Creating ARCHER: A 3D Hopping Robot with Flywheels for Attitude Control

Thesis by
Eric R. Ambrose

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy in the Department of Mechanical Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended May 17th, 2022

Eric R. Ambrose
ORCID: 0000-0001-9433-5650

ACKNOWLEDGEMENTS

I chose to enter grad school because I wanted to continue learning about the application of robotics to the world of prostheses. After all these years, I've learned so much more than just biomechanics and basic robotics. I want to thank Dr. Ames for providing me the opportunity and funding to work on so many different robots during my time in the AMBER Lab, from hopping robots, to humanoids, and of course prosthetic legs. I also had the unique chance to create multiple iterations of the same platform in both the AMBER humanoid and the AMPRO prosthetic leg. Being able to spend time designing and testing a robot, learn from that process, and then get to immediately redesign that robot was an invaluable experience that I will benefit from forever.

Beyond working on the robots themselves, I had the great pleasure of getting to work alongside so many other wonderful people in the AMBER Lab. At the start of grad school, I was able to work with and learn from the veteran members of the lab like Huihua and Shishir, who both helped introduce me to all the facets of robotics that I had yet to delve into. And thank you to Wen-Loong, Jenna, Tom, Maegan, Noel, Drew, Christian, and Rachel for working with me and helping to keep me sane during this extended journey that has been grad school. I couldn't be any more grateful for what I was able to learn from each of you and all our random casual conversations over the years. Traveling to so many different places during my stay in the AMBER Lab has allowed me to learn what is most important in grad school. While I enjoyed the passionate energy of the student body at Texas A&M, the wide breadth of robotics research at Georgia Tech, and the culture of research and peacefulness at Caltech, the ability to enjoy spending time with the other members of the lab was the critical piece.

My time in this lab has not only prepared me for what comes next, but put me in the perfect position to start my career at the Jet Propulsion Lab here in Pasadena. This next chapter of life is going to be so much fun, getting to work on even more unique robotic systems, in even more challenging environments. I can't wait!

ABSTRACT

The field of robotic hopping began over 40 years ago, when it was first shown that robust hopping could be achieved on real hardware. In the years since then, it's become clear that hopping requires high performance and precision from its actuation and planning, due to its extreme interactions with the environment occurring over periodic, yet very short durations of time. Despite being of lower dimensionality than many other legged robots, hoppers are very underactuated, which only adds to the difficulty of planning motions quickly for real-time needs.

The studies of robotic hopping presented in this thesis start with a look into two different actuation styles for creating vertical periodic motion: a compress-release mechanism and a moving-mass mechanism. The dynamics of each were examined from the perspective of stability and robustness to uncertainties in the model and measurements. The compress-release hopper (CRH) was found to be very stable, simple to control, and robust to all uncertainties, but inherently had some inefficiencies due to the requirement of holding compression during portions of the aerial phase. The moving-mass hopper (MMH) required optimization to generate the proper cyclic motions as well as closed-loop control to make them stable. Furthermore, the original configuration of the MMH was also less energetically efficient and robust to uncertainty than the CRH.

In an effort to improve the efficiency of the MMH, a second-generation robot was designed using the principle of parallel elasticity. This involved placing a second spring in parallel to the actuator which would naturally guide the motion of the moving-mass into an optimal path, eliminating a significant portion of actuation effort and improving the overall efficiency. An added benefit of this change was that the robot no longer required closed loop control to create stable hopping. This new robot was built and tested in the lab showing a dramatic improvement over the previous design. The principle of controlling the compliance in the actuator for efficient motion was then taken one step further by creating custom, nonlinear stiffness springs which would provide a more ideal trajectory of motion. This process utilized a design-in-the-loop optimization strategy that would both design these springs as well as the motions of the moving-mass to yield better actuation efficiency. A set of these springs was created and attached to the second-gen MMH, replacing the lower spring, and tested in the lab. These springs did slightly improve the efficiency of the robot, but were restricted by the material selection of the springs

due to manufacturing limitations.

Moving into the realm of 3-Dimensional hopping, a final robot was designed and built: ARCHER. Unlike traditional hopping robots which use a torso with very large inertia to control the leg motion and balance, ARCHER uses a set of three flywheels. The goal of this robot was twofold: to study the feasibility of using flywheels alone to control attitude, and to take advantage of the principle of decoupled systems. By using strategically placed flywheels, the dynamics of the leg and the attitude subsystems were decoupled, meaning their actuation did not have a direct influence on each other. This allows for simpler motion planning and control. The culmination of this thesis was running experiments with this robot, showing its initial performance and ability to hop with separate controllers for each subsystem.

PUBLISHED CONTENT AND CONTRIBUTIONS

Ambrose, Eric, Wen-Loong Ma, and Aaron D Ames (2021). "Towards the Unification of System Design and Motion Synthesis for High-Performance Hopping Robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7073–7078. DOI: 10.1109/ICRA48506.2021.9561322.
E.R.A created the full hardware and software of this robot used in the work, ran all the experiments, performed some analysis of its performance, and led the authorship of the paper.

Ambrose, E. and A. D. Ames (2020). "Improved Performance on Moving-Mass Hopping Robots with Parallel Elasticity". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2457–2463. DOI: 10.1109/ICRA40945.2020.9197070.
E.R.A created the full hardware and software of this robot used in the work, ran all the experiments, performed all analysis of its performance, and led the authorship of the paper.

Ambrose, E., N. Csomay-Shanklin, et al. (2019). "Design and Comparative Analysis of 1D Hopping Robots". In: *Intelligent Robots and systems (IROS), IEEE International Conference on*. DOI: 10.1109/IROS40897.2019.8967692.
E.R.A created the full hardware and software of this robot used in the experiments of this work, ran all the experiments, led the analysis and the authorship of the paper.

Ambrose, Eric, Wen-Loong Ma, Christian Hubicki, et al. (2017). "Toward benchmarking locomotion economy across design configurations on the modular robot: AMBER-3M". In: *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, pp. 1270–1276. DOI: 10.1109/CCTA.2017.8062633.
E.R.A led the mechanical design, assembly, and testing of the robot, helped to run the experiments involved in this work, and led authorship of the paper.

Azimi, Vahid et al. (2017). "Robust control of a powered transfemoral prosthesis device with experimental verification". In: *American Control Conference (ACC), 2017*. IEEE, pp. 517–522. DOI: 10.23919/ACC.2017.7963005.
E.R.A led the mechanical design and assembly of the robot which was used in the work and help with the initial testing that led to its ability to perform the experiments in this paper.

Ma, W. et al. (2017). "Bipedal Robotic Running with DURUS-2D: Bridging the Gap between Theory and Experiment". In: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, pp. 265–274. DOI: 10.1145/3049797.3049823.
E.R.A created the experimental infrastructure used to test this robot and perform the experiments, designed the pair of running legs which were required to reach this level of performance, and helped author the paper.

Zhao, Huihua, Eric Ambrose, and Aaron D. Ames (2017). "Preliminary results on energy efficient 3D prosthetic walking with a powered compliant transfemoral prosthesis". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1140–1147. DOI: 10.1109/ICRA.2017.7989136.
E.R.A led the mechanical design and assembly of the robot, helped to test and run the experiments involved in this work, and contributed in the authorship of the paper.

Powell, Matthew J et al. (2016). "Mechanics-based design of underactuated robotic walking gaits: Initial experimental realization". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots*. IEEE, pp. 981–986. DOI: 10.1109/HUMANOIDS.2016.7803391.
E.R.A created the experimental infrastructure used to test this robot and perform the experiments, and helped author the paper.

Zhao, Huihua, Ayonga Hereid, et al. (2016). "3D multi-contact gait design for prostheses: Hybrid system models, virtual constraints and two-step direct collocation". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, pp. 3668–3674. DOI: 10.1109/CDC.2016.7798821.
E.R.A led the mechanical design and assembly of the robot which was used as the base model for the robot used in this work, as well as participated in the authorship of the paper.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

Robotics is now a booming industry, stretching over a huge range of fields from industrial robots assembling vehicles (Tran and Ha, 2018) or moving products around a warehouse (Ackerman, 2022), all the way to the medical field where robots are used to sort medications (Hoffmann et al., 2016) and perform surgeries (C. Yang et al., 2019). Creating robots has become achievable by hobbyists, where small robotic components like actuators, sensors, and microprocessors are becoming cheaper and better than ever before.

The fields of robotic hopping and running alone have been around for decades (M. H. Raibert, 1984), studying the ways to create stability in very underactuated situations. These robots undergo rapid changes of contact with the world, resulting in extreme impacts and changes in energy level (M. H. Raibert, H. B. Brown, and Chepponis, 1984). Despite this, these systems have been able to achieve incredible behaviors beyond just hopping in place, such as reaching high speeds and heights (Salton et al., 2010), performing flips (M. Raibert, 1986), and jumping off of walls and over uncertain terrain (Haldane, Yim, and Fearing, 2017; Fiorini and J. Burdick, 2003; C. M. Hubicki et al., 2016). Underlying all of these behaviors is the need to find a balance between planning out upcoming motions and being able to overcome uncertain and unforeseen events.

Finding ways to lessen the impact with the world or extend the duration of contact with the world can allow for more manageable dynamics for the robot and its environment (Yoon et al., 2005; Lens and Stryk, 2012), and more energetically efficient motions in general (Sharbafi et al., 2019). This improvement can usually be achieved through changing the motion of the robot as well as modifying the mechanical system. However, doing motion generation before a robot's design has been finalized is often tedious due to the iterative nature of design. By utilizing optimization techniques, finding more efficient motions and mechanical designs can be accomplished together before the robot has even been made.

Lastly, even in relatively low-dimensional systems like a hopping robot, planning out where to place the foot and how to get around an object can be a challenge (Chang

et al., 2019). With only a fraction of a second between each hop and the limited time in contact with the world, there is very little time to update and take action. The concept of decoupled dynamics and interconnected systems (Wen-Loong Ma, Noel Csomay-Shanklin, and Aaron D. Ames, 2020) has been shown to reduce computation time of a robot's planner or gait generation simply through separating the dynamics into carefully constructed parts. In the case of hopping robots, this can be inherently done by replacing a torso mechanism with flywheels which leads to no change in the center of mass during attitude control. This completely separates the influence of the leg compression and the attitude control, meaning they can be controlled and planned separately. The only information that needs be shared between them could be the timing of events to sync up behaviors.

1.2 Contribution

The contributions of this thesis work is threefold.

First, the designs of multiple robotic legged platforms were created: AMBER-3M (Eric Ambrose, Wen-Loong Ma, C. Hubicki, et al., 2017) and AMPRO-3/4 (Huihua Zhao, Eric Ambrose, and Aaron D. Ames, 2017). The process of designing and testing these systems led to huge improvements during their re-design. After the second version of each platform was created, their usefulness and longevity skyrocketed. Each of these devices have already been used by many other researchers for their work in the 5 years since their creation.

Secondly, throughout the original two generations of hopping robots created in the lab (E. Ambrose, N. Csomay-Shanklin, et al., 2019; E. Ambrose and A. D. Ames, 2020; Eric Ambrose, Wen-Loong Ma, and Aaron D Ames, 2021), methods of improving locomotion efficiency through design were explored. Moreover, some aspects of mechanical design were incorporated into the optimal motion generation process, leading to further reductions of cost of transport. These techniques could be used in future robotic systems, especially in the presence of compliance which is being used more and more now that robotic systems are going out into the real world and interacting with humans and other environments requiring safety. This was all shown experimentally through 3 different hopping robots discussed in Chapters 4 and 5.

Finally, the concept of decoupled dynamics was taken to heart in the design of the 3D hopper: ARCHER. By changing the morphology of this hopping robot, it is now capable of 3D motions with much simpler control schemes which would allow

for quicker planning during real-time motion. Simulations with both simple PD control and more rigorous CLF-QP control methods were run. Experiments were performed to validate the idea of decoupled control for a robot, and well as for the efficacy of a flywheel controlled hopping robot on the platform, ARCHER.

1.3 Organization of dissertation

The layout of the thesis chapters is as follows:

Chapter 2: AMBER-3M: A Modular Planar Biped Platform. The next chapter introduces the AMBER series of robots, and the two iterations that I designed: AMBER-3 and AMBER-3M. This chapter also provides my published work on using the modularity of AMBER-3M to study the locomotion economy of different legs on the robot. At the end you will find a brief listing of the other topics this platform has been used to study.

Chapter 3: AMPRO-3: A Transfemoral Prosthesis. Afterward, a similar introduction of the AMPRO series of transfemoral prosthetic legs is provided. These devices are designed to not only test out behaviors and controllers in the AMBER Lab, but can also be used to help amputees regain functional movement. The design of AMPRO-3 and AMPRO-4 are explained along with a section of published work on the design of the former. There will also be a listing of other papers that used these two devices.

Chapter 4: Analysis of Vertical Hopping Mechanisms. Here, the discussion of these robotic hopping systems begins, starting with a study of two different mechanisms for hopping. This chapter will also introduce the concept of "safe actuation" which was a critical design constraint for all of the hoppers created in this thesis work. The performance and stability of each style of vertical hopping will be detailed along with some experimental results for the Moving-Mass Hopper that was created for this work.

Chapter 5: Improving Hopping Efficiency Through Design. Moving further on the moving-mass hopping mechanism, this chapter studies how this method of hopping could be made more efficient through alteration in the mechanical design. This includes new actuation, a different compliance model, improved electrical hardware, and a design-in-the-loop optimization algorithm. There were two different versions of the same robot created and tested in this work, with experimental results for each being provided throughout the chapter.

Chapter 6: ARCHER: The 3D Hopper. Finally, a 3D hopping robot using flywheels

for attitude control was designed, simulated, and tested in the lab. This chapter runs through to design of each subsystem and explains how this style of robot benefits from decoupled dynamics and can exploit this for simpler modelling, control, and motion planning. Full-body simulations were created to design and observe a pair of controllers for this flywheel-based attitude subsystem. The control and design of the robot were then tested on hardware with the robot, ARCHER. The experimental results of these tests are shown.

Chapter 2

AMBER-3M: A MODULAR PLANAR BIPED PLATFORM

The AMBER series of bipeds were created as a humanoid platform to test controllers, behaviors, and walking gaits created in the similarly named AMBER Lab at Texas A&M University. To date, all iterations of this robot have been planar bipeds, meaning they are constrained to the Sagittal plane of motion. While this requires the robot be supported in the lateral direction, the robots are still responsible for carrying their own eight, and controlling their forward motion. This greatly simplifies the scale of robot's dynamics, allowing for a quicker path to test out new research with a higher chance of success, given the difficulty of implementing some control methods on hardware.

The first two generation of this series, AMBER-1 (Yadukumar, Pasupuleti, and Aaron D Ames, 2012) and AMBER 2 (H.-H. Zhao et al., 2014), were small scale humanoids consisting of two legs and a simple torso which was mostly just the hip joints. All of the computation and power was delivered from off-board on these robots, in order to reduce the weight being carried by the joint actuators. Experiments were run on these two robots up until around 2015, when the lab moved to Georgia Institute of Technology and they had to be left behind. In 2013 and 2014 AMBER-2 was used to experimentally realize multi-contact walking behavior, for improved walking efficiency. This behavior was an incredible push forward for the field.

However, around this same, the lab saw the need for a larger-scale humanoid that could generate higher joint torques and speeds, allowing for more strenuous behaviors like stair climbing or walking over rough terrain. This led to the design of AMBER-3, the next iteration of the AMBER series. The robot was designed over the course of late 2013 through 2014, being assembled and tested for the first time in late 2014. While the initial testing showed promise, this robot also needed to be left behind during the move to Georgia Tech. Once there, the robot was redesigned, taking the opportunity to fix some of the problems and inconveniences discovered during that brief testing of the original.

This new robot changed the placement of many electronics, allowing for more accessible control hardware. The structural pieces of the robot were lightened

to allow for less required effort by the actuators. The actuators themselves were given a slightly larger reduction, increasing their peak torque by roughly 14%. The feedback on the torso angle was also increased to improve the performance of controllers during experiments. Lastly, this robot's joint layout was modified to make the legs easily "swap-able," meaning different types of legs could be tested as well. Originally the plan for this was specifically to allow for both point feet, and fully-actuated morphologies to be tested. Later it was expanded to include compliant legs as well, but there options were endless.

This modular feature led to research on the comparison between different legs, and how they effected the performance and efficiency of legged locomotion. The following sections are taken directly from the research paper discussing this topic (Eric Ambrose, Wen-Loong Ma, C. Hubicki, et al., 2017).

2.1 Review of Locomotion Efficiency

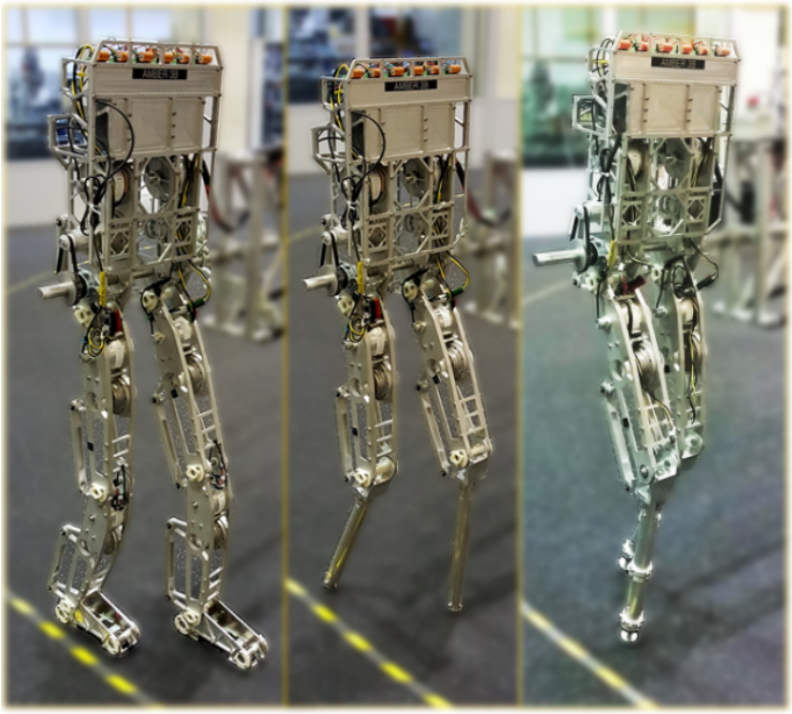


Figure 2.1: AMBER-3M: Modular Bipedal Robot. A robot designed to test energy cost across three configurations: Flat-Foot (left), Point-Foot (center), and Spring-Foot (right).

In the field of bipedal robotics, a significant importance has been placed on optimizing a robot's motion for improved locomotion economy (Collins et al., 2005), (Seok et al., 2013). In these situations, it is common to modify control methods

and joint trajectories as a means of lowering this transport cost. However, there are many factors that have been shown to contribute to a robot's energy economy, such as using different controllers (Sreenath, H. Park, Poulakakis, et al., 2011), (H. Zhao, J. Reher, et al., 2015), actuation methods (Seok et al., 2013), (Martin, Post, and J. Schmiedeler, 2014), morphologies (Sobajima et al., 2013), or behaviors (A. Kuo, 2007). When comparing energy economy across multiple morphologies and behaviors, it can be difficult to guarantee that the differences seen are only from those changes. This is especially true when separate robots are being used in the comparison, since differing hardware can inherently affect the measured economy.

This work parallels research in the field of biomechanics, which has for long studied the energy costs of animal locomotion. For decades, experiments were done suggesting that an animal's cost of transport seems to be a function of their leg length and mass (Pontzer, 2007). Beyond this, they also performed tests showing an energetically optimal speed for animals during walking (Wickler et al., 2000), (Bastien et al., 2005). Further studies compared animals executing different gaits at the same speed (Hoyt and Taylor, 1981), showing an optimal behavior at each speed, based on energy usage. Adopting a similar methodological approach could provide insight into robotic behaviors and morphologies as well.

Performing behavioral comparisons in robotics requires a machine capable of performing varied gait patterns (e.g. heel-toe vs. point-foot walking). Robots have already been used to perform various locomotion behaviors across ranges of energy cost (Bhounsule et al., 2014), speed (Sreenath, H. Park, and J. Grizzle, 2014), and terrain (Manchester et al., 2011). Despite this wider array of behaviors, the observed differences could be partly induced by changing actuation or controllers. Other groups have begun working towards using a single robot in this way through different means. One approach compares behaviors on a simplified simulation model using optimal control (Srinivasan and Ruina, 2006), (Xi, Yesilevskiy, and Remy, 2015). Another method models an existing robot shown to walk efficiently in experiments (J. Reher, E. Cousineau, et al., 2016), and systematically explores the energetics of it running in simulation (W. Ma, A. Hereid, et al., 2016). Additional work explored the energetic benefits of incorporating new modes of actuation into economical locomotion, such as a motorized reaction wheel (T. L. Brown and J. P. Schmiedeler, 2016). These experiments motivate the creation of our own platform for assessing how modulating a behavior can affect the cost of transport.

In this paper, we demonstrate the AMBER-3M platform's capability of experi-

mentally assessing the locomotion economy of varied behaviors and mechanical configurations. The following sections of the report provide a detailed description of the mechanical design and modularity of the platform, our behavior-generalized control, modeling, and gait generation procedures, the experimental setup, and comparative hardware results.

2.2 The Design of AMBER-3M

With the aim of testing a variety of bipedal robot design configurations, we developed the planar bipedal robotic platform, AMBER-3M. The following section describes the mechanical and modular design which allows for effective switching between behaviors.

Mechanical Design of AMBER3M

AMBER-3M, seen in Fig. 2.1, is a planar bipedal robot created in the AMBER Lab of the California Institute of Technology. Each actuated joint is articulated via identical drive systems, consisting of a motor, gearbox, belt, and linkage. The drive motors (MOOG BN34-25EU-02) are capable of applying 355 W of power, 2.2 Nm peak torque, and 0.66 Nm of continuous torque. The maximum motor speed is rated at 785 rad/s. The output of the motors is connected to a Harmonic gearhead (CSG-20-80-2UH-LW) through a timing belt with a 7:8 speed reduction. The Harmonic gearhead itself has a 1:80 gear reduction and an efficiency of roughly 65%. The full system, including efficiency losses, is capable of a peak torque of 130 Nm, continuous torque of 39.2 Nm, and a max speed of 8.6 rad/s. The output of the gearhead is connected to the joint by linkages with a 1:1 input to output position ratio. The presence of linkages allows for the leg segments to be any length and a variety of shapes, without changing the actuation mechanics. An exploded view of these components is shown in Fig. 2.2. Rotor position and velocity feedback is provided by incremental encoders (Renishaw RM22IC) mounted to the back of each motor.

Since AMBER-3M is a planar robot, a support structure is used to constrain it to the sagittal plane. This is accomplished through a 3.43 m radius circular boom which allows the robot to walk on an unending track, while still being fixed in the plane. The distal end of the boom connects to the robot via a rod passing through bearing mounts in the torso, as seen in Fig. 2.1. The forward progression of the boom arm and the relative angle at the torso-boom connection are also measured by encoders. The boom is counterbalanced with weights while the robot is disconnected

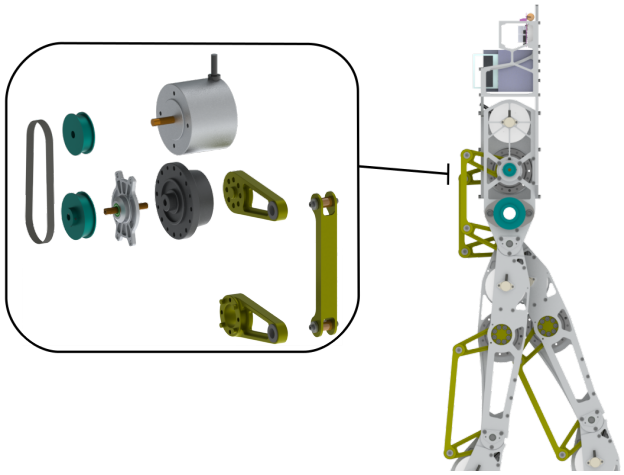


Figure 2.2: An exploded view of the drive system for each actuated joint on AMBER-3M.

to counteract the weight of the arm itself.

The torso of AMBER-3M acts as the control center for the robot. It houses all of the control and power distribution electronics. A National Instruments (NI) cRIO-9033, running LabVIEW2015, is used as the high-level controller, while the low-level control is provided by the motion controllers (ELMO G-SOL-WHI). Fig. 2.3 shows the placement of the control and drive components within AMBER-3M.

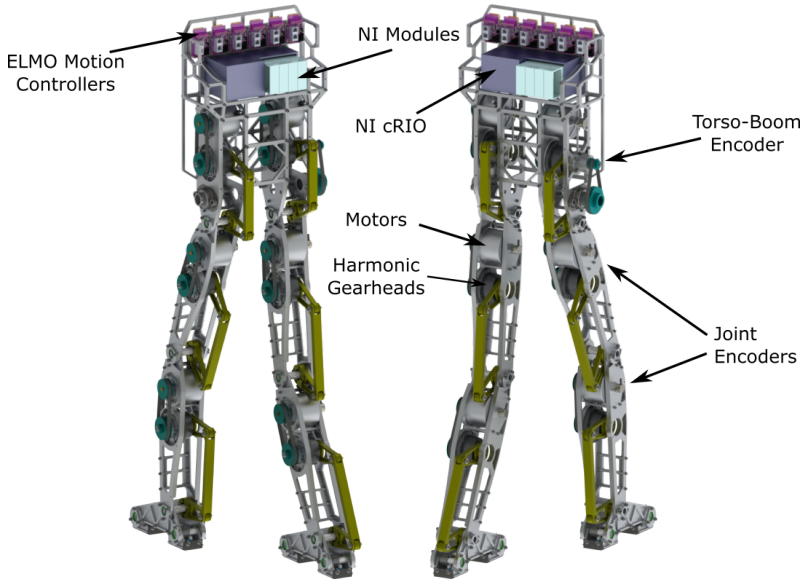


Figure 2.3: The key drive and control components within AMBER-3M.

Modularity

The key design feature of AMBER-3M is the modularity of the legs. Each of the leg segments, from thigh to feet, is able to be removed from the rest of the robot individually. Furthermore, the connection points of all joints on the robot have the same shape and functionality, with only two shafts holding a module to the one above it. The electrical connections running down the legs can also disconnect individually at the joints, for complete modularity. This simple connection style allows for portions of the leg to be swapped out in a matter of minutes. For the proof of concept experiments described later in this paper, three different leg configurations were used: Flat-Foot, Point-Foot, and Compliant Point-Foot (Spring Foot). A CAD model view of each is provided in Fig. 2.4 and the general configuration properties are shown in Table 2.1.

Figure 2.4: The three leg configurations of AMBER-3M: Flat Foot (left), Point Foot (center), and Compliant Point Foot (Right).

Flat-Foot Walking. In this standard leg configuration, AMBER-3M has fully actuated ankle joints with feet and, therefore, six total actuators. These feet can be used to walk with a more complex hybrid multi-contact behavior, as in (H. Zhao, W.-L Ma, et al., 2014), but are only used for flat-foot walking in the presented experiments. The ankles use the same drive system described previously, and have pivoting toe and heel sections which allow them to conform to the ground geometry.

Point-Foot Walking. Alternatively, a pair of rigid calves can be placed on the robot for when underactuated behaviors are desired. These calves do not have ankles or feet attached, but instead have a small rounded surface on bottom. This allows for a small point of contact and easy rotation between the legs and ground. It should be noted that the center of mass is above the hip in this configuration.

Compliant Point-Foot Walking. While these legs have a similar function to the

previous behavior, these calves now include a compression spring. The springs have a stiffness of 17 000 N/m and can deflect up to 3.5 cm, capable of storing up to 10.4 J during each step. There are position (LVDT) sensors placed within each calf to measure the spring deflection (but are not utilized for the experiments presented here). For ease of viewing these components of the calf in Fig. 2.4, the shell of the left leg has been removed.

Configuration	Weight	Height	CoM (from hip)
Flat-foot	29.3 kg	1.400 m	−0.161 m
Point-foot	21.9 kg	1.373 m	0.025 m
Spring-foot	23.5 kg	1.430 m	−0.024 m

Table 2.1: A list of robot properties for each configuration.

2.3 Control Methodology

In an effort to explore the properties of walking behaviors, we sought a formulaic approach to generating comparable control for multiple modes of locomotion. Here, we generalized the control framework from modeling to gait optimization on AMBER-3M across the three mechanical configurations. We built this control framework upon the *hybrid zero dynamics* (HZD) method for its generalizability to both fully- and under-actuated systems, and its track-record of successful hardware implementations (Aaron D. Ames et al., 2015; A. Hereid, E. A. Cousineau, et al., 2016; Wen-Loong. Ma et al., 2017).

Controlled Dynamics

From the view of hybrid zero dynamics, control is the successful satisfaction of a set of *output* equations on the robot via motion of its actuators. Importantly, these outputs are designed to be *impact invariant*, i.e., they must satisfy an HZD condition at all times. We will focus on presenting these output equations in a fashion which generalizes to all three AMBER-3M hardware configurations.

Hybrid Control System. The configuration-modular multi-domain hybrid control system is defined as a tuple (A. Hereid, E. A. Cousineau, et al., 2016),

$$\mathscr{HC} = (\Gamma^{\mathrm{i}}, \mathcal{D}^{\mathrm{i}}, \mathcal{U}^{\mathrm{i}}, S^{\mathrm{i}}, \Delta^{\mathrm{i}}, FG^{\mathrm{i}}) \tag{2.1}$$

where $\Gamma^{\mathrm{i}} = (V^{\mathrm{i}}, E^{\mathrm{i}})$ is a directed cycle and $\mathrm{i} \in \{f, p, s\}$ denotes the design configuration (flat-, point-, and spring-foot models respectively). $v \in V^{\mathrm{i}}$ are vertices indicating domains of admissibility, where $V^f = V^p = \{ss\}$, $V^s = \{ss, ds\}$ and ss,ds represent the single-support (only one leg on ground) and double-support phases of

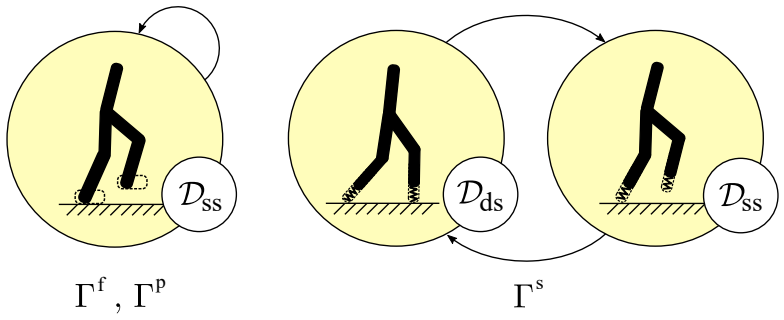


Figure 2.5: The flow of domains for the three configurations.

locomotion (both feet in ground contact). Vertices are connected by edges $e \in E^{i}$, where $e = \{v \rightarrow v_{+}\}$, and v_{+} is the sequential domain for a given configuration i. These are shown as arrows in Fig. 2.5.

In the hybrid control system, $\mathcal{D}^{i} = \{\mathcal{D}_{v}^{i}\}$ is the set of admissible domains of the system for a given configuration, written as the coordinates $(q^{i}, \dot{q}^{i}) \in \mathcal{D}_{v}^{i}$, where q^{i} represent positions and $\dot{q}^{i}$ are velocities. The actuation inputs, $\mathcal{U}^{i} = \{\mathcal{U}_{v}^{i}\}$ are a set of admissible controls with n_{u}^{i} control inputs where $\mathcal{U}_{v}^{i} \subseteq \mathbb{R}^{n_{u}^{i}}$. Particularly, $n_{u}^{p} = n_{u}^{s} = 4$ and $n_{u}^{f} = 6$. In addition, $FG^{i} = \{FG_{v}^{i}\}$ is the set of control systems in the admissible domains $\mathcal{D}_{v}^{i}$. On the edges, we define the switching surfaces $S^{i} = \{S_{e}^{i}\}$, where $S_{e}^{i} \subset \mathcal{D}_{v}^{i}$ and reset maps, $\Delta^{i} = \{\Delta_{e}^{i}\}$, that smoothly map between domains. The predefined flow of these domains is illustrated in Fig. 2.5.

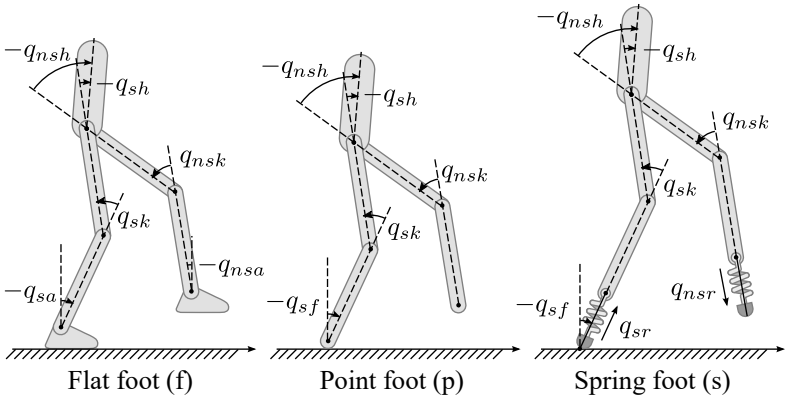


Figure 2.6: Depiction of the generalized coordinates for the three AMBER-3M design configurations.

With this setup, we now specialize to the robot under consideration. In this case the generalized coordinates, q^{i}, which differ across configurations, are $q^{i} \in Q_{v}^{i} \subset \mathbb{R}^{n^{i}}$ where $n^{f} = 6$, $n^{p} = 5$, and $n^{s} = 7$. In particular, we define q^{i} with specific

design configuration $\mathrm{i} \in \{f, p, s\}$ as $q^f = (q_{sa}\ q_{sk}\ q_{sh}\ q_{nsh}\ q_{nsk}\ q_{nsa})$, $q^p = (q_{sf}\ q_{sk}\ q_{sh}\ q_{nsh}\ q_{nsk})$, and $q^s = (q_{sf}\ q_{sr}\ q_{sk}\ q_{sh}\ q_{nsh}\ q_{nsk}\ q_{nsr})$, as labeled in Fig. 2.6.

Continuous Dynamics. The Equations of Motion over a continuous domain $\mathcal{D}_v^{\mathrm{i}}$ is determined by the Euler-Lagrange equation and holonomic constraints (J. W. Grizzle, Chevallereau, A. D. Ames, et al., 2010):

$$D(q^{\mathrm{i}})\ddot{q}^{\mathrm{i}} + H(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) = B_v u_v^{\mathrm{i}} + (J_v^{\mathrm{i}}(q^{\mathrm{i}}))^T F_v^{\mathrm{i}} \tag{2.2}$$

$$J_v(q^{\mathrm{i}})\ddot{q}^{\mathrm{i}} + \dot{J}_v(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}})\dot{q}^{\mathrm{i}} = 0 \tag{2.3}$$

where $D(q^{\mathrm{i}})$ is the inertia matrix, $H(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}})$ contains the Coriolis-centrifugal and gravity terms, B_v is the actuation distribution matrix, and $J_v(q^{\mathrm{i}})$ is the Jacobian of the holonomic constraints. Manipulation of these equations of motion yields the control systems $FG_v^{\mathrm{i}} = (f_v, g_v)$ given by:

$$\dot{x}^{\mathrm{i}} = f_v(x^{\mathrm{i}}) + g_v(x^{\mathrm{i}})u_v^{\mathrm{i}} \tag{2.4}$$

where $x^{\mathrm{i}} = (q^{\mathrm{i}}, \dot{q}^{\mathrm{i}})$ is the system state for a given robot configuration $\mathrm{i} \in \{f, p, s\}$.

Outputs. With the goal of synthesizing controllers and the dynamics defined, we now consider the *outputs* (or *virtual constraints*) (J. W. Grizzle, Chevallereau, Sinnet, et al., 2014; A. D. Ames, 2013) of the form:

$$\begin{aligned} y_v^{\mathrm{i}}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) &= \begin{bmatrix} y_{1,v}^{\mathrm{i}}(q^{\mathrm{i}}) \\ y_{2,v}^{\mathrm{i}}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}, \alpha_v^{\mathrm{i}}) \end{bmatrix} \\ &= \begin{bmatrix} y_{1,v}^{a,\mathrm{i}}(q^{\mathrm{i}}) - y_{1,v}^{d,\mathrm{i}} \\ y_{2,v}^{a,\mathrm{i}}(q^{\mathrm{i}}) - y_{2,v}^{d,\mathrm{i}}(\tau(q^{\mathrm{i}}), \alpha_v^{\mathrm{i}}) \end{bmatrix} \end{aligned} \tag{2.5}$$

where $y_{1,v}^{a,\mathrm{i}} : Q^{\mathrm{i}} \to \mathbb{R}$ and $y_{2,v}^{a,\mathrm{i}} : Q^{\mathrm{i}} \to \mathbb{R}^{n_{o,v}^{\mathrm{i}}}$ define the "actual" relative degree 1 and 2 outputs, respectively (see (A. D. Ames, 2014) for justification). Here, $n_{o,v}^{\mathrm{i}}$ indicates the dimension of relative degree 2 outputs, with $n_{o,ss}^f = 5$, $n_{o,ss}^p = 4$, and $n_{o,ss}^s = 4$, $n_{o,ds}^s = 3$. For the flat-foot, single support walking, we have $y_{2,ss}^{a,f}(q^{\mathrm{i}}) = [q_{sk}, q_{nsk}, \theta_{tor}(q^{\mathrm{i}}), \delta m_{nsl}(q^{\mathrm{i}}), q_{nsf}]^T$, where $\theta_{tor}(q^{\mathrm{i}})$ and $\delta m_{nsl}(q^{\mathrm{i}})$ are defined in Ma et al., 2014. For the other cases, actual degree 2 outputs are simply joint angles: $y_{2,ss}^{a,p}(q^{\mathrm{i}}) = [q_{sk}, q_{sh}, q_{nsh}, q_{nsk}]^T$, $y_{2,ss}^{a,s}(q^{\mathrm{i}}) = [q_{sk}, q_{nsk}, q_{sh}, q_{nsh}]^T$, and $y_{2,ds}^{a,s}(q^{\mathrm{i}}) = [q_{sk}, q_{sh}, q_{nsh}]^T$. The actual relative degree 1 output, $y_1^{a,f}(q^{\mathrm{i}})$ is the linearized hip velocity (see (Ma et al., 2014)) for flat-foot walking. Note that there are no relative degree one outputs for under-actuated walking (point- and spring-foot).

