gradient descent to solve Eq. (O) (specifically

in the outer optimization, as opposed to the inner trajectory optimization). That is, we sample a set of tasks from the distribution Γ and optimize the averaged sample cost over the model parameters θ.

The full approach to (O) is outlined in Algorithm 2. Starting from an initial parameter seed θ_0, N tasks are sampled, and the cost for each task $\mathcal{J}_{\gamma_j}(\theta)$ is evaluated by solving the corresponding trajectory optimization problem (TO).

To compute the gradient $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta)\right]$, we previously (Chen and Posa, 2020) adopted an approach based in sequential quadratic programming. It introduced extra parameters (e.g. tolerance for determining active constraints) and required solving a potentially large and ill-conditioned system of linear equations which can take minutes to solve to good accuracy. Here, we take a new approach where we apply the Envelope Theorem and directly derive the analytical gradient $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta)\right]$ shown in Corollary 1 (also see Section 2.8).

Proposition 1 (Differentiability Condition (Jin et al., 2021)). *Assume $\tilde{h}$ and $\tilde{f}$ are continuously differentiable functions, and consider an optimization problem*

$$\begin{aligned} \tilde{\mathcal{J}}(\theta) = \min_{w} \quad & \tilde{h}(w, \theta) \\ s.t. \quad & \tilde{f}(w, \theta) \leq 0, \end{aligned} \tag{5.5}$$

where $\tilde{\mathcal{J}}(\theta)$ is the optimal cost of the problem. Let $w^*(\theta)$ be the optimal solution to Eq. (5.5). w^* is differentiable with respect to θ if the following conditions hold:

1. *the second-order optimality condition for Eq. (5.5),*
2. *linear independence constraint qualification (LICQ), and*
3. *strict complementarity at w^*.*

Theorem 1 (Envelope Theorem (Riley, 2012)). *Assume the problem in Eq. (5.5) satisfies the*

differentiability condition. The gradient of the optimal cost $\tilde{\mathcal{J}}(\theta)$ with respect to θ is

$$\nabla_\theta \left[\tilde{\mathcal{J}}(\theta)\right] = \frac{\partial \tilde{h}(w^*, \theta)}{\partial \theta} + \lambda^{*T} \frac{\partial \tilde{f}(w^*, \theta)}{\partial \theta}, \tag{5.6}$$

where λ^* is the dual solution to Eq. (5.5)*.*

Proof. We provide our own proof for the Envelope theorem in this thesis, since others' (Takayama and Akira, 1985) are quite complex, and also for self-contained purpose.

The primal-dual solution (w^*, λ^*) to (5.5) should satisfy the following KKT conditions (Kuhn and Tucker, 1951)

$$\frac{\partial \tilde{h}(w^*, \theta)}{\partial w^*} + \lambda^{*T} \frac{\partial \tilde{f}(w^*, \theta)}{\partial w^*} = 0 \tag{5.7}$$

$$\tilde{f}(w^*, \theta) \leq 0, \tag{5.8}$$

$$\lambda^* \geq 0, \tag{5.9}$$

$$\lambda^{*T} \tilde{f}(w^*, \theta) = 0, \tag{5.10}$$

where (5.8), (5.9), and (5.10) are the primal feasibility, dual feasibility, and complementarity conditions, respectively. In the following proof, we occasionally write the primal and dual solution as $w^* = w^*(\theta)$ and $\lambda^* = \lambda^*(\theta)$ for the sake of clear exposition, as they both implicitly depend on θ through the KKT condition. Due to (5.10), one has

$$\mathcal{J}_\gamma(\theta) = \tilde{h}(w^*, \theta) = \tilde{h}(w^*(\theta), \theta) + \lambda^{*T} \tilde{f}(w^*(\theta), \theta). \tag{5.11}$$

and thus,

$$\begin{aligned} \nabla_\theta \left[\mathcal{J}_\gamma(\theta)\right] = & \underbrace{\left(\frac{\partial \tilde{h}(w^*, \theta)}{\partial w^*} + \lambda^{*T} \frac{\partial \tilde{f}(w^*, \theta)}{\partial w^*} \right)}_{=0} \frac{dw^*(\theta)}{d\theta} \\ & + \frac{\partial \tilde{h}(w^*, \theta)}{\partial \theta} + \lambda^{*T} \frac{\partial \tilde{f}(w^*, \theta)}{\partial \theta} + \tilde{f}(w^*, \theta)^T \frac{d\lambda^*(\theta)}{d\theta}. \end{aligned} \tag{5.12}$$

Here, the first term vanishes because of (5.7). Next, we will show that the last term $\tilde{f}(w^*,\theta)^T \frac{d\lambda^*(\theta)}{d\theta}$ will also vanish due to the strict complementarity.

Suppose $\tilde{f}(w^*,\theta) \in \mathbb{R}^l$. Strict complementarity says that the i-th ($i = 1, 2, \ldots, l$) entry of $\tilde{f}(w^*,\theta)$, written as $\tilde{f}(w^*,\theta)[i]$, and i-th entry of λ^*, written as $\lambda^*[i]$, satisfy (5.10), rewritten as

$$\tilde{f}(w^*,\theta)[i] \cdot \tilde{f}(w^*,\theta)[i] = 0, \tag{5.13}$$

but cannot be $\tilde{f}(w^*,\theta)[i] = 0$ and $\lambda^*[i] = 0$ simultaneously, for any $i = 1, 2, \ldots, l$. Differentiating (5.13) with respect to θ on both side, yielding

$$\tilde{f}(w^*,\theta)[i]\frac{d\lambda^*(\theta)[i]}{d\theta} + \lambda^*(\theta)[i]\frac{\partial \tilde{f}(w^*,\theta)[i]}{\partial\theta} = 0. \tag{5.14}$$

Due to the strict complementarity, from (5.14), we can find

$$\frac{d\lambda^*(\theta)[i]}{d\theta} = 0 \quad \text{if} \quad \tilde{f}(w^*,\theta)[i] < 0. \tag{5.15}$$

Back to the last term in (5.12) , one has

$$\tilde{f}(w^*,\theta)^T \frac{d\lambda^*(\theta)}{d\theta} = \sum_{i=1}^{l} \tilde{f}(w^*,\theta)[i] \cdot \frac{d\lambda^*(\theta)[i]}{d\theta} = 0. \tag{5.16}$$

This is because if $\tilde{f}(w^*,\theta)[i] < 0$, $\tilde{f}(w^*,\theta)[i] \cdot \frac{d\lambda^*(\theta)[i]}{d\theta} = 0$ due to (5.15); and if $\tilde{f}(w^*,\theta)[i] = 0$, $\tilde{f}(w^*,\theta)[i] \cdot \frac{d\lambda^*(\theta)[i]}{d\theta} = 0$ trivially holds. Thus, (5.12) becomes

$$\nabla_\theta \left[\mathcal{J}_\gamma(\theta)\right] = \frac{\partial \tilde{h}_\gamma(w^*,\theta)}{\partial\theta} + \lambda^{*T}\frac{\partial \tilde{f}_\gamma(w^*,\theta)}{\partial\theta},$$

which is (5.17). This completes the proof.

□

Algorithm 2 Reduced-order model optimization

Input: Task distribution Γ and step size α
Output: θ^*

Model initialization
$\theta \leftarrow \theta_0$
Model optimization
repeat
  Sample N tasks from $\Gamma \Rightarrow \gamma_j,\ j = 1, ..., N$
  for $j = 1, \ldots, N$ **do**
    Solve (TO) to get $\mathcal{J}_{\gamma_j}(\theta)$
    Compute $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta)\right]$ by Eq. (5.17)
  end for
  Average the gradients $\Delta\theta = \frac{\sum_{j=1}^{N} \nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta)\right]}{N}$
  Gradient descent $\theta \leftarrow \theta - \alpha \cdot \Delta\theta$
until convergence
return θ

Corollary 1. *The gradient of the optimal cost of* (TO) *is*

$$\nabla_\theta \left[\mathcal{J}_\gamma(\theta)\right] = \lambda^{*T} \frac{\partial \tilde{f}_\gamma(w^*, \theta)}{\partial \theta}, \tag{5.17}$$

where w^* and λ^* are respectively the primal and the dual solution to (TO).

Proof. The proof follows directly from Theorem 1. Note that the cost function in (TO) is independent of θ, in which case the first term of Eq. (5.6) becomes 0. □

We note that there is, in general, no guarantee on global convergence when using Eq. (5.6) in a gradient descent algorithm, except for simple cases where $\tilde{h}$ and $\tilde{f}$ are convex functions in (w, θ) (Boyd and Vandenberghe, 2004). As for local convergence towards a stationary point, the gradient descent with Eq. (5.6) is guaranteed to converge with a sufficiently small step size.

While there is no guarantee that the differentiability condition in Proposition 1 holds everywhere (in fact we expect it to fail under certain conditions), in practice we have observed that Algorithm 1 reliably converges. Additionally, the accuracy of the gradient $\nabla_\theta \left[\mathcal{J}_\gamma(\theta)\right]$ is bounded by the accuracy of the primal and dual solutions to (TO) (Jin et al., 2021). In practice, we observed that the gradient

Example #	1	2	3	4	5
Stride length (m)	[-0.4, 0.4]			[0.3, 0.4]	[0.0, 0.42]
Pelvis height (m)	[0.87, 1.03]				0.8
Ground incline (rad)	0				[-0.35, 0.35]
Turning rate (rad/s)	0				[-0.72, 0.72]
Stride duration (s)	0.35				0.35
Parameterize (r, g)?	both (r, g)				only g
Monomial order n_ϕ	2	4	2	2	4
Dominant cost in J_γ	u	u	$\dot{v}$	$\dot{v}$	u
Cost reduction	22.8%	20.7%	27.6%	38.2%	22.4%

Table 5.1: Examples of model optimization. This table includes the task space used to train models (uniform task distribution), the highest order of the monomials of basis functions, the dominant term of the cost function J_γ, and the cost reduction percentage (relative to the cost of the initial model).

was accurate enough (showing local convergence behavior) with the default optimality tolerance and constraint tolerance given by solvers like SNOPT (Gill et al., 2005).

In Algorithm 2, the sampled tasks can sometimes be infeasible for the trajectory optimization problem due to a poor choice in ROM or numerical difficulties when solving (TO). In these cases, we do not include these samples in the gradient update step. This is a reasonable approach as we expect that optimizing the ROM for nearby tasks simultaneously improves performance for the failed task by continuity. This does have the potential to break the optimization process if large regions of the task space were infeasible, but in practice we have found this sample-rejection procedure robust enough to the occasional numerical difficulties.

The model optimization in Algorithm 2 is deemed to have converged if the norm of the average gradient of the sampled costs falls below a specified threshold. This threshold can be set on a case by case basis, depending on the robot models, tasks, etc. In our experiments, we simply look at the cost-iteration plots (e.g. Fig 5.2) and terminate the optimization when the cost has stopped decreasing visibly.

5.1.4. Examples of Model Optimization

In the trajectory optimization problem in Eq. (TO), we assume the robot walks with instantaneous change of support. That is, the robot transitions from right support to left support instantaneously, and vice versa. We consider only half-gait periodic motion, and so include right-left leg alternation in the impact map Δ.

We solve the problems (TO) in parallel in each iteration of Algorithm 2 using the SNOPT toolbox (Gill et al., 2005). All examples were generated using the Drake software toolbox (Tedrake, 2019) and source code is available in the link provided in the Introduction.

Initialization and parameterization of ROM

To demonstrate Algorithm 2, we optimize for 3D reduced-order models on Cassie. The models are initialized with a three-dimensional LIP, of which the generalized position y is shown in Fig. 2.2.

We choose basis functions such that they not only explicitly include the position of the LIP, but also include a diverse set of additional terms. That is, the basis set ϕ_e includes the CoM position relative to the stance foot, and monomials of $\{1, q_7, ..., q_{19}\}$ up to n_ϕ-th order. Similarly, the feature set ϕ_d includes the terms in LIP dynamics (i.e. $c_g y_1/y_3$ and $c_g y_2/y_3$) and monomials of $\{1, y_1, y_2, y_3, \dot{y}_1, \dot{y}_2, \dot{y}_3\}$ up to n_ϕ-th order. With these basis functions, the ROM parameters θ can be trivially initialized to match the LIP model's.

Optimization Examples and Result

We demonstrate a few examples of model optimization and compare their results. The examples are shown in Table 5.1 along with their detailed settings. The optimization results are shown in Fig. 5.2, where the costs are normalized by the optimal cost of (TO) without ROM embedding (i.e. without the constraints g_c).

The cost function h_γ was chosen to be the weighted sum of squares of the robot input u, the generalized velocity v and acceleration $\dot{v}$. In Examples 1, 2 and 5, we heavily penalize the input term which is a proxy of a robot's energy consumption. For the other examples, we heavily penalize

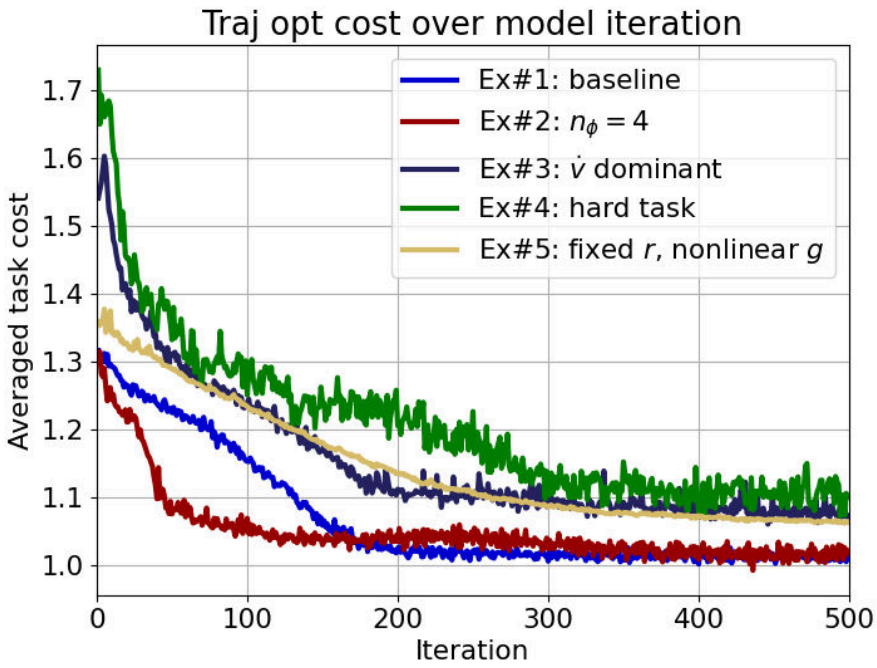


Figure 5.2: The averaged cost of the sampled tasks of each model optimization iteration in Examples 1 to 5. Costs are normalized by the cost associated with the full-order model (i.e. the cost of full model trajectory optimization without any reduced-order model embedding). Therefore, the costs cannot go below 1. The costs at iteration 1 represent the averaged costs for the robots with the embedded initial reduced-order models, LIP. Note that the empirical average does not strictly decrease, as tasks are randomly sampled and are of varying difficulty.

the acceleration $\dot{v}$. We observed that Cassie's motions with the initial ROM are very similar among all examples. In contrast, the motions with optimal ROMs are mostly dependent on the cost function h_γ, given the same ROM parameterization. Compared to Example 1, the optimal motion of Example 3 shows more vertical pelvis movement.