Desired relative degree 2 outputs, $y^{d,\mathrm{i}}_{2,v}(\tau(q^{\mathrm{i}}), \alpha^{\mathrm{i}}_v)$ are represented by a series of 4^{th} order Bézier polynomials (Sreenath, H. Park, and J. Grizzle, 2014), whose parameters α^{i}_v are a set of constant matrices representing a specific gait and will be solved by an optimization algorithm.

$$\tau(q^{\mathrm{i}}) = \frac{\delta p_{hip}(q^{\mathrm{i}}) - p^{\mathrm{i},+}_v}{p^{\mathrm{i},-}_v - p^{\mathrm{i},+}_v} \tag{2.6}$$

where $\delta p_{hip}(q^{\mathrm{i}})$ is the linearized x-position of the hip, whose initial and final values, $p^{i,+}_v$ and $p^{i,-}_v$, are predefined.

Partial Hybrid Zero Dynamics. To drive the outputs to zero, we used an input-output feedback linearizing control law A. D. Ames, 2013 for our hybrid control systems:

$$\begin{aligned} u^{\epsilon}_v(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}, \alpha^{\mathrm{i}}_v) = -(\mathcal{A}^{\mathrm{i}}_v)^{-1} \Bigg(\begin{bmatrix} 0 \\ L^2_{f_v} y^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}, \alpha^{\mathrm{i}}_v,) \end{bmatrix} \\ + \begin{bmatrix} L_{f_v} y^{\mathrm{i}}_{1,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) \\ 2\epsilon L_{f_v} y^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}, \alpha^{\mathrm{i}}_v) \end{bmatrix} + \begin{bmatrix} \epsilon y^{\mathrm{i}}_{1,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) \\ \epsilon^2 y^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}, \alpha^{\mathrm{i}}_v) \end{bmatrix} \Bigg) \end{aligned} \tag{2.7}$$

where $\mathcal{A}^{\mathrm{i}}_v = [L_{g_v} y^{\mathrm{i}}_{1,v}(q, \dot{q}) \quad L_{g_v} L_{f_v} y^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}, \alpha^{\mathrm{i}}_v)]^T$ and $\epsilon > 0$. Applying this controller results in linear output dynamics $\dot{y}^{\mathrm{i}}_{1,v} = -\epsilon y^{\mathrm{i}}_{1,v}$ and $\ddot{y}^{\mathrm{i}}_{2,v} = -2\epsilon \dot{y}^{\mathrm{i}}_{2,v} - \epsilon^2 y^{\mathrm{i}}_{2,v}$, which yield the *partial zero dynamics surface*, given by:

$$\mathbf{PZ}^{\mathrm{i}}_v = \{(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) \in \mathcal{D}^{\mathrm{i}}_v | y^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}) = 0, \dot{y}^{\mathrm{i}}_{2,v}(q^{\mathrm{i}}, \dot{q}^{\mathrm{i}}) = 0\} \tag{2.8}$$

showing asymptotic stability. Furthermore, for any $e \in E^{\mathrm{i}}$, if there exists an α^{i}_v such that

$$\Delta^{\mathrm{i}}_e(S^{\mathrm{i}}_e \cap \mathbf{PZ}^{\mathrm{i}}_v) \subset \mathbf{PZ}^{\mathrm{i}}_{v_+}, \tag{2.9}$$

then the system is *hybrid invariant*. In such cases, stability of the full order system reduces to that of the *partial hybrid zero dynamics* (PHZD). Next, our mission is to find the parameters α^{i}_v (the gait) that guarantee PHZD conditions for each design configuration.

2.4 Efficient HZD Gait Optimization

To generate proper walking gaits for the modular robot AMBER-3M, we generalized and used a collocation based optimization algorithm. Our gait-optimization problem

is defined by the following nonlinear programming (NLP):

$$\mathbf{Z}_v^{\mathrm{i},*} = \underset{\mathbf{Z}}{\operatorname{argmin}} \quad \sum_v \mathcal{J}(\mathbf{Z}_v^{\mathrm{i}}) \tag{2.10}$$

$$\text{s.t} \quad \mathbf{Z}_{\min} \leq \mathbf{Z}_v^{\mathrm{i}} \leq \mathbf{Z}_{\max}, \tag{2.11}$$

$$\mathbf{C}_{\min} \leq \mathbf{C}(\mathbf{Z}_v^{\mathrm{i}}) \leq \mathbf{C}_{\max}, \tag{2.12}$$

where, $\mathbf{Z}_v^{\mathrm{i}}$ is a vector of decision variables, $\mathbf{Z}_{min}$ and $\mathbf{Z}_{max}$ are the enforced limits of each decision variable, and $\mathbf{C}(\mathbf{Z}_{v^{\mathrm{i}}}^{\mathrm{i}})$ is a vector of nonlinear constraint functions.

For the purpose of energy-efficient walking, we formulate the objective function $\mathcal{J}_v(\mathbf{Z}_v^{\mathrm{i}})$ as the mechanical cost of transport (2.13). In this modular optimization problem, the constraints $\mathbf{C}(\mathbf{Z}_{v^{\mathrm{i}}}^{\mathrm{i}})$ are chosen to be the HZD invariant condition, foot height clearance, torso moving range, and ZMP constraints (ZMP only applies to flat-foot case). To generate gaits with a specified forward velocity (for upcoming experiments), we enforced an average x-velocity as a nonlinear equality constraint. Due to the complex nonlinearity of this problem, we employed a *pseudospectral method* (Gottlieb and Orszag, 1977) to improve computational efficiency and robustness. This pseudospectral method numerically approximates the time solution of the dynamics by trigonometric or orthogonal polynomials at chosen collocation points. This method was previously applied to single-domain, under-actuated walking in (A. Hereid, S. Kolathaya, and A. D. Ames, 2016), but was extended to all three configurations in this paper.

2.5 Experimental Evaluation

To validate the modularity in the robot design experimentally, this framework also requires modularity of the control structure, including mechanical, electrical, and controller implementation. This section will introduce the electrical system of AMBER-3M first, then an experimental control procedure for walking is detailed. The effectiveness of the modular components will be shown through tests of each behavior. Finally, we present a more thorough analysis of point-foot walking locomotion economy.

Experiment Configuration

The control framework of AMBER-3M is composed of three levels: the high level controller, the low level controller and the data logger. In the high level, the onboard cRIO running RTLinux serves as the master board, communicating with the low level controllers in a real-time fashion. Experimental data is sent to a remote

desktop, which acts as the data logger. On the low level, motor drivers deliver the desired torque-driven motion to their corresponding BLDC motors. Other modes of experimental feedback, such as the rotary boom states, torso rotation, and electrical power consumption, are collected by an embedded FPGA board.

Algorithm 1 Real Time Controller

Input: AMBER-3M design configuration: $i \in \{f, p, s\}$;
Input: AMBER-3M model parameters;
Input: Domain/step switching flag: $\tau^i_{max,v}$;
Input: Optimization Parameters: $p^{i,+}_v, p^{i,-}_v, q^{i,+}, \alpha^i_v$;
Input: PD controller gains: K^i_p, K^i_d;
Input: System states (Left/Right): $(q^i_{LR}, \dot{q}^i_{LR})^T$;
Input: L/R stance; Encoder status; Drive status;
Output: Enable/Disable motor drives;
Output: Desired torque for each BLDC motor;

Enable motor drives;
repeat
    Wait till all motor drives are enabled
until (Drive-Status == Enable)
while (¬ Stop-RT) **do**
    if Control type == Initialization **then**
        $q_d = q^{i,+}$; $\dot{q}_d = \mathbf{0}$;
    else
        Map $(q, \dot{q})^T$ from Left/Right to Stance/nonStance;
        Calculate phase variable $\tau = \tau(q^i)$;
        if $i == f$ **then**
            Calculate $(\xi_1,\ \xi_2)$ based on τ;
        end if
        Calculate $\dot{\tau}$;
        Calculate $(y_d,\ \dot{y}_d)$;
        Calculate $(q_d,\ \dot{q}_d)$ using PHZD reconstruction Ma et al., 2014;
    end if
    Apply PD controller:
    $u^i = K^i_p(q_a - q_d) + K^i_d(\dot{q}_a - \dot{q}_d)$;
    Map u^i from Stance/nonStance to Left/Right;
    if $\tau > \tau^i_{max,v}$ **then**
        Switch to next domain v_+ or step.
    end if
    Sending torque command to motor driver;
    Log data into remote desktop;
end while

Demonstrating Multiple Walking Behaviors

For each leg configuration, an optimized gait can be reliably produced by the aforementioned optimization method. To realize the motion experimentally on AMBER3M, a real time controller was programmed in C++ and compiled onto the cRIO. A detailed pseudocode is presented in Algorithm 1 to illustrate this control method. Further explanations for the algorithm can be found in (Ma et al., 2014).

It is very important to notice that, for each configuration, the control method was purely PD feedback control without any pre-defined feedforward component. Due to the modularity of this control structure, switching among design configurations is accomplished without restructuring code. To demonstrate the efficacy of the modular framework, gaits were generated for each leg configuration and realized on AMBER-3M using Algorithm 1. The robot was able to walk three laps (64.65 m) on the boom with each tested gait, at which point they were deemed successful. Tiles of a single step from each behavior are shown in Fig. 2.7. A phase portrait of the gaits' experimental and simulated performance is provided in Fig. 2.8, demonstrating that periodic walking was achieved.

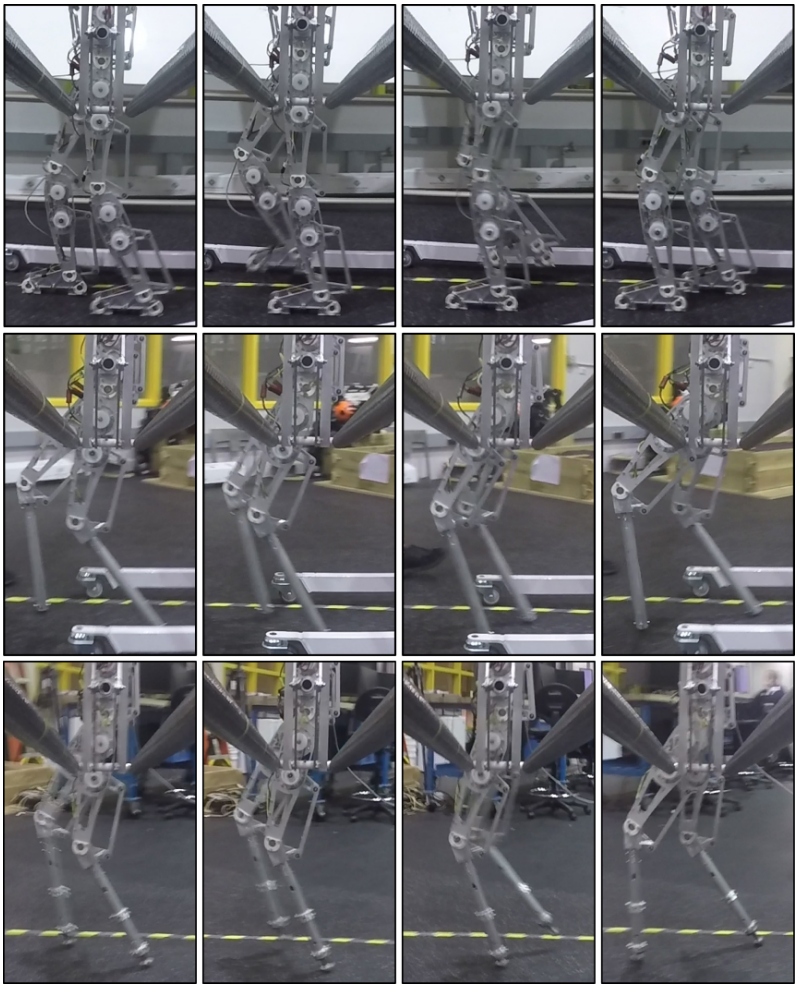


Figure 2.7: Walking tiles from each configuration trial: Flat-foot (top), Point-Foot (middle), and Compliant Point-Foot (bottom).

Point-Foot Locomotion Economy

Here, we assess the locomotion economy of point-foot walking across speeds with systematic experiments. The objective was to discover the energetic optimal limit for point-foot walking over a range of transportation speeds, as well as any trend in this performance. 36 walking gaits were generated, covering a speed range of 0.34 m/s to 0.94 m/s, and tested on AMBER-3M. Note that these speeds are the simulated average forward velocity of the center of mass of the robot. Due to underactuation and model uncertainty, experimental velocities differed from the simulation. For instance, at very slow speeds (< 0.3 m/s), unmodeled friction has a more significant effect, resulting in a reduced speed in experiment.

For each experiment, AMBER-3M again walked three laps while data was logged. Since the robot walks in a circle, the gait is not symmetric between the left and right legs. To circumvent this issue, COT was analyzed for every two steps, or one stride, to average these asymmetries. The mechanical cost of transport for the j^{th} stride was calculated as:

$$MCOT_j = \frac{1}{Mgd_j} \int_{t_j^-}^{t_j^+} \mid u \mid^T \mid \dot{q} \mid dt \tag{2.13}$$

where M is the total mass of the robot, g is the gravitational constant, and d_j is the distance traveled by the center of mass in the kinematic model. The torque u is estimated from the measured current going to the motor, and the angular velocity $\dot{q}$ is measured by the incremental encoder at each joint. These measurements were then plotted against their corresponding average forward velocities for each step in Fig. 2.9.

As shown in the plot, it can be seen that six gaits occupied the lowest region of the plot which have been highlighted in the figure. Using these six gaits, a Pareto frontier can be created to represent the locomotion economy extreme for point-feet walking on AMBER-3M. A significant ($p < 0.005$) trend was found in these optimal gaits through regression analysis, showing that the mechanical energy required to walk increases with speed.

The electrical power consumption of the motor drivers was measured directly from the DC power supply. We calculated the electrical cost of transport for the j^{th} stride by

$$ECOT_j = \frac{1}{Mgd_j} \int_{t_j^-}^{t_j^+} IE \; dt \tag{2.14}$$

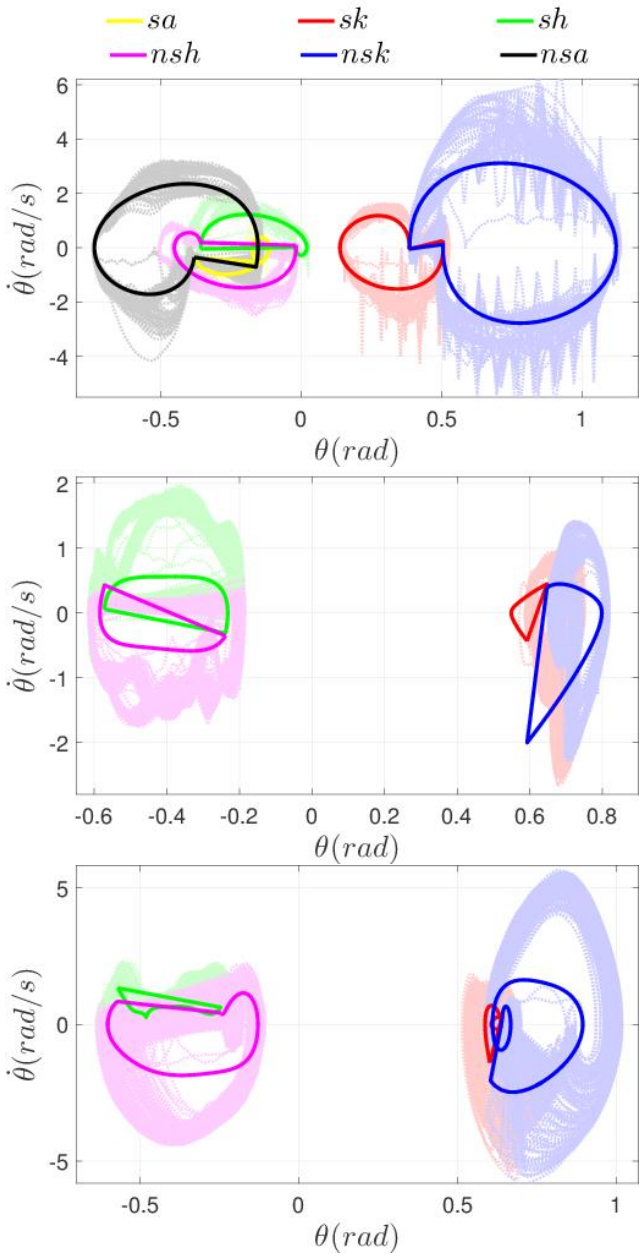


Figure 2.8: Phase portraits for each behaviors: Flat-foot (top), Point-foot (middle), and Compliant Point-Foot (Bottom).

where $E = 48.3V$ is the DC voltage of the power supply, and I is the total current going to the motor drivers. The electrical cost of transport for the six optimal gaits is shown in Fig. 2.10. It should be noted that this power data does not include the power consumed by the sensors and on-board computers. Because these additional power costs are relatively constant, they would not effect any trends seen in the data. In comparison with Fig. 2.10, where higher velocity tends to require higher mechanical energy, electrical power shows a significant ($p < 0.01$) decrease in cost

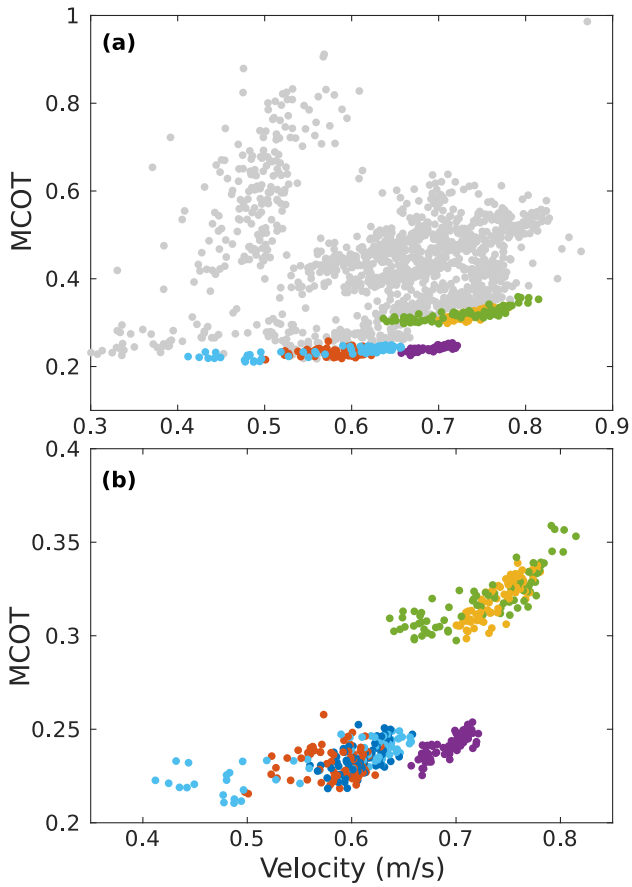


Figure 2.9: Plot of mechanical cost of transport vs walking speed for **(a)** all 36 successful gaits and **(b)** the six optimal gaits. Each color is a distinct gait and each dot is a single stride.

versus speed.

A video of the experiments is shown in n.d.

2.6 Conclusions

A modular robot design was presented as a means of comparing locomotion economy for multiple bipedal gait behaviors. To show the physical capability of AMBER-3M to viably walk with all its leg modules, three different walking behaviors were implemented experimentally. These preliminary results demonstrated that a single robotic platform and control methodology can successfully realize stable walking across multiple behaviors. A total of 36 optimized gaits were used to further study locomotion economy for point-foot walking over a wide walking speed range. A Pareto-optimal frontier for locomotion economy was distilled, illustrating the peak energy performance of this design configuration at various speeds. Significant, yet opposite, trends were seen in the mechanical and electrical costs. Future work will

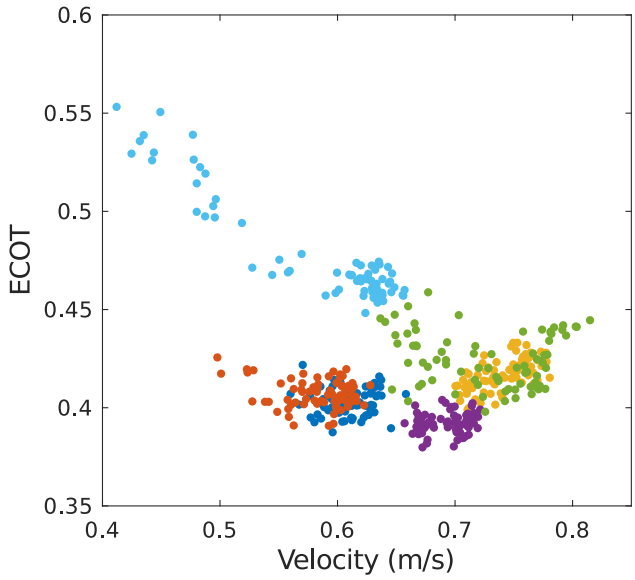


Figure 2.10: Plot of electrical cost of transport vs walking speed.

focus on similar in-depth analysis for the flat- and spring-foot configurations, with the intent of further investigating the energetics of robotic bipedal locomotion.

2.7 Lessons Learned and Later Work

This work was a great look into how the physical shape, size, and configuration of a robot was directly influencing the limit of a robot's capability, regardless of how optimizations are used in the motions and controllers. This idea would come to play a part in the later work with hopping robots and how to improve their locomotion efficiency through mechanical design.

Around the same time that this work on robotic locomotion efficiency was being performed, AMBER-3M was also being used to test out new control and motion generation methods (M. J. Powell et al., 2016). This work was looking into using the angular momentum state of the system as the control target when designing gaits for AMBER-3M, which was a great candid for the initial tests given its model simplicity.

In the years since then, AMBER-3M has been used in many other research applications including Data-driven control (Tabuada et al., 2017), walking over slippery surfaces (Wen-Loong Ma, Or, and Aaron D Ames, 2019), Preference-based learning (Noel Csomay-Shanklin, Tucker, et al., 2021; Tucker et al., 2021), Control barrier functions (Noel Csomay-Shanklin, Cosner, et al., 2021; Rodriguez et al., 2022), and Model predictive control (Galliker et al., 2022).

Chapter 3

AMPRO-3: A TRANSFEMORAL PROSTHESIS

Figure 3.1: AMPRO-3 in use on a treadmill in the AMBER Lab at Caltech.

3.1 Introduction of the AMPRO Series

Throughout 2013 and 2014 the AMBER Lab designed and created the first two iterations of the prosthetic devices known as AMPRO-1 and AMPRO-2, while at Texas A&M University. These prostheses were intended for people that had an amputation or injury above the knee, somewhere in the thigh. The goal of the devices was to attach to the user's leg and allow them to regain the ability to move efficiently. Each prosthesis contained two actuators to drive the extension/flexion degree of freedom of the both knee and ankle in a controlled manor. This would allow the user to not only walk, but move over terrain such as stairs where the height of the surface is changing and the knee would preferably need to be bending while also exerting torque. This was different from commercial devices which would generally only control the ankle, while leaving the knee motion as passive with a built in locking feature to support weight during stance.

Throughout 2014 and parts of 2015 AMPRO-2 was successfully tested by both the students responsible for designing it and an amputee, yielding results for many

papers. However, when the AMBER lab moved to the Georgia Institute of Technology in 2015, these two devices remained behind. In order to continue this area of research after this point, a new iteration of the AMPRO series was created: AMPRO-3. Between the desires to have a more capable robot and to expand beyond previous research, this new device was created with a few extra features and considerations. This chapter of the thesis will discuss those novel changes incorporated into AMPRO-3 and some of the results/papers that arose from its creation.

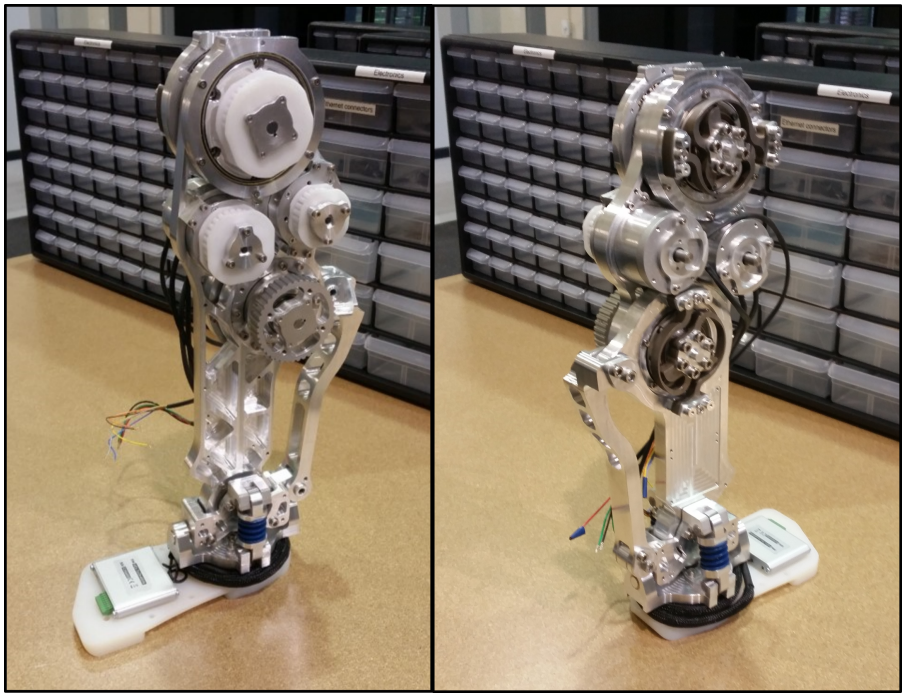


Figure 3.2: AMPRO-3 assembled for the first time after manufacturing was complete.

3.2 AMPRO-3

AMPRO-3, seen in Fig. 3.2, began getting designed in Fall 2015 out of need for a new device. After all of the testing done with the previous iterations of the device, it was decided that a few changes were in order. Specifically, higher torque capability, more user comfort, and improved ability to walk over terrain, all while still maintaining the electrical framework of AMPRO-2 for ease of transferring code to the new device.

The improved torque required a slight increase in gear reduction at each joint, especially for the ankle. Additionally, it was found that the actuator and transmission layout on AMPRO-2 had some noise issues and were quite wide, making the device a little more cumbersome for the user. To fix all of these issues, the actuator was

split into sections. Rather than having the motor and gearbox in line, with a belt leading to the joints, the new layout would have the motors and gearboxes offset with timing belts placed between. This solved the large width problem as well as the noise complaints. The overall gear reduction was also increased by having the timing belt pulleys perform an additional reduction, increasing the peak torque of each joint. In order to keep the center of mass for the leg higher, a linkage was used to drive the ankle torque down to the joint rather than the chains used on AMPRO-2.

In order to tackle the goal of improving user comfortability, springs were added at the output of each actuator. As will be shown later, these springs would give some deflection under the users weight, leading to lower impacts felt by the user. The device would still be able to sense this compliant motion through the use of additional sensors on the joint outputs. These springs can be seen as dark spirals in the right hand side of Fig. 3.2 above.

To give the device the ability to comform to terrain as well as add more user comfort, a third joint was incorporated in the design: ankle roll. This joint would allow the foot to passively match the angle of the terrain in the frontal plane, while the actuated ankle pitch joint could handle the rest. This joint took the longest to design, since the goal was to still keep it compact around the ankle joint. In the end, small springs and a pair of bushings were placed to give the joint its compliant range of motion. This third joint can be seen in the lower portion of Fig. 3.2, just above the foot. This joint had its own sensor placed on the front to provide feedback to the controller if desired. Further, a 6-axis load cell was also placed just below the ankle to give any ground reaction information needed for the controller or assessment of the motion.

Lastly, most of the frame of the device was customized to reduce the mass of the overall device. While this meant the manufacturing time and cost was higher, it meant that the device was able to be kept just below the typical leg mass that it was replacing. These custom frame pieces can seen throughout the device in Fig. 3.2. The device was finished getting manufactured around May 2016, at which time testing began inearnest. The design of the AMPRO-3 was first presented in (Huihua Zhao, Eric Ambrose, and Aaron D. Ames, 2017), where the device, its functionality, and its initial performance were documented. The section on the design from this paper is shown next.

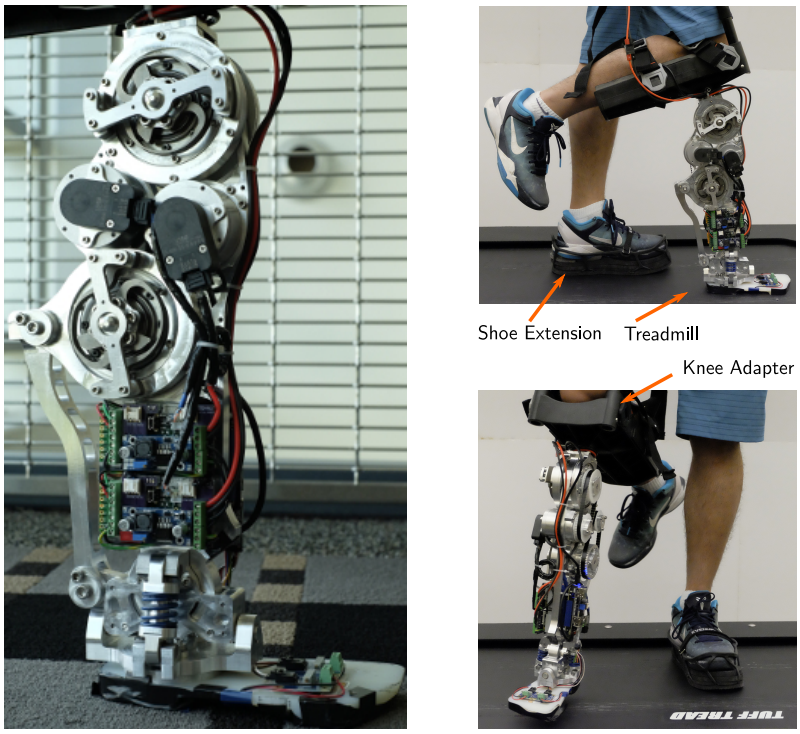


Figure 3.3: Assembled hardware of AMPRO3 (left) and the experimental testing setup (right).

3.3 Excerpt on the Design of AMPRO3

To experimentally validate the systematic control methodology—including hybrid system modeling, nonlinear control and gait generation—a new powered transfemoral prosthesis, AMPRO3, is designed and built from ground up. The mechanical design of AMPRO3 is discussed here with a special focus on the various compliant components. The electrical on-board computation and sensing of AMPRO3 are then illustrated in detail.

Mechanical Design of AMPRO3

The design of AMPRO3, seen in Fig. 3.3, provides the convenience of on-board power alongside the advantages of highly dynamic motion and sensing capabilities, all in a compact design. The device weighs in at 5.95 Kg without the knee adapter, and reaches a total height of 451 mm. Both the knee and ankle flexion/extension joints are actuated and actively controlled on this device. A closer look at the transmission drives is shown in Fig. 3.4. Each of these joints contains a 206 W brushless DC motor (MOOG BN23) capable of just over 1 Nm peak torque. In order to achieve the desired torques associated with human walking, harmonic Gearheads with a 1:100 reduction are placed between the motor and the joint.

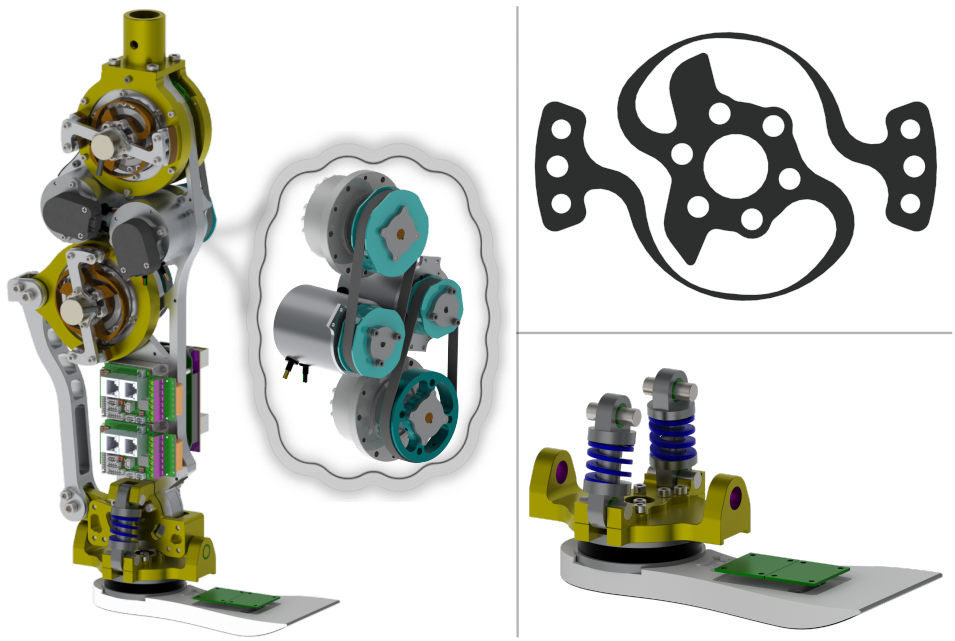


Figure 3.4: The novel design features of AMPRO3, including: Transmission (Left), Torsion Springs (top right), and Ankle Roll Joint (bottom right).

Unlike previous AMPRO models, which had the motor and gearhead mounted in line with each other (H. Zhao, Horn, et al., 2016a), AMPRO3 has the gearhead separated from the motor and then connected via a timing belt. Adding this belt connection between the two allows for separate management of the gear reduction at each joint, while still using the same motors and gearheads for both. The belts are also used to produce the last reduction between the motor and joints. The belt ratio in the ankle and knee transmissions are 4:7 and 5:6, respectively. Table 3.1 shows the final gear ratio, peak torque (T_p), continuous torque (T_c), and max speed (V_{max}) for each joint's transmission. The joint torques listed include an approximate efficiency of the harmonic gearheads of 70%. In addition to these changes, two major design changes were made in AMPRO3 from previous versions. These key changes are discussed separately in the following paragraphs.

SEA Joints. The first major mechanical design change within AMPRO3 is the

Table 3.1: Joint Transmission Capabilities

Joint	Gear Ratio	T_p(Nm)	T_c(Nm)	V_{max}(rad/s)
Ankle	1:175	123	38.5	4.0
Knee	1:120	85	26.5	5.8

addition of compliance at each joint. For the two actuated degrees of freedom, this compliance is represented in the form of a planar torsional spring mounted between the gearhead and the joint. The resulting compliance from the springs decreases the effects of impact on the motors and user, leading to smoother and more comfortable movement. The shape of the torsion springs is based on the design of (Ihrke et al., 2010), modified only for mounting purposes and to reach the desired stiffness of 20Nm/deg. The spring model is shown in Fig. 3.4. The stiffness of these springs was chosen based on the joint torque profile seen in simulation. Adding in this compliance between the motor and the joint creates the possibility of a SEA. In cases like walking, where impacts happen at predictable moments within the gait, it becomes possible to control the timing of energy storage and release in coordination with desired motion, thus saving energy (Paluska and Herr, 2006).

Compliant Passive Ankle Roll Joint. Motivated by the fact that lateral ankle movement plays an important role for a person's preferred motion during walking (for example, walking on uneven terrains) (Ficanha, Rastgaar, and Kaufman, 2014), AMPRO3 also contains a third joint, ankle roll, as another major change from previous designs. This joint is not actuated, although motion is passively controlled through compression springs as shown in Fig. 3.4. These springs are linear and placed on both sides of the joint, leading to a roll joint stiffness of about 2.1Nm/deg. The stiffness of these springs was chosen through a control-in-the-loop method of design, as in (J. Reher, E. A. Cousineau, et al., 2016). The optimization for the prosthetic model was run with multiple spring constants, and the final choice was made based on gait parameters like joint torques and cost of transport.

While the addition of this roll joint provides further compliance between the environment and the user, it also allows for a wider variety of gaits and control methods through available 3D movement (Bauby and A. D. Kuo, 2000). The user can change the width of their stance or walking, and navigate sloped terrain. Based on the designed gaits in simulation, the range of motion for this joint is limited through hard-stops to provide up to 5 degrees of adduction and 2 degrees of abduction. The latter is limited to ensure the user does not move their center of pressure outside their base of support.

On-board Sensing and Computation

A BeagleBone Black (BBB) micro-controller runs the three-level control architecture—including sensing and computation—on board the prosthetic. The low-level control

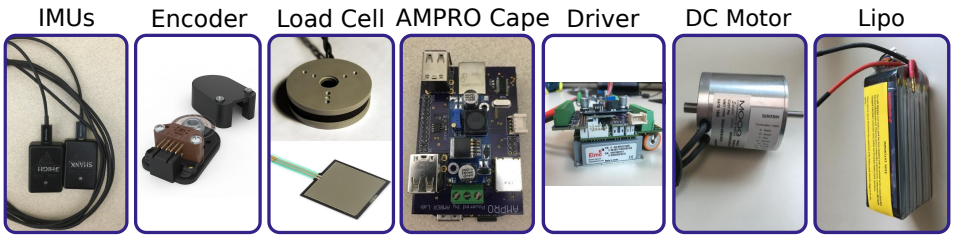


Figure 3.5: Electric components of 3D prosthesis AMPRO3.

of motor driving comes from two ELMO motion controllers (Gold Solo Whistle). The motion controllers are actively controlling the motion of the ankle and knee joints using feedback from two encoders at each joint and input from the BBB. An incremental encoder placed on the motor side of the joints and an absolute encoder placed on the joint output side can be used to measure any deflection of the torsion spring. User sensory feedback is provided to the device through an Inertial Measurement Unit (IMU) fixed to the knee adapter. Further environment sensing is gained from a 6-axis load cell in the foot. Two flex force sensors can also be mounted at the heel and toe of the foot to provide on-and-off ground contact feedback during more complex foot motion (e.g., multi-contact with heel strike and toe off). The whole system is powered through a 9-cell (33.3 V), 3900 mAh Li-Po battery (ThunderPower).