A comparison between Example 1 to Example 2 shows the effect of the order of the monomials n_ϕ in the basis function. We can see in Fig. 5.2 that these two examples share the same starting cost, because the initial weights on the monomials are zeros, making the trajectory optimization problems identical. Additionally we can also see that parameterizing the ROM with second-order monomials seems sufficient for the task space of Examples 1 and 2, since the final normalized cost is close to 1.

A comparison between Example 1 to Example 3 shows the effect of different choices of cost function

h_γ. The initial cost of Example 3 is much higher than Example 1's, which we can interpret as the LIP model being more restrictive under the performance metric of Example 3 than that of 1.

A comparison between Example 3 and Example 4 shows the effect of the task space. Example 4's task space is a subset of Example 3's, specifically the part of the task space with bigger stride length. We would expect the LIP model does not perform well with big stride length, and indeed Fig. 5.2 shows that the initial cost of Example 4 is higher than that of Example 3. Fortunately, a high initial cost provides us with a bigger room of potential improvement. As we see in Table 5.1, Example 3 has higher cost reduction than Example 1, and Example 4 has the highest cost reduction.

In Example 5, the dimension of the task space[4] is increased compared to the other examples, and we only parameterize the ROM dynamics g. That is, the embedding function r remains to be a simple forward kinematic function – the center of mass position relative to the stance foot. In this case, the algorithm was again able to find an optimal model, and the result shows that parameterizing only the ROM dynamics g is sufficient enough for achieving near full model performance (about 5% higher than full-order model's performance).

The optimized models are capable of expressing more input-efficient motions than the LIP model, better leveraging the natural dynamics of Cassie.The reduced-order model optimization improves the performance of the robot, while maintaining the model simplicity. We note that the optimal model, unlike its classical counterpart, does not map easily to a physical model, if the embedding function r contains abstract basis functions such as monomials. While this limits our ability to attach physical meaning to y and τ, it is a sacrifice that one can make to improve performance beyond that of hand-designed approaches.

5.2. MPC for a Special Class of ROMs

After a ROM is optimized, we embed it in the robot via an MPC to achieve desired tasks, depicted in Fig. 5.1. Specifically, in this and the next section (Sections 5.2 and 5.3), we build upon Example 5. Example 5 limits the ROM to a fixed embedding function r, the CoM position relative to the

[4]The dimension here is the non-degenerate dimension, meaning the task dimension with non-zero volume. In Example 5, the dimension is 3 because we vary the stride length, ground incline and turning rate.

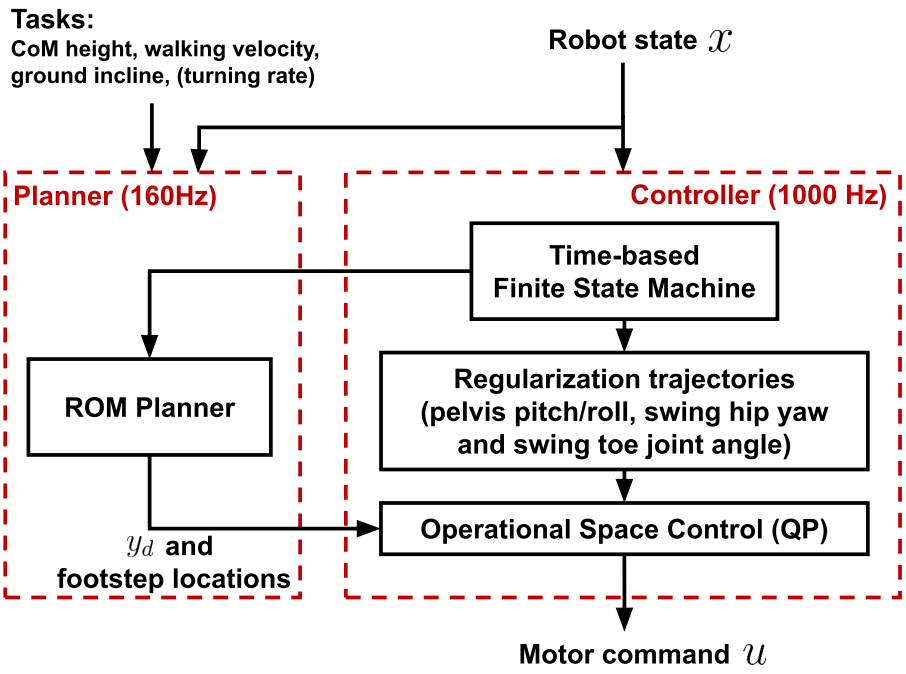


Figure 5.3: The diagram of the MPC introduced in Section 5.2. The MPC is composed of the controller process and the planner process, and it contains a time-based finite state machine which outputs either left or right support state. This finite state machine determines the contact sequence of the high-level planner and the contact mode of the low-level model-based controller. The high-level planner solves for the desired reduced-order model trajectories and swing foot stepping locations, given tasks (commands) and the finite state. For reduced-order models without body orientation (e.g. CoM model without moment of inertia), we send the turning rate command to the controller process instead of planner process. Inside the controller process, the regularization trajectories are used to fill out the joint redundancy of the robot. These regularization trajectories are derived from simple heuristics such as maintaining a horizontal attitude of the pelvis body, having the swing foot parallel to the contact surface, and aligning the hip yaw angle with the desired heading angle. All desired trajectories are sent to the Operational Space Controller (OSC) which is a quadratic-programming based inverse-dynamics controller (Sentis and Khatib, 2005; Wensing and Orin, 2013).

stance foot. This physically-interpretable embedding simplifies the planner and enables a richer performance analysis in Section 5.3. The planner for a general ROM will be introduced in Section 5.4.

The MPC structure is shown in Fig. 5.3. It contains a high-level planner in the reduced-order space (Section 5.2.1) and a low-level tracking controller in the full-order space (Section 5.2.2). The high-level planner receives the robot state and tasks, and plans for the desired ROM trajectory

and the footsteps of the robot. The controller tracks these desired trajectory and footsteps, while internally using nominal trajectories to handle the system redundancy.

Trajectory y_i^{osc}	dim y_i^{osc}	cost weight W			K_p			K_d		
		x	y	z	x	y	z	x	y	z
reduced-order model	3	0.1	0	10	10	0	50	0.2	0	1
pelvis orientation	3	2	4	0.02	200	200	0	10	10	10
swing foot position	3	4	4	4	150	150	200	1	1	1
swing leg hip yaw joint	1	0.5			40			0.5		
swing leg toe joint	1	2			1500			10		

Table 5.2: Trajectories and gains in the Operational Space Control (OSC)

5.2.1. Planning with Reduced-order Models

We formulate a reduced-order trajectory optimization problem to walk n_s strides, using direct collocation method described in Section 2.6 to discretize the trajectory into n knot points. Under the premise that the ROM embedding r is the CoM, we further assume the ROM does not have continuous inputs τ (e.g. center of pressure) but it has discrete inputs $\tau_{fp} \in \mathbb{R}^2$ which is the stepping location of the swing foot relative to the stance foot. Let $z = [y, \dot{y}] \in \mathbb{R}^{2n_y}$, and let z^- and z^+ be the reduced state of pre- and post-touchdown event, respectively. The discrete dynamics is

$$z^+ = z^- + B_{fp}\tau_{fp} \tag{5.18}$$

with

$$B_{fp} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}^T .$$

The first two rows of Eq. (5.18) correspond to the change in stance foot reference for the COM position. The last three rows are derived from the assumption of zero ground impact at the foot touchdown event.

To improve readability, we stack decision variables into bigger vectors $z = [y, \dot{y}] \in \mathbb{R}^{2n_y}$, $Z = [z_0, z_1, ..., z_n] \in \mathbb{R}^{2n_y(n+1)}$, and $\mathrm{T}_{fp} = [\tau_{fp,1}, ..., \tau_{fp,n_s}] \in \mathbb{R}^{2n_s}$. The cost function of the planner is

quadratic and expressed in terms of Z and T_{fp}. The planning problem is

$$\begin{aligned} \min_{Z,\mathrm{T}_{fp}} \quad & \|Z - Z_d\|^2_{W_Z} + \|\mathrm{T}_{fp}\|^2_{W_\mathrm{T}} \\ \text{s.t.} \quad & \text{ROM continuous dynamics (Eq. (2.5b))}, \\ & \text{ROM discrete dynamics (Eq. (5.18))}, \\ & C_{kinematics}(Z, \mathrm{T}_{fp}) \leq 0, \\ & z_0 = \text{current feedback reduced-order state}, \end{aligned} \tag{5.19}$$

where W_Z and W_T are the weights of the norms, Z_d is a stack of desired states which encourage the robot to reach a goal location and regularize velocities, and $C_{kinematics} \leq 0$ is the constraints on step lengths and stepping locations relative to the CoM. After solving Eq. (5.19), we reconstruct the desired ROM trajectory $y_d(t)$ from the optimal solution Z^*, and we construct desired swing foot trajectories from T^*_{fp} with cubic splines.

5.2.2. Operational Space Controller

A controller commonly used in legged robots is the quadratic-programing-based operational space controller (QP-based OSC), which is also referred to as the QP-based whole body controller (Sentis and Khatib, 2005; Wensing and Orin, 2013). Assume there are N_y number of outputs $y_i^{osc}(q)$, with desired outputs $y_{i,d}^{osc}(t)$, where $i = 1, 2, ...N_y$. For each output (neglecting the subscript i), we can derive the commanded acceleration as the sum of the feedforward acceleration of the desired output and a PD control law

$$\ddot{y}^{osc}_{cmd} = \ddot{y}^{osc}_d + K_p(y^{osc}_d - y^{osc}) + K_d(\dot{y}^{osc}_d - \dot{y}^{osc}).$$

At a high level, the OSC solves for robot inputs that minimize the output tracking errors, while respecting the full model dynamics and constraints (essentially an MPC but with zero time horizon).

The optimal control problem of OSC is formulated as

$$\min_{\dot{v},u,\lambda,\epsilon} \quad \sum_{i=1}^{n_y} \|\ddot{y}_i^{osc} - \ddot{y}_{i,cmd}^{osc}\|_{W_i}^2 + \|u\|_{W_u}^2 + \|\epsilon\|_{W_\epsilon}^2 \tag{5.20a}$$

$$\text{s.t.} \quad \ddot{y}_i^{osc} = J_i\dot{v} + \dot{J}_i v, \quad i = 1, ..., N_y \tag{5.20b}$$

$$\text{Dynamics constraint (Eq. (2.1))} \tag{5.20c}$$

$$\epsilon = J_h\dot{v} + \dot{J}_h v \tag{5.20d}$$

$$u_{min} \leq u \leq u_{max} \tag{5.20e}$$

$$\text{Contact force constraints} \tag{5.20f}$$

where $\|\cdot\|_W$ is the weighted 2-norm, (5.20d) contains Cassie's four-bar-linkage constraints, fixed-spring constraints and relaxed contact constraints (relaxed by slack variables ϵ), and (5.20f) includes friction cone constraints, non-negative normal force constraints and force blending constraints for stance leg transition.

Table 5.2 shows all trajectories tracked by the OSC and their corresponding gains and cost weights. The trajectories of the reduced-order model, pelvis orientation and swing foot position are all 3 dimensional, while the hip yaw joint and toe joint of the swing foot are 1 dimensional. The symbols (x,y,z) in Table 5.2 indicate the components of the tracking target. They do not necessarily mean the physical (x, y, z) axes for the reduced-order model, since the model optimization might produce a physically non-interpretable model embedding r.

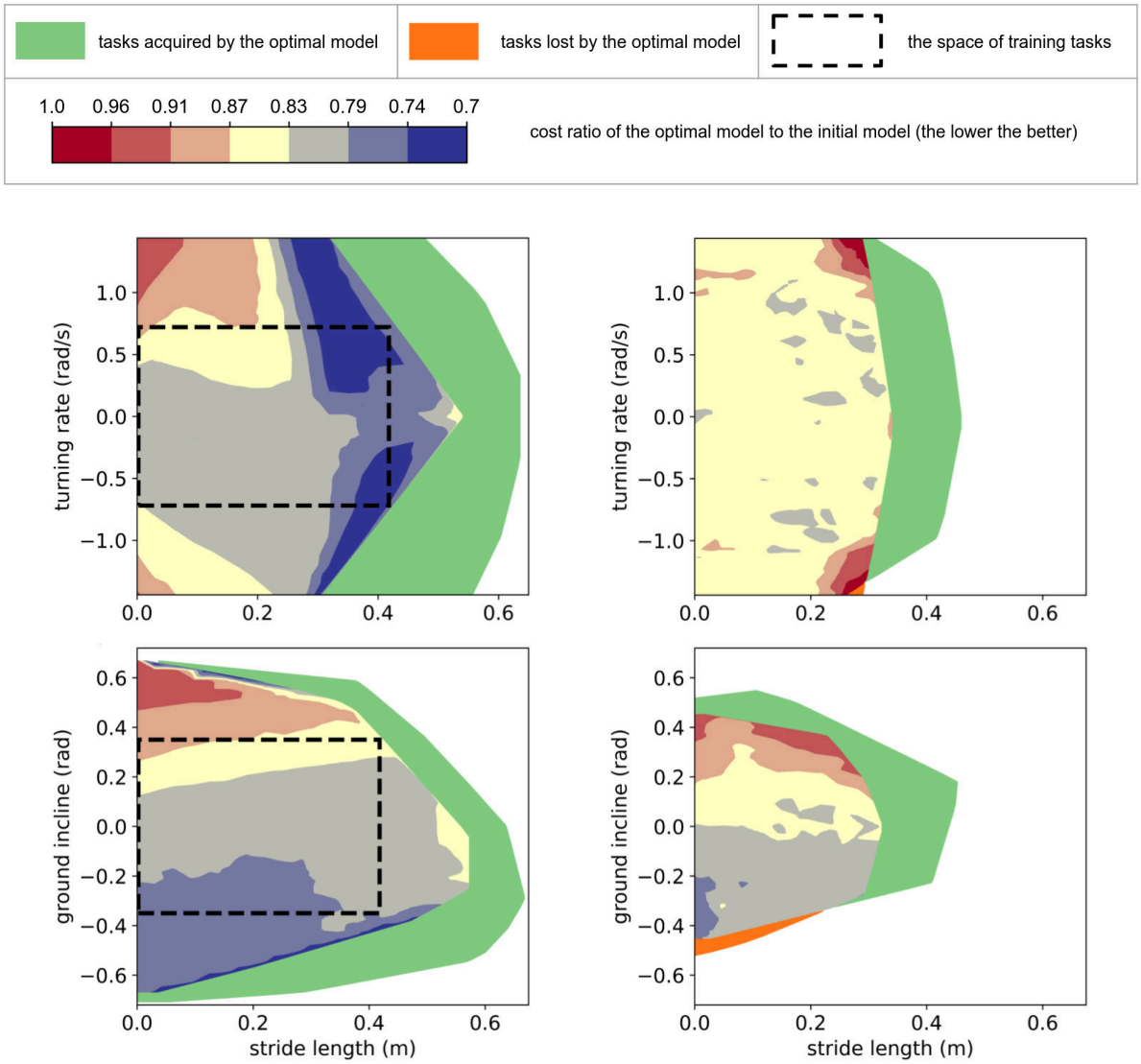
In the existing literature of bipedal robots, robot's floating base position (sometimes the CoM position) and orientation are often chosen to be control targets. They have 6 degrees of freedom (DoF) in total. In the case of fully-actuated robots (i.e. robots with flat feet), there is enough control authority to servo both the position and orientation. For underactuated robots, the existing approaches often give up tracking the trajectories in the transverse plane (x and y axis), because it is not possible to instantaneously track trajectories whose dimension is higher than the number of actuators (or we have to trade off the tracking performance). In this case, motion planning for discrete footstep locations is used to regulate the underactuated DoF. In our control problem, we

also face the same challenge since Cassie has line feet. The total dimension of the desired trajectories in Table 5.2 is 11, while Cassie only has 10 actuators. Following the common approach, we choose not to track the second element of the ROM in OSC, because it corresponds to the lateral position of the CoM for the initial model (a competing tracking objective to the pelvis roll angle) and maintaining a good pelvis roll tracking is crucial for stable walking. Instead, the second element of ROM is regulated by the desired swing foot locations via the planner in Eq. (5.19), even though the OSC does not explicitly track it.