To expand the functionality of the BBB, a custom printed circuit board (PCB): AMPRO Cape, is designed. The Cape adds in a CAN bus chip (for communicating with ELMO drives), 4 USB ports (for communicating with load cell, force sensors, and IMUs), and a 5 V voltage regulator (for power supply of the BBB and USB ports). For the purpose of better wire organization, we also designed a PCB for the ELMO motion driver. This ELMO board contains a) a voltage converter to power the logic board of the ELMO, and b) serial CAN bus and power connectors to allow multiple ELMOs to be connected in serial. The result is a compactly organized self-contained prosthesis. The major electric components are shown in Fig. 3.5 and the assembled hardware of AMPRO3 is shown in Fig. 3.3. The three-level control architecture of AMPRO3 is similar to AMPRO1, which can be referred to (H. Zhao, Horn, et al., 2016b). The high level controller of AMPRO3 is coded into C++ packages and runs on the ROS. The detailed discussion is omitted here and can be referred to in the discussion of AMPRO1 (H. Zhao, Horn, et al., 2016a).

3.4 Later Work on AMPRO

Beyond the work performed in (Huihua Zhao, Eric Ambrose, and Aaron D. Ames, 2017), AMPRO-3 was used for two other papers around this same time (Azimi, Shu, Huihua Zhao, Eric Ambrose, et al., 2017) and (Huihua Zhao, Ayonga Hereid, et al., 2016), which looked more into the motion generation for prosthesis-human combined gaits and some novel controller testing. In parallel to all of this, a new collaboration was initiated with another lab at Georgia Tech that wanted to use AMPRO-3. It was decided that a new version of the device would be designed, and two copies would be created; one for each lab. This final version of the device is AMPRO-4. Only a few changes were made for this device from AMPRO-3, but they greatly improved the ease of use for the device. These changes included adding in a disconnect point in the leg, where spacers could be added to allow the height of the device to change depending on the user. Before, the user would have to add spacers to their other leg to match the device, and this would remove the need for that. This disconnection point can be seen in Fig. 3.6. The other changes were mainly to fix small issues with assembly and functionality that were discovered while using AMPRO-3.

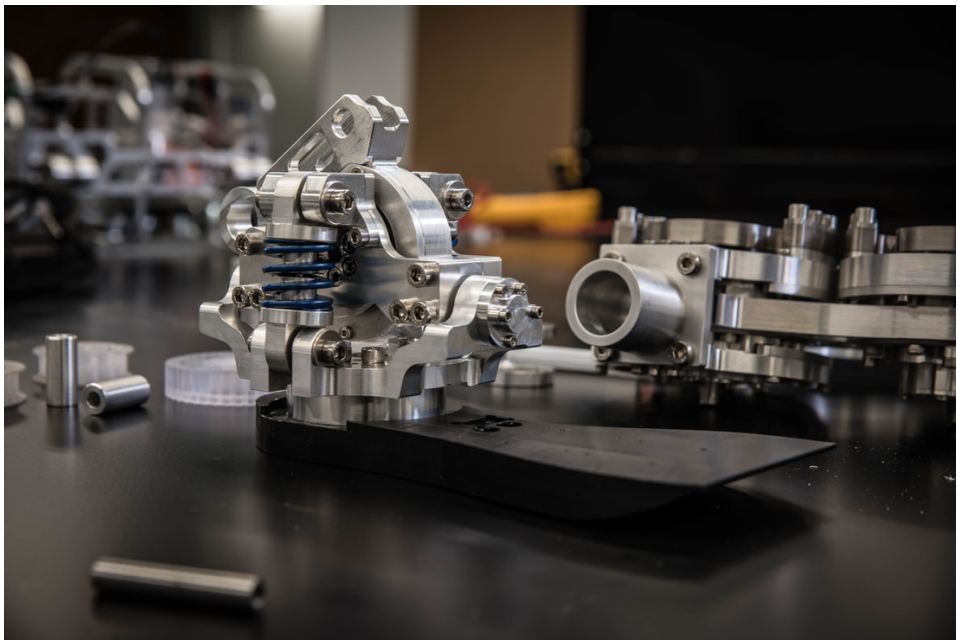


Figure 3.6: Assembly of the foot from AMPRO-4 disconnected from the rest of the device.

3.5 Other Work Using AMPRO-3 and AMPRO-4

The hardware of AMPRO-3 and AMPRO-4 have been used on many papers since 2017. AMPRO-3 has gone on to facilitate six more papers in the AMBER lab, for a variety of topics such as Adaptive control (Azimi, Shu, Huihua Zhao, Gehlhar, et al., 2019), developing human-machine interface technology (Yu et al., 2020), recurrent neural network control (Gao et al., 2020), testing musculoskeletal models (K. Li et al., 2021), and model-dependent control (Gehlhar, J.-h. Yang, and Aaron D Ames, 2021; Gehlhar and Aaron D Ames, 2021).

The version of AMPRO-4 that was used by the EPIC Lab at Georgia Tech was modified to fit a multitude of amputees and retrofitted with more sensing to assess performance. This device has since been used by them for many research topics (Bhakta, Camargo, Donovan, et al., 2020; Bhakta, Camargo, Kunapuli, et al., 2019; Bhakta, Camargo, and A. J. Young, 2018; Bhakta, Camargo, Compton, et al., 2021; A. Young and Krishan, 2019).

Unfortunately, the mechanical design side of the AMPRO project has been dormant since the fourth iteration was created, and the maker has moved on to focus on hopping robots. Many of the lessons and findings of the work with prostheses made there way towards influencing the design of these hopping robots. This can been seen in the research of internal compliance and the use of new actuators, on-board power, and custom-machined parts within these hoppers. These hopping robot projects will be examined in the following chapters, as they make up the majority of this thesis.

Chapter 4

ANALYSIS OF VERTICAL HOPPING MECHANISMS

After the lab moved from Georgia Tech to Caltech, a new project in partnership with Disney Research and Development was started. This project involved creating a hopping robot that would mesh with a few key design principles that Disney stands by such as safe-actuation. The work discussed in this and the next two chapters will cover the multiple generation of hopping robots that were designed, built, and tested in the AMBER Lab, opening doors for new behaviors and directions of research. This chapter presents the work published in (E. Ambrose, N. Csomay-Shanklin, et al., 2019).

4.1 Introduction to hopping

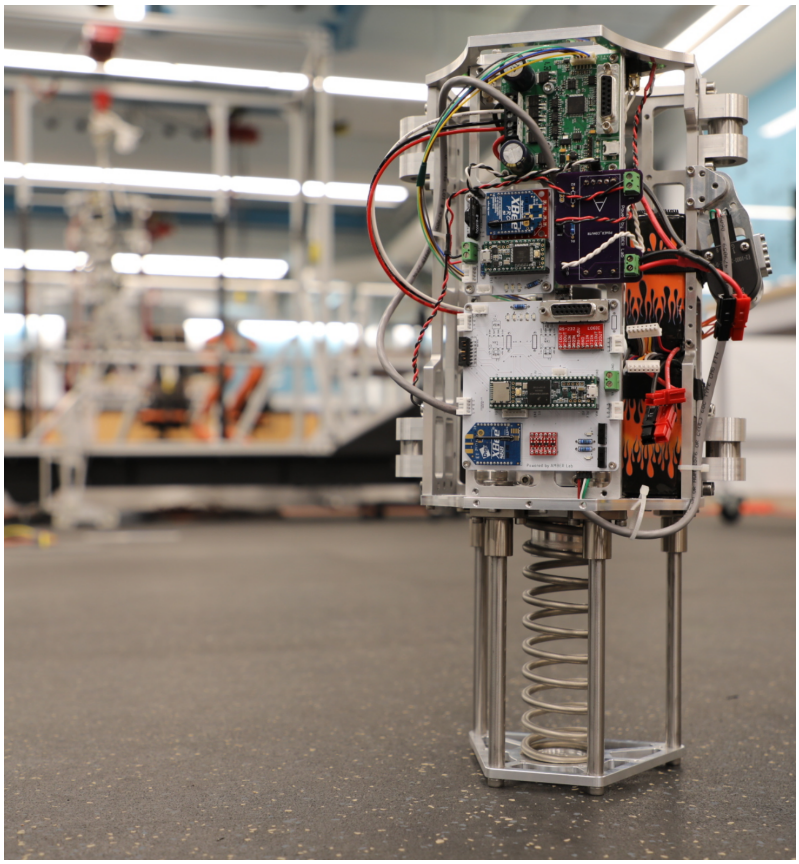


Figure 4.1: A 1-Dimensional, moving-mass hopping robot used to experimentally demonstrate the results of this paper

The hopping motion is fundamentally hybrid in nature, consisting of multiple phases

of differing governing dynamics as well as discrete events (Westervelt, J. W. Grizzle, and Koditschek, 2003; Sinnet et al., 2011). Similar to the case of bipedal running (W. Ma, Kolathaya, et al., 2017; J. Park et al., 2014), there are two different domains of continuous dynamics separated by very dramatic discrete impacts. These impacts cause energy to be lost beyond what would normally be lost in a robot due to friction alone. In order to make up for this disruption and reach a desired hopping height, the robot must add energy back in through actuation. The easiest way to do this is by applying a force on the world while the robot is in contact with the ground, and then spend the time in the air preparing for that action. However, the duration of the ground phase tends to be much smaller than that of the aerial phase, leading to a need for very brief and powerful actuation during this time.

Robotic hopping is not a new problem, but one that has been studied for decades (M. H. Raibert, 1984; H. B. Brown and Zeglin, 1998; Fiorini and J. Burdick, 2003), with many approaches to the problems of actuation and control. Two methods of overcoming the issue of inputting energy during the short ground phase have been followed. The first is the idea of decreasing the amount of energy lost at each impact, to therefore reduce the amount that must be input back to the system by the actuator. This has been accomplished by decreasing the mass of the robot's foot (Zeglin, 1999), or by taking shorter and higher frequency hops (Zeglin, 1999; M. H. Raibert, H. B. Brown, and Chepponis, 1984). The second way of improving actuation is to increase the amount of time spent on the ground to allow for longer durations of energy input. This is typically done by using a leg with a large range of motion and an elastic element (Haldane, Yim, and Fearing, 2017; Hwangbo et al., 2018). By increasing the duration of the ground phase, the on-board actuators can be less powerful and lighter while still having sufficient time to provide the required energy to the system. Work has also been done to increase the capabilities of robots to jump higher (Salton et al., 2010; BostonDynamics, 2012) or over difficult terrain (C. M. Hubicki et al., 2016). For the case of the Urban Hopper and the Sand Flea robot, extreme jump heights were achieved using compressed energy that could be released at a chosen moment. However, storing the amount of energy needed to jump to these heights required more time than the robot was in the air, preventing the immediate succession of another jump. This type of energy release could also be unsafe due to the high impulse force.

With the end goal of developing robots which can freely hop around in safe ways, the group at Disney Research and Development began working towards hopping

with safe actuation via their LEAP robot (DisneyResearchHub, 2016). The idea of safe actuation entails placing an elastic element between the robot and the world and, in doing so, creating a limit on the impulse that can be transmitted between the two. This can be best seen in the patent (Smoot et al., 2018), detailing a robotic bouncing ball consisting of an elastic shell and internal actuators. Given the size constraints of this elastic element and the robot itself, further limits on the duration of the ground phase, the range of motion for the robot, and total amount of energy storage are placed on the system. Such constraints prevent the use of certain energy input strategies during the ground phase.

The goal of this work is to begin examining methods of hopping which will work in such a scenario while still achieving sufficient stability in the process. The two methods analyzed here for storing energy in the spring are a clutch-release method, similar to the style of the Bow Leg hopper (H. B. Brown and Zeglin, 1998), and a moving-mass method, as in (Aguilar and Goldman, 2016), to generate force during the ground phase.

4.2 Models of 1D Hoppers

Two hopping robots are examined in this work, with different designs and methods of actuation. The coordinates, q, of each model represent the heights of the individual entities from the ground, as shown in Fig. 4.2. The corresponding configuration space is given by $q \in Q \subset \mathbb{R}^n$, where n is the number of coordinates for the given model. For the tangent bundle with coordinates $(q, \dot{q}) \in TQ \subset \mathbb{R}^{2n}$, the hybrid control system is defined as the tuple,

$$\mathscr{HC} = (\Gamma, \mathcal{D}, \mathcal{U}, S, \Delta, FG) \tag{4.1}$$

- $\Gamma = \{V, E\}$ is a *directed cycle* containing vertices V and edges E
- $\mathcal{D} = \{\mathcal{D}_v\}_{v \in V}$ is the set of admissible domains
- $\mathcal{U} \in \mathbb{R}$ is the set of admissible control inputs
- $S = \{S_e\}_{e \in E}$ is the set of guards for domains, which represents the transition point between domains
- $\Delta = \{\Delta_e\}_{e \in E}$ is the set of reset maps between domains
- FG is the set of vector fields representing continuous dynamics of the domains

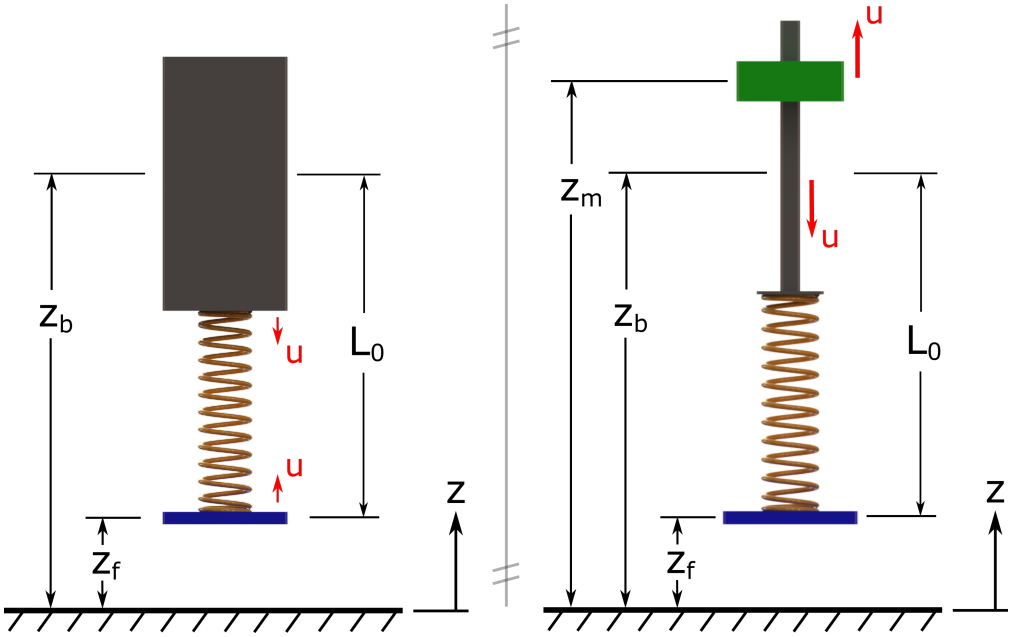


Figure 4.2: Model coordinates for the Compress-Release (left) and Moving-Mass (right) hopping robots.

Fig. 4.3 contains a visual representation of the hybrid domain cycle for each model.

The dynamics during each domain can be written as

$$M\ddot{q} + H(q, \dot{q}) = Bu + J_v^T(q)F_v \tag{4.2}$$

where $M \in \mathbb{R}^{n \times n}$ is the mass matrix, $H \in \mathbb{R}^n$ is a matrix containing the gravity, damping, and spring forces, $B \in \mathbb{R}^n$ is the actuation distribution matrix, and $u \in \mathbb{R}$ is the input force from the actuator. In addition, $J_v \in \mathbb{R}^n$ is the Jacobian of the holonomic constraint in D_v and $F_v \in \mathbb{R}$ is the magnitude of that holonomic constraint force. The method of defining these constraints and their forces is based on the work in (Murray, Z. Li, and Sastry, 1994). During the ground phase this holonomic constraint represents the foot being held in place by ground forces. The beginning of the aerial domain is subject to another holonomic constraint in the form of a hardstop around the spring, preventing the spring from extending past its equilibrium length. As a consequence of this constraint, the foot and body of the robot will move as one entity together until a compressive force is placed on the spring again. The purpose of this hardstop is to allow for a simpler spring design and to cause the robot to leave the ground at a known moment, i.e. when the spring reaches its equilibrium length.

Some of the transitions between domains involve what are assumed to be perfectly plastic impacts, leading to reset maps governing sudden changes of velocity for some

coordinates. In Fig. 4.3 the reset maps at landing and take-off are represented by Δ_L and Δ_T, respectively. The method of getting a reset map is based on the work in (J. W. Grizzle, Chevallereau, A. D. Ames, et al., 2010), where the velocity change is deemed a result of an instantaneous impulse force. The map for landing sets the velocity of the foot to zero instantly, while the reset map at take-off is a conservation of momentum equation between the foot and body given by

$$\dot{z}_{f+} = \dot{z}_{b+} = \Delta_T \dot{z}_{b-} = \frac{M_b}{M_b + M_f} \dot{z}_{b-} \tag{4.3}$$

where M_b and M_f are the masses of the body and foot, and the scripts $-$ and $+$ represent information from before and after the reset, respectively. This means that as the body of the robot rises up and the spring reaches the hardstop, then the foot and body velocities will instantaneously change to a new and equal value.

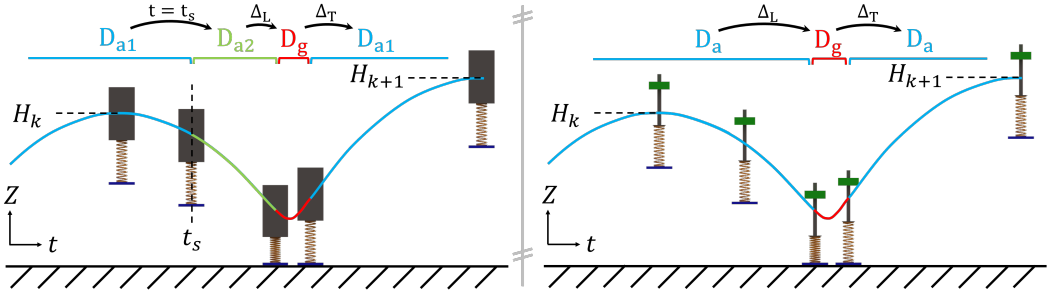


Figure 4.3: Hybrid cycles for both the Compress-Release Hopper (left) and the Moving-Mass Hopper (right).

The following subsections will further detail each robot and their corresponding hybrid control system.

Compress-Release Hopper (CRH)

The CRH hopper uses an actuator within the body to compress a spring and store energy while in the air, and then releases that energy during the ground domain. In order to simplify analysis, a hardstop is placed around the spring which prevents extension past its equilibrium length. This results in the air phase being split between two parts. The first aerial domain has the hardstop active and the spring un-compressed, while the second aerial domain is the spring compression phase when the hardstop is no longer active. The end result is a hopping motion with three domains, as seen in the left half of Fig. 4.3. This cycle starts with the robot entering the first air domain, D_{a1}, with the hardstop in effect until a certain time, $t = t_s$. The robot then begins compressing the spring, removing contact with the hardstop

and entering a second air domain, D_{a2}, where there is no internal constraint. This continues until the foot contacts the surface, beginning the ground domain, D_g. While on the surface, the robot reverses direction and leaves the ground when the body extends to the hardstop length, L_0. At this moment, time is reset and the cycle repeats. The definitions for these domains and their boundary guard surfaces are

$$D_{a1} = \{(t, q, \dot{q}) \in \mathbb{R} \times TQ : z_f \geq 0, t \leq t_s\} \tag{4.4a}$$

$$S_{a1} = \{(t, q, \dot{q}) \in D_{a1} : t = t_s\} \tag{4.4b}$$

$$D_{a2} = \{(t, q, \dot{q}) \in \mathbb{R} \times TQ : z_f \geq 0, t > t_s\} \tag{4.4c}$$

$$S_{a2} = \{(t, q, \dot{q}) \in D_{a2} : z_f = 0, \dot{z}_f < 0\} \tag{4.4d}$$

$$D_g = \{(q, \dot{q}) \in TQ : z_f = 0, z_b \leq L_0\} \tag{4.4e}$$

$$S_g = \{(q, \dot{q}) \in D_g : z_b = L_0, \dot{z}_b > 0\} \tag{4.4f}$$

where here, $q = [z_b, z_f]^T$ and $Q \subset \mathbb{R}^2$.

For the CRH model, there is a specific amount of pre-compression to generate in the spring during the aerial phase, δ^*, that will yield the correct return height of the robot after a full period of motion. Once δ^* has been calculated, as done in Section 4.3, a cubic Bézier polynomial is used as the desired profile for the compression of the spring given by

$$B(\tau) = \delta^*(3(1-\tau)\tau^2 + \tau^3) \tag{4.5}$$

$$\tau = mod(t, t_s)/t_i \tag{4.6}$$

where t_s is the time after entering D_{a1} to begin spring compression, and t_i is the anticipated time to impact from t_s which can be calculated from height and velocity information. To minimize energy consumption of the motor, it is best to set t_s as large as possible; however, in order to ensure that the hardware is capable of following the given trajectory, $B'(\tau) \leq v_{max}$ and $B''(\tau) \leq a_{max}$ need to be enforced over the compression profile, where v_{max} and a_{max} are the maximum velocity and acceleration of the motor, respectively.

Moving-Mass Hopper (MMH)

The second hopper utilizes a moving mass within the body of the robot, which is actuated in series with the spring via a ball screw in order to apply a force to the spring during the ground phase, as seen in Fig. 4.2. The dynamics for this hopper are setup in the same way as in (4.2), but with a few minor changes. The inherent differences between this model and the CRH model revolve around this third mass:

the mover. Firstly, there are now three coordinates used to represent the hopper as seen in Fig. 4.2, so $q = [z_b, z_f, z_m]^T \in \mathbb{R}^3$ and $Q \subset \mathbb{R}^3$. Secondly, the actuator is now a motor moving itself along a ball screw and is therefore acting between the body and mover, which will be reflected in the actuation matrix, B. Lastly, there is no actuator force across the spring so there is only one domain within the aerial phase leading to a simpler domain cycle, as seen in 4.3. There are still the same two impacts due to the ground and a hardstop around the spring, but the aerial domain and its guard are now defined as

$$D_a = \{(q, \dot{q}) \in TQ : z_f \geq 0\} \tag{4.7a}$$

$$S_a = \{(q, \dot{q}) \in D_a : z_f = 0, \dot{z}_f < 0\}\}. \tag{4.7b}$$

4.3 1D Hopping Mechanism Analysis

In this section, orbital stability analysis will be performed for each hopper model using a Poincarémap (M'Closkey and J. W. Burdick, 1993). The Poincarésection is placed at the hop apex, given by

$$\Sigma = \left\{(q, \dot{q}) \in TQ : z_f > 0, \dot{z}_b = 0\right\} \tag{4.8}$$

Four cases of damping and control action will be examined in simulation. The first is with actuation assuming there is no damping in the robot joints and the simulation also includes no damping. The second case will take this same action but in simulation containing damping to see how it performs. The third trial will use actuation based on a robot model with damping during the ground phase and in a simulation with that exact same amount of damping present. The final case will show how the actuation from the third case performs with a larger amount of damping in the simulation.

CRH Model Analysis

The value of compression needed in the spring, δ^*, is found through calculating the amount of spring energy that will make up for the losses from friction and impacts, as well as any desired height change. The complexity of determining this compression value gets increasingly difficult when including the damping terms of the dynamics. In order to analyze the stability of the hopper and its control methods, we will begin with the case where damping is not considered and then work towards the case with damping during the ground phase. By not considering these damping terms, we can determine whether or not the robot is able to converge to the goal height with corresponding model uncertainty.

Due to the spring hardstop, the foot coordinates are fixed relative to the body during flight, which leads to the Poincarésection of this model being a one-dimensional curve in the z_b coordinate. The simplicity of this section allows for the Poincarémap to be easily plotted as a scalar map, as seen in the subplots of Fig. 4.4.

The control input for this system is the amount of potential energy to generate through spring compression during the second aerial domain. In order to calculate these controllers, we also define the following height characteristics:

- H_k, the actual apex height of the current hop
- H^*, the desired height to reach at the apex of the next hop
- H_{k+1}, the actual apex height reached on the next hop
- H_c, the value given to the controller as the apex height of the current hop

Note that this H_c variable is used as a means of simulating uncertainty in the state measurement for the body height. By giving incorrect information to the system, we can observe any bounds on the allowable uncertainty.

Undamped Cases

In order to find the amount of energy needed to input, we must first find how much energy is expected to be lost during our next hop if we were to reach the desired height. We will do this by calculating the energy lost in each impact and also the energy required to make any anticipated height change. In the case without damping, these are given by

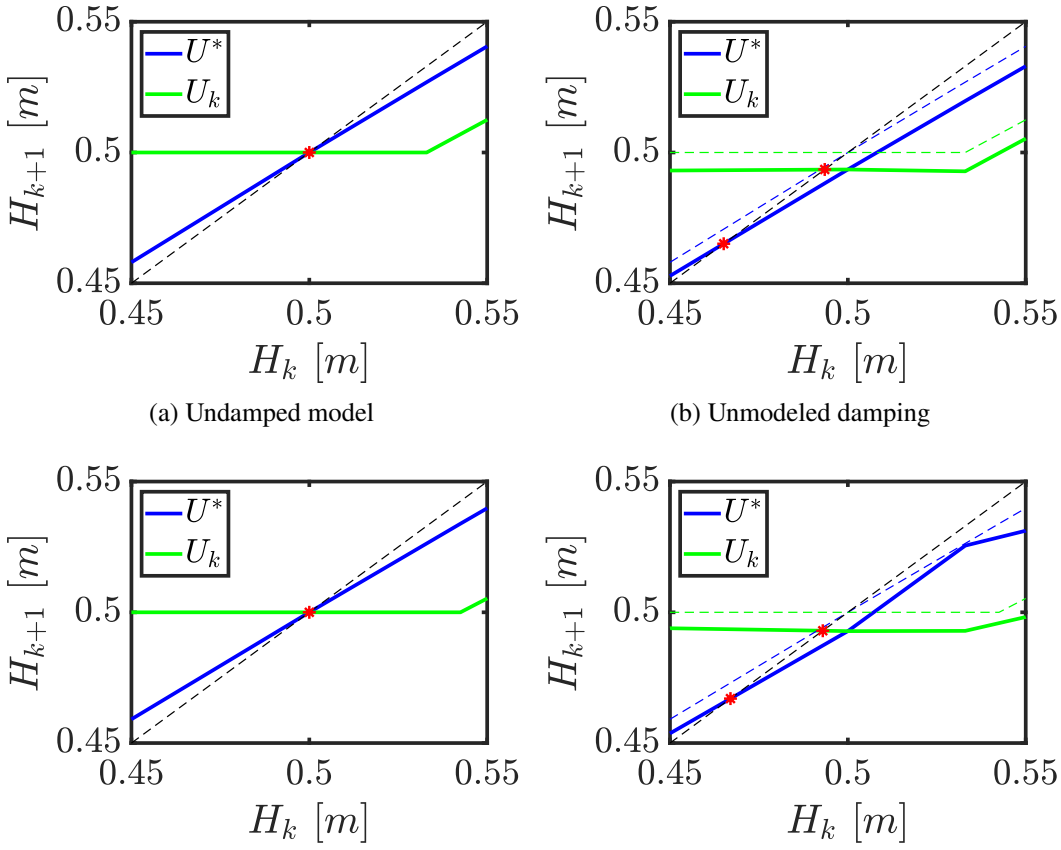
$$E_L(H_c) = M_f g(H_c - L_s) \tag{4.9a}$$

$$E_T(H^*) = (M_b + M_f)g(H^* - L_s)\left(\frac{M_f}{M_b}\right) \tag{4.9b}$$

$$E_H(H_c, H^*) = (M_b + M_f)g(H^* - H_c) \tag{4.9c}$$

where E_L is the energy lost in the landing impact due to the foot contacting the ground, E_T is the energy lost in the take-off impact due to the foot being picked back up off the ground, and E_H is the potential energy required to change height from the current hop height to the next. In order to reach the desired hop height, the energy we input in the spring, $u(H_c, H^*)$, must be equal to the sum of these three energy values

$$u(H_c, H^*) = E_L(H_c) + E_T(H^*) + E_H(H_c, H^*) \tag{4.10}$$



(a) Undamped model

(b) Unmodeled damping

(c) Damped model

(d) Additional damping

Figure 4.4: Poincarémaps for the CRH.

and from this, we get the desired compression:

$$\delta^* = \sqrt{\frac{2u(H_c, H^*)}{k_s}} \tag{4.11}$$

where k_s is the stiffness of the spring.

With this controller in mind, we can assess the stability around a desired hop height. A simple closed-form solution exists for this undamped case. We are able to calculate the total amount of energy lost from impacts for a hop using the given input spring energy, and then determine the final hop height. The total amount of energy added into the system after impact losses is

$$\Delta E = H^* g(M_b + M_f) - \frac{H_c M_b^2 g}{M_b + M_f} - \frac{H_k g(2M_b M_f + M_f^2)}{M_b + M_f}. \tag{4.12}$$

Using this added energy value, we can find the corresponding height change

$$\Delta H = \frac{\Delta E}{(M_b + M_f)g}. \tag{4.13}$$

With this, we can find the actual final height for the system

$$H_{k+1} = P(H_k, H_c, H^*) = H_k + \Delta H \tag{4.14}$$

which is also the Poincarémap of the system under this controller. Substituting (4.12)-(4.13) into (4.14) yields the full form of the Poincarémap

$$P(H_k, H_c, H^*) = H^* - \frac{M_b^2(H_c - H_k)}{(M_b + M_f)^2}. \tag{4.15}$$

Illustrated in Fig. 4.4a is the Poincarémapping for control action taken based on two different choices of H_c: either $H_c = H^*$, labeled as U^*, or $H_c = H_k$, labeled as U_k. In the case of the former, we would not be measuring the actual height of the robot, but instead always choose the control input that is associated with our theoretical periodic solution. On the other hand, when we set $H_c = H_k$ we are using the exact information of the current hop height. The convergence of each of these cases, i.e. the stability of the Poincarémap, can be seen in the figure, or found by taking the derivative of the Poincarémap with respect to the current hop height:

$$\frac{\partial P}{\partial H_k} = \frac{M_b^2}{(M_b + M_f)^2}(1 - \frac{\partial H_c}{\partial H_k}). \tag{4.16}$$

When $H_c = H_k$, this derivative evaluates to 0, which shows that our controller would drive the robot to the desired height in a single hop as a deadbeat controller. For the case where $H_c = H^*$, the derivative is slightly less than 1, which still yields asymptotic convergence, albeit slowly.

Due to the damping present in the hardware, the plots in Fig. 4.4a do not show an accurate representation of the return height for a hop starting from each H_k. In order to see the performance of the controlled system with damped equations of motion, these same controllers were used in simulation which integrated the dynamics through each domain to give the return height. The same controllers from before were used, and yet still gave convergence as seen in Fig. 4.4b. Here the solid lines are the curves from the damped dynamics, while the dashed lines are that from the previous case. It can be seen that both controllers still lead to convergence, just to a different point then our desired height. Now, using the actual previous height yields not only faster convergence, but convergence to a height that is closer to our desired height. As the damping value increases away from the ideal un-damped model, the convergence point will start to move away from the desired height.

Ground Damping Cases

We now consider the case when damping during the ground phase of hopping is added in the form of spring material damping. Since the air domains here remain deterministic, the key calculation only relates to the dynamics of the ground domain. To this end, only the height of the body needs to be used, as the foot is always fixed to the ground during this time. To simplify notation in this section, the body height coordinate is referred to as z. The dynamics of the ground phase become that of a single degree of freedom spring-mass-damper system, which has a known solution of

$$z(t) = e^{at}(z(0)\cos(bt) + \frac{-az(0) + \dot{z}(0)}{b}\sin(bt)) \tag{4.17}$$

$$a = -c_s/(2M_b)$$
$$b = (k_s/M_b - c_s^2/(4M_b^2))^{1/2}$$

where $z(0)$ and $\dot{z}(0)$ are the starting position and velocity of the body, and c_s is the damping coefficient of the joint. Note that this is the solution around the equilibrium length of the spring. Therefore, the L_0 term will be excluded until later since this will only shift the curve. For our case, (4.17) represents the equation of motion during the time between impact and take-off. For a given hop height, H_k, and spring compression, δ_k, the height of the next hop can be calculated from the known initial conditions at the landing impact, $z_L, \dot{z}_L$, and moving forward in time to the known boundary condition at take-off, $z_T = 0$. With this information, the duration of the ground domain can be found analytically through the expression

$$t_g = -\frac{2}{b}\arctan\left[\frac{c}{b} + \frac{a}{b} - \frac{\dot{z}_L}{bz_L}\right] \tag{4.18}$$
$$c = \sqrt{a^2z_L^2 - 2az_L\dot{z}_L + b^2z_L^2 + \dot{z}_L^2}$$

Substituting t_g into the derivative of (4.17), the velocity of the robot before take-off, $\dot{z}_{T-}$, can be found. The height of the next hop is then found using this velocity and the impact map of take-off yields

$$H_{k+1} = \frac{(\Delta_T\dot{z}_{T-})^2}{2g} + L_0 \tag{4.19}$$

Conversely, the method of determining the required amount of compression to reach a certain hop height involves using the desired states at take-off, and then working

backwards towards the initial conditions at landing. The take-off states are found by using the same boundary conditions as before. The landing states are unknown, but each are a function of the desired spring compression, δ^*. Taking these conditions along with (4.17) and its derivative, and simplifying yields the equation

$$e^{2at_g}\left(\frac{a^2}{b^2}\sin^2(bt_g)+\cos^2(bt_g)\right)=\frac{2g}{v_0}\left(\frac{e^{at_g}}{a}\sin(bt_g)-\frac{H_k-L_0}{v_0}\right) \tag{4.20}$$

which is only a function of the ground phase duration, t_g, that is now a negative value. Solving this transcendental equation for t_g must be done numerically. Substituting that value into (4.17) along with the take-off conditions, will yield the desired compression needed to overcome even the energy lost from damping. Using the fsolve function in MATLAB to find this value, a similar Poincaréplot is created and shown in Fig. 4.4c. Once again, exact convergence can be achieved in a range around the desired hop height with proper height measurement, and quick convergence elsewhere. Even in the case where steady-state control is used, the convergence is slightly faster than that of the un-damped case which is shown by dashed lines in the figure.

Simulation was also performed for the case with damping set 50% higher than the controller model. The results of this in Fig. 4.4d, show that once again the point of convergence is lowered from the desired point. The dashed lines here represent the curves from the ideal damped case. The exact convergence height relative to the desired is based on the level of uncertainty for both the height measurement and model damping. These results show that the robot should be stable under this control method, even with poor measurement of the hop height and some uncertainty in the damping estimation. It is clear that having a better estimate of the actual damping in the system will allow for a better choice of compression actuation.

MMH Model Analysis

This model uses an additional moving mass inside the body to input energy, which gives a third coordinate in the dynamics, z_m, and another source of internal friction. The presence of this moving mass inherently allows for many possible motions that will lead to a desired hop height change, since this is determined by the energy propagated to the spring during the ground phase. However, due to the indirect method of adding energy into the system and the extra friction, it becomes non-trivial to find those solutions. These issues point towards using trajectory optimization to generate the desired motions, and then tracking those trajectories during simulation

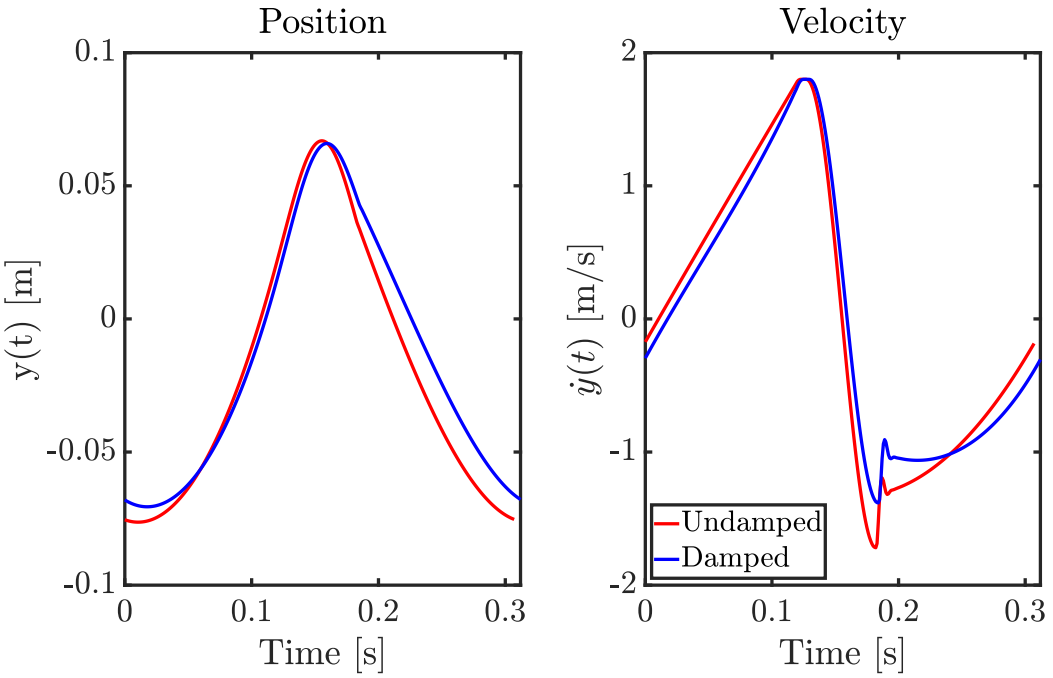


Figure 4.5: Example trajectories for the y coordinate for both the damped and undamped models.

or in practice. Consequently, characterizing the stability of these motions will be done numerically in MATLAB.

Generating Periodic Motions

The trajectory optimization package used in this project is *GPOPS-II* (Patterson and Rao, 2014), which utilizes the *ipopt* solver. This software package allows users to specify the continuous and discrete dynamics, constraints, initial conditions, and the cost function in a convenient way within MATLAB and then maps those setup criteria into the form needed by the solver. An initial guess for the solution can be given in this framework as well. For this problem, it was found that the initial guess needed to be feasible, but not accurate in order to yield a solution.

The general optimization formulation is

minimize:

$$J(t) = \int_{t_0}^{t_f} U^2(t)dt \tag{4.21}$$

subject to:

$$\begin{aligned} \dot{x} &= f(x) + g(x)U(t) \\ x_{min} \leq x &\leq x_{max} \\ y_{min} \leq y &\leq y_{max} \\ \dot{y}_{min} \leq \dot{y} &\leq \dot{y}_{max} \\ -u_{max} \leq U(t) &\leq u_{max} \\ x(t_0) &= x(t_f) \end{aligned}$$

where $U(t)$ is the computed trajectory of the input force, $x = (z_b, z_m, z_f, \dot{z}_b, \dot{z}_m, \dot{z}_f)^T$ is the system state, and $y = z_m - z_b$ is the relative degree of freedom that is directly controlled by the actuator. For a given hop height, the initial conditions for the body and foot states are fixed, while the remaining mover states are only given bounds within the optimization framework. Note that the trajectory $U(t)$ is different from the control input u, so that the controllers may be formulated using this trajectory as a feed-forward term.

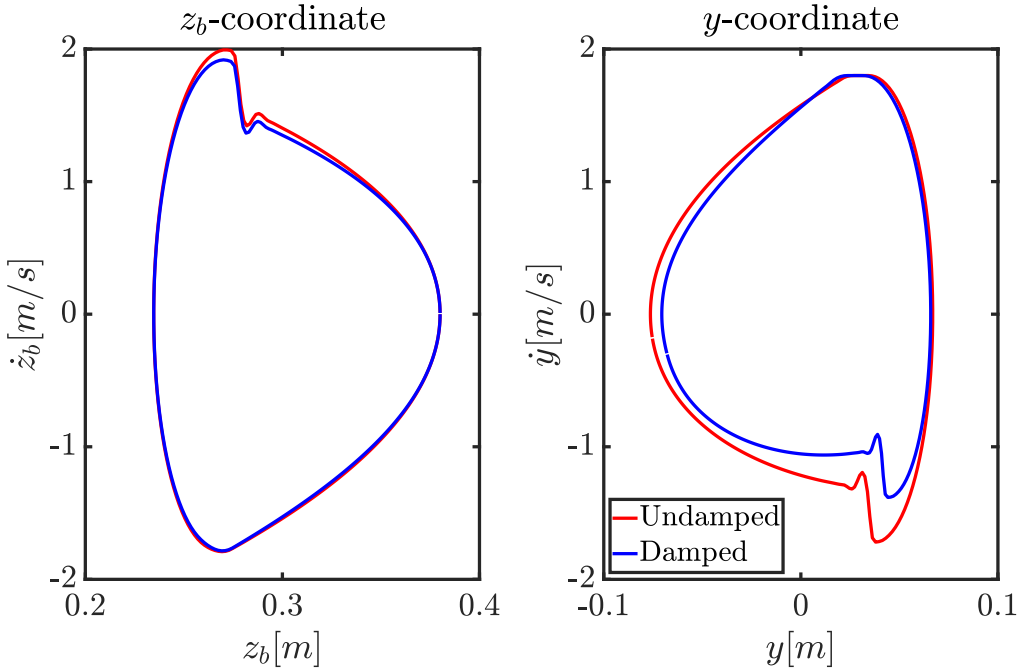


Figure 4.6: Phase space representation for the z_b and y coordinates for both damped and undamped models.

Using this method and the model parameters from hardware, trajectories were generated for the robot spanning a hop height range 0.32 m to 0.42 m. For this section, trajectories for a 0.38 m height with and without damping will be explored

as an example. Fig. 4.5 shows the trajectory of the y coordinate for these example motions, while Fig. 4.6 shows the phase space orbits of both the body and relative coordinates of the robot.

Stability Analysis

Two controllers were run in simulation: open-loop (4.22), and open-loop with PD feedback on the y coordinate (4.23).

$$u(t) = U(T) \tag{4.22}$$

$$u(t) = U(T) + k_p(y_a - y_d) + k_d(\dot{y}_a - \dot{y}_d) \tag{4.23}$$

where $T = mod(t, t_f)$ and t_f is the duration of the generated hop motion. Once again, Poincaréanalysis was used to assess the stability of the system under these controllers. With the additional coordinate, the Poincarésection for this model becomes a three dimensional surface along $(z_b, z_m, \dot{z}_m)^T$. Despite that the value of the mover coordinate is not critical for reaching the desired height, it still needs to be stable within the allowable range of motion due to hardware constraints. This necessitates achieving stability for all three dimensions of the Poincarésection, which can be determined by looking at the Jacobian of the section and its eigenvalues, λ_i, which are found using the MATLAB function *fsolve*. The system is stable if $|\lambda_i| < 1$ for all eigenvalues.

This was first done with the open-loop controller. The results in Table 4.1 show that the undamped system is not stable, while the optimal motion for the damped model is stable. However, the open-loop controller on this damped system had a small domain of attraction and was not robust to uncertainty in the damping terms. From here, a PD controller was added to improve stability and the domain of attraction. The gains of this controller were tuned to give optimal stability, while maintaining the input limits of the hardware. The right side of Table 4.1 shows that stability was dramatically improved with this controller. The domain of attraction also improved, now allowing for a wider range in initial conditions that lead to convergence.

This stability analysis shows that (4.23) is robust to uncertainty in the hop height measurement, but it does not take model uncertainty of the damping terms into account. With simulation, it was shown that the chosen gains were not able to keep the system stable with uncertainty greater than 5% for the damping terms of the spring and the internal relative joint. The right-most column of Table 4.1 contains the stability analysis of the controller when gains were tuned to perform better with

Table 4.1: Eigenvalues of PoincaréMap Jacobian

	OL Undamped	OL Damped$_1$	OL+PD Undamped	OL+PD Damped$_1$	OL+PD Damped$_2$
λ_1	1.4941	0.5177	0.0368	0.0757	0.5942
λ_2	0.8647	0.6395	0.0170	0.0312	0.3942
λ_3	1.4733	0.0630	0.0952	0.1277	0.0054

unmodeled damping. Compared with the ideal case, this controller took a larger number of hops to stabilize, but was able to do so with parameter uncertainty of up to 20% in the damping terms.

4.4 Moving-Mass Hopper Experiments

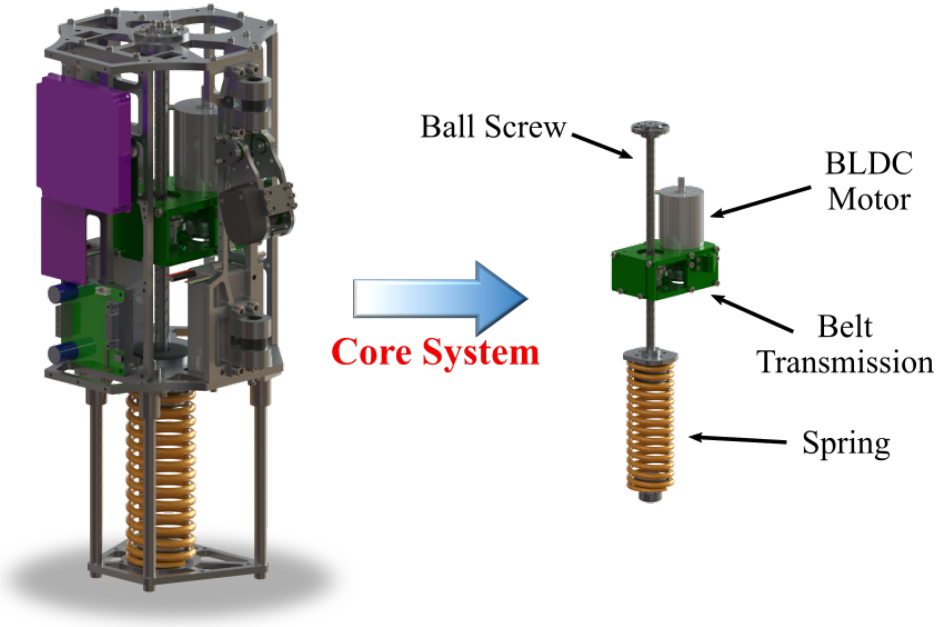


Figure 4.7: CAD model of the MMH, with a clear view of the core components.

Running experiments with robots is valuable for showing the performance of these concepts in the real world. Currently, a robot for the MMH model has been custom built for this purpose. Fig. 4.7 shows the CAD model for the robot and a clear view of the inner workings. The force is input from a brushless DC motor connected to a ball screw via pulleys and a belt. This motor moves itself and some additional mass vertically along the ball screw, generating force onto the body of the robot, and therefore, onto one end of the spring. Using the known specifications of this motor and the ball screw transmission, max linear force and velocity of the actuator

were found to be 100 N and 1.75 m/s, respectively. Due to the size of the body, the relative coordinate, y, is limited to ± 0.1 m, restricting the range that the mover can deviate from the body. The breakdown of mass within the robot is 55% in the body, 31% in the mover, and 14% in the foot.

These model parameters and constraints were given to the optimization described in Section 4.3, resulting in hopping motions up to 0.42 m. The max force of the motor and mass of the foot prevent the robot from hopping any higher. These trajectories were taken and given to the controller on the robot in the form of desired input versus time. Experiments were run for each control method described previously, with the initial conditions being provided by the user dropping the robot from a chosen height. Data was collected from two on-board encoders monitoring the height of the body and the y coordinate. The hopper is constrained to be vertical at all times by linear bearings and rails on either side of the robot.

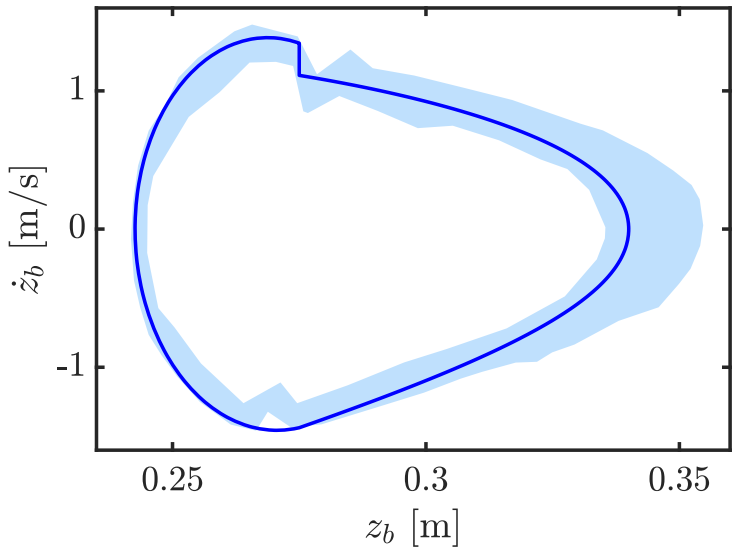


Figure 4.8: Experiment hopping data (shaded blue) compared with simulated trajectory (dark curve), showing the existence of a periodic orbit.

Experiments were run for at least 30 consecutive hops to show long term stability. It was seen in experiment that the initial condition of the drop did have to be fairly close to the ideal, but did not need to be perfect. This was apparent since the hopper needs to settle into the periodic orbit. This could take anywhere from 1 to 15 hops, depending on the timing and height of the initial drop. Fig. 4.8 shows data collected from an experiment for a hop height of 0.34 m, containing about 40 hops and an initial condition that was too high. The robot took about 8 hops to settle into its stable hop height. The data shows that the robot was able to remain around the simulated periodic orbit throughout the experiment. The two sections of data near

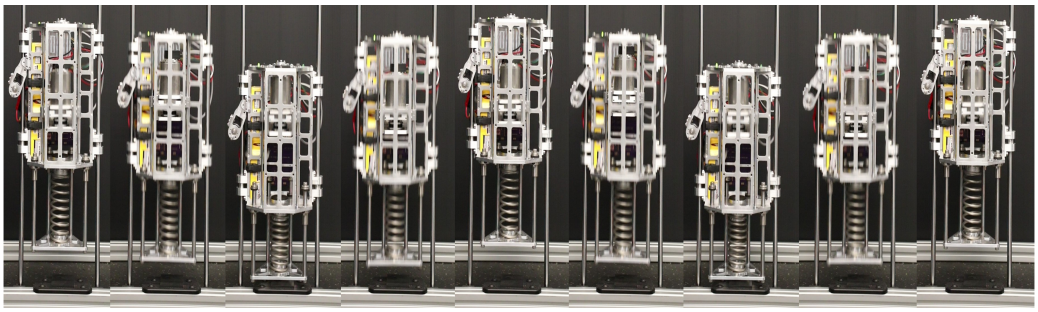


Figure 4.9: Motion tiles of MMH robot over two consecutive hops.

the top and bottom of the plot show an oscillation in velocity of the robot that is not seen in simulation. This is likely due to the assumption of perfectly plastic impacts being not entirely true, showing that these impacts had a small bounce before coming to completion. Fig. 4.9 shows motion tiles of a two consecutive hops from the same experiment on our rail system. The video of these experiments can been seen in (E. Ambrose, 2019).

Chapter 5

IMPROVING HOPPING EFFICIENCY THROUGH DESIGN

After the preliminary efforts described previously studying hopping mechanisms, this work transitioned toward trying to improve the efficiency of the moving-mass mechanism in particular. The work in this chapter will talk about the many changes made on both hardware and in the optimization space towards this goal. The work presented here was originally published in (E. Ambrose and A. D. Ames, 2020; Eric Ambrose, Wen-Loong Ma, and Aaron D Ames, 2021).

5.1 Review of Efficient and High-performance Hopping

Hopping is a mode of locomotion that undergoes rapid changes in contact with the world; as a result, advanced methods from nonlinear and hybrid dynamics are well-suited for its analysis (Sinnet et al., 2011), (Westervelt, J. W. Grizzle, and Koditschek, 2003). Just as in the case of running with legged robots (W. Ma, Kolathaya, et al., 2017), (J. Park et al., 2014), hopping consists of long airborne phases and shorter ground contact phases separated by dramatic discrete impacts. These impacts, along with friction and other forces, cause energy to be lost at every hop. This necessitates adding energy back into the system through its actuators in order to reach the desired hop height in a controlled manner. The most straight forward way to accomplish this is to apply force on the world during the ground contact phase to assist the robot's ascent. Due to the limited time on the ground, this leads to a need for brief and high power actuation coupled with compliant elements to store and return energy.

Over the past few decades, robotic hoppers have been developed with many approaches to the challenges of design and control (M. H. Raibert, 1984), (M. H. Raibert, H. B. Brown, and Chepponis, 1984), (Fiorini and J. Burdick, 2003). These robots successfully hop for long periods of time, avoid large obstacles, and perform extreme behaviors such as flips (M. Raibert, 1986). Beyond this, they helped set the standard for the level of control to be expected for such a dynamic type of locomotion. One of the main focuses of hopping robot design has been to improve the efficiency of locomotion (H. B. Brown and Zeglin, 1998), (Hwangbo et al., 2018), (Haldane, Yim, and Fearing, 2017) to allow for long-term hopping using on-board power. These improvements were achieved through a variety of means such as

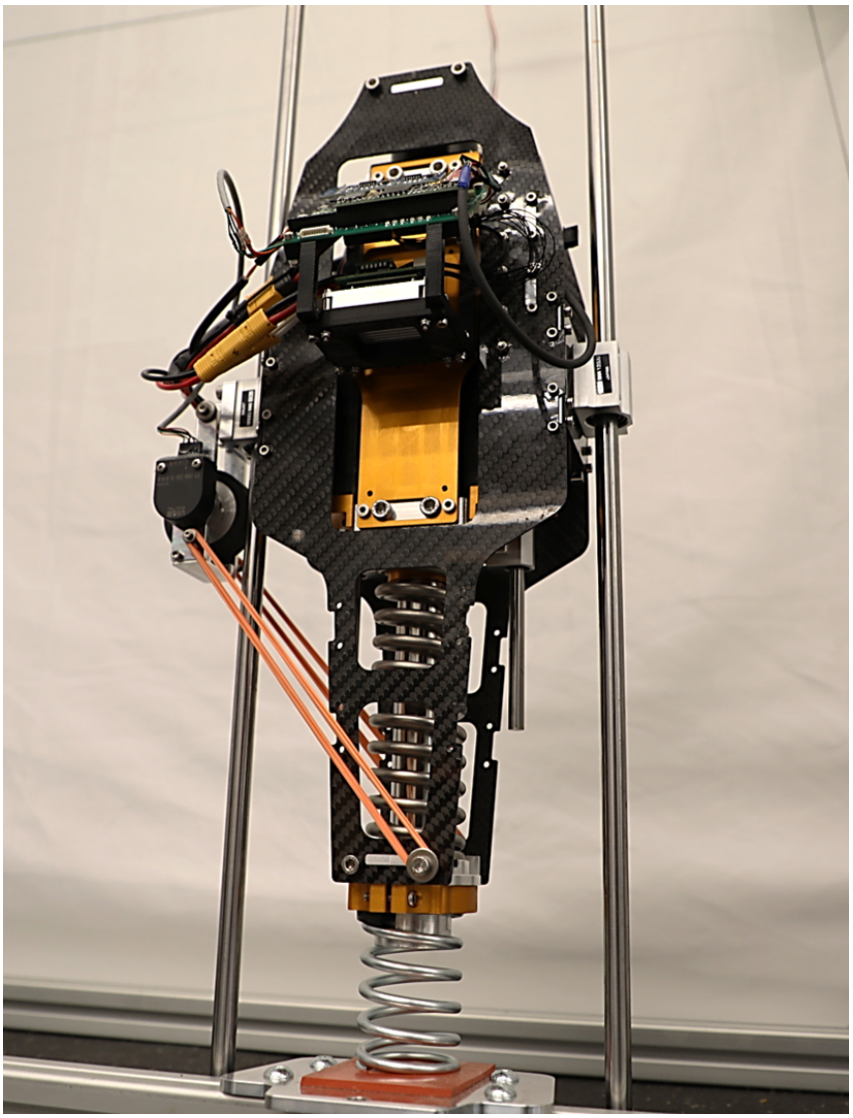


Figure 5.1: A 3-mass, vertically-constrained hopping robot with two springs: one in parallel with the actuator and one below in series with the actuator.

lessening the mass of components, using more direct actuation methods, or using more efficient elastic elements. With improved efficiency of actuation, researchers have been able to push the capabilities of the robots further towards higher hops (Hwangbo et al., 2018), (Haldane, Yim, and Fearing, 2017) and over complex terrain (C. M. Hubicki et al., 2016).

When using high-power actuation in critical environments such as when humans are present, the task of safety becomes a critical concern. The issue of robotic safety has been brought up in the field of robotic manipulators, due to the more recent prominence of manufacturing and surgical robots. Measures have been to

taken to deal with this issue in a variety of ways, including the addition of elasticity (Yoon et al., 2005), (Lens and Stryk, 2012). In the realm of robotic hopping, safety can be improved by placing a spring between the robot and the world, as was done with the LEAP robot developed by Disney Research and Development (DisneyResearchHub, 2016). The goal of putting an elastic element here is to limit the impulse force transferred from the robot to prevent any possible damage or harm on the surroundings. This can be further seen in the patent of a robotic bouncing ball (Smoot et al., 2018), where all actuators and mechanisms are contained within the ball itself.

With the goal of finding robotic hopping solutions that fit within the paradigm of this safe actuation, previous research (E. Ambrose, N. Csomay-Shanklin, et al., 2019) examined two methods of hopping: a clutch-release hopper similar to that of the Bowleg Hopper (H. B. Brown and Zeglin, 1998), and a moving-mass hopper akin to that used in (Aguilar and Goldman, 2016). It was shown that both types of hopping could reach a stable desired hop height through control methods in the span of a few hops. The previous model for the moving-mass hopper had a single spring below the body of the robot which would act as a safe barrier between it and the world, while also providing the necessary energy storage to achieve periodic hopping. However, the results with this model showed a need for large actuation force in order to reach a desired hop height.

This work focuses on improving the design of the moving-mass hopper model through the addition of a second spring, placed in parallel with the actuator. Research groups have shown a possible improvement in efficiency to robot locomotion with the incorporation of parallel elasticity (Sharbafi et al., 2019), (Eslamy, Grimmer, and Seyfarth, 2012). Furthermore, it has been shown that parallel elastic actuators can be used to improve the tracking of motion trajectories (Mettin et al., 2010), such as those used in the original moving-mass model. A new robot was created for this work, with the addition of the second spring in parallel in order to examine the benefits of parallel elasticity.

5.2 The Double-Spring Hopper Model

The original moving-mass model was made up of three masses and a single spring, acting between the body and foot masses. The third mass, the mover, was contained within the body and could be moved vertically with the use of linear actuation. This arrangement had the spring in series with the actuator which would serve to redirect

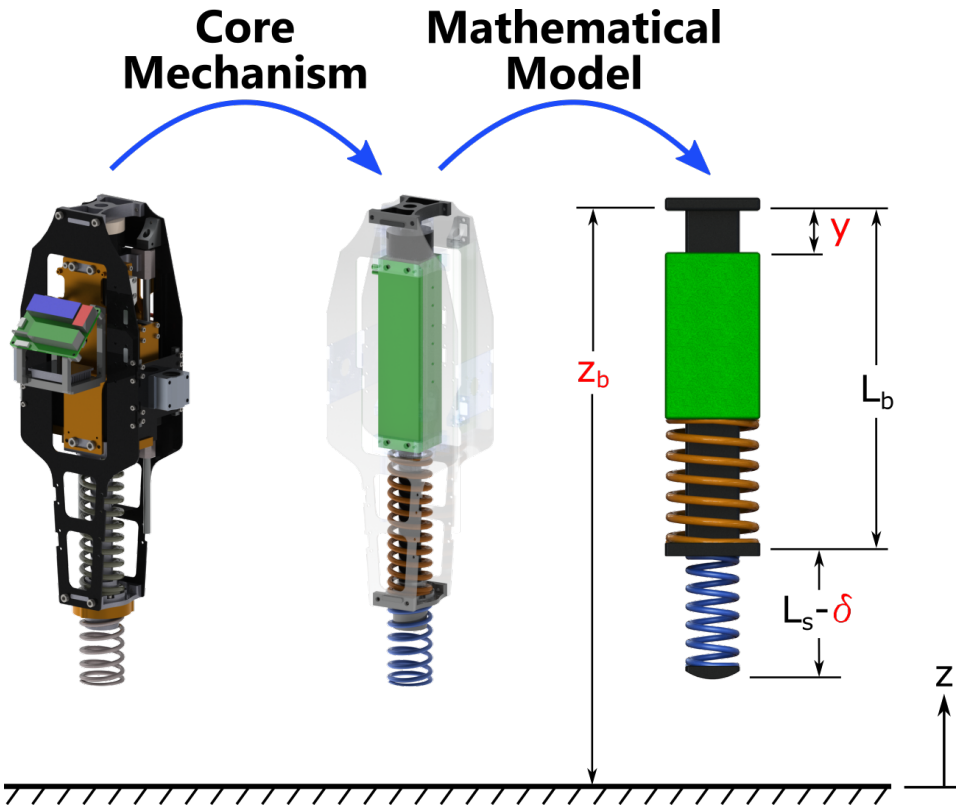


Figure 5.2: Model coordinates and design for the double-spring, moving-mass robot, with a look at the core mechanism.

the motion of the body back upwards after an impact with the ground. The actuator could slide the mover impulsively while the robot was on the ground in order to assist the reversal of the body's velocity. The actuator also needed to apply force in order to keep the mover from reaching the end of its range of motion and creating an internal impact which could reduce the energy in the system and lead to an unwanted loss of hop height.

Unlike this previous model, the new model presented here contains an additional spring between the body and mover which acts in parallel with the actuator (Fig. 5.1). By adding this second spring here, the mover will now undergo its own passive reversal of velocity during the ground phase. A caveat of this design, is that the mover will now contact the body when the spring is at its equilibrium length, meaning that the spring will only compress and not extend past this point. This leads to internal contact occurring between the body and mover at the end of the second springs compression cycle.

Dynamics

In order to model this new double-spring hopping robot, the coordinates of the configuration space are chosen to be $q = (z_b, y, \delta) \in Q \subset \mathbb{R}^3$, where z_b is the height of the body from the ground, y is the deflection of the upper spring in parallel, and δ is the deflection of the lower spring in series. These are shown in red within Fig. 5.2, along with two of the constant length parameters of the robot model. The unpinned dynamics were derived using these coordinates and Lagrangian mechanics, then put in the form

$$\underbrace{\begin{bmatrix} M_0 & -M_m & M_f \\ -M_m & M_m & 0 \\ M_f & 0 & M_f \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \ddot{z}_b \\ \ddot{y} \\ \ddot{\delta} \end{bmatrix}}_{\ddot{q}} + \underbrace{\begin{bmatrix} c_b \dot{z}_b + M_0 g \\ F_p - M_m g \\ F_s + M_f g \end{bmatrix}}_{H(q,\dot{q})} = \underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{B} u \tag{5.1}$$

where M_0 is the total mass of the robot, M_m is the mass of the mover, M_f is the mass of the foot, c_b is the damping coefficient of the body moving vertically, F_p is the extension force in the upper spring in parallel, F_s is the extension force of the lower spring in series, and g is the gravitational acceleration constant. The full form of the spring forces includes a term of the stiffness force and a term for the damping force, i.e. $F_p = k_p y + c_p \dot{y}$, where k_p and c_p are the stiffness and damping constants of the spring in parallel. The 'foot' of the robot is a virtual mass at the lower end of the spring in series, containing a third of that spring's mass. The height of the foot is defined as $z_f(q) = z_b - L_b - L_s + \delta$.

The sole difference in the unpinned dynamics of this robot model from the single-spring case comes from the F_p term, which is highlighted in (5.1). For the case of the single-spring model, this term was just the damping force between the mover and body. Despite this small difference in the unpinned dynamics, the complexity from the internal contact is more significant, as will be shown in the following subsection.

Hybrid Structure

Beyond the dynamics described in (5.1), there are also external and internal forces acting on the robot intermittently during each hop, which come from the contact with the ground, as well as a range of motion limit placed around the spring in parallel within the body. This hardstop prevents the upper spring from extending past its equilibrium length. The mover is able to move downwards within the body, compressing the upper spring, but will impact the body when it reaches the

equilibrium position again. At the point of some of these discrete events an impact occurs, such as when the foot reaches the ground. Due to these changes in the dynamics and the presence of discrete impact events, the dynamical model can be re-written as a hybrid control system. For the tangent bundle with coordinates $(q, \dot{q}) \in TQ \subset \mathbb{R}^6$, the hybrid control system is defined as the tuple,

$$\mathscr{HC} = (\Gamma, \mathcal{D}, \mathcal{U}, S, \Delta, FG) \tag{5.2}$$

- $\Gamma = \{V, E\}$ is a *directed cycle* containing vertices V and edges E. Fig. 5.3 shows the cycle for the double-spring model
- $\mathcal{D} = \{\mathcal{D}_v\}_{v \in V}$ is the set of admissible domains
- $\mathcal{U} \in \mathbb{R}$ is the set of admissible control input forces
- $S = \{S_e\}_{e \in E}$ is the set of guards for domains, which represents the transition point between domains
- $\Delta = \{\Delta_e\}_{e \in E}$ is the set of state reset maps between domains, defined in (5.7)
- FG is the set of vector fields representing the control system dynamics for each domain, defined in (5.3)-(5.6)

For the case of this double-spring hopper model, there are four possible domains for the robot as shown in Fig. 5.3. These are based on the state of ground contact and internal hardstop contact, which are visually represented in the figure by the gray animation effect lines. The arrows in the figure represent the possible domain switching that can occur. Note that there is only one diagonal arrow in the diagram. This is due to an assumption that two contact switching events will not occur at the same time, with the exception of if the robot hits the ground while applying zero actuator force, in which case it will transition straight from domain 1 to domain 3. Each contact manifests in the dynamics through a holonomic constraint, which can be added onto the end of our equations of motion to yield the new set of equations augmenting (5.1):

$$M\ddot{q} + H(q, \dot{q}) = Bu + J_v^T F_v \tag{5.3}$$

$$J_v \ddot{q} + \dot{J}_v \dot{q} = 0 \tag{5.4}$$

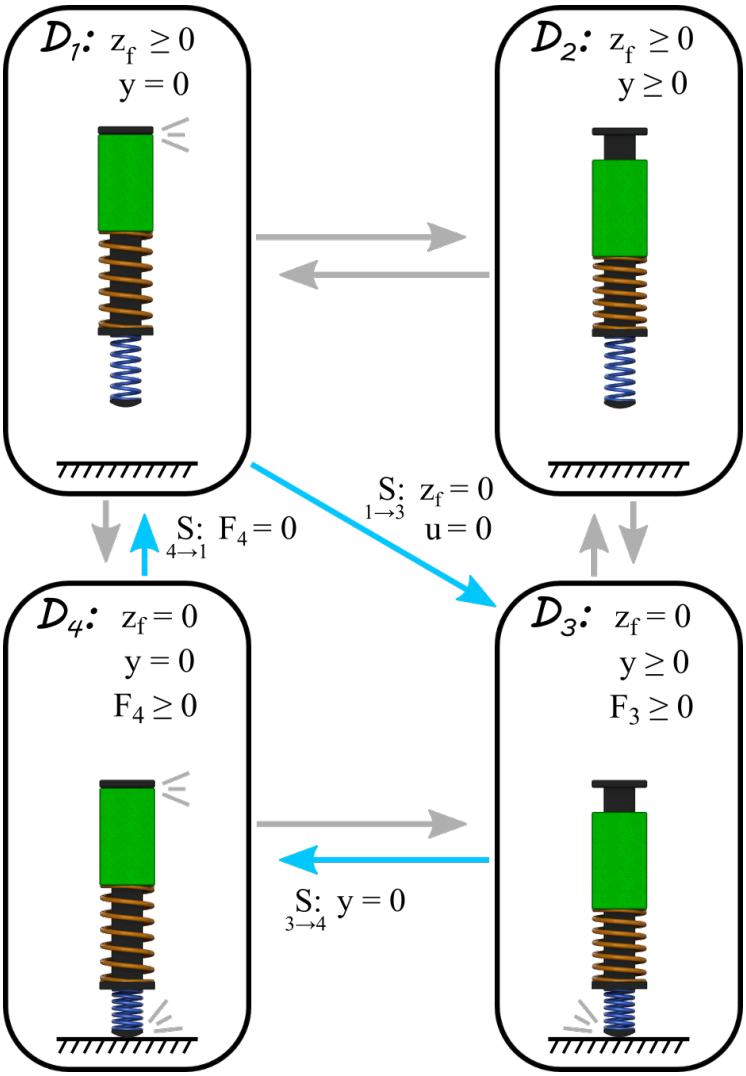


Figure 5.3: Directed cycle for the hybrid system, with vertices depicted as boxes and edges shown as arrows. Blue arrows show the specific domain map for this work, and gray arrows show other possible switches.

where J_v is the Jacobian of the holonomic constraints in domain, D_v, and F_v are the forces of those same constraints. Domain D_2 does not include any contacts, unlike the other three domains. The Jacobians for the contact domains are:

$$J_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \; J_3 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \; J_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{5.5}$$

The method of defining these Jacobians and their forces is based on the concepts presented in (Murray, Z. Li, and Sastry, 1994) where the constraint force, F_v, is found using the following calculation:

$$F_v(q, \dot{q}) = -(J_v M^{-1} J_v^T)^{-1} [J_v M^{-1} (Bu - H(q, \dot{q})) + \dot{J}_v \dot{q}] \tag{5.6}$$

As mentioned before, there are impacts which occur at some of the domain transitions. This includes any time that the foot contacts the ground and anytime that the mover reaches its range of motion limit against the body of the hopper. The impact in each case is assumed to be perfectly plastic, so the corresponding reset map will result in a instantaneous velocity change for some of the coordinates. The method of calculating the reset map, Δ_e, is based on the work in (J. W. Grizzle, Chevallereau, A. D. Ames, et al., 2010), and is defined by the equation

$$\dot{q}^+ = \Delta_e \dot{q}^- = [I - ((M^{-1}J_{v+}^T)/(J_{v+}M^{-1}J_{v+}^T))J_{v+}]\dot{q}^- \tag{5.7}$$

where I is the $3x3$ identity matrix, J_{v+} is the Jacobian of the constraint for the upcoming domain, and $\dot{q}^-$ and $\dot{q}^+$ represent the coordinate velocities before and after the reset, respectively. The result of this reset mapping for ground contact is the foot velocity will be immediately set to zero after the impact, while that for the mover hardstop is based on conservation of momentum principles for the mover and body so they have the same velocity after impact.

5.3 Simulation of the Double-Spring Hopper

Motion Generation

Due to the complexity of hybrid system models—and thereof hopping—that combines discrete and continuous dynamics, motion generation becomes non-trivial. To this end, trajectory optimization is used to find the ideal actuation input to follow. The method of optimization utilized here is similar to that used in (W.-L. Ma, Hamed, and A. D. Ames, 2019), with the nonlinear programming formulated as

$$\begin{aligned} \min_{q(t),u(t)} \quad & \int_{t_0}^{t_f} u^2(t)dt \\ s.t. \quad & \mathbf{C_1}.\ pinned\ dynamics \\ & \mathbf{C_2}.\ hybrid\ continuity \\ & \mathbf{C_3}.\ physical\ feasibility \end{aligned} \tag{5.8}$$

where constraint $\mathbf{C_1}$ is the dynamics from (5.3) and (5.4), $\mathbf{C_2}$ represents the directed cycle continuity and desired initial conditions constraints, and $\mathbf{C_3}$ includes torque/range of motion limits present in the experimental hardware. The optimization package used for this work was *GPOPS-II* (Patterson and Rao, 2014), which is MATLAB based and allows for multiple domains and discrete events as are required for hopping systems.

From previous work with the single-spring model, only two domains were planned in optimization. This was assuming that the mover would never contact the body,

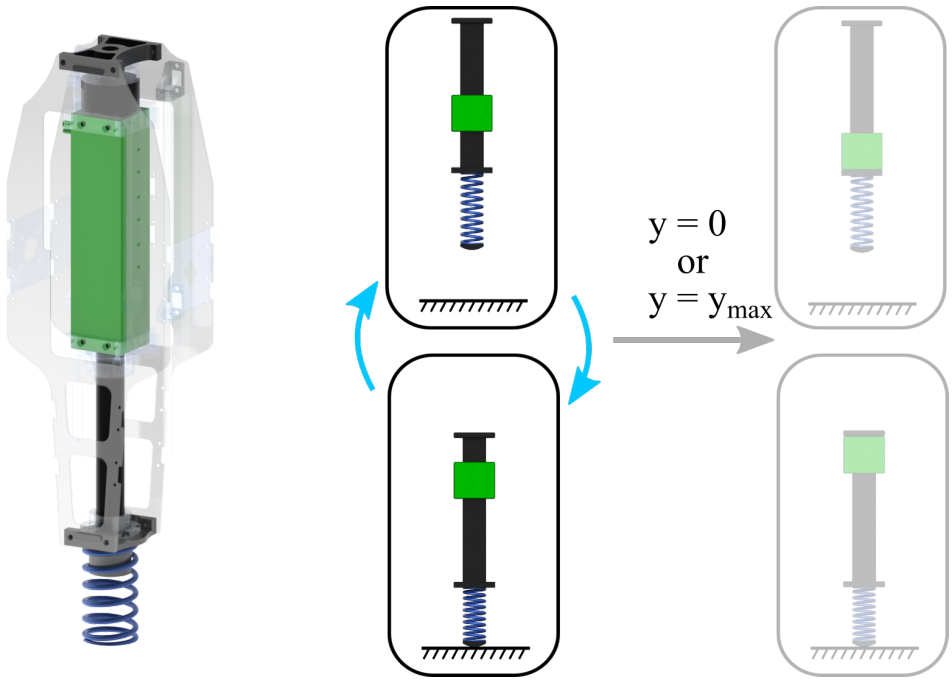


Figure 5.4: The model for the single-spring hopper and its hybrid system, with the possible other domains that were considered, but found sub-optimal shown as greyed-out.

restricting the hybrid structure to only be based on foot contact with the ground. Alternative hybrid domain cycles were explored, allowing for this internal contact and various orders of events; see Fig. 5.4. However, it was found that the optimal choice was in fact to restrict the mover to never reach contact with the body, and again to only use the two-domain structure as done previously. The optimization was re-run for the single-spring model using parameters consistent with the robot to be used in experiment, but with the absence of the second spring and its resulting damping on the motion of the mover.

For the double-spring case, multiple hybrid cycles were examined to find the optimal discrete sequence of events, i.e., the optimal graph Γ to use in the hybrid system model. The best hybrid structure was found with stiffness constants of the two springs being close together and then skipping over domain 2, following the path of $D_1 \rightarrow D_3 \rightarrow D_4 \rightarrow D_1$. The process of going straight from D_1 to D_3 involves applying no input from the actuator until the foot reaches the ground, and then making sure that the initial input force is non-negative. This aligned with the preferred direction of input during the ground phase, requiring no extra constraint in the optimization. Note that in this particular hybrid cycle, there is only one domain when the mover is not locked to the body and when actuation will occur: domain 3.