5.2.3. Hardware Setup and Solve Time

We implement the MPC in Fig. 5.3 using the Drake toolbox (Tedrake, 2019), and the code is publicly available in the link provided in the Introduction. In hardware experiment, the MPC planner runs on a laptop equipped with Intel i7 11800H, and everything else (low-level controller, state estimator, etc) on Cassie's onboard computer. These two computers communicate via LCM (Huang et al., 2010). A human sends walking velocity commands to Cassie with a remote controller. Cassie is able to stably walk around with both the initial ROM and the optimal ROM (shown in the supplementary video).

The planning horizon was set to 2 foot steps with stride duration being 0.4 seconds. With cubic spline interpolation between knot points, we found that 4 knot points per stride was sufficient. IPOPT (Wächter and Biegler, 2006) was used to solve the planning problem in Eq. (5.19), and the solve time was on average around 6 milliseconds with warm-starts. We observed that this solve time was independent of the reduced-order models (initial or optimal) in our experiments. In contrast to the ROM, similar code required tens of seconds for the simplified Cassie model for a single foot step. As the following sections will show, Cassie's performance (with respect to the user-specified cost function) with the optimal model is better than with the initial model. This demonstrates that the use of ROM greatly increases planning speed, and that the optimized ROM improves the performance of the robot.



(a) Trajectory optimization, (C1).

(b) Simulation, (C2).

Figure 5.4: Cost comparison between the initial model (R2) and the optimal model (R3). Each plot shows the ratio of the optimal model's cost to the initial model's cost. For these examples, the ROM reduces the cost across the entire task space. The color scheme red-to-blue illustrates the degree to which the ROM shows improvement, with red corresponding to a minimal improvement and blue to a 30% reduction. The scales of the axes are the same between the trajectory optimization (C1) and the simulation (C2) for ease of comparisons.

5.3. Performance Evaluation and Comparison

In this section, we evaluate the performance of the robot (with respect to a user-specified cost function $\tilde{h}_\gamma$ in Section 5.1.2) in the following ROM settings:

(R1) without reduced-order model embedding,

(R2) with initial reduced-order model embedding,

(R3) with optimal reduced-order model embedding.

Additionally, the evaluation is done in the following cases:

(C1) trajectory optimization (open-loop),

(C2) simulation (closed-loop),

(C3) hardware experiment with real Cassie (closed-loop),

where (C1) is labeled as open-loop, while the others are considered closed-loop, because trajectory optimization is an optimal control method that solves for control inputs and feasible state trajectories simultaneously, without requiring a feedback controller. Table 5.3 lists the experiments conducted in this section. We note that (C1) is the same as Eq. (TO) but with a different task distribution, and that (R1) is only evaluated in (C1) because it serves as an idealized benchmark for comparison.

5.3.1. Experiment Motivations

Motivation for (C1)

In Section 5.1, we optimized for a reduced-order model given a task distribution. Here, one objective is to evaluate how well the model generalizes to out-of-distribution tasks. Additionally, trajectory optimization provides the ideal performance benchmark for the closed-loop system to compare to.

Motivation for (C2)

Trajectory optimization is used in Eq. (TO) to find the optimal model based on the open-loop performance. (C2) evaluates how well the cost reduction in open-loop can be translated to the closed-loop system with the MPC from Section 5.2.

	(**R1**) no ROM	(**R2**) initial ROM	(**R3**) optimal ROM
(**C1**) Traj. Opt.	X	X	X
(**C2**) Simulation		X	X
(**C3**) Experiment on real Cassie		X	X

Table 5.3: Experiments conducted in Section 5.3 (marked with x)

Stride length variation	< 2 cm
Side stepping variation	< 3 cm
Pelvis height variation	< 3 cm
Pelvis yaw variation	< 0.1 rad
Window size	4 consecutive footsteps

Table 5.4: Criteria to determine periodic walking gaits

Motivation for (C3)

(C3) evaluates how well the performance improvement can be translated to hardware.

5.3.2. Experiment Setups

In all experiments in this section, we use the initial and the optimal ROM from Example 5. Additionally, for all performance evaluations, we use the cost function $\tilde{h}_\gamma$ from Example 5 which mainly penalizes the joint torques.

(C1) Trajectory Optimization

We evaluate the open-loop performance by running the full-model trajectory optimization in Eq. (TO) over a wide range of tasks (stride length, turning rate, ground incline, etc). As a special case, (C1) combined with (R1) corresponds to Eq. (TO) without the constraint $g_c = 0$.

(C2) Simulation

We use Drake simulation (Tedrake, 2019). The MPC horizon is set to two footsteps, and the duration per step is fixed to 0.35 seconds which is the same as that of open-loop. Similar to (C1), we evaluate the performance at different tasks. For each desired task, we run a simulation for 12

seconds and extract a periodic walking gait based on a set of criteria listed in Table 5.4. Then we compute the cost and the actual achieved tasks (stride length, turning rate, etc) of that periodic gait.

(C3) Hardware Experiment

Some heuristics are introduced to the MPC in order to stabilize Cassie well. For example, we add a double-support phase to smoothly transition between two single-support phases by linearly blending the ground forces of the two legs. This is critical for Cassie, because unloading the springs of the support leg too fast when transitioning into swing phase can cause foot oscillation and bad swing foot trajectory tracking. The double-support phase duration is set to 0.1 seconds, and the swing phase duration is decreased to 0.3 seconds, compared to the nominal 0.35 seconds of stride duration in the trajectory optimization.

The hardware setup is described in Section 5.2.3. During the experiment, we send commands to walk Cassie around and make sure that the safety hoist does not interfere with Cassie's motion. After the experiment, we apply the criteria listed in Table 5.4 to extract periodic gait for performance evaluation.

5.3.3. Turning and Sloped Walking in Simulation

The goal of this section is to evaluate model performance in simulation. We use the initial and the optimal model from Example 5 of Section 5.1.4. The training task distribution of this model covers various turning rates, ground inclines and positive stride lengths, shown in Table 5.1. During performance evaluation (both (C1) and (C2)), we increase the task space size to two times of that of training stage in order to examine the optimal model's performance on the both seen (from training) and unseen tasks.

To visualize the performance improvement, we compare the cost landscapes between the initial and the optimal model. For (C1) and (C2), we first derive the cost landscapes of both models using the cost function $\tilde{h}_\gamma$ and then superimpose them in terms of cost ratio (i.e. ratio of (R3)'s cost to (R2)'s cost). The cost landscape comparisons are shown in Fig. 5.4. The red-blue color bar represents

different levels of performance improvement in terms of $\tilde{h}_\gamma$. Green color corresponds to the tasks acquired by the optimal model (i.e. the task that the initial model cannot execute). Orange color corresponds to the task lost by using the optimal model. We see in Fig. 5.4 that the optimal ROM performs better than the LIP in all tasks shown in the figure (since the cost ratio is always smaller than 1). The maximum cost reduction is 30% in trajectory optimization and 23% in simulation. This means that Cassie is able to complete the same task with 23% less joint torque in simulation.

Besides the improvement in terms of cost ratio, we also observe in simulation that the optimal ROM gained new task capability (indicated by the green area). For example, Cassie is capable of walking 54% faster on a slope of 0.2 radian when using the optimal ROM. Cassie also gets better in climbing steeper hills. At 0.1m stride length, Cassie can climb up a hill with 32% steeper incline. Overall, the task region gained is much bigger than the lost in simulation.

Comparing the cost landscape between the trajectory optimization and simulation, we can see that the task region gained are similar[5]. Cassie in general can walk faster at different turning rates and on different ground inclines. We also observed that the cost landscapes of the open- and closed-loop share a similar profile in ground incline. Both show bigger cost reduction in walking downhill than uphill. In contrast, the landscapes in turning rate look different between the open and closed loop. This is partially because there is only one stride (left support phase) in trajectory optimization while we average the cost over 4 strides in the simulation. Additionally, there is a difference in stride lengths between the open- and closed-loop, showing a control challenge in stabilizing around high-speed nominal trajectories. This gap can be mitigated by increasing the control gains in simulation. However, we avoid using unrealistic high gains because they do not work on hardware.

In Fig. 5.6, the dashed-line boxes represent the training task space used in the model optimization stage. There is not a strong correlation between the cost ratio and the training space. However, we can observe that the performance of the optimal ROM generalizes well to the unseen task (region outside the dashed-line box). For example, the cost reduction ratio stays around 16% at different

[5]In the case of (C1), there was not a clear threshold for defining the failure of a task. We picked a cost threshold at which the robot's motion does not look abnormal.

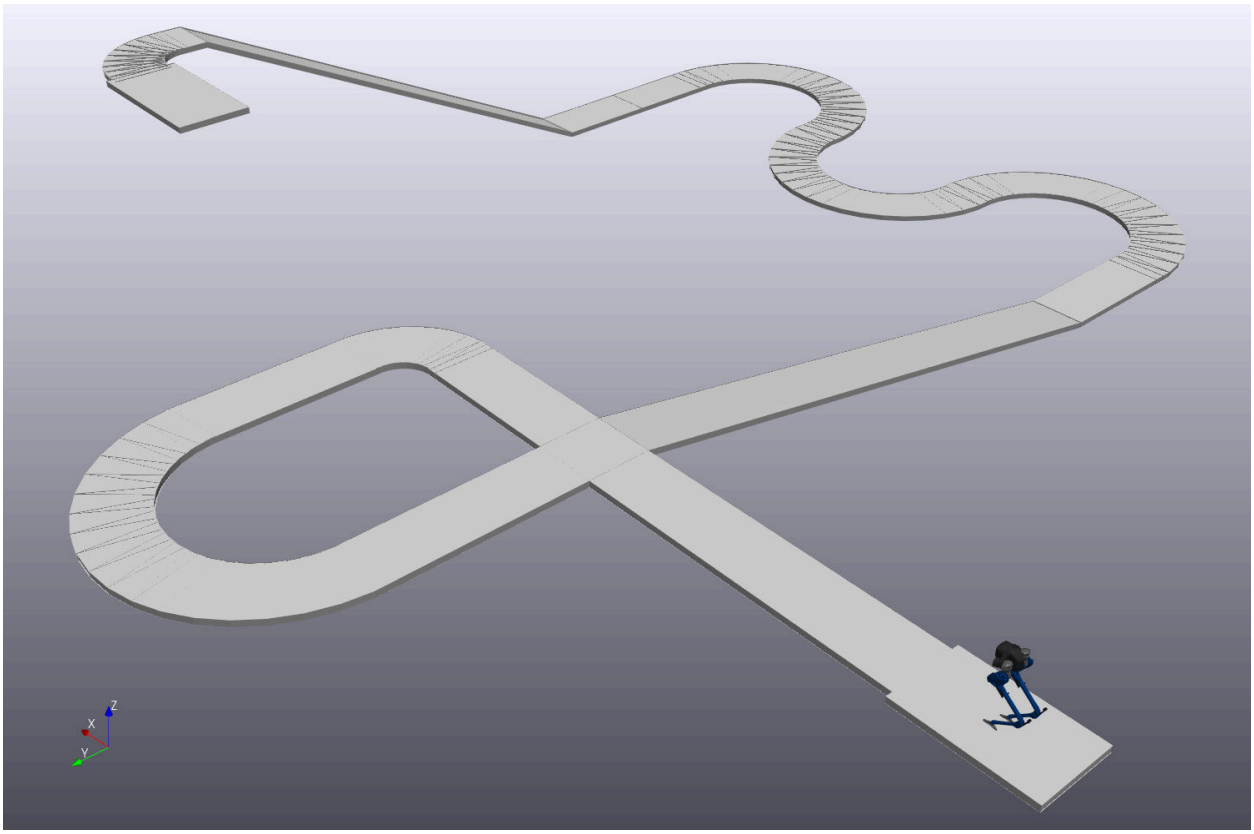


Figure 5.5: A track designed to showcase the performance difference between the LIP and the optimal ROM in simulation. The video of Cassie finishing the track can be found in the supplementary materials.

	LIP	Optimal ROM	Speed Improvement
Straight line (5m)	5.72 (s)	4.05 (s)	41%
Fast 90-degree turn	1.65 (s)	1.2 (s)	38%
Downhill (20%)	11.67 (s)	8.4 (s)	39%
S-turns	20.37 (s)	14.47 (s)	41%
Uphill (50%)	failed	17.73 (s)	-

Table 5.5: Completion time for some segments of the course.

turning rates in simulation.