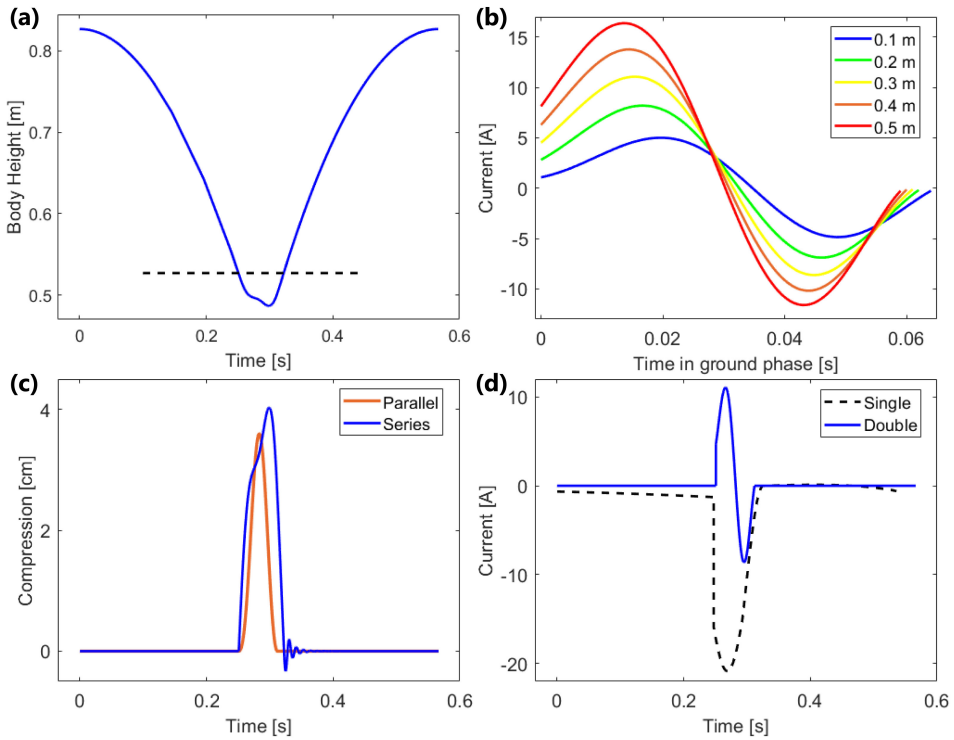


Figure 5.5: (a) Optimal trajectory for the body height with ground contact threshold. (b) Input current profiles for 5 hop heights. (c) Optimal trajectory for both spring deflections (y and δ). (d) Input force comparison for both models.

This optimization was run for multiple hop heights with foot clearance ranging from 0.1 m to 0.5 m. Fig. 4(a) shows the optimal trajectory of the body height coordinate for a foot clearance of 0.3 m, while Fig. 4(c) shows the corresponding optimal spring deflection trajectories. Fig. 4(b) shows the optimal input force trajectories for all 5 hop heights over the time interval of domain 3, when the upper spring is not locked in contact with the body. Fig. 4(d) shows a comparison between the optimal force profiles of the single-spring and double-spring models, both for a hop with foot clearance of 0.3 m. It can be seen that the optimal force trajectory for the double-spring model follows a path similar to that of the spring in parallel, assisting its compression and extension.

Table 5.1 shows further comparisons between both versions of the moving-mass model in these optimizations. Specifically, it compares peak force, mechanical efficiency, and electrical efficiency. Mechanical energy used in each hop is a function of motor force and mover velocity given by

$$E_{MECH} = \int_{t_0}^{t_f} |u\dot{y}|dt \tag{5.9}$$

Current and voltage were estimated as functions of the force and velocity of the

	Single-Spring			Double-Spring		
H_f	F_{max}	η_{MECH}	η_{ELEC}	F_{max}	η_{MECH}	η_{ELEC}
0.1	95.2	25%	17%	50.9	78%	58%
0.2	183.4	26%	16%	98.3	77%	55%
0.3	250.1	29%	16%	132.6	76%	54%
0.4	303.0	30%	15%	165.2	76%	52%
0.5	348.1	30%	15%	196.4	76%	51%

Table 5.1: Key comparison data for the optimization results.

motor using the measured resistance and back emf of the motor. Using these values of current and voltage, the total electrical energy provided in each hop was

$$E_{ELEC} = \int_{t_0}^{t_f} |IV| dt \tag{5.10}$$

where I is the current in the motor, and V is the voltage being sent to the motor from the motor controller. These energy consumptions were compared to the amount of energy lost in the system when the actuator provided zero force throughout an entire hop, in order to get the energetic efficiencies.

It was found that the optimal input of the double-spring model required 40% less peak force than the single-spring model, due to the fact that the motor only assisted the natural motion of the upper spring rather than manipulate the mover completely on its own. The double-spring model also had about $2.5x$ better mechanical efficiency and $3x$ better electrical efficiency, despite the fact that the additional spring resulted in higher damping between the mover and body and, therefore, slightly more inherent energy loss during each hop.

Stability of the Simulation

Hopping was simulated for both models in MATLAB starting with open-loop playback of the input trajectories. At the onset of every hop, time is reset and the trajectory starts again from the beginning in order to observe stability with minimal state feedback. Poincarémap analysis (M'Closkey and J. W. Burdick, 1993) is used to provide a metric to the stability, where the system is considered stable if $|\lambda_i| < 1$ for all eigenvalues of the Poincarésection Jacobian. Just as discovered in (E. Ambrose, N. Csomay-Shanklin, et al., 2019), the single-spring model with updated parameters is not stable using this open-loop plus time reset control method alone. For example, in the case of hopping 0.3 m off the ground, the maximum eigenvalue magnitude was $\lambda_{max} = 1.711$. It is possible to achieve stability by adding a closed-

loop controller around the y coordinate; PD control is used in this example. By closing the loop around this coordinate, the max eigenvalue magnitude becomes $\lambda_{max} = 0.332$, allowing the robot to stabilize from errors in initial conditions.

In contrast, the double-spring model is stable simply through open-loop playback and time reset. The eigenvalue magnitude for the same example hop height is $\lambda_{max} = 0.803$, without the use of a stabilizing controller. This is due to the spring in parallel forcing the y coordinate along a path very similar to that desired, since the the optimal trajectory acts only to assist this deflection. Any small deviations from the optimal path are guided back to the trajectory by the spring's natural dynamics. Furthermore, the fact that control input is only occurring during a single domain lasting roughly 60 ms also limits the negative effects of any error as well.

5.4 Double-Spring Hopper Experiments

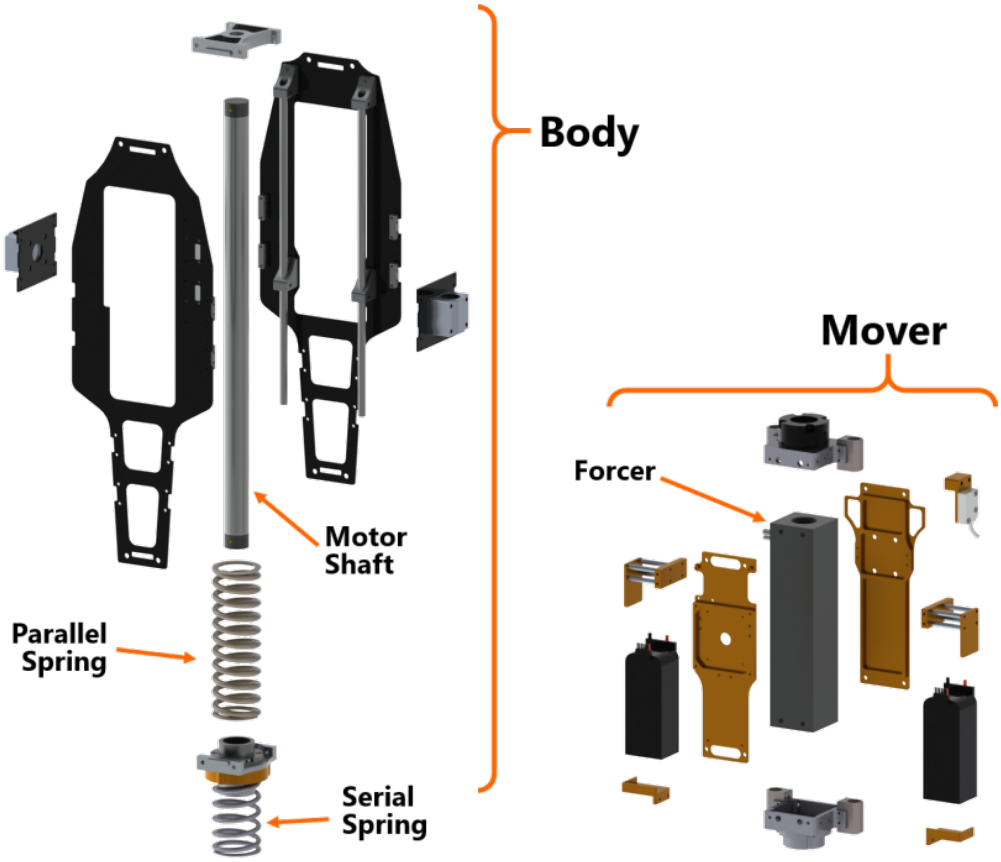


Figure 5.6: An exploded view of the CAD model of the robot, calling out the key internal components.

Hopping Robot Hardware

The one-dimensional hopping robot shown in Fig. 5.1 was developed to demonstrate these concepts on real hardware. The robot is fixed to the vertical axis through the use of two linear rails and ball bearings on either side of the robot. The core of the system is built around a linear brushless DC motor from Nippon Pulse (Pulse, 2019), capable of providing up to 450 N of peak force and 75 N of continuous force. This motor is made up of two parts: a forcer, which is a box shaped 'stator', and a shaft, which acts as the moving 'rotor'.

As shown in Fig. 5.6, the body of the hopping robot is built around the motor shaft, accounting for the majority of the robot's 52 cm height. The mover is connected to the motor's forcer and also contains most of the robot's electronics, including the batteries. This allows for a high percentage of the robot's mass to be on the mover, rather than with the body. The upper spring is placed around the motor shaft to act in parallel with the actuation. In order to minimize the mass of the foot, the lower spring is only held at the top using a clamp mechanism while the lower end is unconstrained. This reduces the amount of energy lost during each impact with the ground, but allows for some off-axis deflection in the spring. Parameter estimation for the system was performed with drop tests of the robot, while the motor was commanded to maintain zero current. The data from these tests was used to identify the three damping coefficients and two spring stiffness constants in the model. While estimates of the stiffness constants were provided by the manufacturer, it was found that the actual values differed slightly and were updated accordingly. Final values the model parameters were: M_0 =6.025 kg, M_m =3.3 kg, M_f =0.05 kg, k_s =17 330 N/m, k_p =18 000 N/m, c_b =2 N s/m, c_s =10 N s/m, and c_p =25 N s/m.

Experiments

Experiments were performed with this robot using the force input trajectories gathered from optimization for hopping motions with foot clearances ranging from 0.1 m to 0.4 m. As done in the simulations for this model, the robot attempted to playback the open-loop trajectory of current to the motor while on the ground. From the initial tests, it was seen that the robot was not returning to its starting height after the impacts with the ground. It was found that this was due to the internal impact between the mover and body not being perfectly plastic. Instead, there was a some bouncing happening at that domain transition leading to less uniform energy transfer. This was corrected by having the motor apply a negative current until lift-off occurred to 'lock' the body and mover together. With this change, the robot was

able to hop back up to its intended height, with noticeably less bouncing during this impact.

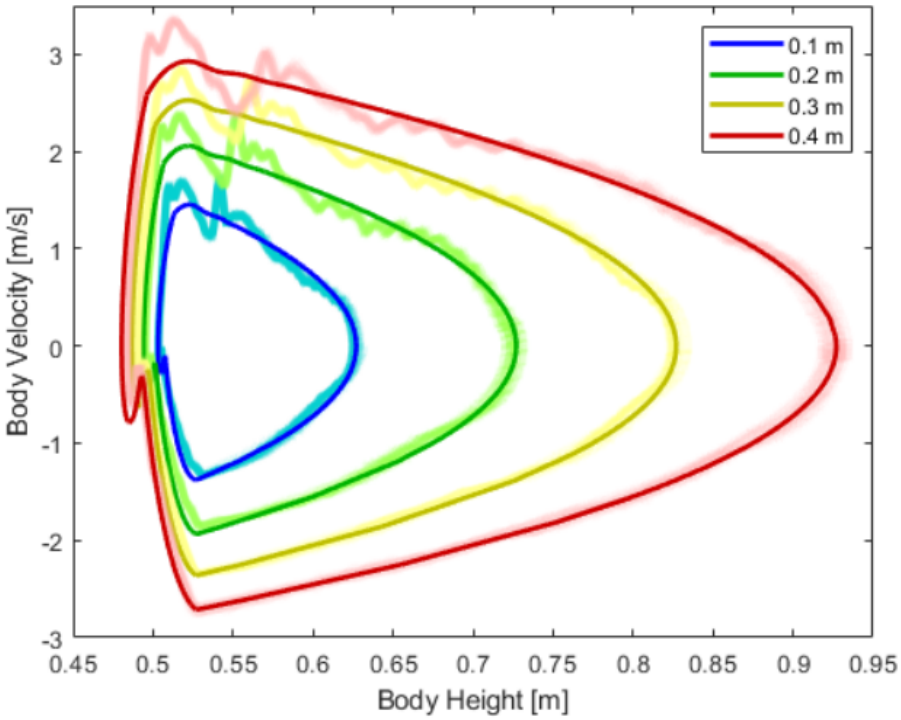


Figure 5.7: Phase portraits of the z_b coordinate during 20 consecutive hops in experiment for each height. The dark curves represents the ideal trajectories from optimization, and the light curves are the experimental sensor data.

Experiments were run at four different heights corresponding to foot clearances of 0.1 m, 0.2 m, 0.3 m, and 0.4 m (experiments can be seen in the supplemental video (E. Ambrose, 2020)). A phase portrait of data from each experiment is provided in Fig. 5.7, showing the periodic nature of the hopping motion. The data further confirms the stability inherent to this hopping model as seen by the very consistent path from the robot during the experiments. The efficiency of the experiments were lower than in simulation, partially due to the locking feature added onto the trajectories. The actual electrical efficiency ranged from 46% to 50%, depending on hop height.

It can be seen that there is significant deviation from the optimal trajectory between the time of the internal impact until the time just after the take-off of the robot. This deviation causes oscillations around the ideal path and is eventually damped out by the time the robot reaches its apex. This is due to the way the robot is attached to the rails in experiment and the fact that the lower spring can laterally deflect while on the ground. The linear bearings are able to rotate slightly from the rails via

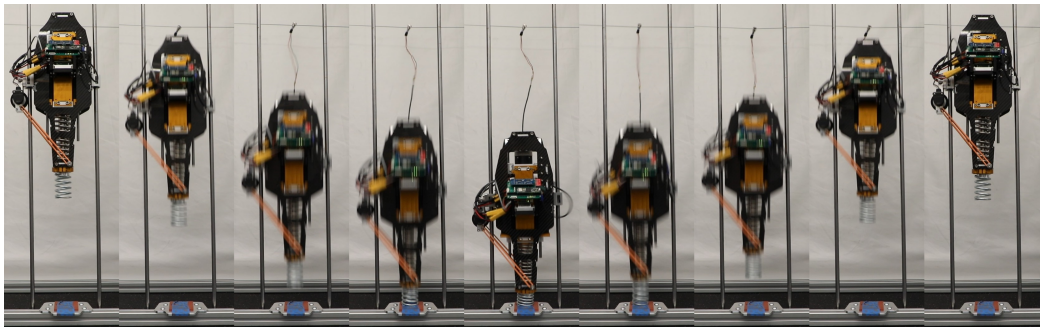


Figure 5.8: Tiles from the 0.3 m experiment over a single hop.

backlash, so the whole robot is able to rock back and forth marginally, leading to oscillations in the body velocity during ascent. This phenomenon can also be seen in the experimental video: (E. Ambrose, 2020).

5.5 Conclusions and Next Steps

A new double-spring version of the moving-mass hopper model was presented and compared with the previously used single-spring model. Trajectory optimization was used to identify the optimal actuation for each model, showing two different patterns of force input. The single-spring case had large force impulses during the ground phase acting in time with the spring in series. Conversely, in the double-spring case, the actuator acted in time with the natural compression of the upper spring in parallel. This led to a prolonged deflection of the spring in series, but an overall more condensed period of actuation. It was shown in simulation that adding this spring in parallel improved hopping performance through greatly increasing energy efficiency, reducing peak actuator effort by roughly 40%, and providing stability to the hopping motion without the need for closed-loop control.

This model was further demonstrated on hardware using a novel, vertically constrained hopping robot designed and built in the lab. The robot was able to reach hop heights up to 40 cm off the ground, while maintaining the improved energetic efficiency. This max height was only limited by the length of the lower spring, which was reaching its solid length at higher hop heights. In the future, an alternative elastic component will be made to replace this spring, allowing for larger deflections and increased energy storage. Additional actuators will also be added to provide the balancing mechanisms necessary for hopping in 3-dimensional space.

The study of system design optimization has a long and rich history in both academia and industry. Shape design in aerodynamics (Lyu, Kenway, and Martins, 2015), (Jameson, Martinelli, and Pierce, 1998) using computational fluid dynamics (CFD)

focuses on designing aircraft shape against airflow from the front and lateral directions. Logic optimization was used by the automated electronic design industry in circuit design given volume constraints for optimal manufacturing cost (Buchfuhrer and Umans, 2011). Structural optimization (Haftka and Gürdal, 1992) designs mechanical systems according to metrics such as minimal weight or increased strength according to specific application scenarios. Lastly, the genetic algorithm uses evolution of population to guide the design of static components for complex environments (Lohn, Hornby, and Linden, 2005).

One such example of design optimization study was from the realm of industrial robotic manipulation tasks, which could greatly benefit from even small improvements to efficiency (Schmit and Okada, 2013). Here, springs were placed in parallel to joint motors and optimized alongside the motion trajectories to boost efficiency. In this case, researchers were able to find a closed-form solution for the optimal spring parameters in terms of the trajectories, which allowed for the optimization to again be considered as a classic trajectory optimization problem. In another example from the field of robotic hopping, model-free design optimization techniques were used to streamline the design iteration process (Saar, Giardina, and Iida, 2018). This method involved hopping experiments, using a robot which could have certain design parameters changed quickly between separate tests rather than using prior simulation to yield a single final design.

However, most traditional system design optimization problems often craft the design variables for a specific task. The goal of this paper is to make a first step towards simultaneously generating high-performance motions for robotic systems and finding their optimal design parameters. This is achieved through controlling and designing the hopping robot in Fig. 5.9 (right side). A related example is the leg design of quadrupedal robots such as (Chadwick et al., 2020).

Previous studies of designing motions and control laws for compliant bipeds (Jake Reher, Wen-Loong Ma, and Aaron D. Ames, 2019) and hoppers (Chang et al., 2019) have demonstrated the effectiveness of model-based approaches using collocation-based optimization (A. Hereid, C. M. Hubicki, et al., 2018). We have also explored different design methodologies of spring-loaded hoppers (E. Ambrose, N. Csomay-Shanklin, et al., 2019), and hopping with parallel elasticity (E. Ambrose and A. D. Ames, 2020). These first two hopping robots can be seen in the left and center positions of Fig. 5.9. In this paper, we further exploit the use of high-performance mechanisms via curved plate springs to enhance performance. Plate springs have the

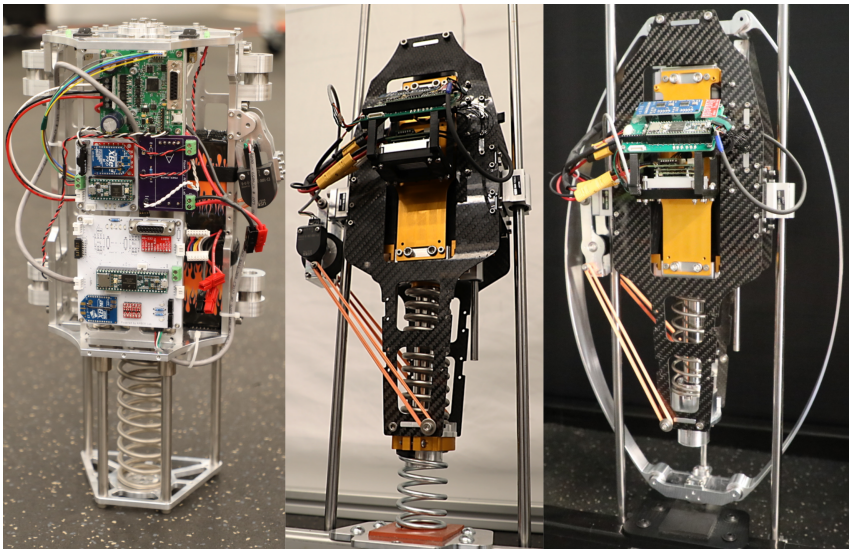


Figure 5.9: Evolution of the moving-mass hopping robot. Mark I (left, E. Ambrose, N. Csomay-Shanklin, et al., 2019), Mark II (center, E. Ambrose and A. D. Ames, 2020), and now Mark III (right) is modified to feature the curved-springs for this work.

advantage of being easy to design for a variety of nonlinear stiffness profiles, which can provide for improved energy storage and acceleration profiles (H. B. Brown and Zeglin, 1998), (Kooi, 1994). However, the nonlinear and compliant nature of these springs cannot be easily characterized by a traditional relationship, such as Hooke's law, and is thus a memoryless system. Instead, beam theory (Barber, 2010) must be used to develop the spatial-domain dynamical equations that govern the springs. To incorporate this with traditional time-domain dynamics of the torso, which are largely used to characterize robotic systems (Featherstone, 2008) and legged locomotion (J. W. Grizzle, Chevallereau, A. D. Ames, et al., 2010), we take inspiration from interconnected dynamical systems (Antonelli, 2013). Through shared variables of spring force and length, the torso and the spring subsystems are dynamically coupled. We represent this formulation as an *interconnected time-spatial dynamical system*. A similar idea was used to decouple quadrupedal dynamics as two coupled bipedal subsystems (Wen-Loong Ma, Noel Csomay-Shanklin, and Aaron D. Ames, 2020).

Further, we can configure the curved spring design as decision variables for an optimization algorithm. This allows the optimization to concurrently find the solution to the interconnected dynamical system and the optimal spring design parameters. The end result is a custom-made hopping robot driven by the full-body dynamics and optimal design, which can hop stably and with greater energy efficiency.

Our main contributions are twofold. First, we propose a time-spatial dynamic modeling of the hopping robot that utilizes the curved springs in Fig. 5.9. Based on this, a generic optimization formulation is presented to solve for a solution to this dynamical system with a corresponding hopping motion, and find the optimal design variables for the curved springs. Secondly, the construction of a hopping robot, on which experiments are conducted and stable hopping is achieved. Therefore, we demonstrate the complete process of system and controller design, and experimental validation.

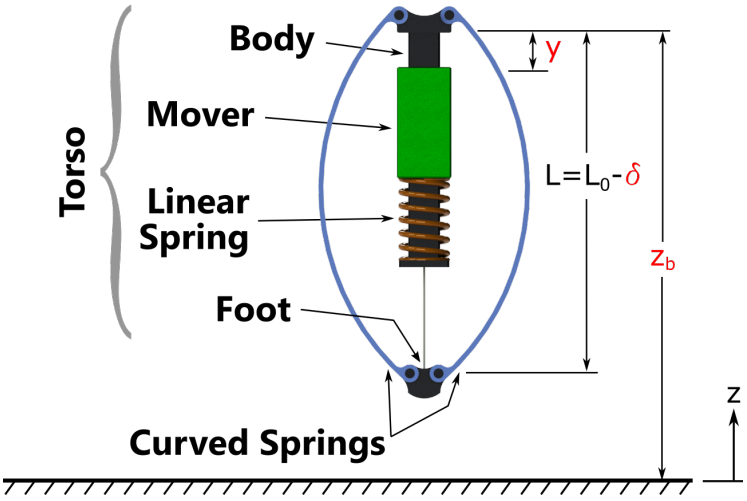


Figure 5.10: The system's configuration coordinates, with a look at the core mechanism features.

5.6 Interconnected Dynamics

The dynamics of the hopping machine in Fig. 5.10 are composed of two main parts: the time-domain dynamics of the "torso", whose electric motor's controller is our *control target*; and the spatial-domain dynamics of a curved spring, whose physical parameters are our *design target*. The two subsystems are interconnected through the shared spring force and length variables, F_s and L. We will introduce each subsystem first, then show the interconnected full-body dynamics. Note that we used the following notations to distinguish the two types of derivative:

$$\dot{\square} \triangleq \frac{\partial \square}{\partial t}, \quad \square' \triangleq \frac{\partial \square}{\partial s}. \tag{5.11}$$

Time-domain dynamics of the torso

As Fig. 5.10 shows, the torso contains all of the robot other than the curved springs. Hence, as detailed in (E. Ambrose and A. D. Ames, 2020), we can obtain the

torso's *time-domain* dynamics with configuration coordinates $q = (z_b, y, \delta)^\top \in \mathbb{R}^3$ and using the Euler-Lagrangian methods to obtain the equations of motion. Without considering any constraint, we have

$$M(q)\ddot{q} + H(q, \dot{q}) + B_s F_s = Bu + J_v^\top(q)\lambda_v, \tag{5.12}$$

with

$$M(q) = \begin{bmatrix} M_0 & -M_m & M_f \\ -M_m & M_m & 0 \\ M_f & 0 & M_f \end{bmatrix}, \; H(q, \dot{q}) = \begin{bmatrix} c_b \dot{z}_b + M_0 g \\ F_p - M_m g \\ M_f g \end{bmatrix},$$

and the actuation matrices $B_s^\top = (0, 0, 1), B^\top = (0, 1, 0)$. M_m, M_f, and M_0 are the mass of the mover, foot, and total robot. c_b is the damping of the body as it moves vertically. The internal spring creates a standard linear extension force given by $F_p = k_p y + c_p \dot{y}$, where k_p and c_p are the stiffness and damping coefficients, respectively. F_s is the extension force of the curved springs being examined in this work.

There exist four different dynamic domains, denoted by $\mathcal{D}_v$ with $v \in \{1, 2, 3, 4\}$. Each domain has a holonomic constraint representing certain locking mechanisms. Fig. 5.11 shows the domain map for this hybrid system. The hopping motion followed here contains three continuous-time domains: one aerial domain, $\mathcal{D}_1$, and two ground domains, $\mathcal{D}_3$ and $\mathcal{D}_4$. The two ground domains are distinguished by whether the mover is in contact with the body. During each domain, a holonomic constraint is used to represent the contacts with internal components and the ground. In this case, these are

$$h_1(q) \triangleq (y, \delta)^\top, \; h_3(q) \triangleq (z_b + \delta), \; h_4(q) \triangleq (y, z_b + \delta)^\top. \tag{5.13}$$

and the corresponding Jacobian matrices can be defined as

$$J_v \triangleq \frac{\partial h_v(q)}{\partial q}.$$

Therefore, we can explicitly solve the constraint force λ_v for domain $\mathcal{D}_v$, and (5.12) simply becomes

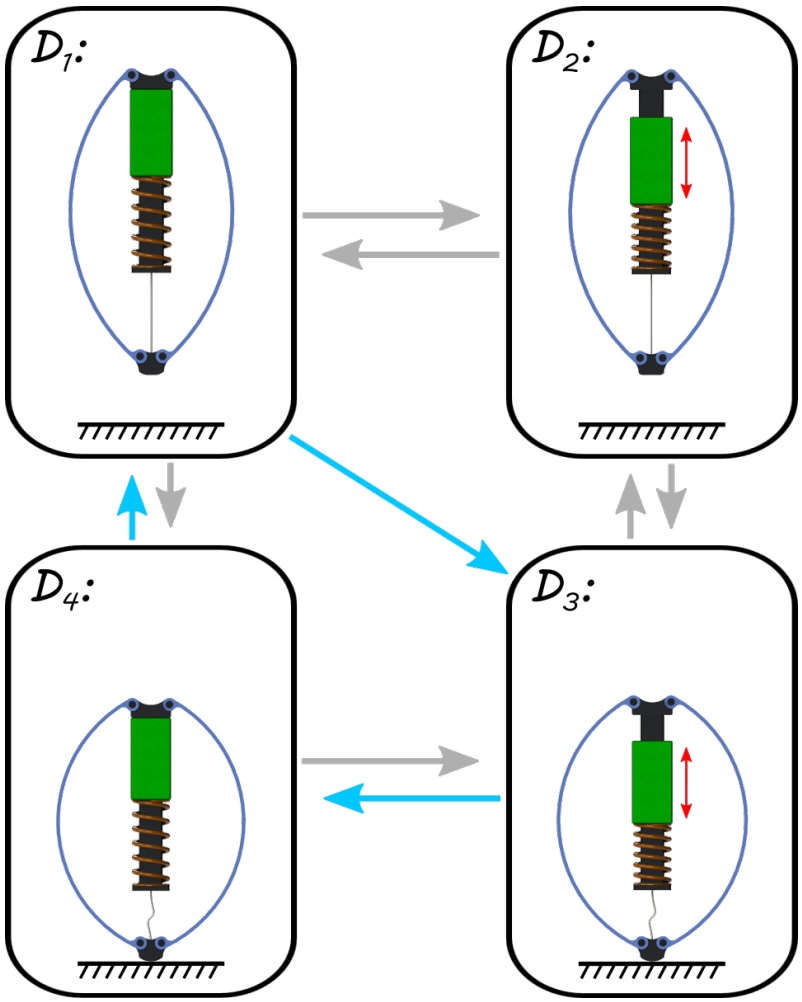


Figure 5.11: Directed graph for the hybrid hopping system. Each phase of dynamics can be represented by a interconnected system as given by (5.24)-(5.25). The blue arrows indicate the order of phases followed in this work.

$$M(q)\ddot{q} + H_v(q, \dot{q}) + B_{s,v}F_s = B_v u. \tag{5.14}$$

In $\mathcal{D}_1$ the mover and foot are each locked to the body due to hard-stops present in the system. In both cases the spring forces will tend to keep the coordinates y and δ fixed to that of the body coordinate, z_b. When the robot contacts the ground, the system enters $\mathcal{D}_3$ and the foot becomes fixed to the surface, leading to a holonomic constraint acting on the foot, while the mover continues downward. During this phase the actuator will input force to the system until the mover reestablishes contact with the body, and the system enters $\mathcal{D}_4$. This final ground domain is very short and lasts until the curved springs extend back to their equilibrium length and the

hardstop engages, at which time the system re-enters the air domain, $\mathcal{D}_1$, and the cycle repeats.

In order to distinguish the effects of the actuator input and curved spring forces from the other dynamics, we write the equations of motion from (5.14) in an ODE format:

$$\dot{x}_1 = \underbrace{\begin{bmatrix} \dot{q} \\ -M^{-1}H_v \end{bmatrix}}_{f_v(x_1)} + \underbrace{\begin{bmatrix} 0 \\ M^{-1}B_v \end{bmatrix}}_{g_v(x_1)} u + \underbrace{\begin{bmatrix} 0 \\ -M^{-1}B_{s,v} \end{bmatrix}}_{\bar{g}_v(x_1)} F_s, \tag{5.15}$$

where $x_1 = (q, \dot{q})^\top \in \mathbb{R}^6$ are the time-domain states.

Spatial-domain dynamics of the curved springs

Normally for linear springs, the spring dynamics are given as a linear map: $F_s = -kL - c\dot{L}$ according to Hooke's law, where k is the stiffness and c is the coefficient of an additional damping term. However, the curved spring of interest here is governed by spatial-domain dynamics, which will be detailed next. Designing the springs from curved plates serves as a way of controlling their stiffness profile. This will allow for the timing of the ground phase to be synced with a desirable force input from the actuator, resulting in a more efficient motion than would be possible by having two coils springs in the system as was done in (E. Ambrose and A. D. Ames, 2020).

Fig. 5.12 details the basic model of the curved, plate spring that will be used in this problem. Based on the work in (Georgiev and Joel Burdick, 2018), the internal relations for the moment along the beam and its derivative can be described in terms of the *parameterized arclength*, s, as

$$M(s) = (A(s)Eer_n)\gamma'(s) \tag{5.16}$$

$$M'(s) = -N(s) \tag{5.17}$$

where $s \in [0, s_0]$ is the position along the arc length of the beam's neutral surface, which is shown as a grey, dashed arc within the beam in Fig. 5.12. $M(s)$, $N(s)$, and $\gamma(s)$ are the internal moment, force, and resultant deflection at that point due to the loading force, F_s. $F_s \in \mathbb{R}$ is the external force acting on the spring along the vertical direction. Note: $\alpha(s)$ represents the natural angle of the beam at point s when the spring is unloaded. The deflection angle $\gamma(s)$ is relative to this $\alpha(s)$ value. Furthermore, E represents the Modulus of Elasticity of the spring material,

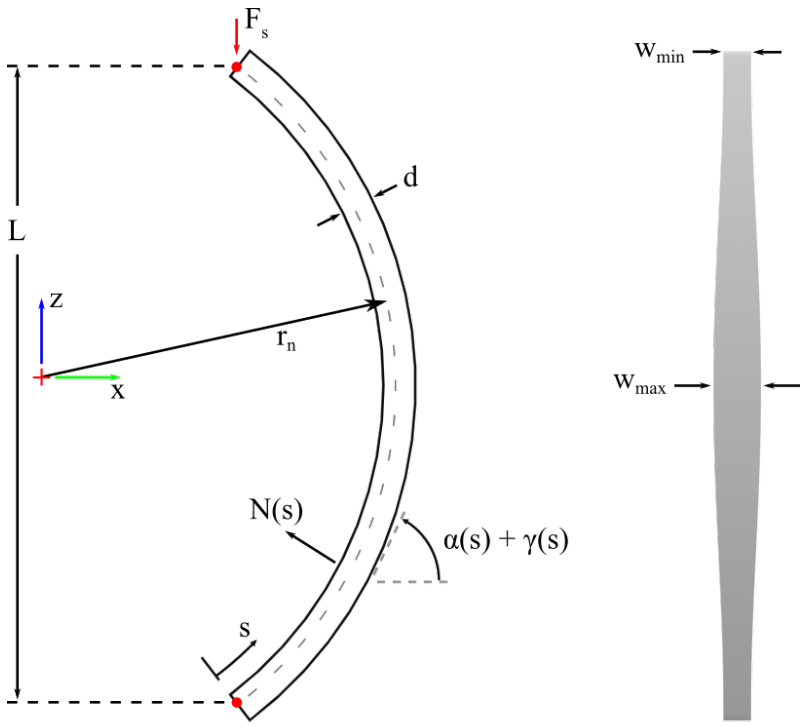


Figure 5.12: The geometry properties of the curved spring from a front view (left) and side view (right).

r_n is the radius of the unloaded spring's neutral surface, and e is the eccentricity of the neutral surface from spring center due to the curvature of the spring.

Throughout this paper, the radius of the neutral surface and the in-plane thickness of the spring, d, will be held constant, while the width, $w(s)$, is allowed to vary along the length of the spring. From these spatial metrics, the cross sectional area of the spring at the given position, $A(s) = dw(s)$, can be found. For this work, we define the variable width to be

$$w(s) = \left(\frac{w_{\min} - w_{\max}}{2}\right) \sin\left(\frac{2\pi}{s_0} s\right) + \left(\frac{w_{min} + w_{max}}{2}\right),$$

where w_{min} and w_{max} are the minimum and maximum width of the spring. This width profile was chosen to make the ends of the spring narrow while the middle is wider. This will allow for a better distribution of stress along the beam (H. B. Brown and Zeglin, 1998). The internal normal force $N(s)$ is given by

$$N(s) = -F_s \cos(\alpha(s) + \gamma(s)) \triangleq \hat{N}(s) F_s. \tag{5.18}$$

Rearranging the governing equations (5.16)-(5.18) gives the final deformation as:

$$\gamma''(s) \triangleq \frac{\partial^2 \gamma}{\partial s^2} = -\frac{N(s) + Eer_n A'(s)\gamma'(s)}{A(s)Eer_n}$$
$$= -\frac{A'(s)}{A(s)}\gamma'(s) - \frac{\hat{N}(s)}{A(s)Eer_n}F_s. \tag{5.19}$$

We now define the spring's states as $x_2 = (\gamma(s), \gamma'(s))^\top \in \mathbb{R}^2$ and the static parameters as $\rho = (d, w_{min}, w_{max}, E) \in \mathbb{R}^4$, i.e., design variables. We then can write (5.19) as in the form of an ordinary differential equation (ODE):

$$x_2' = f_\rho(s, x_2) + g_\rho(s, x_2)F_s. \tag{5.20}$$

For this system, the independent variable is now s rather than time, and the dependent variable is the deflection angle $\gamma \triangleq \gamma(s)$. Hence the solution to (5.20) can be expressed as

$$x_2(s)^\top = (\gamma(s), \gamma'(s)), \quad \forall\, s \in [0, s_0]. \tag{5.21}$$

Since s is a spatial characterization of the spring dynamics, we call (5.20) the spatial-domain dynamics of the curved-spring subsystem.

Boundary condition and numerical solution.

For the subsystem (5.20), a boundary condition is required to uniquely determine the solution. Since the ends of this spring are pinned, there is no moment transferred at these points (Georgiev and Joel Burdick, 2017), and we can set the boundary condition as $\gamma'(0) = \gamma'(s_0) = 0$. We then can solve for the numerical solutions $x_2(s^j)$ for a specified grid $j \in \{0, 1, 2 \ldots, \mathrm{K}\}$, where $s^j = js_0/\mathrm{K}$. This will be detailed in the next section.

Output.