Lastly, we designed a track shown in Fig. 5.5 for Cassie to finish as fast as possible to showcase the capability of the optimal ROM. The track includes various segments requiring Cassie to turn by different angles and walk on different sloped grounds. To enable Cassie to race through the track, we

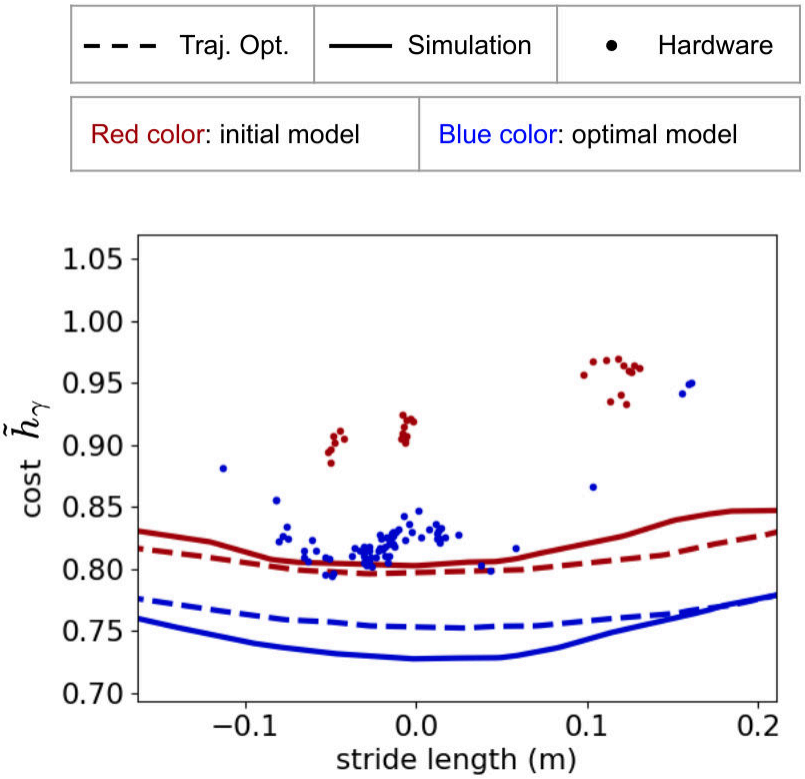
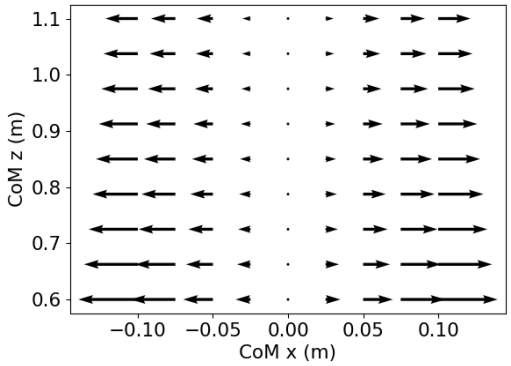


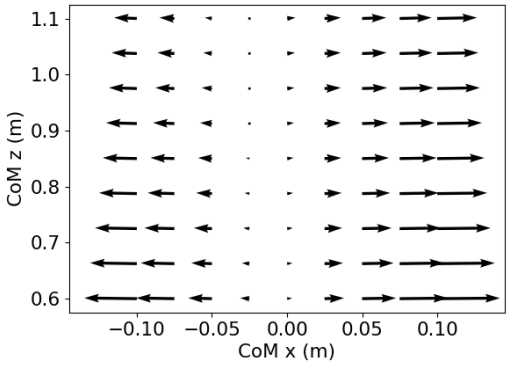
Figure 5.6: Cost comparison between the initial model (R2) and the optimal model (R3). The cost $\tilde{h}_\gamma$ is the cost function in Eq. (TO). For trajectory optimization and simulation, we densely sampled the tasks and interpolated the costs. For the hardware experiment, we plot the costs of collected data points directly in the figure without interpolation. To collect the data on hardware, we used a remote control to walk Cassie around, and we applied a moving window of 4 foot steps to extract periodic gaits according to Table 5.4. We note that there was about 2 cm of height variation in the hardware experiment, but we normalized every extracted data point to the same height using the height-cost relationship from the trajectory optimization to make fair comparisons.

implemented a high-level path following controller that sends commands (such as walking velocity) to the MPC. We tune the controller parameters so that Cassie can finish many segments as fast as possible without falling off the track. We can see in Table 5.5 that the Cassie on average can walk about 40% faster with the optimal ROM. This speed improvement reflected the periodic-walking result shown in the cost landscape plots in Fig. 5.4. Additionally, we observed that for the task of 50% ground incline Cassie with LIP exhibited many stop-and-go motions and eventually fell, while Cassie with the optimal ROM was able to complete the incline steadily.

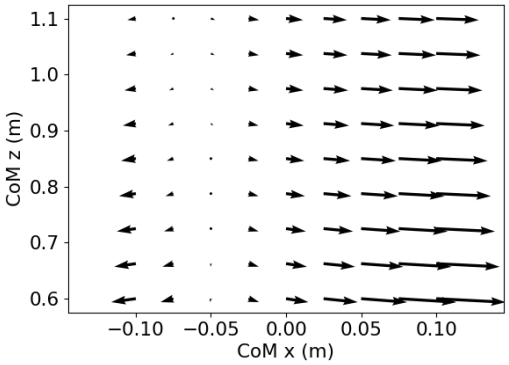
5.3.4. Straight-line Walking on Hardware



(a) Initial model (R2). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).



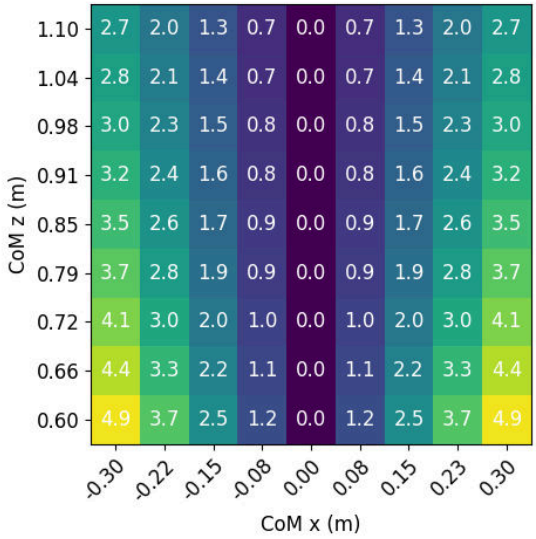
(b) Optimal model (R3). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).



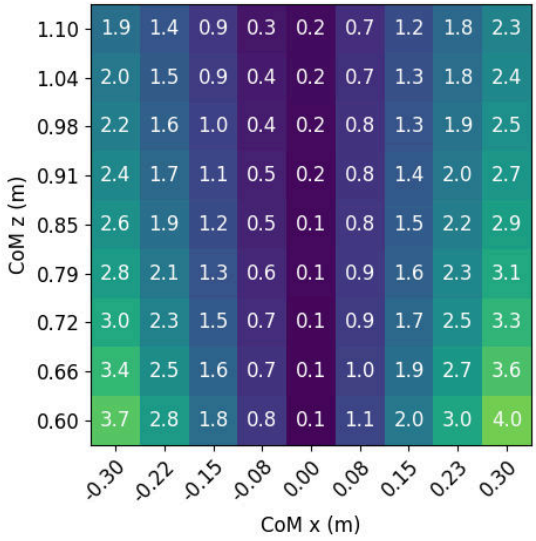
(c) Optimal model (R3). 2D slice at CoM position = -0.2m in y axis (when the CoM is to the right of the left stance foot, for example).

Figure 5.7: The vector fields of the ROM dynamics g over the CoM x and z position. In this example, the dynamics is the acceleration of the CoM, which is a function of the CoM position and velocity defined in Eq. (2.5b). The first plot is the initial model's dynamics, while the latter two are that of the optimal model at two different slices of CoM y position. In all plots, the CoM velocity is 0. We note that the size of the vectors only reflects the relative magnitude. The absolute magnitude of the vectors for the first two plots are shown in Fig. 5.8, although the scales of the x axis are different.

In this section, we aim to evaluate the model performance on the real robot. We evaluate the cost for different stride lengths while fixing the pelvis height (0.95 m), and then plot the costs of both the initial and the optimal model directly in Fig. 5.6. We also conducted the same experiment but in trajectory optimization and simulation for comparison. Additionally, in order to maximize the controller robustness for the hardware experiment, we constrained the center of pressure (CoP) close to the foot center during the model optimization stage, although this limits the potential



(a) Initial model (R2). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).



(b) Optimal model (R3). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).

Figure 5.8: The magnitude of ROM dynamics g over the CoM x and z position. The settings of Fig. 5.8a and 5.8b are the same as Fig. 5.7a and 5.7b, respectively. The magnitude plots show that the optimal model has smaller CoM accelerations, which implies smaller ground reaction forces, particularly in the x axis (implied by the vector fields in Fig. 5.7). We observed in the experiments that these force vectors align more closely with the normal direction of the ground.

performance gain of the optimal ROM.

In Fig. 5.6, we can see that the performance of the optimal ROM (evaluated by Cassie's joint torque squared in this case) is better than that of the initial ROM. On hardware, the performance improvement is around 10% for low-speed and medium-speed walking. As a comparison, the cost improves by up to 8% in open-loop and improves by up to 14% in simulation using the same optimal ROM. We can also observe that the hardware costs are higher than those of open-loop and simulation, which might result from additional torques required to track the desired trajectories due to sim-to-real modeling errors. Nonetheless, the improvement percentages are fairly similar across the board. This demonstrates that the model performance was successfully transferred to the hardware via the MPC, despite the modeling error in the full-order model.

5.3.5. Optimal Robot Behaviors

In order to understand the source of the performance improvement, we look at the motion of the robot and the center of mass dynamics (the ROM dynamics g). The discussions in this section are based on the straight-line walking experiments in Section 5.3.4.

We observe in the full-model trajectory optimization that the average center of pressure (CoP) stays at the center of the support polygon when we use LIP on Cassie (R2). In contrast, the CoP moves toward the rear end of the support polygon where there is no ROM embedding (R1). Interestingly, this CoP shift emerges when using the optimal model (R3) in the open-loop trajectory, and we observe the same behavior in both simulation and hardware experiment. On hardware, we confirm this by visualizing the projected CoM on the ground when Cassie walks in place. The projected CoM is close to CoP, since there is little centroidal angular momentum for walking in place. The projected CoM of the hardware data indeed shifts towards the back of the support polygon when using the optimal model.

To understand why the projected CoM moves backward, we plot the ROM dynamic function g in Fig. 5.7 for both the initial model and the optimal model. In the case of the initial model (LIP), we know that the dynamics should be symmetrical about the z axis, specifically the acceleration

should be 0 at x = 0 (Fig. 5.7a). This vector field profile, however, looks different in the case of the optimal model. As we can see from Fig. 5.7b, the area with near-zero acceleration shifts towards the -x direction (i.e. to the back of the support polygon), and interestingly it also slightly correlates to the height of the CoM. The higher the CoM is, the further back the region of zero acceleration is. Additionally, we know that the LIP dynamics in the x-z plane is independent of the CoM y position. That is, no matter which slice of the x-z plane we take, the vector field should look identical. For the optimal model, the dynamics in the x-z plane is a function of the CoM y position. The further away the CoM is from the stance foot (bigger foot spread), the further back the zero acceleration region is.

Aside from the vector field plots of the CoM dynamics, we also visualize the absolute value of the vectors in Fig. 5.8. We can see that the magnitude of the CoM acceleration in general becomes smaller when using the optimal model. This implies that the total ground reaction force is smaller with the optimal model, even if the robot walks at the same speed. Given the same walking speed, the robot decelerates and accelerates less in the x axis when using the optimal model (i.e., the average speed is the same, but the speed fluctuation becomes smaller after using the optimal model). We hypothesize that the decrease in ground force magnitude partially contributes to the decrease in the joint torque in the case of Cassie walking. The fact that there is less work done on the CoM during walking with the optimal ROM might have led to the decrease in torque squared which is a proxy for energy consumption.

The experiments in Section 5.3 demonstrate two things. First, the optimal behavior and the performance are transferred from the open-loop training (left side of Fig. 5.1) to a closed-loop system (right side of Fig. 5.1). Second, the optimal reduced-order model improves the real Cassie's performance, while the low dimensionality permits a real time planning application.

5.4. MPC for General ROMs

Sections 5.2 and 5.3 limit the ROMs to a predefined embedding function r which simplifies the planner and enables real time planning results. In this section, we present an MPC formulation for a general ROM, where full-order states at swing foot touchdown events were used to provide

physical meaning to the resulting plan. Given an ongoing steady improvement in computational and algorithmic speed, we believe this general MPC will soon be solvable in real time on hardware.

5.4.1. Hybrid Nature of the Robot Dynamics

Shown in Eq. (2.2), the dynamics of the full model is hybrid – it contains both the continuous-time dynamics and discrete-time dynamics due to the foot collision. In contrast, many existing reduced-order models assume zero ground impacts at foot touchdown. This is partially due to the fact that the exact embedding of a reduced-order discrete dynamics does not always exist. For example, we could have two pre-impact states x of the full model that correspond to the same reduced-order states, but the post-impact states of the full model map to two different reduced-order states. In this case, the reduced-order discrete dynamics is not an ordinary (single-valued) function. Therefore, in order to capture the exact full impact dynamics in the planner, it is necessary to mix the reduced-order model with the discrete dynamics from the full-order model. We note that the traditional approaches to reduced-order planning and embedding must also grapple with approximations of the impact event.

In addition to the issue above, the mix of reduced- and full-order models also seems necessary if we do not retain the physical interpretability of the embedding r when planning for the optimal footstep locations in the planner. This results in a low-dimensional trajectory optimization problem, a search for $y_j(t)$ and $\tau_j(t)$, with additional decision variables $x_{-,j}, x_{+,j}$, representing the pre- and post-impact full-order states. The index j refers to the j-th stride. The constraints relating the reduced-order state to the full-order model and the impact dynamics are

$$\begin{aligned} y_j(t_j) = r(q_{-,j};\theta_e), \qquad & \dot{y}_j(t_j) = \frac{\partial r(q_{-,j};\theta_e)}{\partial q_{-,j}}\dot{q}_{-,j}, \\ y_{j+1}(t_j) = r(q_{+,j};\theta_e), \quad & \dot{y}_{j+1}(t_j) = \frac{\partial r(q_{+,j};\theta_e)}{\partial q_{+,j}}\dot{q}_{+,j}, \\ \text{and} \qquad & C_{hybrid}(x_{-,j}, x_{+,j}, \Lambda_j) \leq 0, \end{aligned} \tag{5.21}$$

where t_j is the impact time (ending the j-th stride), C_{hybrid} represents the hybrid guard S and the impact mapping Δ without left-right leg alternation[6] (Westervelt et al., 2003).

[6]The impact mapping Δ can be simplified to identity if we assume no impact (i.e. swing foot touchdown velocity

5.4.2. Planning with ROM and Full-order Impact Dynamics

Similar to Section 5.2, we formulate a reduced-order trajectory optimization problem to walk n_s strides. However, we replace the discrete footstep inputs T_{fp} with the full robot states $x_{-,j}$ and $x_{+,j}$. The key difference between the planning problems in Section 5.2 and this section is that here we introduce new variables

- full-order states $x_{-,j}$, $x_{+,j}$ and ground impulses Λ_j (to capture the exact full-order impact dynamics), and
- the ROM's continuous-time input τ.

To improve readability, we stack decision variables into bigger vectors $\mathrm{T} = [\tau_1, ..., \tau_n] \in \mathbb{R}^{n_\tau n}$ and $X = [x_{-,1}, ..., x_{-,n_s}, x_{+,1}, ..., x_{+,n_s}] \in \mathbb{R}^{2n_x n_s}$.

Costs are nominally expressed in terms of $[y, \dot{y}]$ and τ, though the pre- and post-impact full-order states can also be used to represent goal locations. In addition to the constraints in Eq. (5.21), we impose constraints $C_{kinematics}(X) \leq 0$ on the full model's kinematics such that the solution obeys joint limits, stance foot stays fixed during the stance phase, and legs do not collide with each other.

The planning problem with the general ROM is

$$\begin{aligned} \min_{w} \quad & \|\mathrm{T}\|^2_{W_\mathrm{T}} + \|Z - Z_{reg}\|^2_{W_Z} + \|X - X_{reg}\|^2_{W_X} \\ \text{s.t.} \quad & \text{Reduced-order dynamics (Eq. (2.5b))}, \\ & \text{Hybrid constraints (Eq. (5.21))}, \\ & C_{kinematics}(X) \leq 0, \\ & x_0 = \text{current feedback full-order state}, \end{aligned} \tag{5.22}$$

where $w = [Z, \mathrm{T}, x_0, X, \Lambda_1, ..., \Lambda_{n_s}] \in \mathbb{R}^{n_w}$, W_T, W_Z and W_X are the weights of the norms, Z_{reg} is the regularization state for the reduced model, and X_{reg} is the regularization state for the full state and contains the goal location of the robot. After solving Eq. (5.22), we reconstruct the desired

is 0 in the vertical axis).

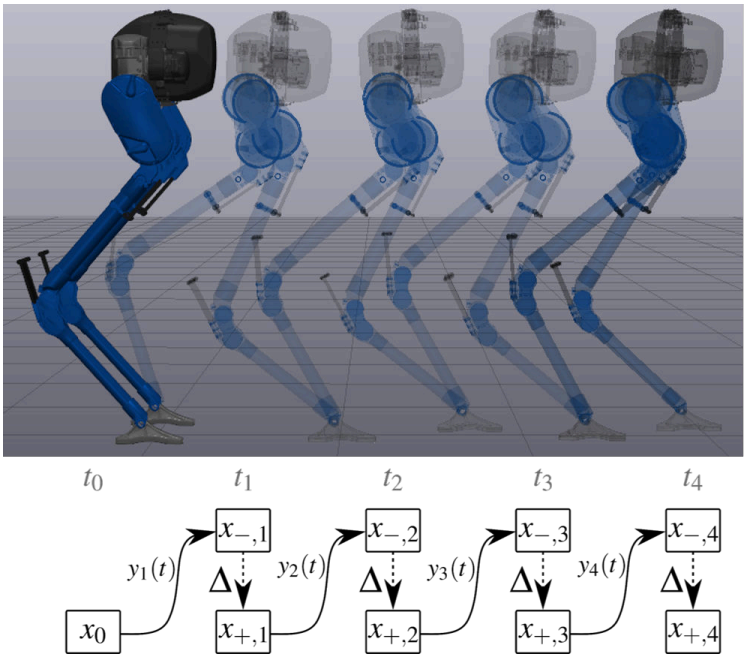


Figure 5.9: An example of the real time planner in Eq. (5.22). Given a task of covering two meters in four steps starting from a standing pose, we rapidly plan a trajectory for the reduced-order model. The high-dimensional model is used to capture the hybrid event at stepping, as illustrated in the diagram.

ROM trajectory $y_d(t)$ from the optimal solution Z^*. Different from Section 5.2, the optimal solution w^* here also contains the desired full-order states X^* at impact events, from which we derive not only the desired swing foot stepping locations but also the desired trajectories for joints such as swing hip yaw and swing toe joint. Additionally, since there are full states in the planner, we can send the turning rate command directly to the ROM planner.

Fig. 5.9 visualizes the pre-impact states in the case where the robot walks two meters with four strides, connected by the hybrid events and continuous low-dimensional trajectories $y_j(t)$. Although there is no guarantee that the planned trajectories $y_j(t)$ are feasible for the full model except those at the hybrid events, we were able to retrieve $q(t)$ from $y_j(t)$ through inverse kinematics, meaning the embedding existed empirically. We note that classical models like LIP also provide no guarantees (Iqbal et al., 2022). For instance, there is no constraint on leg lengths in the ROM which could lead to kinematic infeasibility. The formulation in Eq. (5.22) preserves an exact representation of the hybrid dynamics, but results in a significantly reduced optimization problem.

5.4.3. Implementation and Experiments

We implement the MPC using Eq. (5.22) for both simulation and hardware experiments (the hardware setup is the same as Section 5.2.3). In simulation, we were able to transfer the open-loop performance to closed-loop performance with this new MPC. However, on hardware, the off-the-shelf solvers IPOPT and SNOPT were not capable of solving the planning problem in Eq. (5.22) fast enough or well enough to enable a high-performance real time MPC. With IPOPT, the planner simply did not run fast enough. With SNOPT, even though the solve time can be decreased down to 30ms with loose optimality tolerance and constraint tolerance, we sacrificed the solution quality too much to achieve big stride length on Cassie. Nonetheless, we believe that the general MPC will soon be solvable within reasonable time constraints on hardware, as computer technology advances steadily. Additionally, a well-engineered custom solver can also help enable real time planning. Boston Dynamics has shown their success in the nonlinear MPC solver with the centroidal momentum model and full model configurations[7] (Bos).

5.5. Discussion

5.5.1. Model Parameterization

Trade-off between planning speed and performance

A fundamental trade-off exists between planning speed and model performance. As the model (the functions r and g) becomes more expressive, we see slower planning speeds and better performances. This trade-off has been shown by existing works (Li et al., 2021; Norby et al., 2022). Additionally, we found that, for some choice of task space, a linear model performs close to a full model. This linear reduced-order dynamics transforms the MPC in Eq. (5.19) into a quadratic optimization problem, allowing for rapid planning. This linear model also renders a closed-form solution and makes it suitable for existing techniques in robust control design and stability analysis. For challenging or complex task spaces, linear basis functions sacrifice significant performance when compared with those of higher degree. We emphasize that our method can be used to optimize models of any

[7]The centroidal momentum model (Orin and Goswami, 2008; Wensing and Orin, 2016) describes the actual dynamics without imposing constraints on the full model. However, this momentum model lacks positional representations, thus requiring the incorporation of full model configurations in planning.

chosen degree, and leave such selection to the practitioner.

Alternative basis functions

Beside different orders of monomials, we also experimented with trigonometric monomials (e.g. $\sin^a(x)\cos^b(x)$ where $a, b \in \mathbb{N}$). However, we found no notable difference with this basis set. Since quadratic basis approaches optimal performance in model optimization in Section 5.1.4, we leave a broader exploration of choices of basis functions as a possible future work.

Physical Interpretability of ROMs

Classical ROMs often maintain some level of physical interperability, because they are built from mechanical components like springs and masses. Our approach, which uses more general representations, does sacrifice this connection to human intuition. However, we have found it beneficial to manually select the embedding function r. This has the benefit of ensuring that the reduced-order state remains human interpretable, which is useful for specifying objectives for planning and control in Section 5.2. One might also imagine restricting the space of reduced-order dynamics functions g to maintain physical connections (e.g. specifying nonlinear or velocity-dependent springs, inspired by human studies (Pequera et al., 2023)), though we leave this to future work.

5.5.2. Performance Gap Between Open-loop and Closed-loop

The proposed approach to model optimization uses full-model trajectory optimization. This has a few advantages. First, it allows us to embed the reduced-order model into the full model exactly via constraints. Second, it is more sample efficient than the approaches in reinforcement learning (such as (Pandala et al., 2022)). However, using trajectory optimization leaves a potential performance gap between the offline training and online deployment, because trajectory optimization is an open-loop optimal control method which does not consider any controller heuristics by default. For instance, our walking controller constructs the swing foot trajectory with cubic splines, and we found the open-loop performance can be transferred to the closed-loop better after we add this cubic spline heuristic as a constraint in the trajectory optimization problem.

While we have seen the performance of the robot improve, we also observed that the solver would

exploit any degree of freedom of the input and state variables during the model training stage. For example, the center of pressure turned out to play an important role in improving the performance (reducing joint torques) of Cassie in Section 5.3. If we had not regularized the CoP (or the CoM motion) during the model optimization stage, the solver would have moved the CoP all the way to the edge of the support polygon. Although this exploitation can potentially lead to a much bigger cost improvement, it also hurts the robustness of the model. Under hardware noise and model uncertainty, tracking the planned trajectories of this optimal model cannot stabilize the robot well, and thus the performance cannot be transferred to the hardware. One principled way of fixing this issue is to optimize the robustness of the trajectory alongside the user-specified cost function (Dai and Tedrake, 2012a; Zhu et al., 2022).

For the general optimal ROM MPC in Section 5.4, one place where the performance gap can enter is the choice of the cost function in Eq. (5.22). In our past experiments, we simply ran a full model trajectory optimization and used this optimal trajectory for the regularization term in Eq. (5.22). It worked well, although there was a slight improvement drop. To mitigate the gap, one could try inverse optimal control to learn the MPC cost function given data from the full model trajectory optimization.

Since our robot has one-DOF underactuation caused by line feet design, it is not trivial to track the desired trajectory of the reduced-order model within the continuous phase of hybrid dynamics. We noticed in the experiment that there was a noticeable performance difference between whether or not we tracked the first element of the desired ROM trajectory. Therefore, we conjecture that if we use a robot with a finite size of feet, we could translate the open-loop performance to the closed-loop performance better and more easily.

Lastly, we found that empirically it is easier to transfer model performance from open-loop to closed-loop when we fix the embedding function r, although the open-loop performance improvement is usually much bigger (sometimes near full-model's performance) when we optimize both the embedding function r and dynamics g. As a concrete example, in some model optimization instances the optimal ROM position y can be insensitive to the change of CoM position, which makes it difficult

to servo the CoM height. In the worst case, this insensitivity could lead to substantial CoM height movement and instability of the closed-loop system.

5.5.3. Limitations of Our Framework

In our bilevel optimization approach, the initial ROM must be feasible for the inner-level trajectory optimization to obtain a meaningful gradient for the outer-level optimization. This means that we must initialize the ROM to one capable of walking, potentially limiting our ability to use stochastic initialization to explore the entire ROM space. Despite this potential drawback, we note that random task sampling in Algorithm 2 can help escape certain local minima (effectiveness depends on the cost landscape of the model optimization). Future work could explore the role of this initialization, for instance by evaluating performance when starting from multiple existing hand-designed ROMs.

Our approach requires the user to determine the dimension of the ROM. Increasing the dimension theoretically strictly improves model performance, at the cost of MPC computational speed. As a result, this defines a Pareto optimal front, without a simple way to automatically determine the dimension. That said, there are recent works which attempt to select between models of varying complexity (Li et al., 2021; Khazoom et al., 2023; Norby et al., 2022), which we believe might be applied to our framework.

5.5.4. Generality of Our Framework

This chapter focuses on applying the optimal ROMs to the hardware Cassie, but throughout the project we observe that LIP performs reasonable well for Cassie, particularly over relatively simple task domains such as straight-line walking. We hypothesize that this is due, in part, to the fact that Cassie's legs are relatively light. As an experiment, we investigated the effect of foot weight on the performance improvement. When increasing the foot's mass to 4kg (the robot weighs 40kg in total), we observed that the LIP cost relative to the full model's increased from 1.3 to 1.8 and offered a greater room for improvement, resulting in 40% torque cost reduction for tasks similar to Example 1's. Beside this investigation, we also saw more than 75% of cost reduction for the five-link planar robot in our prior work (Chen and Posa, 2020).

Furthermore, the proposed framework is agnostic to types of robots and tasks (e.g. quadrupeds and dexterous manipulators). This has implications all over robotics, given the need for computational efficiency and the prevalence of reduced-order models in locomotion and manipulation.

5.6. Conclusion

In this chapter, we directly optimized the reduced-order models which can be used in an online planner that achieved performance higher than that of the traditional physical models. We formulated a bilevel optimization problem and presented an efficient algorithm that leverages the problem structure. Examples showed improvements up to 38% depending on the task difficulty and the performance metric. The optimal reduced-order models are more permissive and capable of higher performance, while remaining low dimensional. We also designed two MPCs for the optimal reduced-order models which enable Cassie to accomplish tasks with better performance. In the hardware experiment, the optimal ROM showed 10% of improvement on Cassie, and we investigated the source of performance gains for this particular model. We demonstrated that the use of ROM greatly reduces planning time, and that the optimized ROM improves the performance of the robot beyond the traditional ROMs.

Although the model optimization approach presented in this chapter has the advantage of optimizing models agnostic to low-level controllers, it does not guarantee that the performance improvements from these optimal models can be transferred to the robot via a feedback controller as discussed in Section 5.5.2. In an attempt to fix the this issue, Chapter 6 optimizes the model in a closed-loop fashion, so that the model optimization accounts for the controller heuristics and maintains the closed-loop stability. This would also potentially ease the process of realizing the optimal model performance on hardware. Additionally, discussed in Section 5.4.1, an approximation is necessary if we were to find a low dimensional representation of the full-order impact dynamics. In this chapter, we only circumvented the hybrid problem by either using a physically-interpretable ROM or mixing the full impact dynamics with the ROM. Finding an optimal low-dimensional discrete dynamics for a robot still remains an open question.

CHAPTER 6

REINFORCEMENT LEARNING FOR REDUCED-ORDER MODELS OF LEGGED LOCOMOTION

This chapter improves upon Chapter 5. Similar to Chapter 5, we optimize a ROM given a user-specified task distribution and objective function, but different from Chapter 5 we also take into account the feedback controller used with the ROM. Specifically, we cast the ROM optimization problem as a model-based reinforcement learning problem. All videos and code are available at https://sites.google.com/view/ymchen/research/rl-for-roms.

Additionally, the work in this chapter also provides an avenue to bridge classical model-based control and model-free RL for bipedal robots, as outlined in Section 3.3.