The output of the curved-spring subsystem is the vertical length of the spring, i.e. $L(t, s)$ for at a given time $t \in [0, T]$. This can be calculated from the natural shape of the spring in conjunction with the deflection angles as,

$$y(s) \triangleq L(t, s) = \int_0^{s_0} \sin(\alpha(s) + \gamma(s))ds \tag{5.22}$$
$$\approx s_\Delta \sum_j \sin(\alpha(s^j) + \gamma(s^j)) \tag{5.23}$$

Note that (5.22) defines the exact value while (5.23) is the trapezoidal approximation.

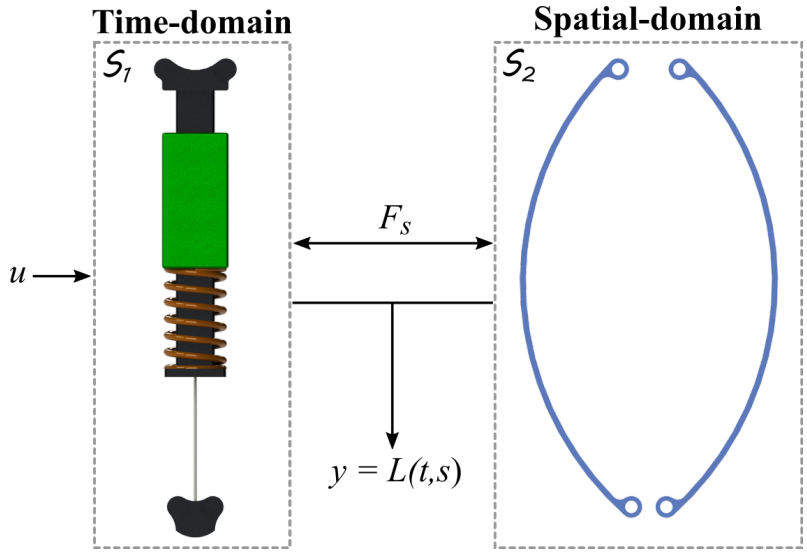


Figure 5.13: The signal-flow diagram of the interconnected full-body dynamics.

Full-Body Dynamics

Due to the fact that the curved-spring subsystem evolves according to a spatial-domain, independent variable s, the full-body dynamics of the hopper in design becomes a *interconnected time, spatial-domain system*. This is achieved through an interconnected mechanism, characterized by the shared variables $L(t,s), F_s$. Concretely, we summarize the full-body dynamics from (5.15) and (5.21) as:

$$S_1: \quad \dot{x}_1 = f_v(x_1) + g_v(x_1)u + \bar{g}_v(x_1)F_s \tag{5.24}$$

$$S_2: \begin{cases} x_2' = f_2(s,x_2) + g_2(s,x_2)F_s \\ L \approx \dfrac{s_0}{K_2}\sum_j^{K_2} \sin(\alpha(s^j) + \gamma(s^j)) \end{cases} \tag{5.25}$$

where $x_1(t) \triangleq (q,\dot{q})^\top$, $x_2(t) \triangleq (\gamma,\gamma')^\top$, $j = 0,1,\ldots,K_2$, and $s^j = js/K_2$. As the block diagram shown in Fig. 5.13, this interconnection between the two subsystems happens at the output level instead of the states level, resulting in a coupled control system (Wen-Loong Ma, Noel Csomay-Shanklin, and Aaron D. Ames, 2020). Since the curved springs only deflect while the robot is on the ground, we only need to match the shared variables during phases $\mathcal{D}_3$ and $\mathcal{D}_4$. Throughout $\mathcal{D}_1$, $L \equiv L_0$ and $F_s \equiv 0$.

5.7 System-Level Optimization

The goal of this section is to use optimization (Betts, 1998), (Kelly, 2017) to simultaneously design highly-dynamic hopping motions and a curved spring of nonlinear stiffness which allows for such motions with high energy efficiency of the electric

motor. Leveraging the previous construction of the multi-phase interconnected dynamics, we introduce a general optimization formulation that solves for a solution to the full-body dynamics, given by (5.24)-(5.25), and find the optimal spring parameters ρ. These parameters will be later used to guide our manufacturing of a hopping robot.

Decision variables.

We first discretize the time horizon as an evenly distributed grid $\{t^i\}_{i=0,1,\ldots,K_1}$ where $t^{K_i} = T$ for the time-domain dynamics. Then we similarly define the spatial grid as $\{s^j\}_{j=0,1,\ldots,K_2}$. We denote the time-domain states at time t^i as x_1^i and the spatial-domain states at s^j as x_2^j. Correspondingly, the shared spring force at time t^i is defined as F_s^i. Define the decision variable as:

$$X \triangleq \{x_1^0, P, (\gamma^0)^i, F_s^i, \rho\} \quad \text{with } i = 0, 1, \ldots K_1 \tag{5.26}$$

where x_1^0 is the initial condition of (5.24) when $t = 0$, P is the vector of Bezier polynomial coefficients describing the desired motion of the mover coordinate, y, and γ^0 is a boundary value of (5.25). Note that $(\gamma')^0 = 0$ was a given condition, and $(\gamma^0)^i$ is the boundary value at time t^i. The mover coordinate is the degree of freedom which is directly controlled by the actuator, and will be the key trajectory to track on the actual hardware in experiments.

Optimization problem.

We denote $p(x_i) \leq 0$ as the path constraints for the time-domain dynamics, then we have the optimization problem using a direct-shooting method as:

System-Level Design Optimization:

$$\min_X \; J(X) \tag{OPT}$$

$$\text{s.t. } \mathrm{C}_1.\; x_1^{i+1} = x_1^i + \int_{t^i}^{t^{i+1}} \left(f_v(x_1) + g_v(x_1)u(x_1) + g_v(x_1)F_s^i \right) dt \qquad i = 0, \ldots K_1\text{-}1$$

$$\mathrm{C}_2.\; \begin{cases} x_2^{j+1} = x_2^j + \int_{s^j}^{s^{j+1}} f_\rho(s, x_2) + g_\rho(s, x_2)F_s^i & j = 0, \ldots K_2\text{-}1 \\ \dfrac{s_0}{K_2} \displaystyle\sum_j^{K_2} \cos(s^j + \gamma^j) - L^i = 0 & i = 0, \ldots, K_1 \end{cases}$$

$$\mathrm{C}_3.\; (\gamma')^0 = 0,\; (\gamma')^{K_2} = 0$$

$$\mathrm{C}_4.\; p(x_1^i) \leq 0 \qquad i = 0, \ldots, K_1$$

$$\mathrm{C}_5.\; b(x_1^i) = 0 \qquad i = 0, \ldots, K_1$$

Essentially, for the initial condition x_1^0 and every spring force F_s^i picked by the optimization algorithm, C_1 solves for its states at time t^i with the assumed closed-loop controller $u_v(x_1)$ (details of the tracking controller can be found in (E. Ambrose, N. Csomay-Shanklin, et al., 2019)) for all phases $\mathcal{D}_v$. We post the boundary condition as an equality condition C_3, so that for each sampling time t^i and the shared spring force F_s^i from C_1, the dynamics of (5.25) is then solved. Its output L^i is also shared by (5.24) as one of the torso's states. Note that this characterization of F_s^i and L^i is the same as the Lagrange multiplier. Further, we posted a range of inequality constraints such as hopping height and max spring deflection, δ_{max}, in C_4. This max deflection was based on the max allowable travel given the size of the robot, as well as the max stress that the plates can handle before failure. A set of equality constraints C_5 is additionally used to enforce the boundary conditions of the torso dynamics to match each other through the ground impact, so that the optimization solution is periodic within the interconnected dynamics. This periodic solution then can be implemented on the hardware as a cyclic hopping motion. Lastly, the cost function was chosen to be $J(X) = \int_0^T u(x_1)^2 dt$, to minimize the effort of the actuator over a single hop within the constraints described.

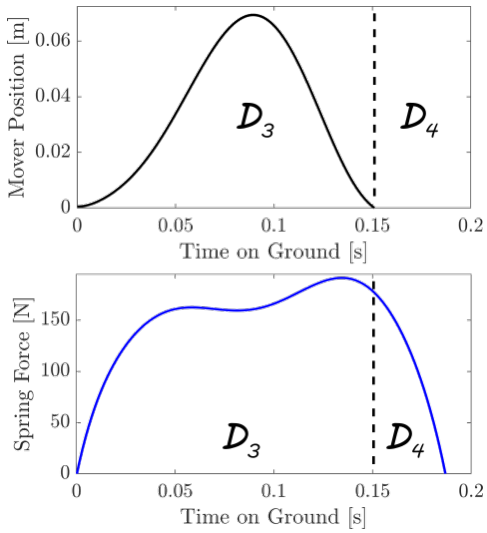


Figure 5.14: Output results of the optimization: desired trajectory of the actuated mover coordinate (top), and expected spring force in each curved spring (bottom).

Optimal hopping motion and design variables.

This optimization problem was implemented as described in *MATLAB* using *fmincon*. The average run time for the optimization to complete was 5 min. In order to facilitate manufacturing of the springs, the material's elastic modulus of the curved-springs was set for Al-7075 ($E = 72GPa$), while the other spring design parameters

were allowed to vary. Other materials such as steel and titanium would be better suited for large strain deformation like this, but were not considered here for the sake of time and limited manufacturing resources at hand. In order to stay within the yield stress limit of Al-7075, $\sigma_y = 503\,\text{MPa}$, the feasible height of hopping was limited. The optimization was able to generate results for hop heights up to 17 cm off the ground, before the max stress of the material reached unsafe levels. Fig. 5.14 shows the mover trajectory and temporal spring force profile for the result of a 9 cm hop height, which requires a max input force of 49 N.

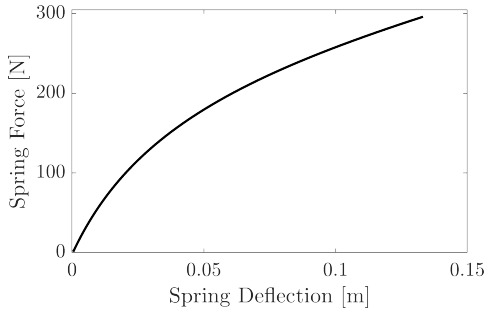


Figure 5.15: Final spring force profile as a function of deflection showing the softening nature of the designed springs.

The chosen spring design was based on this same 9 cm hop result which gave the geometric spring properties of $d = 3.25\,\text{mm}$, $w_{min} = 24\,\text{mm}$, and $w_{max} = 41\,\text{mm}$. The resultant spring force profile for this spring design is shown in Fig. 5.15. The stiffness of these springs vary from $2390\,\text{N/m}$ at zero deflection to $1080\,\text{N/m}$ at 10 cm of deflection. At this 10 cm mark, the springs are storing 16.45 J and the max stress within the plates is 283 MPa. The duration of the ground phase using this result is 0.18 s, which is almost 3 times longer than what was seen in previous work with a stiff coil spring. The ground phase elongation alone allows for lower peak actuator force and less stressful impacts on the robot. This resultant spring and motion design was taken and implemented as described in the next section.

5.8 Final Moving-Mass Hopper with Custom Springs

The robot used for the experimental validation of this spring design and hopping motion was previously constructed using two linear coil springs (E. Ambrose and A. D. Ames, 2020). To replace the lower coil spring, a set of curved springs were machined out of Aluminum 7075-T651 and mounted to this robot through slight modifications to the top and bottom of the robot's body (see Fig. 5.9). A linear guide was also attached in order to constrain the spring deflection to the vertical direction.

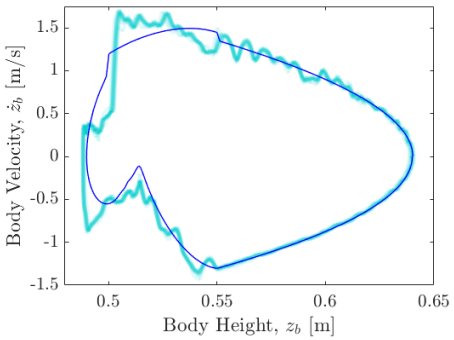


Figure 5.16: Phase portrait of the body coordinate with simulated trajectory (dark blue) and experimental results from 20 consecutive hops (light blue).

The hopping robot was connected to a frame of vertical rails to facilitate its 1-dimensional motion. Experiments were run utilizing an open-loop input of motor force as a time-based trajectory as in (E. Ambrose and A. D. Ames, 2020). The trajectory used here was generated through optimization for a hop height of 9 cm off the ground. Experiments were performed for trials of 20 consecutive hops. A phase portrait of the body coordinate, z_b, for one of these trials is shown in Fig. 5.16. Vibration of the rails and curved springs was visually observed during the trials and was present in the state tracking data as seen in the phase portrait. The plot shows the consistent nature of both the overall hop height and the effects of the ground impact and system vibrations, despite the motion being driven by an open-loop playback of the input trajectory. Motion tiles of a single hop are shown in Fig. 5.17, which show the full extent of the deflection within the plate springs. The supplemental video shows the experiments in real time and slow-motion (E. Ambrose, 2021).

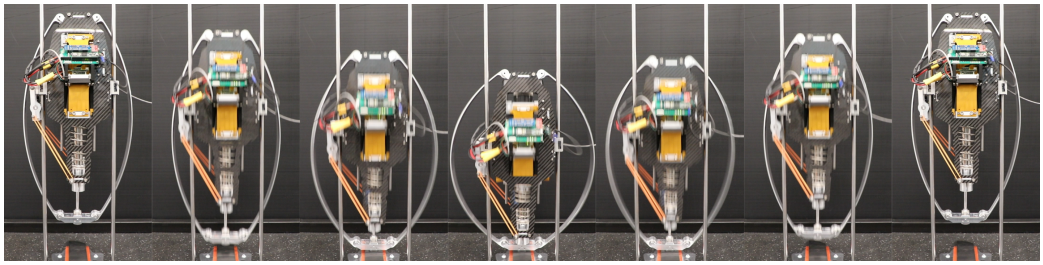


Figure 5.17: Tiles from the experiment over a single hop.

Chapter 6

ARCHER: THE 3D HOPPER

Moving beyond the work presented in the last two chapters on 1 Dimensional hopping robots, this chapter presents the expansion into the realm of 3D hopping. First, the two subsystem of the robot will be presented along with the details of how they were simulated, designed, and finally combined to form the final robot that is ARCHER. Next, simulations of the full robot were created to assess performance of controllers for the attitude subsystem. Finally, the robot was built and tested in the lab to show its initial performance.

6.1 Compression Actuator Design

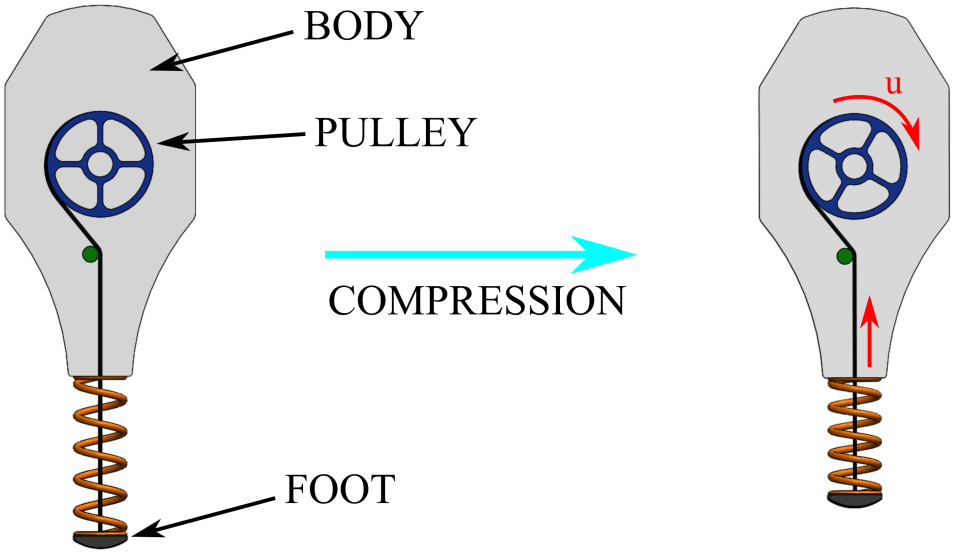


Figure 6.1: Basic model used to simulate the compression motion

The goal of this robot is to show the effectiveness of the flywheels to control attitude, and the ability to decouple the leg from this attitude control. With this goal in mind, this final robot switches to using the compress-release method for controlling the vertical component of hopping. While the moving-mess method was able to be vastly optimized through design and motion planning, the CRH method is still far simpler to control and plan. Switching to this method will allow any on-board computation to focus the majority of its efforts on the motion planning of the overall robot and the attitude control.

Design Planning through Simulation

In order to plan out the design of the compression actuator, a set of MATLAB simulations were created. The first would be used to simulate the basic 1-Dimensional hopping motion for this type of mechanism, and the second would simulate the compression and release motion of the actuator. Previous work had shown that selecting an actuator that was both strong enough to compress the spring, and quick enough to release the cable completely within the short ground phase was a challenge. Rather than sticking with the idea of using a clutch to do the releasing, the new idea was to use a higher power motor in combination with a low reduction gearbox to reach the desired performance of both motions directions directly.

Fig. 6.1 shows the an example of the basic model used for these simulations. The actuator here will have a pulley connected at its output, which will be used to pull up on a cable connected to the foot of the robot. As the pulley spins, the cable will either act to apply a compressing force on the spring or will let slack back out of the cable allowing the spring to expand, depending on the direction of rotation. The equations of motion here are just as they were for the original CRH hopper in Chapter 4. The actuation in these equations, u, will be treated separately in the second simulation. Here u is will just be a simple PD+FF controller on the deflection of the spring, while in the air-decent phase, and $u = 0$ during the ground phase. Reiterating the equations of motion used previously, we have

$$\underbrace{\begin{bmatrix} M_b + M_f & M_f \\ M_f & M_f \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \ddot{z}_b \\ \ddot{\delta} \end{bmatrix}}_{\ddot{q}} + \underbrace{\begin{bmatrix} (M_b + M_f)g + c_b\dot{z_b} \\ k_s\delta + c_s\dot{\delta} + M_f g \end{bmatrix}}_{H(q,\dot{q})} = \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u. \tag{6.1}$$

Since the design of this actuator will affect the mass and inertia of this robot, the model parameters will change throughout the process until the final design is reached. These model parameters were chosen to approximate the final mass at first, then iterated a few times throughout. Similarly, the stiffness of the spring was also a variable of this process as well since it would need to change along with the actuator to yield the desired performance within our constraints of max deflection and minimum ground phase duration. With this in mind, the simulation process became as follows:

1. Select approximate model info
2. Simulate the generic 1D hopping motion to determine the required compression to maintain a certain hop height for that model

3. Collect the data of that sim, including: air/ground phase duration, required compression, and hop height

4. Run a second simulation of the motor compression and release cycle, given desired current/voltage limits and the model of the actuator

5. Observe the minimum time to compress and release and compare with the max duration of the air and ground phases

6. Iterate as desired to get required performance/safety margins

During the design process, both low and high hops were simulated in order to see the limits of the actuator. Low heights show the smallest time intervals in which to compress, while the higher hops would show the limit of the compression the actuator could perform, as well as the max height that the spring would be capable of handling. Next, the details of each simulation will be discussed in more detail.

With the model being as simple as it is, there are just a few parameters to be set before a simulation can be run such as mass and damping terms. The mass terms will get updated as the model becomes more solidified. Meanwhile the stiffness of the spring will be adjusted during the iteration process, attempting to minimize the stiffness without going past the max allowable deflection during the higher hops. The goal with the spring stiffness selection is to be stiff enough to prevent deflections too large for foot's range of motion in hardware, but also soft enough to prolong the ground phase duration.

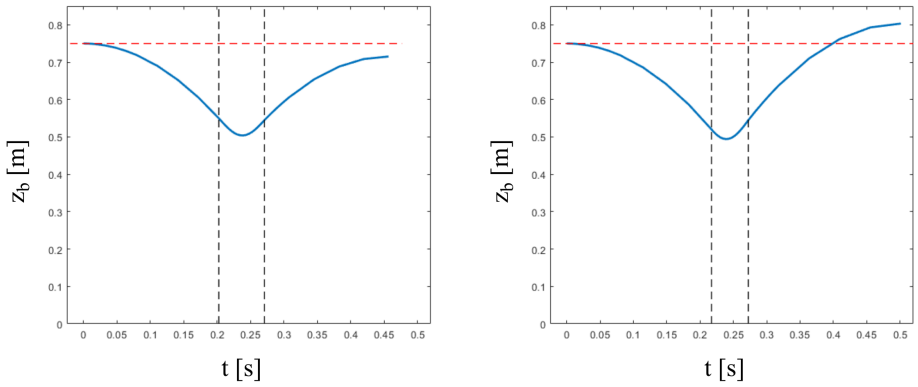


Figure 6.2: Effect of pre-compression on the hop height of the hopper. The left side plot shows the result of under compression, and the right side plot shows a hop with over compression.

With this model, the hopping simulations were set to start from an initial hop height, with a given pre-compression to reach at impact with the ground. That will play out until the next hop apex is reached. Fig. 6.2 shows some examples of the vertical motion throughout a single hop, and how different amounts of pre-compression control the hop height. A simple optimization process was wrapped around this simulation, which will determine the ideal pre-compression, δ_0, to yield a specific final hop height given the initial height. At the end of the optimization, the duration of both phases is recorded along with δ_0.

This timing information gets used in the second simulation step next, where the actuator is assessed under chosen electrical limits for max current and voltage. The goal here is to see how quickly the actuator can compress the spring to the deflection δ_0, and also how quickly it can "release" the slack under no load (assuming the ground contact will remove the load from the actuator). These times will be compared to the maximum values which are the duration of the phases gathered previously.

The dynamics of this second model involves the motor-gearbox-pulley-spring system. In an effort to make this gearbox be customizable yet achievable on hardware, it was chosen to be a planetary design as seen in Fig. 6.3. The layout of the planetary gearbox in the compression actuator places the Sun gear onto the rotor of the motor, and the carrier of the gearbox at the output and connected to the pulley. The ring gear will be fixed to the frame along with the stator of the motor. With this configuration, the gear ratio of the drive is defined as $R = \frac{r_s}{2(r_s+r_p)}$, where the output (carrier-side) of the gearbox has a higher torque, but lower angular velocity than the motor side. Here, r_s and r_p are the radii of the Sun gear and Planet gear, respectively.

This system will be treated as a single degree of freedom, despite the presence of a flexible cable. In order to place the dynamics of the spring load in the same equation as the electro-mechanical motor dynamics, there is a need for a relation for how torque is transmitted across the gearbox via the equivalent inertia of the system as a whole. As derived in (Borders, 2009), this equivalent inertia of the system felt by the motor is given as

$$J_{eq} = (J_s + J_{rot}) + \frac{n_p r_s^2}{4}\left(\frac{J_p}{r_p^2} + M_p\right) + J_c R^2 \tag{6.2}$$

where J_s, J_p, J_c, and J_{rot} are the inertias of the sun gear, planet gears, carrier, and rotor of the motor, respectively. Additionally, n_p is the number of planets and M_p is their mass.

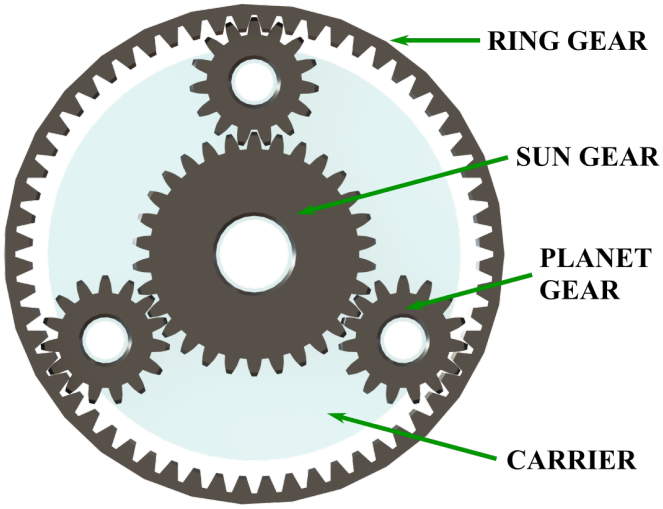


Figure 6.3: Basic profile of the planetary gearbox

The general equation of motion for this actuator can be written using this equivalent inertia as

$$J_{eq}\ddot{\theta} + c\dot{\theta} + \frac{r_{pul}R}{n}F_s = k_v I \tag{6.3}$$

where θ is the motor angle, c is a damping coefficient, r_{pul} is the radius of the pulley, n is the efficiency of the gearbox, F_s is the spring force, k_v is the torque constant of the motor, and I is the system input of motor current.

In order to determine the quickest possible compression and release, the control in these simulations will be bang-bang at the chosen current limit. Voltage limits are also present, but they end up not coming into play since the motor never gets up to a high speed during these motions and the motor's low terminal resistance. This simulation has two phases: compression and release. The compression phase will have the actual spring force working against the motor, while the release phase will have no spring force throughout. The states of motor velocity and position will be continuous across the phase change, which will lead to a little bit of overshoot in the deflection before beginning to extend again.

Fig. 6.4 shows an example of one of these simulations via a graph of deflection over time. The red curve represents the absolute minimum rate of compression and release given the phase durations found in the first simulation, which is based on the data from a stable hop height of 0.75 m. The blue curve shows the actual deflection of the actuator undergoing the motion with the chosen control. In this case, the motor current throughout the first phase is set at 20 A, while in the second phase it

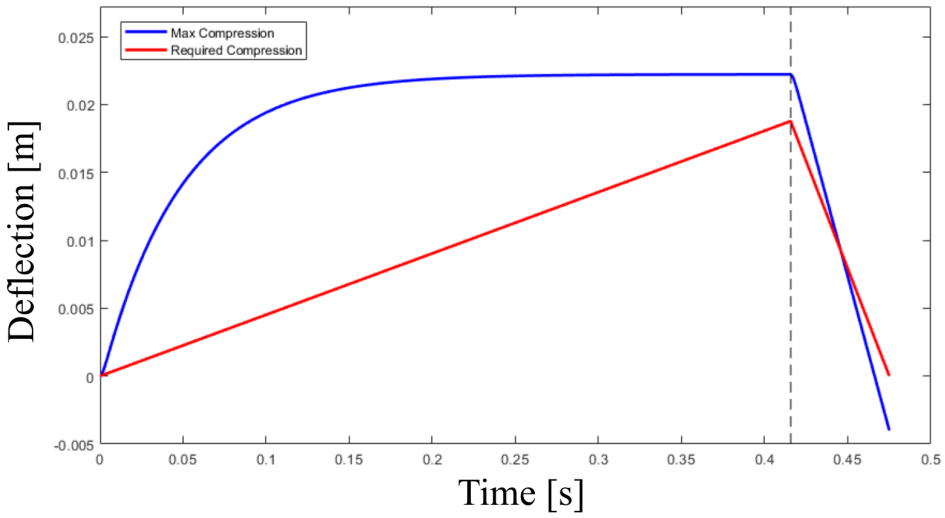


Figure 6.4: Example motion from the compress-release cycle simulation, for a hop height of 0.75m and a current limit of 20A.

is -20 A. It can be seen that with this current limit, the actuator can compress more than the required amount for this hop height, and is also able to release the cable more than quickly enough during the ground phase.

Hardware Realization

From here, the design of the actuator can be iterated on by changing the motor or gearbox parameters, and the spring stiffness could be shifted as desired. Some factors are limited by available products, such as gear ratios are based on the standard gearing options, particularly the ring gears which are difficult to find in more than just a few sizes. It was quickly seen that the best route to take was pairing a more powerful motor with a low-reduction gearbox, due to the limited duration of the ground phase. In order to prioritize speed, a torque-dense motor was needed to make up for the lower gearbox benefit. Luckily, drone motors are amazing now, so this wasn't difficult to do and a final design was reached.

Solidworks was used as a key tool during the design process. Each time a new motor or gear ratio was selected for a simulation trial, it was modeled up in CAD in order to determine the inertias for the components. Fig. 6.5 shows the final design along with the overall dimensions. The overall mass of the actuator came out to 950 g, and is capable of generating 13 N m of torque and rotating at 600RPM.

The motor selected for this actuator is the U10-plus from T-Motor (T-Motor, 2021b), with a custom wiring to reach a back-emf constant of 50RPM/V. The frame of the stator on the motor was also customized so it would have a thicker aluminum section

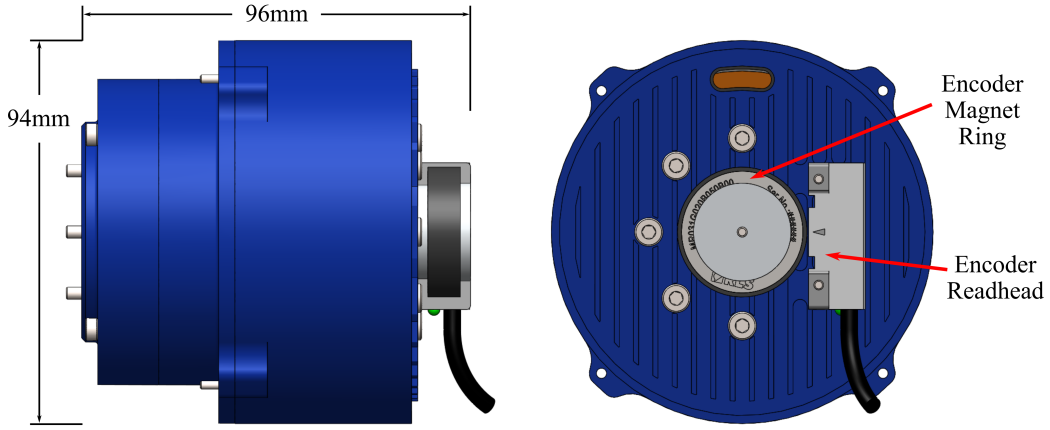


Figure 6.5: Views of the final compression actuator giving the overall size and the location of the incremental encoder on its back.

leading to better heat capacity and heat transfer. Furthermore, in order to not push the temperature of the motor, the current is limited to 25 A. This motor, as with the flywheel motors, is controlled with a Gold Solo Twitter from ELMO-MC (ELMO-MotionControl, 2021). The specific model used was the *G-SOLTWIR50/100SE1S*, which is capable of outputting up to 50 A and 100 V. This controller will use an incremental encoder as its feedback source on the motor's motion. The encoder used here and for the flywheels is the *LM13* from Renishaw (Renishaw, 2021). This sensor works by reading the rotation of a circular magnet that is attached to the rotor of the motor, in this case that magnet is the *MR031C*.

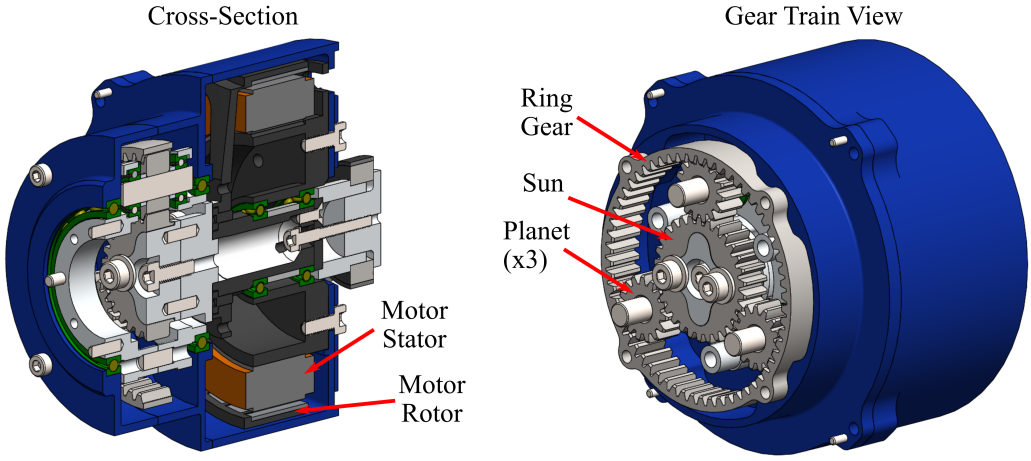


Figure 6.6: Internal views of the compression actuator, to see the layout of the components and specifically the gear train.

A better view of the internal layout can be seen in Fig. 6.6. The larger section of the actuator contains the motor, with the back plate being the mounting point for

Parameter	Value	Parameter	Value
r_s	0.015 m	r_p	0.0075 m
J_{rot}	0.000 134 041 kg m^2	n_p	3
J_s	0.000 007 150 kg m^2	R	$\frac{1}{3}$
J_p	0.000 000 335 kg m^2	η_{eff}	90%
M_p	0.0154 kg	c	0.1 Nms/rad
J_c	0.000 023 273 kg m^2	k_v	0.191 N m/A
J_{pul}	0.000 075 000 kg m^2	r_{pul}	0.0375 m

Table 6.1: Final parameter selection for the compression actuator.

the motor and the encoder. This back plate also has fins over the exterior in order to further help with the heat transfer away from the motor, since the motor temperature is the limiting factor on performance and capability. The front, smaller section of the actuator contains the gear box and supporting bearings. There are two halves of the carrier of the gearbox, each with its own bearing between it and the frame to force proper alignment and keep things running efficiently. Similarly, there are smaller bearings on either side of the planets, connecting them to the carrier. The final consideration for improving gear-train efficiency was to allow the sun gear to *float*, meaning it can slightly deviate from the rotor center in order to properly center itself between the planets. This is more important than staying concentric with the motor, as long as the motor can still transmit torque through the sun gear, which it can due to the spline profile between the two.

Table 6.1 provides the specific parameters selected for this actuator, including the inertias and radii for each key component, the approximate damping coefficient, and overall torque efficiency. The radius of the output pulley is provided here, since it is used in the compression simulation, and will be discussed later on along with the rest of the cable and foot system. Plots of the electrical and mechanical capabilities of this actuator can be seen in Fig. 6.7. The two limits in place within these plots are for the max current and max power of the motor. Both of these limits are selected to prevent the motor coil temperature from getting too high which would lead to shorts and other permanent damage. As mentioned previously, for this hardware the current limit is set to 25 A, while the motor power will be limited to 250 W.

A cable system is used to connect the output of the compression actuator to the foot of the robot. This is accomplished through two pulleys and a flexible, Teflon-reinforced cable. Fig. 6.8 shows the general layout of these components with the rest of the robot hidden. The upper pulley is attached directly to the output of the

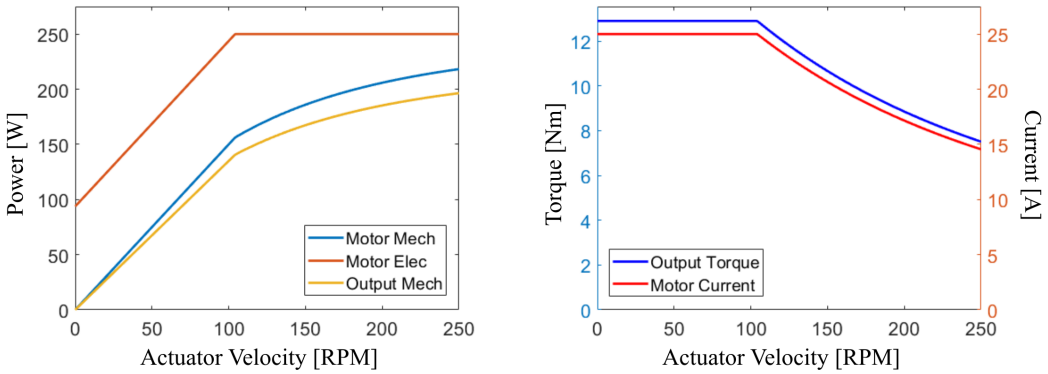


Figure 6.7: Plots of the actuator performance, showing output torque, motor current, and various power specs vs output speed.

compression actuator, and spins in order to take up or let out slack in the cable. The lower end of the cable is attached to a fixed pulley connected to the top of the foot. Between the two pulleys, just below the upper one, there is a small bearing placed to force the cable to align directly above the lower pulley and cause the compression force to be in-line with the spring.

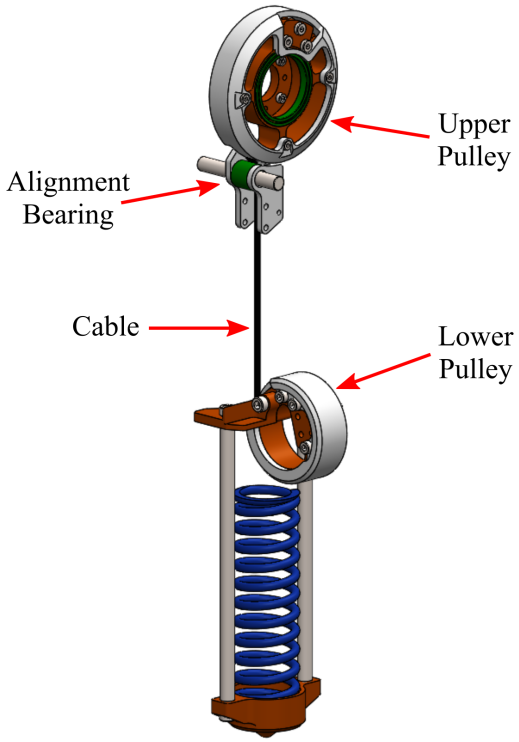


Figure 6.8: CAD view of the compression sub-system, including the pulleys, cable, foot, spring, and alignment features.

The cable itself is clamped to each pulley at its ends. In order to prevent slipping

of the cord within the clamp, it is wrapped around the pulley multiple times which allows the friction between the pulley/cable to provide extra force rather than relying completely on the clamp. There are grooves in the rim of the pulleys that force the cable to sit consistently while under load. Fig. 6.9 shows some close-up views of the upper pulley as an example of these grooves and clamps.