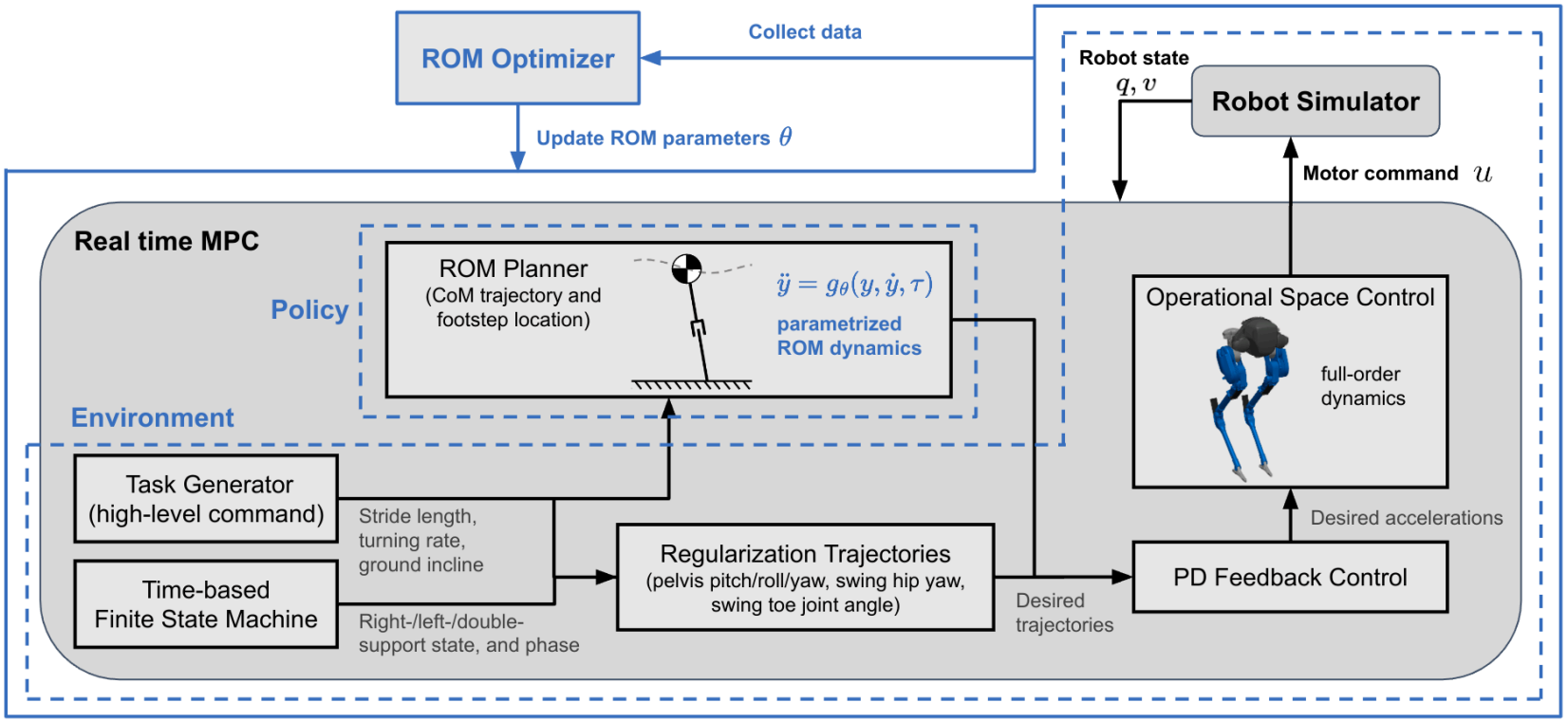


Figure 6.1: The diagram of our reinforcement learning framework for reduced-order models (ROMs) of legged locomotion. We learn a ROM in simulation where the robot is controlled via a real time MPC presented in Section 5.2. The MPC follows a high-level command (task) and operates based on a time-based finite state machine (FSM), governing footstep timing in left-support, right-support, or double-support state. The MPC contains two trajectory generators – a reduced-order model planner and a regularization trajectory generator to fill out the joint redundancy of the robot. All desired trajectories are converted to desired acceleration command via PD feedback control before being sent to the Operational Space Controller (OSC), a quadratic-programming-based inverse-dynamics controller (Sentis and Khatib, 2005; Wensing and Orin, 2013). In this learning framework, the ROM planner is the policy, while everything else in the closed-loop system, such as the simulation and OSC, constitutes the environment. We parameterize the policy using ROM parameters, specifically the parameters of ROM dynamics. The optimizer collects data from simulation rollouts and updates the model (policy) parameters according to a user-specified reward function. To ensure that the optimizer collects data at a consistent rate, we maintain a fixed update rate of 20Hz for the ROM planner, and we downsample the environment state (which operates at 1000Hz) to match this 20Hz rate.

6.1. Problem Statement

Our goal is to find the optimal model parameters that maximize the performance of the robot given a task distribution Γ. In this study, Γ is a uniform distribution over a range of stride length, turning rate and ground incline. Let $u(\theta)$ be a model-based control policy (i.e. a controller) that guides the robot to follow the trajectories of a ROM parameterized by θ. From another viewpoint, this same policy $u(\theta)$ also constrains the robot to behave like the ROM. Let $\{(x_t, u_t) \mid t = 1, 2, ..., T\}$ be the state and input trajectories of the robot completing a task $\gamma \sim \Gamma$ under the policy $u(\theta)$. We evaluate the performance for this task γ and policy $u(\theta)$ using a cost function $h(x, u)$ accumulated over the trajectories:

$$H_{\gamma,u(\theta)} = \sum_{t=1}^{T} h\left(x_t, u_t\right). \tag{6.1}$$

Given this evaluation metric, an optimization problem for finding the model can be formulated as

$$\min_{\theta} \mathbb{E}_{\gamma\sim\Gamma} \min_{u(\theta)} \left[H_{\gamma,u(\theta)}\right], \tag{6.2}$$

where the inner-level optimization minimizes the cost $H_{\gamma,u(\theta)}$ over all possible controllers $u(\theta)$ that constrain the robot to behave like a ROM parameterized by θ, and the outer-level optimization minimizes the expectation of inner-level cost over a task distribution. Eq. (6.2) was proven to be solvable by Chapter 5 with successful results. However, the optimal policy $u(\theta)$ of (6.2) is not computable online in real time, necessitating an alternative policy $u_o(\theta)$ for online model deployment. Thus, while solving (6.2) leads to a ROM capable of maximal performance, this performance is not necessarily realizable via real time control, leading to reduced closed-loop performance. To improve this, in this chapter, we find the model parameters θ while using the control policy $u_o(\theta)$ during offline training:

$$\min_{\theta} \mathbb{E}_{\gamma\sim\Gamma} \left[H_{\gamma,u_o(\theta)}\right]. \tag{6.3}$$

Specifically, this control policy $u_o(\theta)$ is the model predictive control shown in Fig. 6.1. This seemingly simple change necessitates a significant shift in algorithmic approach, which we detail in Section 6.2.

6.2. ROM Optimization via Reinforcement Learning

In this section, we cast the model optimization problem in Eq. (6.3) as a model-based reinforcement learning problem.

6.2.1. Reinforcement Learning Structure

In reinforcement learning, the system comprises a policy and an environment. The policy takes the current environment state s and outputs an action a. Given the current state s and the action a, the environment transitions to the next state s'. Each pair of state and action (s, a) results in a reward r. In this work, the policy is the ROM planner, and the environment includes everything else in the closed-loop system (Fig. 6.1). Given this choice, the state of the environment s includes the robot's state and input (x, u), task γ, and finite state machine information (including the phase of the current state). The policy action a are the CoM states and the foot step locations (the solution of the ROM trajectory optimization). Additionally, we limit the policy's update rate to 20Hz, so that the state and action pairs are collected at a fixed rate.

6.2.2. Policy Parameterization

Reduced-order models for legged locomotion commonly describe the CoM motion, while each model imposes different constraints on the robot to derive its own CoM dynamics (Kajita and Tani, 1991; Blickhan, 1989). Inspired by this, we search for an optimal CoM dynamics g, while using the same embedding function r, the CoM position relative to the stance foot, as LIP and SLIP.

We parameterize the ROM with monomials of the state of the ROM, with linear weights. That is,

$$\ddot{y} = g_\theta(y, \dot{y}, \tau) = \Theta\phi(y, \dot{y}), \tag{6.4}$$

where g_θ is the ROM dynamics parameterized by θ, $\Theta \in \mathbb{R}^{n_y \times n_\phi}$ is the matrix form of θ, and ϕ is the feature vector containing the monomials. Even though we use monomials here, any function approximator (e.g. a neural network) can be used to parameterize the ROM dynamics in Eq. (6.4). We note that the ROM parameters θ are the policy parameters, as the ROM planner is the policy in our RL framework (Fig. 6.1).

In this chapter, we initialize the ROM to an LIP, because the LIP is effective for simple walking tasks, and this initialization speeds up the learning process. To implement this initialization, we augment the feature vector ϕ with the terms in the LIP dynamics function. Additionally, we use monomials of order up to 2. That is, ϕ includes elements such as y_1, y_0^2 and $y_0\dot{y}_1$, where the subscripts denote the indices of each element in y. Given this, θ is of dimension 90. The parameterization choice in Eq. (6.4) preserves physical interpretability, since the learned model describes the CoM dynamics.

6.2.3. Rewards and the RL Problem Statement

Instead of minimizing the cost in Eq. (6.3), our RL formulation maximizes the return (i.e. accumulated rewards) $R = \sum_{t=1}^{T} r_t$, where r_t is the reward at time t. Therefore, to encourage minimizing the cost $h(x, u)$ and achieving a desired task γ, we design the reward function

$$r = \exp(-w \cdot h) + 0.5 \exp\left(-\|\gamma - \gamma_{fb}\|_W\right), \tag{6.5}$$

where w and W are constant weights, γ is the desired task value, and γ_{fb} is the achieved task value (a function of the robot's state). We note that this reward is a function of the environment state s. Furthermore, the accumulated reward structure already incentivizes the robot to finish the entire episode, eliminating the need for penalty terms in case of early simulation termination (e.g. the robot falls). In this chapter, we choose $h = u^T u$ (quadratic cost on motor torques), and the horizon $T = 100$ which is equivalent to 5 seconds of simulation time. Additionally, the user-specified h is ultimately the metric for model performance evaluation.

The objective of our RL problem is to maximize the expected return over the task distribution:

$$\max_{\theta} \mathbb{E}_{\gamma \sim \Gamma}\left[R\right]. \tag{6.6}$$

There are several algorithms for solving Eq. (6.6), such as policy gradient and evolutionary strategy. In this work[8], we choose Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen, 2006). The advantage of this approach includes easy parallelization (Salimans et al., 2017) and easy

[8]Proximal Policy Gradient (PPO) also resulted in successful model training in our experiment, but it required more hyperparameter tuning.

Algorithm 3 Evaluation of return $\hat{R}(\theta, \{\gamma_j\})$

Input: model parameters θ and sampled tasks $\{\gamma_j\}$
1: **for** $j = 1, \ldots, N_\gamma$ **do**
2: Roll out an episode with the MPC in Section 5.2
3: Compute the return $R_{\gamma_j} = \sum_{t=1}^{T} r_t$ of this rollout
4: **end for**
5: **return** $\frac{1}{N_\gamma} \sum_{j=1}^{N_\gamma} R_{\gamma_j}$

Algorithm 4 CMA-ES with curriculum learning

Input: initial mean θ_0 and variance σ_0^2 for the parameter distribution p_θ, and initial task set Γ_d
1: **for** iter $= 1, 2, \ldots$ **do**
2: **if** mod(iter, N_c) $= 0$ **then**
3: Grow the task set Γ_d (Section 6.2.4)
4: **end if**
5: Randomly draw N_γ tasks $\{\gamma_j\}$ from Γ_d
6: Sample N_θ parameters $\{\theta_i\} \sim p_\theta$
7: **for** each sampled θ_i **do**
8: Compute return $\hat{R}(\theta_i, \{\gamma_j\})$
9: **end for**
10: Update the mean and covariance of p_θ (CMA-ES)
11: **end for**

hyperparameter tuning (partially due to the absence of value approximation). We use the package Optuna (Akiba et al., 2019) for the CMA-ES optimizer.

Let p_θ be the probability distribution over θ in the CMA-ES algorithm. The exact problem that CMA-ES solves is

$$\max_{p_\theta} \mathbb{E}_{\theta \sim p_\theta} \left[\mathbb{E}_{\gamma \sim \Gamma} [R] \right]. \tag{6.7}$$

The difference between Eq. (6.7) and (6.3), besides the cost-reward difference, is the stochasticity of parameters θ needed for the exploration of CMA-ES.

6.2.4. Curriculum Learning

We observe that learning a policy for a large task space is difficult, so we implement a curriculum learning similar to (Margolis et al., 2022) to facilitate the learning process. Our method discretizes the continuous task domain of Γ into a set Γ_d. This set Γ_d starts small and expands every N_c iterations by including the adjacent tasks of successful tasks during learning.

To evaluate the inner expected value in Eq. (6.7), we approximate it by randomly drawing N_γ number of tasks from the set Γ_d and averaging the returns of these tasks. Let $\hat{R}$ be this approximate expected return given model parameters θ. The evaluation of $\hat{R}$ is shown in Algorithm 3. As a part of the curriculum learning, we also adjust the number of tasks N_γ, as the size of Γ_d grows larger. Specifically, $N_\gamma = \text{floor}\left(\rho_\gamma |\Gamma_d|\right)$, where ρ_γ is the sampling percentage and $|\cdot|$ counts the number of elements in a set.

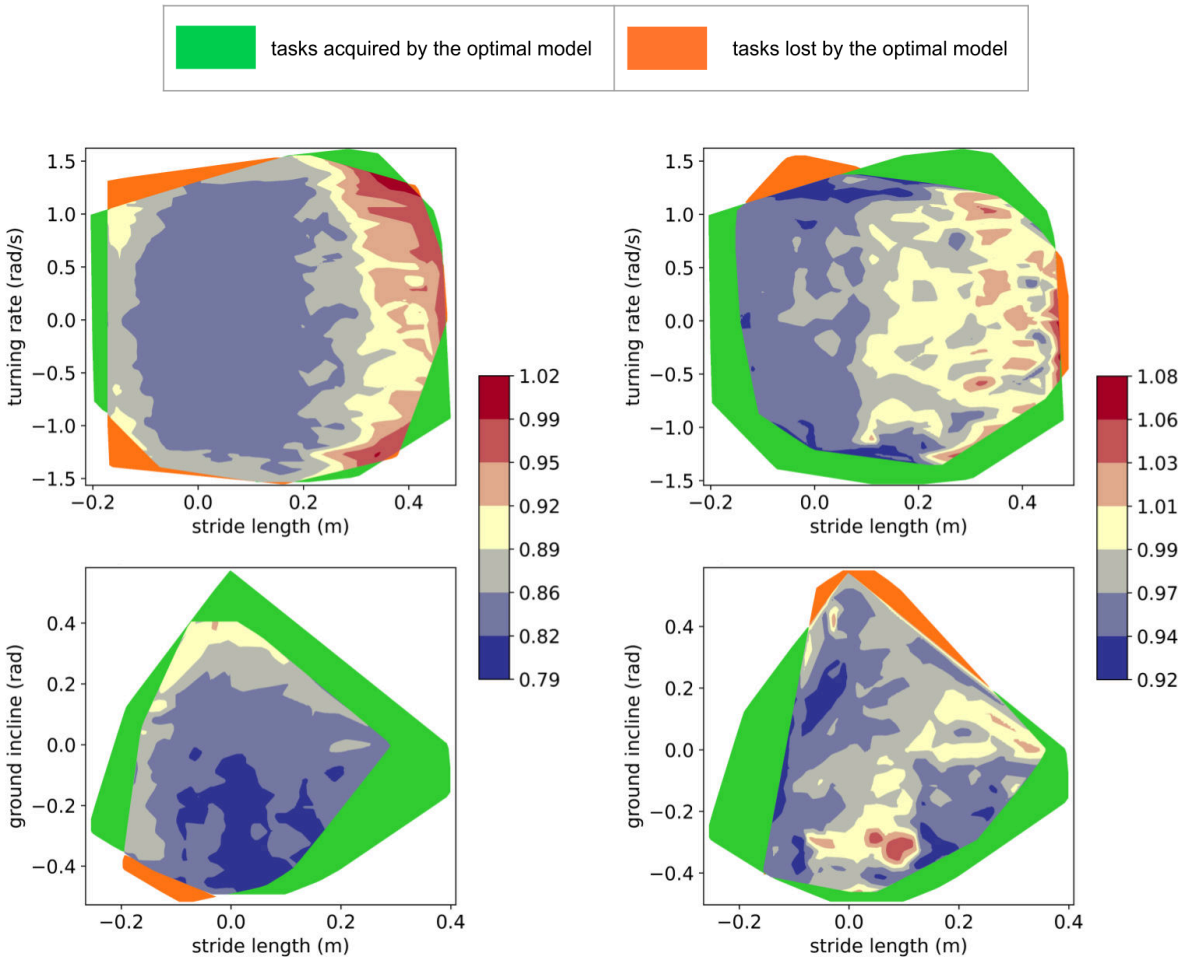
The algorithm for solving Eq. (6.7) with curriculum learning is outlined in Algorithm 4. In every iteration, CMA-ES samples a few model parameters θ's according to p_θ, and evaluates the value (return $\hat{R}$ in this case) for each sampled θ. Given these values, CMA-ES updates the mean and the covariance matrix of the parameter distribution p_θ.

6.3. Experimental Result

We learn a ROM in simulation using Drake (Tedrake, 2019) offline, and deploy it to the same simulation environment for detailed evaluation comparison between the initial model and the optimal model. We also compare this optimal model to the one derived using the prior approach (Chen et al., 2023a). Lastly, we showcase the flexibility in task space reparameterization in Section 6.3.4.

6.3.1. Hyperparameters

The hyperparameters for the RL are shown in Table 6.1. The number of parameters sampled per iteration N_θ is adaptively chosen by the CMA-ES optimizer according to the parameter dimension. The sample density ρ_γ is chosen to be 0.1 which has the benefit of speeding up the learning process compared to evaluating all discretized task samples (i.e. $\rho_\gamma = 1$). The initial standard deviation σ_0 of the parameters is set to a relatively small number, since the initial ROM already works for simple tasks. Moreover, we observe that using a larger standard deviation σ_0 does not show better performance in our experiments, and it has a drawback of potentially diverging from the optimal parameters from time to time. For the curriculum learning, we set the initial task space to be straight-line walking with stride lengths between -0.1 m and 0.2 m. The task space is discretized by 0.1 m, 0.1 rad and 0.45 rad/s in stride length, ground incline and turning rate, respectively. Lastly, we only expand the task space every 30 iterations to learn a model gradually.



(a) Comparison against LIP (initial model)

(b) Comparison against the optimal model of Chapter 5

Figure 6.2: Cost landscape comparisons. Fig. 6.2a compares the optimal model to the initial model. Fig. 6.2b compares the optimal models between this chapter and Chapter 5. For each model comparison, we create landscapes in two sets of experiments – one involving turning rate and stride length (with a constant ground incline of 0 rad), and the other involving ground incline and stride length (with a constant turning rate of 0 rad/s). Each plot shows the ratio of the optimal model's cost to the compared model's cost. The color scheme red-to-blue illustrates the degree to which the optimal ROM shows improvement. Ratio below 1 means the optimal ROM performs better than the compared model, and vice versa. Green color shows the task regions gained by the optimal model, and orange color shows the task regions lost by the optimal model.

6.3.2. Comparing the Optimal Model to LIP (Initial Model)

We compare the performance of the initial model θ_0 and the optimal model from the learning result. To evaluate the model performance, we run the simulation for a wide range of stride lengths, ground inclines and turning rates. We then extract the periodic walking gaits according to the

criteria shown in Table 5.4 which ensure variations over foot steps fall below specific thresholds. Given these periodic trajectories, we compute the cost $H_{\gamma,u(\theta)}$ and plot the landscape of the cost ratio of the optimal model to the initial model, shown in Fig. 6.2a. The color scheme red-to-blue illustrates the degree to which the ROM shows improvement, with red corresponding to a minimal improvement and blue to a 20% cost reduction. In addition to the ratio, we also visualize the task region where either the initial model or the optimal model fails to complete, visualized by green and orange. Green corresponds to the task regions that the optimal model gains, and orange corresponds to the regions that the optimal model loses.

In this example, the optimal ROM reduces the cost across almost the entire task space (up to 21% cost reduction). For flat ground walking tasks, the optimal model shows the largest improvement in the region of small stride lengths. For inclined walking, the optimal model achieves higher performance improvement in downhill tasks than uphill tasks. Additionally, the optimal model increases the task region size by 49% for inclined walking. For example, at an incline of -0.2 rad, the maximum stride length increases from 0.2 m to 0.37m.

6.3.3. Comparison against Chapter 5

We also conducted a set of experiments similar to Section 6.3.2. Instead of comparing against the initial model, we compare the optimal model from Algorithm 4 to the one from Algorithm 2. The cost landscape comparison for this is shown in Fig. 6.2b.

Compared to the prior approach, the optimal model of this chapter shows up to 28% increase in task space along with an average 2.7% cost improvement. For the flat ground walking tasks, the model yields a 22% larger viable task region than that of the prior approach. For the inclined walking tasks, it is 28% larger. We hypothesize that this improvement comes from the fact that in the RL framework we are able to use a Cassie model of high fidelity during training, while the prior approach was limited to a simplified Cassie model for ease of solving the inner-level optimization in Eq. (6.2).

We note that, besides the above numerical comparisons, the RL approach in this chapter is easier

initial standard deviation σ_0	1e-3
number of sampled parameters per iteration N_θ	17
sample density ρ_γ	0.1
number of iterations for task space expansion N_c	30
stride length discretization	0.1 m
turning rate discretization	0.45 rad/s
ground incline discretization	0.1 rad

Table 6.1: Hyperparameters for the model learning

to implement than that in Chapter 5, since it does not require a careful implementation of offline trajectory optimization (inner-level optimization in Eq. (6.2)) in conjunction with the online control policy ($u_o(\theta)$ in Section 6.1) for reducing the gap between the open-loop and closed-loop performances.

6.3.4. Generalization to New Task Space Parameterization

One advantage of using a model-based approach is the flexibility in specifying new tasks, as motivated in Section 3.3. During the training stage of our experiment shown above, we train the model using common tasks including stride length, turning rate and ground incline. We demonstrate that the model can be easily extended to achieve unseen tasks by modifying the MPC diagram in Fig. 6.1. For example, we might want to ensure that a body-mounted sensor is oriented at a target of interest, or we might want a robot to collaboratively carry a table with a human, which requires the robot facing a different direction than the walking direction. To mimic these scenarios for Cassie, we turn Cassie's pelvis to the side while walking forward. We achieve this by simply changing the desired value of the pelvis yaw angle for the regularization trajectory generator in Fig. 6.1. In the accompanying video, we can see that the robot achieves this task without any more offline training.

6.4. Conclusion and Future Work

We formulate a model-based reinforcement learning problem for reduced-order models of legged locomotion. This provides an avenue to bridge the gap between the well-established model-based control and the emerging field of reinforcement learning for legged robots, combining the performance-maximizing capability of RL with the physical interpretability and the task specification flexibility

of model-based approaches. The experiments show that the optimal model reduces the torque cost by up to 21% and improves the viable task region size by up to 49% over the traditional models like LIP. We also compare this work to Chapter 5 which uses full model trajectory optimization during the ROM optimization, and the results show an up to 28% improvement in the viable task region size along with a mild improvement in torque cost.

In this chapter, we solve the RL problem using CMA-ES, but theoretically any reinforcement learning optimizer can be applied to our RL framework in Fig. 6.1, since our policy (ROM planner) is differentiable (Amos et al., 2018). Future work includes exploring more optimizers to find one that results in the highest performance improvement. Additionally, we observed in Chapter 5 that parametrizing the embedding function r along with the ROM dynamics function g (i.e. both Eq. (2.5a) and (2.5b)) lead to higher performance improvement than parameterizing only g. In this chapter, we simplified the problem and the RL formulation by limiting r to a CoM kinematic function, so one future direction is to parameterize both r and g, and learn the model in a similar pipeline.

CHAPTER 7

CONCLUSION

In this thesis, we formulate problems to find optimal reduced-order models, provide algorithms to solve these model optimization problems, and deploy the optimal models to the robots.

In Chapter 4, we look at a specific case of ROM optimization where our goal is to find an orientation coordinate that minimizes the error of the centroidal angular momentum predicted by this coordinate. We found the prediction error was low despite the non-integrability of the angular momentum, and we applied this coordinate in humanoid walking and biped running, resulting in a smaller momentum variation throughout the motions.

In Chapter 5, we formulate the problem such that we can find a model that is optimal with respect to any objective function and task distribution. We optimized models for the bipedal robot Cassie and saw up to 23%, 32% and 53% improvements in torque cost, ground incline and walking speed in simulation, and saw on average a 10% cost improvement on hardware. The model optimization results showed a bigger benefit in hard tasks (or a big set of tasks) and in robots with heavy legs.

In Chapter 6, we improve upon Chapter 5 to further include the feedback controller into the model optimization, which eases the implementation process and reduces the performance gap between the training and model deployment. The results showed an improvement over the approach in Chapter 5 with a bigger viable task region and a lower torque cost.

This work introduces a avenue for advancing the control of legged locomotion, providing a framework for optimizing ROMs and ultimately improving the performance of legged robots across diverse tasks and scenarios. As the framework is general, it has implications that extend beyond the field of legged locomotion and into the broader realm of robotics.

7.1. Future work

Our definition of a ROM in this thesis considers only the continuous time dynamics without the discrete mapping from the ground impact event. This implicitly assumes zero ground impacts at the contact event. This assumption simplifies the design of the MPC, but also at the same time restricts the scope of the explored ROM, resulting in a potential lost in performance. One future direction is to find a high-performance hybrid ROM which captures both the continuous time and discrete time dynamics of the robot.

In some of our experiments, we found that linear models perform surprisingly well in specific sets of tasks. A linear model renders a convex MPC which can be solved with a guaranteed global optimality and demands significantly less computational resources than a nonlinear MPC. This is potentially useful for the ongoing humanoid development, particularly when there is limited onboard computing power or a thermal restriction for the computing unit. Exploring the space of linear ROMs under different sets of tasks space (including more dynamical motions) is another interesting direction.

In this thesis, we improve the robustness of ROMs with heuristics such as restricting the center of pressure near the center of the foot. There are principled ways to design a robust policy such as using offline robust trajectory optimization (Dai and Tedrake, 2012b; Kong et al., 2021), sums of squares (Posa et al., 2017b), and domain randomization (Xie et al., 2020). Additionally, we can apply existing model-based methods that provide stability or safety guarantees (Ames et al., 2019; Nguyen et al., 2016; Khazoom et al., 2022) to the high-performance model derived from our model optimization algorithm. These guarantees can facilitate deploying robots into the real world.

BIBLIOGRAPHY

2022 icra legged robots wokshop presentation by robin deits. https://youtu.be/yagQG_b_hfs. Accessed: 2022-10-24.

Nadia humanoid. https://www.ihmc.us/nadia-humanoid/. Accessed: 2022-09-05.

Boston dynamics petman prototype. https://www.youtube.com/watch?v=67CUudkjEG4. Accessed: 2023-09-04.

Evan Ackerman. Boston dynamics' cheetah robot now faster than fastest human. *IEEE Spectrum (Automation Blog).*, 2012.

Evan Ackerman. Durus: Sri's ultra-efficient walking humanoid robot. *IEEE Spectrum (Automation Blog).*, 2015.

Evan Ackerman. Agility robotics introduces cassie, a dynamic and talented robot delivery ostrich. *IEEE Spectrum (Automation Blog).*, 2017.

SN Afriat. Theory of maxima and the method of lagrange. *SIAM Journal on Applied Mathematics*, 20(3):343–357, 1971.

Agility Robotics. Digit. https://agilityrobotics.com/news/2022/future-robotics-l3mjh. Accessed: 2023-08-27.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018.

Ryan Batke, Fangzhou Yu, Jeremy Dao, Jonathan Hurst, Ross L Hatton, Alan Fern, and Kevin Green. Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 714–721. IEEE, 2022.

John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

John T Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Advances in Design and Control. Society for Industrial and Applied Mathematics, 2001.

Gerardo Bledt, Patrick M Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4102–4109. IEEE, 2017.

Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

Reinhard Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22 (11-12):1217–1227, 1989.

Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

Boston Dynamics. Atlas. https://bostondynamics.com/atlas/. Accessed: 2023-08-27.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 9780521833783.

Jerome Bracken and James T McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

Katie Byl and Russ Tedrake. Metastable Walking Machines. *The International Journal of Robotics Research*, 28(8):1040–1064, aug 2009. ISSN 0278-3649. doi: 10.1177/0278364909340446. URL http://journals.sagepub.com/doi/10.1177/0278364909340446.

Yu-Ming Chen and Michael Posa. Optimal reduced-order modeling of bipedal locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8753–8760. IEEE, 2020.

Yu-Ming Chen, Jianshu Hu, and Michael Posa. Beyond inverted pendulums: Task-optimal simple models of legged locomotion. *arXiv preprint arXiv:2301.02075*, 2023a.