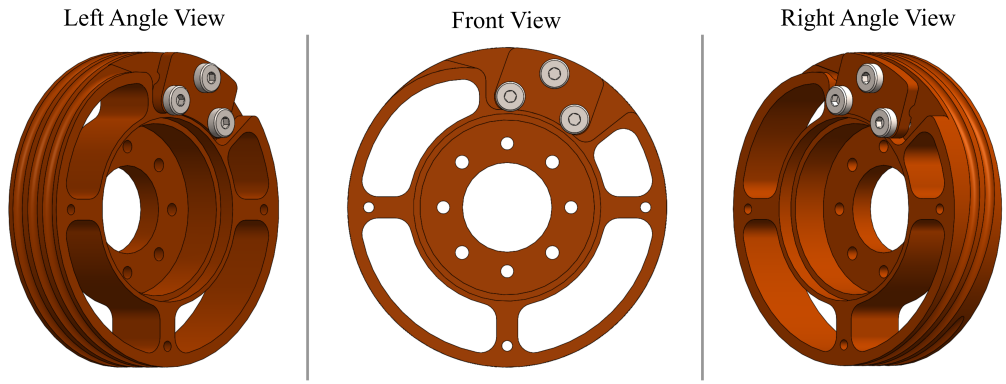


Figure 6.9: Views of the upper pulley which show the groove and clamp features that help hold onto the cable during compression.

To prevent the cable from coming out of the grooves, a light plastic cover is placed over the pulleys once the cable has been wrapped, as can be seen in Fig. 6.10. Cutouts in the covers allow the cord to exit/enter at the required spots, but prevent it from moving at all in most of the wrapped areas, and add extra normal force which will further increase the friction resistance and reduce the likelihood of slipping at the clamps.

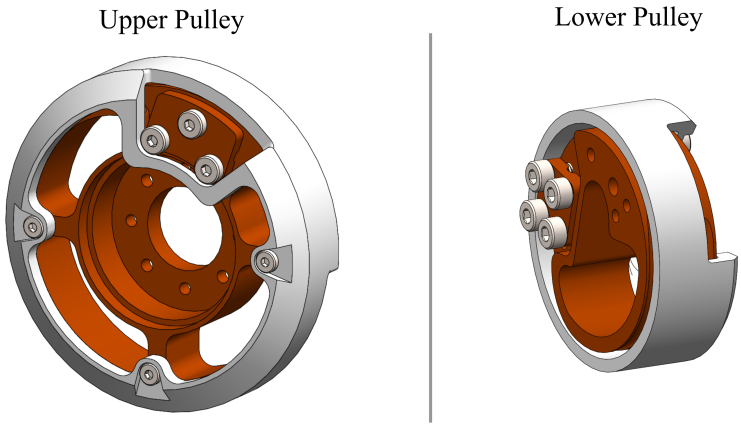


Figure 6.10: CAD model of both pulleys with their covers to keep the cable from slipping out of the grooves.

The foot assembly can also be seen in Fig. 6.8, specifically the foot pad, spring, foot guides, and the hardstop plates. During the process of simulating and designing the

compression actuator, the foot mass was used since it is the main factor in energy loss and motivates the need for pre-compression. Fortunately, its mass doesn't change as most design factors are updated, so it remained a constant throughout the design process. This was due to the fact that springs are generally designed around having a convenient diameter, but modulate stiffness through other dimensions like wire diameter and number of coils. As with the original 1D-CRH robot, this foot contains hard-stops placed at an equilibrium length based on the given weight of the robot. This CRH hopping system would be very conducive to the non-linear spring design optimization done in previous work. Since the control of this system can be treated as a discrete event that occurs in the air and does nothing during the ground phase, the optimization would be much simpler and more likely to give a global optimum for the spring design. A few designs were generated for these springs throughout the process of designing the rest of the robot. For the sake of time on this thesis, it was decided to remain with the standard linear coil spring. Through months of hearing from companies that they couldn't or wouldn't be willing to make custom springs from fiber glass materials like E-glass and S-Glass, it was decided to not make further attempts and slow down the rest of the project.

6.2 Flywheel Simulation and Design

The attitude control of the robot is preformed by a group of 3 flywheels within the torso, which will act in conjunction to keep the torso upright and place the foot as desired. The key metric in planning out this subsystem come down to choosing the mass/inertia of each flywheel, the maximum torque which can be exerted by each flywheel actuator, and the maximum speed that each flywheel can spin before the actuator loses its control bandwidth. This section will describe in detail the process of designing and simulating the attitude subsystem of the robot, to not only determine these key parameters, but also to observe the efficacy of this style of attitude control. The following subsections will delve into a review of attitude control used on hoppers in the past, the reasoning behind the choice of using flywheels on this robot, and their design process including the final choices made for ARCHER.

Review of other 2D/3D Hopping Platforms

Hopping is a mode of periodic, legged locomotion in which the body spends part of the motion cycle on the ground and the rest completely airborne. In between these two phases of motion, there are rapid and extreme changes of contact with the world in the form of impacts. In this way, hopping is alike to running in legged systems

(W. Ma, Kolathaya, et al., 2017), (J. Park et al., 2014). The presence of these contact changes and their energetic consequences motivate the use of methods for dynamics analysis used in classical legged locomotion (M. Raibert, 1986), or also in hybrid systems (Sinnet et al., 2011), (Westervelt, J. W. Grizzle, and Koditschek, 2003). Furthermore, hopping robots have been shown to be very capable despite these conditions, resulting in complex behaviors like flipping (M. Raibert, 1986), or changing terrain (C. M. Hubicki et al., 2016),(Yim, Wang, and Fearing, 2019). Achieving good performance and stability on hopping systems requires intelligent planning to prepare the robot during the air phase for the quick dynamics of the upcoming ground phase.

Researchers in the Leg Lab developed a set of control methods to stabilize hopping and other similar legged systems based on conservation of energy, foot placement, and leg sweeping (M. H. Raibert, 1984), (M. H. Raibert, H. B. Brown, and Chepponis, 1984), (M. Raibert, 1986). These controllers were able to successfully generate hopping motions for both 2D and 3D hopping systems and could maintain balance over a large number of hops and with a variety of disturbances. Furthermore, they required fairly simple state information and computations, making their implementation very straight-forward. One caveat of their hopping robots was the presence of a large torso, which could easily affect the dynamics of the system during the ground phase through a change in its position. This fact was reflected in the control strategies they developed, in both the placement of the robot's foot and the magnitude of the hip's forward velocity being dramatically effected by the torso angle. A benefit of having a torso is that the robot is able to control its center of mass location and apply torques to the leg during the ground phase with a single joint. The down-side of this is that the two dynamic features are coupled, when it could be better to have a separate actuator to control each.

As part of a collaboration with Disney Research and Development, a robotic hopping platform is being created which will adhere to the principles of safe actuation (E. Ambrose, N. Csomay-Shanklin, et al., 2019),(E. Ambrose and A. D. Ames, 2020). This safety concern forces any contact between the world and the robot to occur through an elastic element such as a spring. This requirement, along with those implied by recent patent on a robotic bouncing ball (Smoot et al., 2018), gives rise to many design constraints including the size of the mechanisms within the elastic exterior. In order to fit a torso of the robot inside, the size and range of motion of the link would need to be reduced, which would decrease the controllability of the

robot. Beyond this, the added weight of a torso would entail increasing the stiffness of the spring and lessening the benefits of the elastic boundary.

In order to maintain the controllability of the robot while satisfying the design constraints, flywheels are now being considered as the alternative source of actuation. Flywheels provide many benefits over a torso design within the context of this project. For one, they are significantly lighter, even when including three separate wheels and actuators. Secondly, they are usually small, but can be made up to any size that fits within your size constraint. Thirdly, flywheels are able to spin indefinitely which removes the range of motion constraints on the angle of this degree of freedom. The constraint on the velocity of the flywheel remains and then becomes the limiting factor on controllability of the system. Additionally, the flywheels center of mass is at its joint, which means that the center of mass for the robot is unaffected by the flywheel spinning. This could potentially allow for a separate actuator to be placed on the system to control center of mass location and gain more decoupled control over the dynamics of the robot.

The design of this robot will assume flywheels alone are the acting agent for attitude control. The model of the robot will have these flywheels placed such that their axes of rotation align at a single point which coincides not only with the central vertical axis of the robot leg, but also with the over all center of mass for the body of the robot. The body of the robot would include everything other than the foot and springs elements. To start these flywheels will be aligned with the overall directions of Yaw, Pitch, and Roll for the robots motion with an additional mass placed so that the center of mass remains in the desired location. Later on, this is changed so that the flywheels would be placed symmetrically around the central axis of the robot, removing the need for a large additional mass and lowering the mass of the overall robot.

In order to properly plan out the design of the flywheels, simulations must be run. This will require more than just basic balancing in place, but also in cases where the robot will be hopping around from spot to spot as well as having a constant velocity in a given direction. These simulations which move beyond balancing will require a plan for foot placement, which hadn't yet been studied in this project. The following subsection will describe how other methods were modified for the purpose of a flywheel driven system. The results of the initial simulations using these methods will be presented, which were the key factors used to settle on the design of the flywheel subsystem.

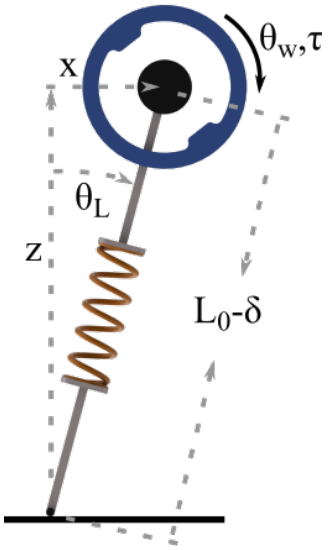


Figure 6.11: The 2D Flywheel-SLIP model.

Initial Simulations of a Flywheel-Controlled Hopper

Historically, methods for controlling the motions of hopping robots have made the assumption that there is a torso present above the leg which has larger inertia than the leg itself. This torso is typically used to modulate the angular position and velocity of the leg during the periodic motion. Flywheels are being examined to assess their possible replacement of such a torso, in order to meet the required performance on this potential hopping robot. To begin, we first introduce the Flywheel Spring-loaded Inverted Pendulum (fSLIP) model seen in Fig. 6.11, as described in Xiong and A. Ames, 2020. This model places a revolute joint and flywheel at the point mass of a typical Spring-loaded Inverted Pendulum (SLIP) model. This model allows for a torque between the leg and flywheel in order to effect the angular degree of freedom. Overall, this model now contains three degrees of freedom. In this work, the three model coordinates are chosen to be the horizontal and vertical position of the mass, x and z, as well as the angle of the flywheel, θ_w. The dynamics for the fSLIP model are given by the following three equations:

$$m\ddot{x} = -\frac{\tau}{r}cos(\theta_L) + F_s sin(\theta_L) \tag{6.4}$$

$$m\ddot{z} = \frac{\tau}{r}sin(\theta_L) + F_s cos(\theta_L) - mg \tag{6.5}$$

$$I\ddot{\theta}_w = \tau \tag{6.6}$$

where m is the combined mass of the flywheel and point mass, I is the inertia of the flywheel, τ is the torque on the flywheel, $F_s = k_s\delta + c_s\dot{\delta}$ is the spring extension

force, and δ is the spring deflection. The model kinematic constraints are on the angle of the leg from vertical, $\theta_L = tan^{-1}(\frac{x}{z})$, and the instantaneous length of the leg, $r = L_0 - \delta$, where L_0 is the free length of the leg.

Vertical Control Vertical control is used to maintain the desired oscillations of the robot between ground and air phases. Actuation along the leg axis is required when energy loss is present in the system or when you wish to change hop height (amplitude of oscillations). Energy loss can come from dissipation forces such as damping or through impacts when there is unsprung mass in the leg of the robot. In the classical methods, actuation is used to counteract energy lost during impacts. It is possible to determine the exact amount that will be lost during the next ground phase, and prepare to add that same amount back into the system in advance to create a net zero loss of energy.

The foot is assumed to be massless for the fSLIP model, which means that energy is conserved through the ground contacts. This leads to no need for any actuation to maintain a given vertical hop height. However, actuation will be used to change hop height due to the inherent difference in energy required for a mass at a given height.

Foot Placement Foot placement is used to effect the dynamics of the robot to provide stability. This will first be shown for the case when you wish to reach zero velocity. In the classical methods, the desired placement of the foot, x_{td}, is given with respect to the hip joint between the leg and torso, where a positive value represents placing the foot in front of the hip joint and a negative value means placing the foot behind the hip joint. Here, the hip joint represents the joint between the leg and flywheel. The calculation for this placement is based on the desire to keep the foot nominally below the robot's center of gravity, with small deviations designed to correct any unwanted motion. Consequently, the equation for x_{td} can be broken up into a term for the center of gravity offset and a second term other for the error correction, given as

$$x_{td} = x_{cg} + x_{err}. \tag{6.7}$$

The error correction term can be made up of multiple feedback metrics added up. The choice made in (M. H. Raibert, 1984) included three such terms given by

$$x_{err} = k_1(\dot{x}_T - \dot{x}_{T,d}) + k_2(\theta_T - \theta_{T,d}) + k_3(\dot{\theta}_T), \tag{6.8}$$

where $\dot{x}_{T,d}$ and $\theta_{T,d}$ are the desired values for the torso's forward velocity and orientation, $\dot{x}_T$ and θ_T, and k_i are controller gains.

In the fSLIP model, the flywheel's center of mass is located at the revolute joint so the angle of the flywheel does not effect the dynamics. Therefore, the second term in (6.8) may be omitted leaving just two state-based terms determining the placement of the foot. This leads to the forward velocity of the flywheel and leg to be equivalent. The modified version of the error correction term is

$$x_{err} = k_1(\dot{x} - \dot{x}_d) + k_2(\dot{\theta}_w). \tag{6.9}$$

Once the foot placement distance has be chosen, the desired angle of landing can be found via

$$\theta_L = sin^{-1}(\frac{-x_{td}}{r}), \tag{6.10}$$

where $r = L_0 - \delta$ is the instantaneous length of the leg, given the current deflection of the spring, δ.

Leg-Sweeping In order to maintain a velocity, the foot placement algorithm must be altered. The method discussed in the previous sub-section assumes that the robot holds position while on the ground, and acts only to pivot about the ground contact. The distance traveled by center of mass during the stance phase is linearly approximated by

$$\Delta x_s = \dot{x}_d T_s, \tag{6.11}$$

where T_s is the duration of the stance phase. Due to the nature of the spring-leg system, the duration of the stance phase closely follows the half-period of the natural frequency oscillation of the spring-mass system given by

$$T_s = \frac{\pi}{\omega_n} = \pi\frac{m}{k_s}. \tag{6.12}$$

The nominal touchdown distance is modified to place the foot underneath in the center of the body traversal as

$$x_{td} = x_{err} + \frac{\Delta x_s}{2}. \tag{6.13}$$

With the desire to keep the body following the same velocity throughout the stance phase, the body center must follow a distinct path. The corresponding foot placement trajectory is given by

$$x(t) = x_{td} - \Delta x_s\frac{t - t_{td}}{T_s}, \tag{6.14}$$

where t_{td} is the time of touchdown and t is the time within the stance phase with respect to t_{td}. This second term acts to track the foot placement throughout stance

and correct for any error during each step. The touchdown angle can now be made a function of this time-varying stance position as

$$\theta_L(t) = sin^{-1}(\frac{-x(t)}{r}). \tag{6.15}$$

In the case where there is a large difference between actual and desired forward velocity, the robot will attempt to place its foot further away from its center of gravity. This will lead to large ground reaction force in the lateral direction, which could violate the friction cone conditions typical of ground forces. To prevent such events, the desired velocity can be modulated to limit the acceleration or deceleration of the robot, just as in (M. H. Raibert, 1984). This modulation method is described by the logical condition

$$\Delta x_s = \begin{cases} (\dot{x} - \Delta\dot{x}_0)T_s, & \text{if } (\dot{x} - \Delta\dot{x}_d) > \Delta\dot{x}_0 \\ (\dot{x} + \Delta\dot{x}_0)T_s, & \text{if } (\dot{x} - \Delta\dot{x}_d) < -\Delta\dot{x}_0 \\ \dot{x}_d T_s, & \text{otherwise,} \end{cases} \tag{6.16}$$

where $\Delta\dot{x}_0$ is the max change in velocity.

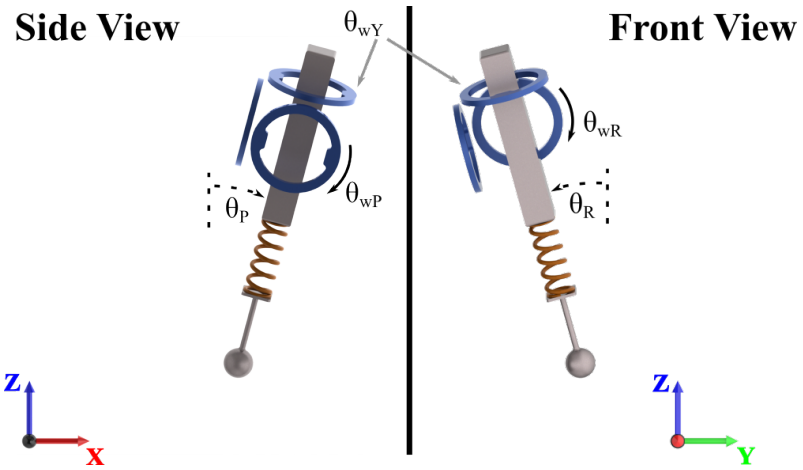


Figure 6.12: Mathematical model of the 3D Flywheel Hopper.

3D Hopping Specialization The foot placement techniques described in the previous section were developed for the ideal 2D-fSLIP model. In this section the controllers will be extended to the 3D hopper model which more closely represents the robotic system that will be soon developed in the lab. The goal is to use these methods to assess the performance of this hopping robot before the design is finalized. Fig. 6.12 shows the 3D hopper model with some of the new coordinates

labelled for clarification. The robot will consist of a body with three orthogonally-mounted flywheels, which will be able to control the full body orientation. This model assumes that the center of mass for the body-flywheel assembly is at the point of intersection for the three flywheel joint axes. This could be reasonably matched on actual hardware later on by adding a counter weight as was modelled in this simulation. Below the body is a linear spring and foot with mass. The contact between the foot and the ground is assumed to be a point contact.

The orientation of the body is modeled as three rotations in the order of Roll (θ_R-rotation about X), Pitch (θ_P-rotation about Y), and then Yaw (θ_Y-rotation about Z). The angle of the flywheel connected along each of these axis will be labeled as θ_{wi}, with $i \in [R, P, Y]$. As was the intention with these methods, control in the X-Z plane and the Y-Z plane will be kept separate and treated as a decoupled system. However, the simulation method described later in the section will follow the full non-linear dynamics of the robot. In order to simplify the section, the modified methods will be described for only control in the X-Z plane which involves controlling around desired values of $\dot{x}$, θ_P, and θ_{wP}. The same methods will be used for the states in the Y-Z plane during simulation.

Due to the presence of the foot's unsprung mass, the system will now lose energy during each ground phase of the hopping cycle. To make up for this loss, a linear actuator is placed in series with the spring. The controller for this linear actuator follows the same principles as in (M. H. Raibert, 1984),(E. Ambrose, N. Csomay-Shanklin, et al., 2019), where the amount of energy in the system is calculated at take-off and compared with the desired energy to reach a chosen height. The linear actuator can then choose how much energy to store and then release during the upcoming ground phase in order to change the current energy level. The equations for calculating the desired spring compression, δ_0 is given as

$$\delta_0 = \sqrt{\frac{2\Delta E}{k_s}}, \tag{6.17}$$

where ΔE is the required input energy to reach the desired hop height during the next air phase. The actuator will compress the spring by this amount during each air phase and release it during the ground phase.

Now that the masses and inertias of the robot model are no longer trivial as in the fSLIP case, the foot placement controller must also be modified accordingly. The changes show up in the estimate of the center of gravity, x_{cg}, since the mass is no

longer condensed to a point. This distance is given by

$$x_{cg} = -\frac{rM_f sin(\theta_P)}{M_0}, \tag{6.18}$$

where M_f is the mass of the foot and M_0 is the total mass of the system. Since this equation depends on the touchdown angle, the formulation of the desired touchdown distance must be solved for in combination with the error correction term described in (6.9). The final equation for the touchdown distance comes out to be

$$x_{TD} = \frac{M_f}{M_0 - M_f} x_{err}. \tag{6.19}$$

From here, the desired touchdown angle can be solved for again using (6.10).

The leg sweeping adjustments to the foot placement algorithm used in the fSLIP model were tried in simulation, but the forward velocity tracking was not very successful and lead to steady-state velocity error. When the feedback gain on the flywheel velocity was lowered, the steady-state error would decrease but then the flywheel velocity would build up to infeasible levels. Conversely, if the gain on body velocity was increased then the hopping became unstable. The end result used in simulation was to add a third feedback term on the desired body position to combat the negative effect of the flywheel velocity gain. The new equation for touchdown distance error correction is given by

$$x_{err} = k_1(\dot{x} - \dot{x}_d) + k_2(\dot{\theta}_w) + k_3(x - x_d), \tag{6.20}$$

where x_d is determined by the initial and desired velocities, as well as the velocity change limits. With this additional term, the robot was seen to track the desired velocity better while still adhering to the flywheel velocity limits.

3D Hopping Simulations In order to assess the performance and stability of the controllers on the 3D hopper platform, a fully non-linear model of the robot was created in *Simscape*. The parameter values for mass, inertia, lengths, and torque/velocity limits were chosen to reflect those of the future hardware system. A few of these values are given here as reference. The mass and max principle inertia of the hopper body were set to 4.8 kg and 0.052 kg m^2. The mass and max principle inertia of the flywheels were set at 0.3 kg and 0.0012 kg m^2. The torque and velocity limits for the flywheels were placed at 2.0 N m and 500 rad/s.

Ground contact was simulated through the addition of a *SphereToPlane* contact from the *Simscape Multibody* Library. This model block treats the ground contact as a

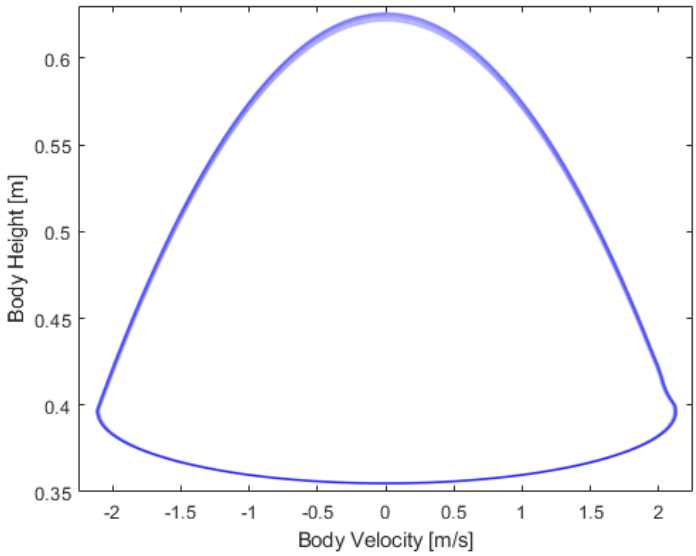


Figure 6.13: Phase portrait for hopping over a period of 25 hops.

spring-damper system, with user-defined stiffness and damping parameters. These were chosen to be fairly high to prevent noticeable bouncing between the foot and surface during the ground phases of the hopping motion. Simulations were run to show the controllers ability to stabilize the robot at a given hop height. Fig. 6.13 shows the phase portrait for 25 consecutive hops in simulation at a height of 0.625 m. This is the height that will be used throughout the following foot placement simulations of the hopper controllers as well.

The foot placement algorithm for the case of stabilizing the robot to zero velocity was tested next for both the 2D and 3D cases. The body was given an initial velocity of 0.5 m/s in either the x axis for the 2D case, or both for 3D. The controller gains were tuned again for this model, and were finally set at $k_1 = 0.08$ and $k_2 = 0.00025$. Due to friction constraints between the foot and ground in simulation, the robot is only able to slow down or speed up in increments. The method of limiting acceleration and deceleration used in (6.16) was implemented here with the max desired change in velocity for a single hop set to 0.2 m/s. Fig. 6.14 shows the simulation results for this behavior in the 2D case over a period of 7 s. The robot was able to bring its body velocity and flywheel velocity down to zero within 15 hops, and easily stayed within the flywheel torque and velocity limits.

A final set of simulations for the hopper were run to test the leg sweeping and velocity tracking controllers. As mentioned previously, the standard form of the error correction terms was not sufficient to track velocity, leading to steady-state

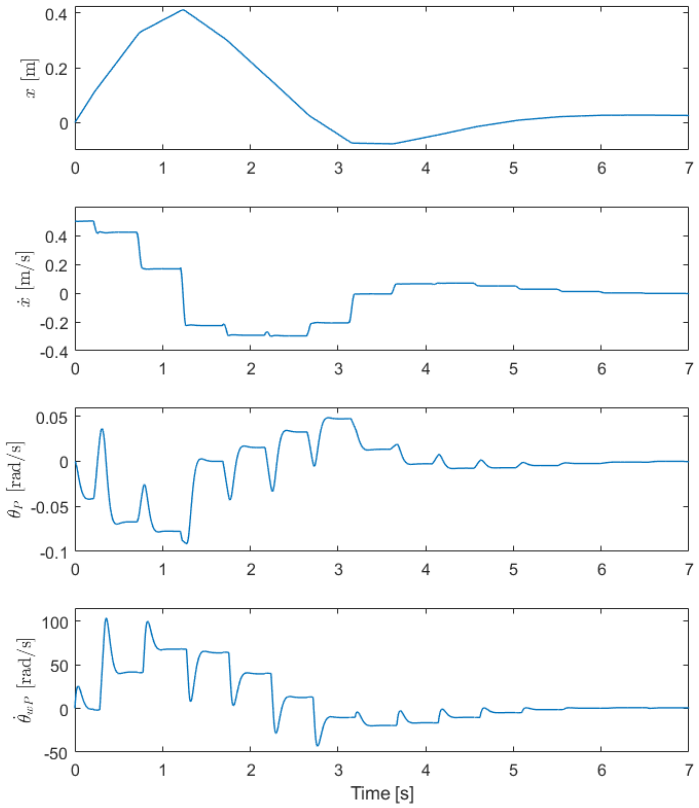


Figure 6.14: Simulation for the foot placement algorithm on the 3D hopper, showing settling after 7 seconds. State data are shown for a) Body forward position, b) Body forward velocity, c) Body pitch angle, and d) Pitch flywheel velocity.

error for the forward velocity. This was fixed through adding the third term with feedback on the desired position in (6.20), essentially acting as an integral term on the velocity. Fig. 6.15 shows the results of a simulation with an initial forward velocity of zero, and a desired velocity of 0.3 m/s.

Final Design of the Flywheel Subsystem

The simulations described previously were used to determine the key parameters of the system through an iterative process. The objective was to select the mass and inertia of the flywheels, as well as the motor used to spin them. Selecting the motor was the first step of this process, since the torque would show how much orientation error could be corrected in a single hop. The simulations were run using different peak torques in order to establish if certain actuators could bring the robot back to rest in a short time. Through a few attempts it was found that 2 N m of torque would

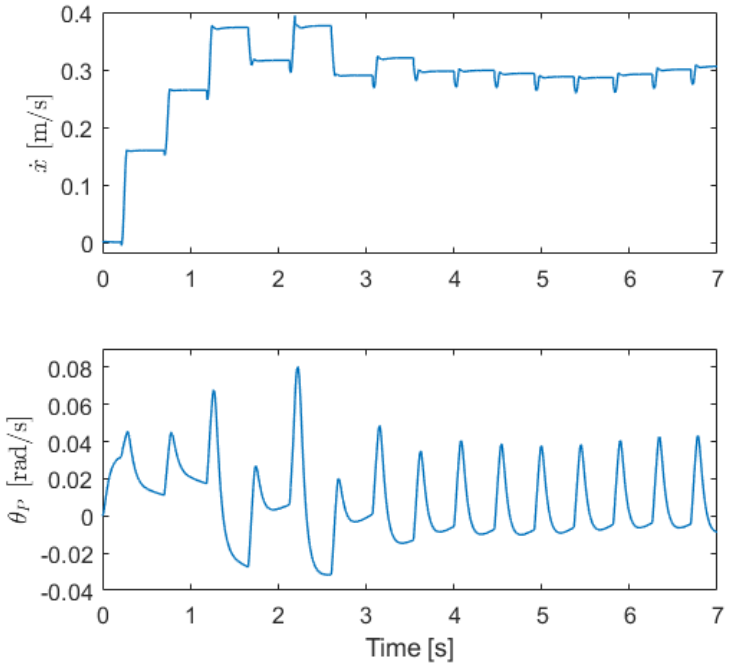


Figure 6.15: Simulation for the foot placement algorithm on the 3D hopper, showing settling after 7 seconds. State data are shown for Body forward velocity (top) and Body pitch angle (bottom).

be enough to achieve this. With the goal of choosing the lightest motor possible, out-runner motors were the style of choice. The specific model chosen was the Anti-Gravity MN7005, KV115 motor from T-Motor (T-Motor, 2021a). This motor was designed to have a very minimal frame, leading to a mass of 188 g while still being capable of the required torque and over 750 W of power.

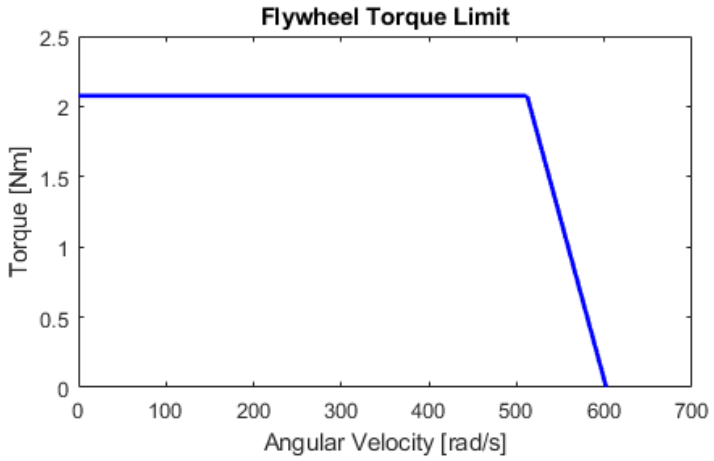


Figure 6.16: Plot of the torque limit of the flywheel motors as a function of their velocity.

With this motor and an assumed max battery voltage of 50 V, the max allowable velocity of the flywheels can be found to be 600 rad/s. However, the current that can be run through the motor will diminish as the motor velocity approaches this max velocity. Fig. 6.16 show this limited torque, which begins once the motor gets above 511 rad/s. Once the motor reaches its ultimate limit, the motor will be unable to apply torque in that direction anymore, severely limiting its control bandwidth.

In order to keep the motor away from the limits as effectively as possible, its better to have a flywheel with large inertia. However, this would also drive up the mass of the flywheel and, therefore, the whole robot as well. The question was how large of an inertia could be reasonably allowed for this robot. The max diameter of the flywheel was already determined from the CAD model as 150 mm, and similarly the thickness was limited to 15 mm. Flywheels of a few different masses were designed in CAD, attempting to get the largest inertia possible within the geometric limits mentioned. Each of these models were examined in the simulations via using their mass/inertia in the Simscape model. It was determined that the flywheel with a mass of 250 g, and inertia of 0.001 25 kg m^2, was the smallest version that could perform the constant velocity simulations up to a forward velocity of 0.5 m/s without crossing the speed threshold of the motors, so this was the size that was selected.

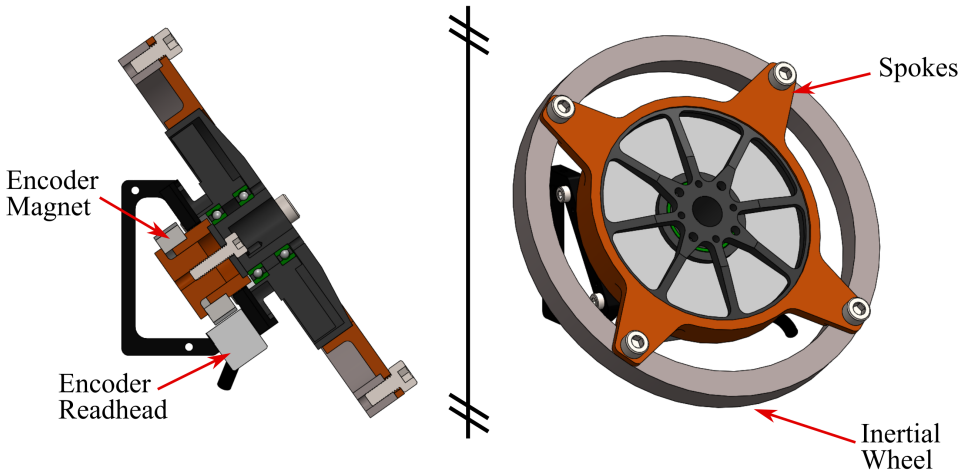


Figure 6.17: CAD model of the flywheel-actuator assembly, including the feedback encoder and frame mounting plates.

Fig. 6.17 shows the final model of the flywheel actuator. The inertia wheel itself is attached to the motor through a thin aluminum spoke frame. This spoke piece is glued directly to the rotor of the motor, while the other end is bolted to the main piece of the flywheel, which is a stainless steel ring in order to drive up the inertia-mass ratio of the overall flywheel. The feedback and motor controller for

these actuators are the same as for the compression actuator, meaning a incremental encoder (Renishaw, 2021) mounted to the back of the motor, and driven by an ELMO Twitter (ELMO-MotionControl, 2021). The flywheel actuators are attached to the main body of the robot through aluminum frame pieces. The frame pieces directly holding the motor and motor controllers have heat sink fins machined into their backs, in the hopes this will help transfer heat away from these components during experiments and allowing the performance of the motors to be less limited by heat.

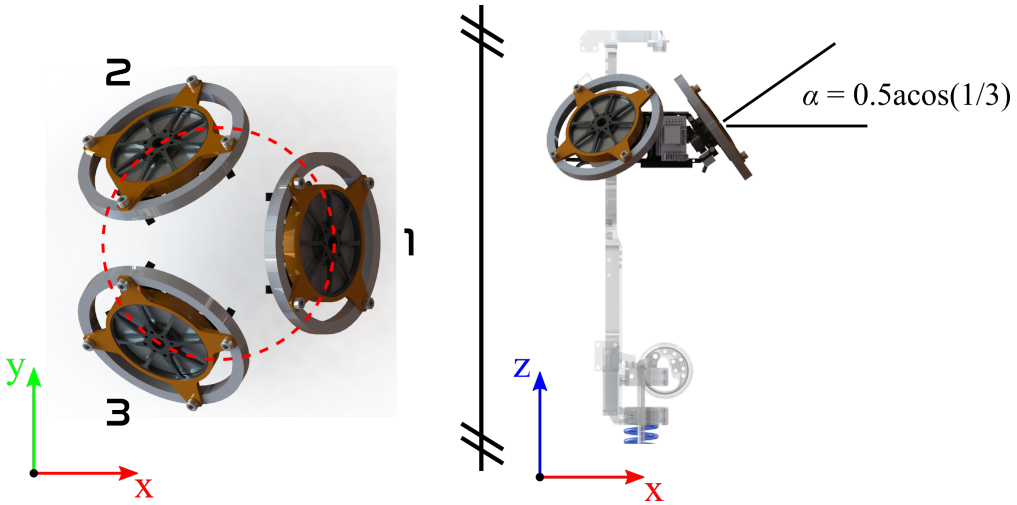


Figure 6.18: Layout of the 3 Flywheels within the robot. In the right-hand image, the angle of the flywheel axis is called out relative to the horizontal plane.

The right side of Fig. 6.18 shows the 3 flywheels within the frame of the final robot CAD, with most of the other features hidden or faded to highlight the flywheels only. The model used previously was adapted to have the flywheels mounted at these final symmetrical positions, rather than in line with the body's Yaw, Pitch, and Roll axes as can be seen in Fig. 6.18. Each of the flywheels are angled upwards from level by an angle of $\alpha = 0.5acos(1/3)$, and spaced 120° apart around the central vertical axis of the robot. The flywheels are still orthogonal to each other, but rotated so they are symmetric with respect to the robot allowing for the robot's center of mass to remain on the center axis. You can imagine that this change of flywheel orientation is like taking a cube that is flat on a table, and rotating it so its standing up on one of its corners. The three flywheels are labelled 1-3 in Fig. 6.18, with flywheel 1 on the front of the robot and having its axis of rotation lying within the x-z plane.

The next section will cover the culmination of the ARCHER design, including the mathematical explanation for the decoupled nature of this system's dynamics.

6.3 Final Design of ARCHER

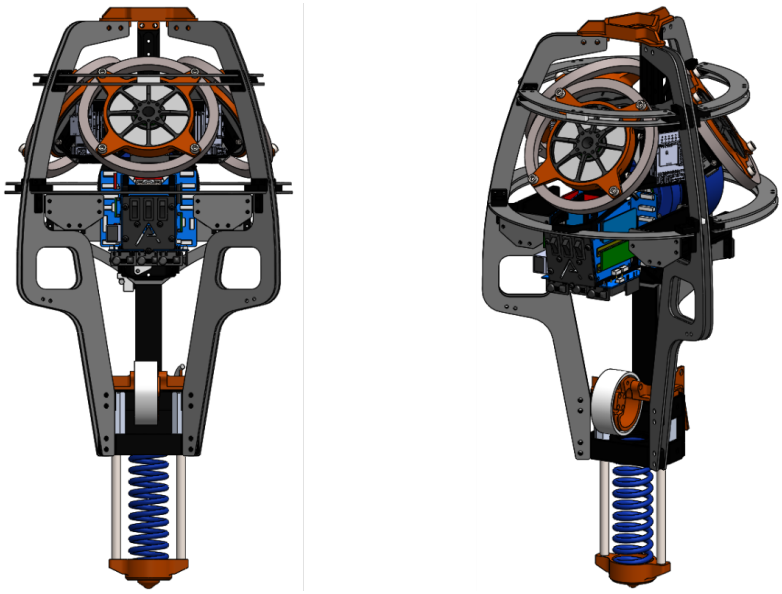


Figure 6.19: Final CAD model for ARCHER, showing the combined leg compression and flywheel balancing subsystems within its frame.