Yu-Ming Chen, Gabriel Nelson, Robert Griffin, Michael Posa, and Jerry Pratt. Integrable whole-body orientation coordinates for legged robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023b.

Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2021.

Devin Crowley, Jeremy Dao, Helei Duan, Kevin Green, Jonathan Hurst, and Alan Fern. Optimizing bipedal locomotion for the 100m dash with comparison to human running. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12205–12211. IEEE, 2023.

Hongkai Dai and Russ Tedrake. Optimizing Robust Limit Cycles for Legged Locomotion on Unknown Terrain. In *Proceedings of the IEEE Conference on Decision and Control*, page 8, Maui, Hawaii, dec 2012a.

Hongkai Dai and Russ Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1207–1213. IEEE, 2012b.

Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body Motion Planning with Centroidal Dynamics and Full Kinematics. *IEEE-RAS International Conference on Humanoid Robots*, 2014.

Jeremy Dao, Kevin Green, Helei Duan, Alan Fern, and Jonathan Hurst. Sim-to-real learning for bipedal locomotion under unsensed dynamic loads. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10449–10455. IEEE, 2022.

Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-based inverse reinforcement learning from visual demonstrations. In *Conference on Robot Learning*, pages 1930–1942. PMLR, 2021.

Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

Wenqian Du, Ze Wang, Etienne Moullet, and Faïz Benamar. Meaningful centroidal frame orientation of multi-body floating locomotion systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3061–3067. IEEE, 2021.

Tom Erez and Emanuel Todorov. Trajectory optimization for domains with contacts using inverse dynamics. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4914–4919. IEEE, 2012.

Salman Faraji and Auke J Ijspeert. 3lp: A linear 3d-walking model including torso and swing dynamics. *the international journal of robotics research*, 36(4):436–455, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2018.

R J Full and D E Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *The Journal of Experimental Biology*, 202:3325–3332, 1999. URL http://jeb.biologists.org/content/jexbio/202/23/3325.full.pdf.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

Mariano Garcia, Anindya Chatterjee, Andy Ruina, and Michael Coleman. The Simplest Walking Model: Stability, Complexity, and Scaling. *Journal of Biomechanical Engineering*, 120(2):281, apr 1998. ISSN 0148-0731. doi: 10.1115/1.2798313. URL http://biomechanical.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.2798313.

Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6724–6731. IEEE, 2022.

Philip E Gill, Walter Murray, and Margaret H Wright. Practical optimization (book). *London and New York, Academic Press, 1981. 415 p*, 1981.

Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

Yukai Gong and Jessy W Grizzle. Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion. *Journal of Dynamic Systems, Measurement, and Control*, 144(12):121006, 2022.

Jessy W Grizzle, Christine Chevallereau, Ryan W Sinnet, and Aaron D Ames. Models, feedback control, and open problems of 3d bipedal robotic walking. *Automatica*, 50(8):1955–1988, 2014.

Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102, 2006.

Ross L Hatton and Howie Choset. Geometric motion planning: The local connection, stokes' theorem, and the importance of coordinate choice. *The International Journal of Robotics Research*, 30(8):988–1014, 2011.

Ross L Hatton and Howie Choset. Nonconservativity and noncommutativity in locomotion: geometric mechanics in minimum-perturbation coordinates. *The European Physical Journal Special Topics*, 224(17-18):3141–3174, 2015.

Kathrin Hatz, Johannes P Schloder, and Hans Georg Bock. Estimating parameters in optimal control problems. *SIAM Journal on Scientific Computing*, 34(3):A1707–A1728, 2012.

Ayonga Hereid and Aaron D Ames. Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726. IEEE, 2017.

Ayonga Hereid, Omar Harib, Ross Hartley, Yukai Gong, and Jessy W Grizzle. Rapid bipedal gait

design using c-frost with illustration on a cassie-series robot. *arXiv preprint arXiv:1807.06614*, 2018.

Alexander Herzog, Stefan Schaal, and Ludovic Righetti. Structured contact force optimization for kino-dynamic motion generation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2703–2710. IEEE, 2016.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062. IEEE, 2010.

Yildirim Hurmuzlu and Dan B Marghitu. Rigid body collisions of planar kinematic chains with multiple contact points. *The international journal of robotics research*, 13(1):82–92, 1994.

Marco Hutter, Mark A Hoepflinger, Christian Gehring, Michael Bloesch, C David Remy, and Roland Siegwart. Hybrid operational space control for compliant legged systems. *Robotics*, page 129, 2013.

Amir Iqbal, Sushant Veer, and Yan Gu. Drs-lip: Linear inverted pendulum model for legged locomotion on dynamic rigid surfaces. *arXiv preprint arXiv:2202.00151*, 2022.

Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems*, 33:7979–7992, 2020.

Wanxin Jin, Shaoshuai Mou, and George J Pappas. Safe pontryagin differentiable programming. *Advances in Neural Information Processing Systems*, 34:16034–16050, 2021.

Wanxin Jin, Todd D Murphey, Dana Kulić, Neta Ezer, and Shaoshuai Mou. Learning from sparse demonstrations. *IEEE Transactions on Robotics*, 2022.

S Kajita and K Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. volume 2, pages 1405–1411. IEEE International Conference on Robotics and Automation (ICRA), 1991.

Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. pages 239–246. IEEE International Conference on Intelligent Robots and Systems (IROS), 2001.

Mohammadreza Kasaei, Ali Ahmadi, Nuno Lau, and Artur Pereira. A robust model-based biped locomotion framework based on three-mass model: From planning to control. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 257–

262. IEEE, 2020.

S Mohammadreza Kasaei, Nuno Lau, and Artur Pereira. A reliable hierarchical omnidirectional walking engine for a bipedal robot by using the enhanced lip plus flywheel. In *Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 399–406. World Scientific, 2018.

Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

Matthew Kelly and Andy Ruina. Non-linear robust control for inverted-pendulum 2D walking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4353–4358. IEEE, may 2015. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139800. URL http://ieeexplore.ieee.org/document/7139800/.

Charles Khazoom, Daniel Gonzalez-Diaz, Yanran Ding, and Sangbae Kim. Humanoid self-collision avoidance using whole-body control with control barrier functions. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 558–565. IEEE, 2022.

Charles Khazoom, Steve Heim, Daniel Gonzalez-Diaz, and Sangbae Kim. Optimal scheduling of models and horizons for model hierarchy predictive control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9952–9958. IEEE, 2023.

Alex Khripin and Alfred Anthony Rizzi. Natural pitch and roll, December 13 2016. US Patent 9,517,561.

Nathan J Kong, George Council, and Aaron M Johnson. ilqr for piecewise-smooth hybrid dynamical systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 5374–5381. IEEE, 2021.

Twan Koolen, Tomas de Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113, 2012a.

Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113, 2012b.

Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentum-based control framework and application to the humanoid robot Atlas. *International Journal of Humanoid Robotics*, 13(1), 2016a.

Twan Koolen, Michael Posa, and Russ Tedrake. Balance control using center of mass height variation: limitations imposed by unilateral contact. In *Humanoid Robots (Humanoids), 2016 IEEE-*

RAS 16th International Conference on, pages 8–15. IEEE, 2016b.

Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200500249.

Scott Kuindersma, Robin Deits, Maurice Fallon, Andres Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for Atlas. *Autonomous Robots*, 40(3):429–455, 2016. ISSN 15737527. doi: 10.1007/s10514-015-9479-3.

He Li, Robert J Frei, and Patrick M Wensing. Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, 6(2):3373–3380, 2021.

Yuntao Ma, Farbod Farshidian, and Marco Hutter. Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12149–12155. IEEE, 2023.

Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

Pat Marion and the team. Flipping the script with atlas. https://blog.bostondynamics.com/flipping-the-script-with-atlas.

Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

Akinori Miyata, Sho Miyahara, and Dragomir N Nenchev. Walking with arm swinging and pelvis rotation generated with the relative angular acceleration. *IEEE Robotics and Automation Letters*, 5(1):151–158, 2019.

Yoshihiko Nakamura and Ranjan Mukherjee. Nonholonomic path planning of space robots via bi-directional approach. In *Proceedings., IEEE International Conference on Robotics and Automation (ICRA)*, pages 1764–1769. IEEE, 1990.

Quan Nguyen, Ayonga Hereid, Jessy W Grizzle, Aaron D Ames, and Koushil Sreenath. 3d dynamic walking on stepping stones with control barrier functions. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 827–834. IEEE, 2016.

Joseph Norby, Ardalan Tajbakhsh, Yanhao Yang, and Aaron M Johnson. Adaptive complexity model predictive control. *arXiv preprint arXiv:2209.02849*, 2022.

Cenk Oguz Saglam and Katie Byl. Robust Policies via Meshing for Metastable Rough Terrain Walking. In *Robotics: Science and Systems*, 2014. URL http://roboticsproceedings.org/rss10/p49.pdf.

David E Orin and Ambarish Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 653–659. IEEE, 2008.

Abhishek Pandala, Randall T Fawcett, Ugo Rosolia, Aaron D Ames, and Kaveh Akbari Hamed. Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced- and full-order models. *IEEE Robotics and Automation Letters*, 7(3):6622–6629, 2022.

Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

G Pequera, V Yelós, and CM Biancardi. Reducing cost of transport in asymmetrical gaits: lessons from unilateral skipping. *European Journal of Applied Physiology*, 123(3):623–631, 2023.

Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, pages 2279–2291. PMLR, 2021.

E Philip, Walter Murray, and Michael A Saunders. User's guide for snopt version 7: Software for large-scale nonlinear programming, 2015.

Marko B Popovic, Ambarish Goswami, and Hugh Herr. Ground reference points in legged locomotion: Definitions, biological trajectories and control implications. *The International Journal of Robotics Research*, 24(12):1013–1032, 2005.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact. *The International Journal of Robotics Research*, 33(1):69–81, jan 2013. ISSN 0278-3649. doi: 10.1177/0278364913506757. URL http://ijr.sagepub.com/content/33/1/69.short.

Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 1366–1373, Stockholm, Sweden, may 2016. ISBN 9781467380263. doi: 10.1109/ICRA.2016.7487270.

Michael Posa, Twan Koolen, and Russ Tedrake. Balancing and Step Recovery Capturability via Sums-of-Squares Optimization. In *Robotics: Science and Systems*, 2017a.

Michael Antonio Posa, Twan Koolen, and Russell L Tedrake. Balancing and step recovery capturability via sums-of-squares optimization. 2017b.

Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006.

Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

Marc H. Raibert, H. Benjamin Brown, and Michael Chepponis. Experiments in Balance with a 3D One-Legged Hopping Machine. *The International Journal of Robotics Research*, 3(2):75–92, jun 1984. ISSN 0278-3649. doi: 10.1177/027836498400300207. URL http://journals.sagepub.com/doi/10.1177/027836498400300207.

Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. In *International conference on machine learning*, pages 7953–7963. PMLR, 2020.

Jacob Reher, Eric A Cousineau, Ayonga Hereid, Christian M Hubicki, and Aaron D Ames. Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1794–1801. IEEE, 2016.

Jacob Reher, Wen-Loong Ma, and Aaron D Ames. Dynamic walking with compliance on a cassie bipedal robot. In *2019 18th European Control Conference (ECC)*, pages 2589–2595. IEEE, 2019.

John G Riley. *Essential microeconomics*. Cambridge University Press, 2012.

Boardwalk Robotics. Nadia humanoid. https://www.ihmc.us/nadia-humanoid/. Accessed: 2022-09-05.

Alessandro Saccon, Silvio Traversaro, Francesco Nori, and Henk Nijmeijer. On centroidal dynamics and integrability of average angular velocity. *IEEE Robotics and Automation Letters*, 2(2):943–950, 2017.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Tomoya Sato, Sho Sakaino, and Kouhei Ohnishi. Real-time walking trajectory generation method with three-mass models at constant body height for three-dimensional biped robots. *IEEE transactions on industrial electronics*, 58(2):376–383, 2010.

A L Schwab and M Wisse. Basin of attraction of the simplest walking model. In *ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2001. URL http://bicycle.tudelft.nl/schwab/Publications/vib21363.pdf.

Luis Sentis and Oussama Khatib. Control of free-floating humanoid robots through task prioritization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1718–1723. IEEE, 2005.

Tim Seyde, Apoorv Shrivastava, Johannes Englsberger, Sylvain Bertrand, Jerry Pratt, and Robert J Griffin. Inclusion of angular momentum during planning for capture point based walking. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1791–1798. IEEE, 2018.

AJ Shaiju and Ian R Petersen. Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems. *IFAC Proceedings Volumes*, 41(2):8773–8778, 2008.

Chenggen Shi, Jie Lu, and Guangquan Zhang. An extended kuhn–tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, 162(1):51–63, 2005.

Shuhei Shimmyo, Tomoya Sato, and Kouhei Ohnishi. Biped walking pattern generation by using preview control based on three-mass model. *IEEE transactions on industrial electronics*, 60(11): 5137–5147, 2012.

Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv:2105.08328*, 2021.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

Akira Takayama and Takayama Akira. *Mathematical economics*. Cambridge university press, 1985.

Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091. IEEE, 2009.

Russ Tedrake. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

Matthew Travers, Ross Hatton, and Howie Choset. Minimum perturbation coordinates on so (3). In *2013 American Control Conference*, pages 2006–2012. IEEE, 2013.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

Patrick M Wensing and David E Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *2013 IEEE International Conference on Robotics and Automation*, pages 3103–3109. IEEE, 2013.

Patrick M Wensing and David E Orin. Improved computation of the humanoid centroidal dynamics and application for whole-body control. *International Journal of Humanoid Robotics*, 13(01): 1550039, 2016.

Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *arXiv preprint arXiv:2211.11644*, 2022.

Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE transactions on automatic control*, 48(1):42–56, 2003.

Bong Wie and Peter M Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pages 317–329. PMLR, 2020.

Xiaobin Xiong and Aaron Ames. 3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization. *IEEE Transactions on Robotics*, 38(4):2405–2425, 2022.

Xiaobin Xiong and Aaron D Ames. Dynamic and versatile humanoid walking via embedding 3d actuated slip model with hybrid lip based stepping. *IEEE Robotics and Automation Letters*, 5 (4):6286–6293, 2020.

Xiaobin Xiong, Jenna Reher, and Aaron D Ames. Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2825–2831. IEEE, 2021.

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

Huihua Zhao, Ayonga Hereid, Wen-loong Ma, and Aaron D Ames. Multi-contact bipedal robotic locomotion. *Robotica*, 35(5):1072–1106, 2017.

James Zhu, Nathan J Kong, George Council, and Aaron M Johnson. Hybrid event shaping to stabilize periodic hybrid orbits. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 01–07. IEEE, 2022.

Victor Zordan, David Brown, Adriano Macchietto, and KangKang Yin. Control of rotational dynamics for ground and aerial behavior. *IEEE Transactions on Visualization and Computer Graphics*, 20(10):1356–1366, 2014.