Component Layout and Specifications Fig. 6.19 shows the full CAD model of ARCHER, including its compression actuator, spring leg design, and the recently described flywheel attitude control system. The central spine of the robot along with the frame holding the flywheels is made of anodized aluminum, which was custom machined to allow for the core mechanisms to be precisely aligned. The peripheral frame is made of thin plates of carbon fiber to save on weight, while still providing support. Additional rings of carbon fiber were placed around the perimeter of the upper portion of the robot in order to prevent the flywheels from contacting the ground if it fell over.

Power is provided from two on board LiPo batteries connected in series. The voltage from the batteries is up to 50.8 V, but will often be slightly lower as the batteries drain. This voltage is the source of velocity limit on the flywheels. The batteries are able to deliver over 100 A of current, which is more than the 4 motors of the robot will ever need at one time. These batteries are placed in compartments on either side of the upper pulley, slightly shifted towards the front of the robot to help offset the rear-biased weight of the compression actuator. This can be seen in Fig. 6.20 along with the main power relay of the robot. This relay allows for a switch on the front of the robot to control when power is given to all of the electronics.

The main computation and user interface happens at the front of the robot just below

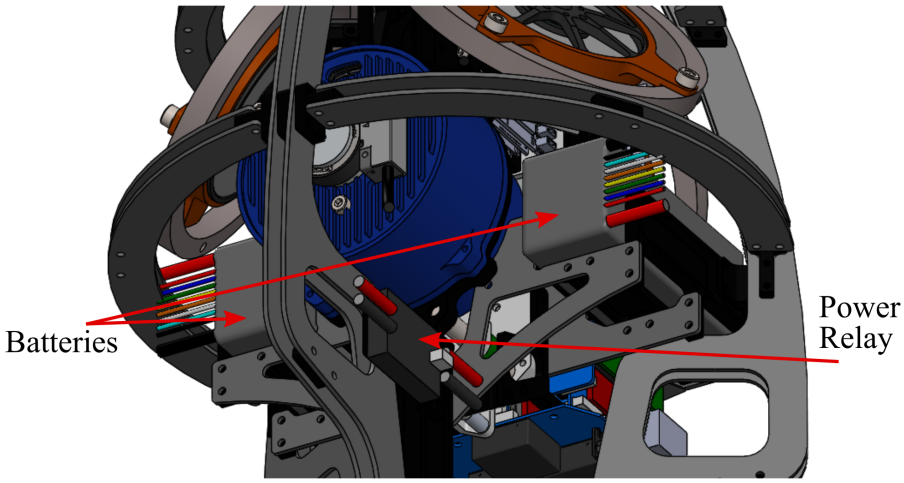


Figure 6.20: Close up of the batteries and power relay, from a perspective looking up at the robot.

the flywheel, as can be seen in Fig. 6.21. The front plate of the electronics contains three switches and an LED. These three switches control the main power, safety-stop for the motor controllers, and an additional user interface switch to interact with the controllers. There are two control circuit boards on the robot; one on each side of the main switch board. These divide the work load of the control into subsystems of the flywheels and the leg actuator. This not only makes the speed of each controller faster, but also helps demonstrate the decoupled nature of this robot's dynamics. The computation on ARCHER was designed to be performed by two Teensy 4.1 Arduino boards (one per control board). These boards work very well for communicating with sensors, motor controllers, and other devices, but are

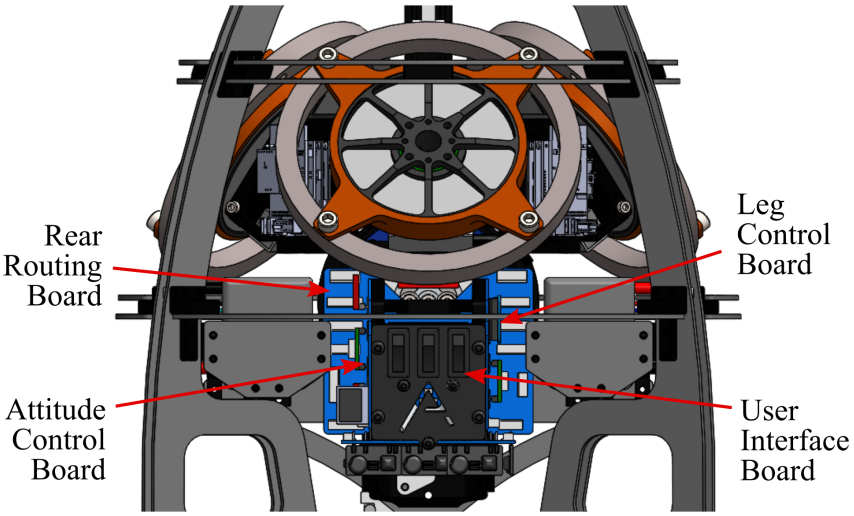


Figure 6.21: View of the robot's front with call-outs for the main electronics.

limited in their computational ability (PJRC, 2021) compared to typical embedded computing options. These two controllers have a signal exchange between them for the purpose of syncing up at the start of the experiment, which makes sure they start their portions of the hopping motion together, as well as have correct relative timing in their data logs. There is an additional circuit board at the back of the control area which routes all of the feedback and communication signals to the motor controllers.

Because the foot is connected to the leg actuator through a cable, it is not possible to always measure the position of the foot using the encoder on the compression actuator. The cable itself is made from a polyester shell with a Teflon core to limit the stretch in the cable, but the cable does still stretch a little despite the use of Teflon, adding to the need for another way to measure the foot position. This is accomplished through a separate encoder placed directly at the foot. This encoder is used to detect ground contact, determine the moment of launch, and measure current foot deflection. This sensor will also be used in a pre-experiment process to find the zero position of the leg actuation. This is critical because of the cable's ability to stretch. Fig. 6.22 shows a close up this encoder and the connection between the foot guides and the spine of the robot. This encoder functions like the ones on the motors, but uses a linear magnetic strip attached to the side of the robot's spine.

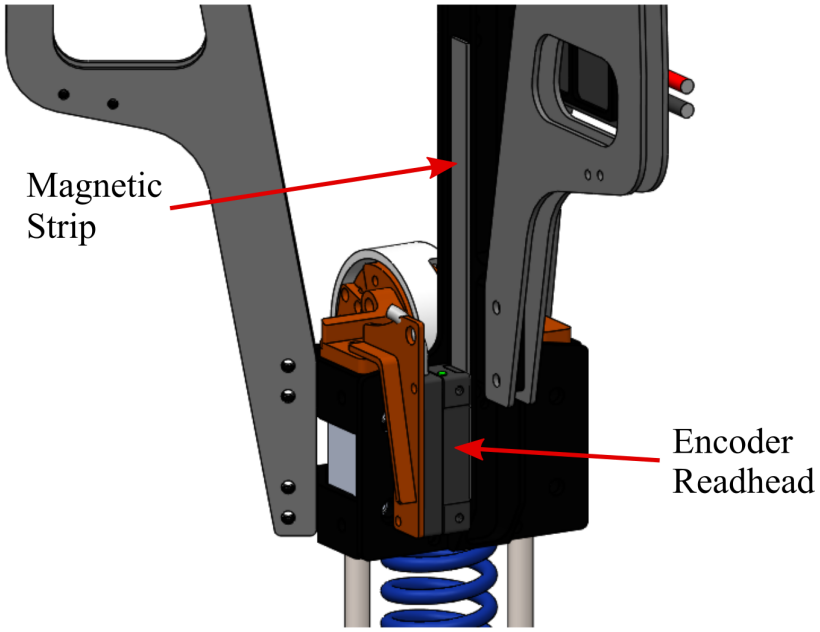


Figure 6.22: Close up of the foot joint on ARCHER from behind. Note the orange frame piece connecting the foot encoder to the rest of the foot assembly.

The total mass of the robot comes out to around 6.5 kg, with a height of 54 cm and a diameter of 26 cm at its widest point. With the known final mass, the spring stiffness was also selected to be 11 732 N/m. Now that the model of the robot being finalized to the point of manufacturing, it was also time to write out the final equations of motion in preparations for the simulations that would be performed next.

Decoupled Dynamics ARCHER has 10 degrees of freedom, with the configuration variables defined as

$$q^\top = (\underbrace{x, y, z}_{p}, \underbrace{\phi_x, \phi_y, \phi_z}_{\phi}, \underbrace{\theta_1, \theta_2, \theta_3}_{\theta}, \delta) \tag{6.21}$$

where $(x, y, z, \phi_x, \phi_y, \phi_z)^\top \in \mathbb{R}^6$ are the floating base coordinates for the robot's body at the center of mass, $(\theta_1, \theta_2, \theta_3)$ are the angles of each flywheel relative to the body, and δ is the deflection of the leg spring. There are 4 actuators on the robot; one directly attached to each flywheel, and one controlling the compression of the leg spring. This gives the control inputs of

$$u^\top = (\underbrace{u_1, u_2, u_3}_{u_1}, \underbrace{u_l}_{u_2}) \tag{6.22}$$

where we are now denoting the split between the subsystems, with the subscript 1 for the attitude control subsystem, and the subscript 2 representing the leg subsystem.

The continuous-time dynamics of the robot are then given as:

$$\mathcal{R} \triangleq \begin{cases} D(q)\ddot{q} + H(q, \dot{q}) = B(q)u + J_v(q)^\top \lambda_v \\ \text{s.t.} \qquad h_v(q) \equiv 0 \end{cases} \tag{6.23}$$

where $h_v(q)$ is the holonomic constraint on the robot during motion, $J_v(q) = \partial h / \partial q$, and λ is the vector of constraint force magnitudes. The holonomic constraints vary depending on the domain of motion, either the stance or flight phase, denoted by the subscript $v \in (s, f)$. During the stance phase, the constraints act to keep the toe of the robot in static contact with the ground, while in portions of the air phase the constraints act to keep the spring from extending beyond its set pre-load length (based on the physical hardstop built into the foot, as in the 1D CRH robot).

Through structuring the actuation matrix, $B(q)$, in a specific way we can observe the important decoupled nature of this robot's dynamics. The actuation throughput

portion of the dynamics can written as

$$B(q)u = \begin{bmatrix} \mathbf{0}_{(3x3)} & B_{p_2}(\phi) \\ B_\phi(\phi, l) & \mathbf{0}_{(3x1)} \\ B_\theta & \mathbf{0}_{(3x1)} \\ \mathbf{0}_{(1x3)} & B_l \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{6.24}$$

Due to the way this actuation matrix was set up, we can note a few important facts:

- B has no terms that depend on p or θ.
- B_{p_2} only depends on ϕ.
- B_θ and B_l are constants.
- u_2 does not directly effect the angular coordinates of the robot.
- u_1 has no effect on the spring deflection.

These last two bullet point show definitively that the two subsystems of the robot can be decoupled, so the control and planning can be separated. This was a large motivation for designing the robot with flywheels. The next section will show the final simulation structure used to plan the motions and control strategies to be used in experiments. These simulations will use the final model parameters from the CAD model to start, and slowly move over to measured values from hardware once the robot is built.

6.4 Simulation and Controller Design

The Simscape simulation framework used previously in the design of the flywheels and the foot placement planning method was adapted to this final full-body model. All of the mass, inertia, and other model parameters were updated to match the CAD model. The final inertia matrix for the body was increased slightly from the CAD in order to account for the mass of the cabling that would be present on hardware. As mentioned in the previous section, the flywheels were realigned to match the hardware rather than remaining in-line with the robot's Cartesian frame. This required that the base frame of the simulation move away from a series of 3 individual rotary joints representing the body's rotation, to a spherical joint which would provide a quaternion as positional feedback. The overall coordinate frame remains the same, with the alignment of x-forward, y-left, and z-up. Lastly, the timing of the foot's spring compression algorithm was updated to be more realistic to future experiments. Rather than constantly tracking the energy level of the system,

this would wait until the robot was beginning to to descend in the air phase, before starting to track the desired compression of the spring. This would place a safety factor of sorts on the compression time to make sure it would be achievable on hardware.

The only remaining piece of the simulations that needed to be developed were the attitude controllers. The next two subsections will delve into the formulation of both a standard linear control method, and a nonlinear CLF-QP control method and their performances in Simscape.

PD Controller

A simple PD controller will be used on the orientation of the robot based around the quaternion feedback along with angular rates, as that is the state information that will be received from the IMU on the actual hardware. Since the flywheels axes do not line up with the primary directions of motion that will guide the robot, there was a need to create a control strategy that will use the properly aligned IMU data. This was essentially accomplished through checking the error between desired and actual quaternions and mapping it to each of the three flywheel axes to yield their contribution to the control action required to correct the error. This subsection will show the math behind this control method and the performance of the simulations.

In general, the behaviors that we will want from the robot's orientation will be stated in terms of yaw (Y - rotation about z-axis), pitch (P - rotation about y-axis), and roll (R - rotation about x-axis) angles. This was based on the formulation of the foot placement planner that was described previously, being in terms of pitch and roll angles to reach a given foot touch-down position. Since the orientation state feedback will be given in terms of quaternion, we will need to relate our desired positions, (Y_d, P_d, R_d) in the same frame. The process of converting our desired YPR angles into a quaternion requires creating 3 vectors:

$$q_1 = (cos(Y_d/2), 0, 0, sin(Y_d/2))^T \tag{6.25}$$

$$q_2 = (cos(P_d/2), 0, sin(P_d/2), 0)^T \tag{6.26}$$

$$q_3 = (cos(R_d/2), sin(R_d/2), 0, 0)^T \tag{6.27}$$

and then multiplying them together as $Q = q_1 * q_2 * q_3$, where each multiplication is done via quaternion multiplication. Given two quaternions, $q = (q_0, q_1, q_2, q_3)^T$ and $p = (p_0, p_1, p_2, p_3)^T$, quaternion multiplication of the two is performed as

follows:

$$q * p = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{6.28}$$

This multiplication is performed sequentially to get the final quaternion of the desired orientation, Q_d. To get the error between our actual quaternion, Q_a, and desired, another multiplication is performed yielding a new quaternion vector

$$Q_e = Q_d * Q_a. \tag{6.29}$$

From here, we need to find the axis vector associated with the error and the magnitude of the error. The magnitude of the error quaternion is found by simply taking the 2-norm of the last 3 entries in $Q_e = (w_e, x_e, y_e, z_e)^T$, specifically $A_e = \sqrt{x_e^2 + y_e^2 + z_e^2}$. The entries of the axis vector of the error quaternion, $V_e = (e_1, e_2, e_3)^T$ are defined as follows:

$$e_1 = x_e/A_e \tag{6.30}$$

$$e_2 = y_e/A_e \tag{6.31}$$

$$e_3 = z_e/A_e \tag{6.32}$$

The flywheel axes will need to be compared to this orientation error axis and the vector of body angular rates, $w = (dY, dP, dR)^T$, in order to determine how much contribution they need to make towards correcting the error. This will require knowing the direction vector of the flywheels. The default axis vectors for the three flywheels when the body is in its zero position are given by

$$f_1 = (cos(\alpha), 0, sin(\alpha))^T \tag{6.33}$$

$$f_2 = (-0.5cos(\alpha), \frac{\sqrt{3}}{2}cos(\alpha), sin(\alpha))^T \tag{6.34}$$

$$f_3 = (-0.5cos(\alpha), -\frac{\sqrt{3}}{2}cos(\alpha), sin(\alpha))^T \tag{6.35}$$

where $\alpha = 0.5acos(1/3)$, listed in Fig. 6.18, is the angle each flywheel axis is angled up from the horizontal plane.

Each of these axis vectors are rotated from their zero position as the robot moves, by the amount defined in the quaternion, Q_a. Converting this quaternion into euler

angles of YPR is done through via

$$Y = atan2(2(qa_0qa_1 + qa_2qa_3), qa_0^2 + qa_3^2 - qa_1^2 - qa_2^2) \tag{6.36}$$

$$P = asin(2(qa_0qa_2 + qa_1qa_3)) \tag{6.37}$$

$$R = atan2(2(qa_1qa_2 + qa_0qa_3), qa_2^2 + qa_3^2 - qa_0^2 - qa_1^2) \tag{6.38}$$

where qa_i is the i-th element of the Q_a vector. With these Euler angles, the default flywheel axes can be rotated to their current position by multiplication with the rotation matrix associated with those Euler angles, defined as

$$R_a = \begin{bmatrix} c(Y)c(P) & c(Y)s(P)s(R) - c(R)s(Y) & s(Y)s(R) + c(Y)c(R)s(P) \\ c(P)s(Y) & c(Y)c(R) + s(Y)s(P)s(R) & c(R)s(Y)s(P) - c(Y)s(R) \\ -s(P) & c(P)s(R) & c(P)c(R) \end{bmatrix} \tag{6.39}$$

where the cosine and sine functions have been abbreviated as $c(x)$ and $s(x)$. The final vectors for each flywheel axis are then found as

$$v_1 = R_a f_1 \tag{6.40}$$

$$v_2 = R_a f_2 \tag{6.41}$$

$$v_3 = R_a f_3 \tag{6.42}$$

With these vectors defined, the final control input for each flywheel can be calculated using the following control scheme:

$$u_1 = k_p||A_e||(v_1 \cdot v) + k_d||w||(v_1 \cdot w) \tag{6.43}$$

$$u_2 = k_p||A_e||(v_2 \cdot v) + k_d||w||(v_2 \cdot w) \tag{6.44}$$

$$u_3 = k_p||A_e||(v_3 \cdot v) + k_d||w||(v_3 \cdot w) \tag{6.45}$$

where k_p and K_d are the proportional and derivative gains in the controller. These inputs are also capped by the max torque of the system, 2 N m, before being sent as the final inputs in the simulation.

PD Controller Performance

Numerous simulations were performed in Simscape using this model and controller. In general, the robot began airborne and was then dropped with various initial conditions of orientation, angular velocities, and linear velocities. The robot was able to correct itself and reach a steady-state hopping in place motion over a wide

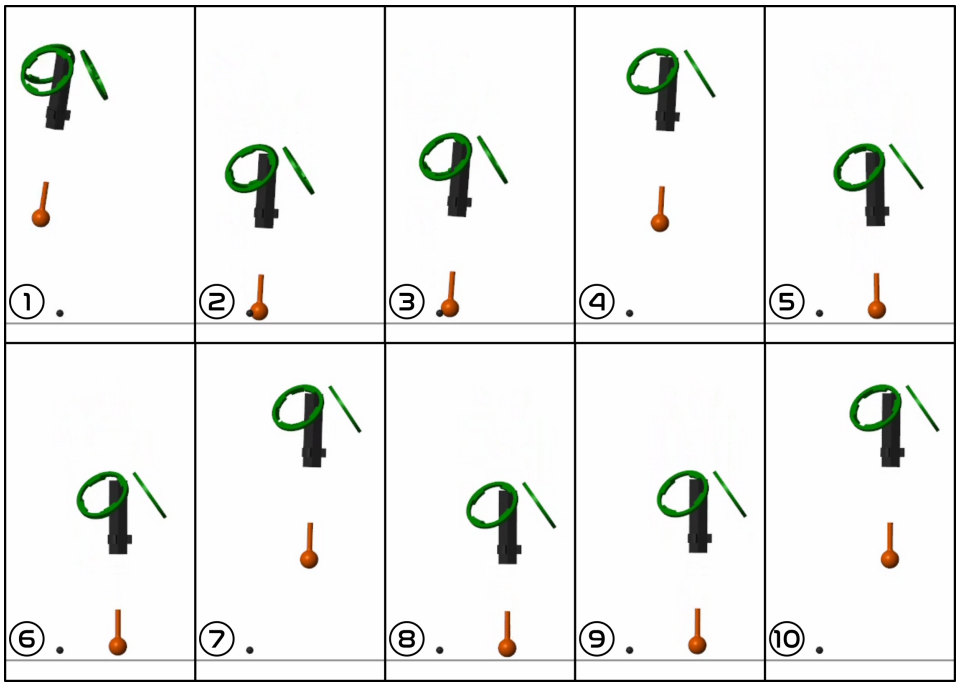


Figure 6.23: Motion tiles from a simulation with PD control (view from the side).

range of initial conditions. Based on the desired behaviors that would be performed in hardware-based experiments later on, the robot was limited to pitch and roll angles within 0.25 rad of vertical and the initial linear velocities were capped at 0.5 m/s. In some cases where the initial orientation and angular velocities were in an antagonistic condition, then the maximum initial linear velocity that could be corrected was lower, sometimes even as low as 0.15 m/s.

Data from a specific run are shown here for reference. In this simulation, the initial conditions were $P_0 = 0.15$ rad, $R_0 = 0.15$ rad, $\dot{x}_0 = 0.2$ m/s, and the robot was attempting to reach zero linear velocity while maintaining the hop height of 0.6 m. Fig. 6.23 shows motion tiles of this simulation over the first 3 hops, during which the robot is able to cancel out almost all of the unwanted forward velocity.

The horizontal velocity was tracked during the simulation, and showed the robot was able to come to a stop, with only small deviations around 0 m/s while hopping in place. Fig. 6.24 shows both the x- and the y-velocity over the first 4 seconds of the simulation, although most of the efforts to correct the initial condition were done throughout the first 3 hops which were shown in the tiles of Fig. 6.23. Despite having no initial y-velocity, the robot did still deviate from zero during the process of correcting its roll angle. Similarly, the orientation of the body was tracked to show its progression over that same time period. Fig. 6.25 shows the initial pitch

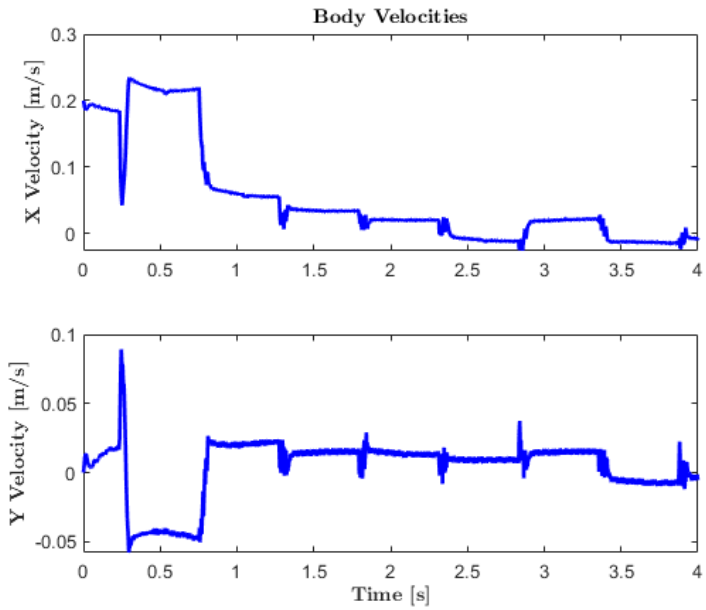


Figure 6.24: Horizontal body velocity data from one of the simulations, showing the robot's ability to limit unwanted initial conditions.

and roll offsets, and how the robot settles back towards its zero-position for all three attitude angles. Again, most of the progress towards leveling out happens within the first few hops. It should be noted here that, as with the body velocities, the robot tends to hover or oscillate around zero, but never truly settling fully.

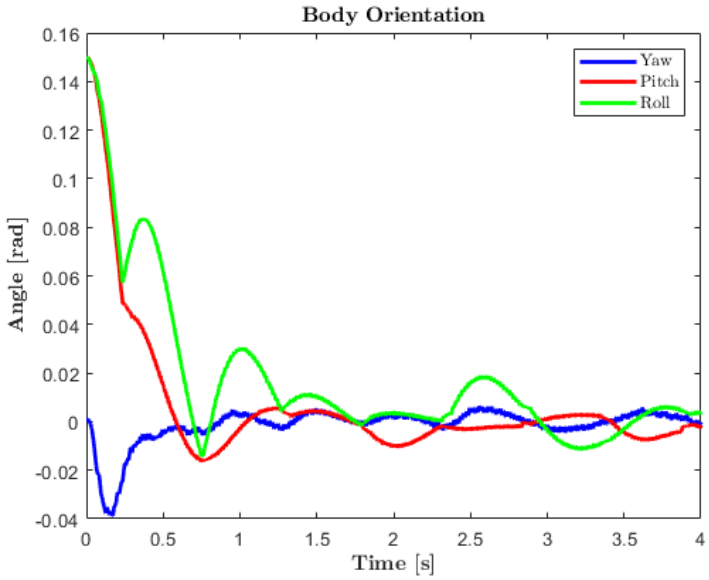


Figure 6.25: Attitude data from the example simulation, showing how the robot was able to successfully recover from initial angular offsets.

Based on the performance of the simulation model, it seems that this type of

controller could be easily adapted to the hardware for experiments, with some gain tuning for the attitude control and foot placement depending on the quality of feedback data being received from the on board inertial measurement unit (IMU). With a desire to test more stable and optimal control methods on hardware, another controller was tested in simulation to assess its efficacy. This is described in the next subsection.

CLF-QP Controller

A second version of the robot simulation was created using a different controller, namely a control Lyapunov function based quadratic program (CLF-QP) (Aaron D Ames and M. Powell, 2013). This was done by creating a few new blocks in the Simscape model, replacing the basic PD controller used previously. The algorithm for this new controller is described next.

Starting with the general dynamics of the robot, written as

$$\dot{x} = f(x) + g(x)u \tag{6.46}$$

$$y = h(x) \tag{6.47}$$

where $x = (q, \dot{q})^T$ are the states of the system, and $q \in \mathbb{R}^n (n = 10)$ are the coordinates of the robot. Further, $y \in \mathbb{R}^m$ is the vector of outputs for the controller. We wish to track the Roll, Pitch, and Yaw angles for the robot's base, so these outputs are defined as the difference between the actual and desired values of those angles, or $y(q) = y_a(q) - y_d$. In this case, there are 3 outputs, i.e. $m = 3$, which are all relative degree 2 outputs. Therefore:

$$\ddot{y} = L_f^2 h(x) + \underbrace{L_g L_f h(x)}_{A} u \tag{6.48}$$

where L denotes the Lie derivative. Using a general FL-IO controller of $u = A^{-1}(-L_f^2 h(x) + \mu)$ we can set

$$\ddot{y} = \mu \tag{6.49}$$

where $\mu \in \mathbb{R}^m$ can be of our design. We can re-write these output dynamics as

$$\dot{\eta} = \underbrace{\begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}}_{F} \eta + \underbrace{\begin{bmatrix} 0 \\ I \end{bmatrix}}_{G} \mu \tag{6.50}$$

where $\eta = (y, \dot{y})^T \in \mathbb{R}^{2m}$.

The goal from here is to set up a control Lyapunov function which will drive these outputs to zero. The general conditions of a stabilizing CLF are

$$\begin{aligned} C_1||x||^2 \leq V(x) \leq C_2||x||^2 \\ \dot{V}(x) \leq -C_3 V(x) \end{aligned} \tag{6.51}$$

where $C_1, C_2, C_3 > 0$ (Aaron D Ames, Galloway, et al., 2014). In regards to the control system listed in (6.50), we can formulate the following continuous time algebraic Riccati equation:

$$F^T P + PF - PGG^T P + Q = 0 \tag{6.52}$$

for $Q = Q^T > 0$ and solution, $P = P^T > 0$. The solution, P, can be used to construct a Lyapunov function of the form $V(\eta) = \eta^T P \eta$, which satisfies the first condition of (6.51). Taking the derivative

$$\dot{V}(\eta) = L_f V(\eta) + L_g V(\eta)\mu, \tag{6.53}$$

it is apparent that by choosing μ we can ensure that the second condition of (6.51) is met as well. Combining (6.53) and (6.50) yields

$$\dot{V}(\eta) = \eta^T (F^T P + PF)\eta + 2\eta^T PG\mu. \tag{6.54}$$

From here, a quadratic program can be formulated which finds an optimal solution for μ.

$$\begin{aligned} u^* = \underset{u}{argmin}\ & ||Au + L_f^2 h(x)||^2 \\ s.t.\ \ & \eta^T (F^T P + PF)\eta + 2\eta^T PG(Au + L_f^2 h(x)) \leq -C_3 \eta^T P \eta \end{aligned} \tag{6.55}$$

The final control actions sent to the flywheel motors takes the form of

$$u = A^{-1}(-L_f^2 h(x) + u^*) \tag{6.56}$$

This control method was implemented in Simscape using a custom control block, and preforming the QP algorithm via the built-in MATLAB *quadprog* function. The terms for the dynamics in (6.46) were generated offline using the built-in functions in the FROST package (Ayonga Hereid and Aaron D Ames, 2017) for generating dynamics, using the same model parameters as the Simscape model. The following section details the performance of the simulations using this controller.

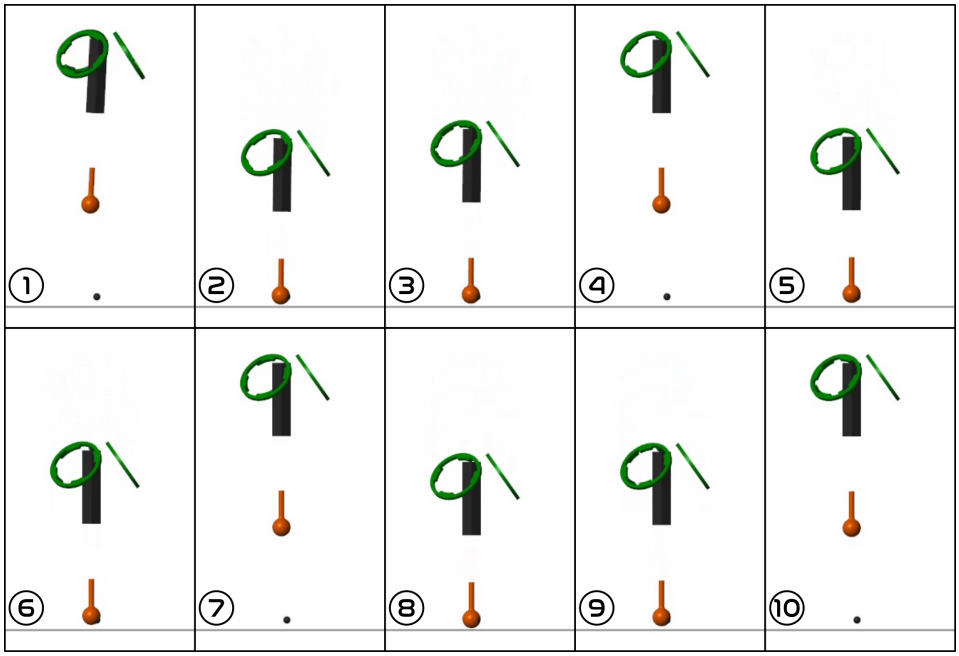


Figure 6.26: Motion tiles of the first three hops from a simulation with the CLF-QP controller (view from the side of the robot).

CLF-QP Controller Performance

Once again, the Simscape simulations were run on the same model as before with this new controller replacing the PD controller block. It was quickly seen that the window of initial conditions from which the robot could stabilize was rather small. When the initial conditions were above just a few degrees for roll and pitch offsets, or the initial x- and y-velocities were above a few cm/s, then the robot would fail after just a few hops. However, when the robot did start within that small domain, then it stabilized very quickly and smoothly. The data shown in the following figures and plots of this section will be from a successful run, with initial conditions near the boundary of that small window. The specific initial conditions in this run were $P_0 = 0.05\,\text{rad}$, $R_0 = 0.05\,\text{rad}$, $\dot{x}_0 = 0.05\,\text{m/s}$, and $\dot{y}_0 = 0.02\,\text{m/s}$.

Fig. 6.26 shows some motion tiles of the first three hops in this example simulation for this control method. Most of the course correction is happening during the first hop, so there is very little to distinguish the tiles of the second and third hop. As with the PD-controlled simulations, the body orientation and linear velocity data was collected as reference for the performance of the robot. As is seen in Fig. 6.27, the robot was able to stabilize itself very quickly throughout the first two hops of the simulation. The rest of the hops within the 4 second window stayed almost perfectly in place, showing just how stable the robot could be. At each impact, there was a small and brief blip in the velocity data, but it was removed almost immediately

upon entering the air-phase again. Fig. 6.28 shows the body attitude data from the same simulation, and once again the speed of stabilization can be seen. The amount of deviation from zero after stabilizing is also much smaller under this controller than with the PD controller.

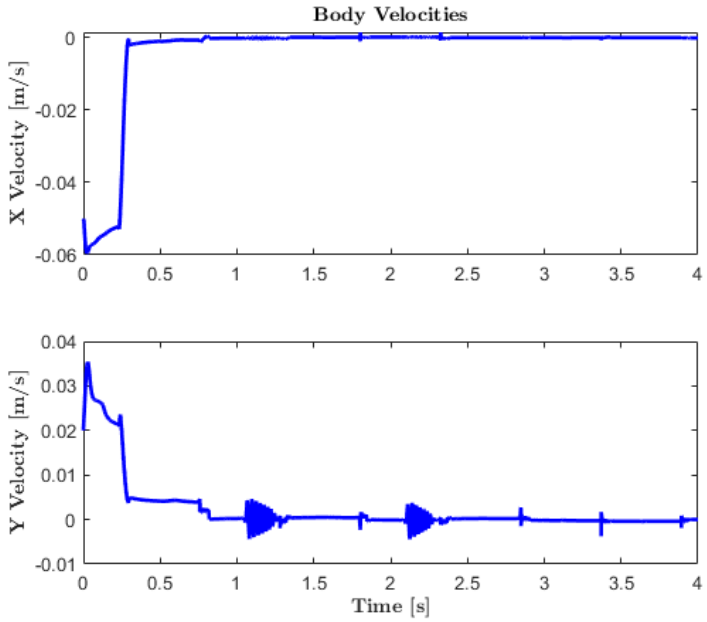


Figure 6.27: Horizontal body velocity data from a simulation using a CLF-QP controller, showing the robot's ability to limit unwanted initial conditions quickly while in its small window of stability.

Due to the limited computational ability of the on-board Teensy 4.1 boards, computation for this quadratic programming method would likely need to be performed by a different computer which would exchange motor input and feedback information with ARCHER in order to properly function in real-time. This inconvenience could be fixed by redesigning the computing electronics on the robot, but given the time frame of the project, this has yet to be done, and will be mentioned in the future work section later on. The following section will move onto the experiments of the robot, showing the results gathered thus far on ARCHER. These experiments will only be using the PD control method described previously.

6.5 Experiments with ARCHER

During the same period of time that the simulations in the last section were being created and run, the process of building and testing ARCHER was also commencing. The hardware of the robot was ordered, manufactured, and procured over the course of a year, with delays in the ordering of key electronics and some of the CNC

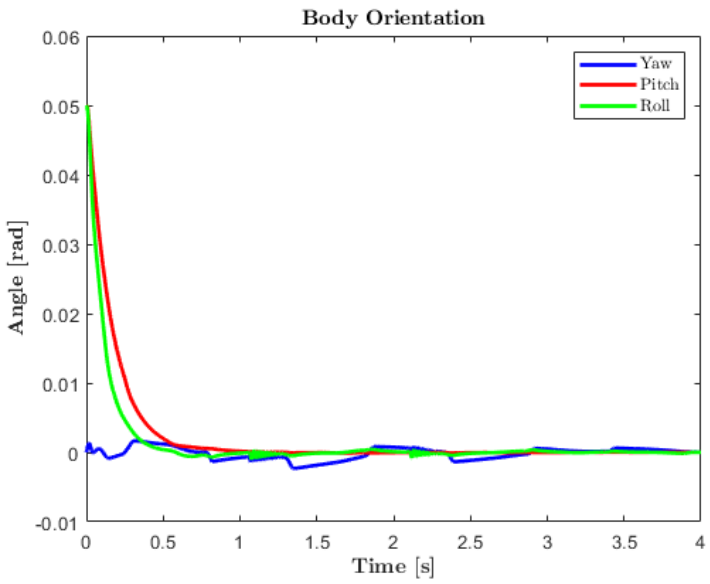


Figure 6.28: Attitude data from the example CLF-QP simulation, showing how the robot was able to successfully recover from initial angular offsets in just 2 hops.

made parts. As the components were coming in, preliminary assembly, testing, and preparations could be made for the final experiments. These will be detailed in the next subsection.

Preparing for Experiments

The custom parts for the compression actuator were manufactured in-house at Caltech before the rest of the custom machined parts arrived, so that it could be assembled and tested while waiting for the rest of the robot's components. This process involved using CNC mills to machine the gears, the inner aluminum transmission parts, and the housing. The motors, motor controllers, and some of the encoders were also among the first components to arrive, so testing of the motors and compression actuator was done first, which also allowed for the set up of the motor controllers to be performed. A torque cell was used to verify the torque constant of the compression motor, since it was custom wound and this parameter is critical to quickly controlling the spring deflection during hopping. The constant was measured to be between $0.175-0.190\,\mathrm{N\,m/A}$, which matched very closely to the expected value.

Next, all three flywheel motors were mounted so that a full test of communication could be done between the the motor controllers, motors, and Arduino. The VectorNav VN-100 IMU (VectorNav, 2021) was also setup within this testing, so that the direction/sign of feedback could be observed. This was put into the loop with the

motors to verify the directions the motors would spin to correct for different attitude errors.

Once all of the manufactured parts were in hand, and modified to meet tolerances, the central compression system was assembled. The compression actuator was tested with the cable and spring system for the first time. The amount of stretch in the cable was observed to be larger than expected. This forced the need of a initialization step for determining the amount of torque from the motor which would lead to spring compression. This step would mark both the torque and position of the motor at the moment when the foot encoder measures a change in deflection. To account for any possible changes to the cable during repeated experiments, this process is performed every time ARCHER starts up, before hopping begins.

The circuit boards for controlling the robot and routing power/signals, were designed using AutoDesk Eagle and sent to an online PCB shop to be fabricated. All component soldering was done in the AMBER Lab once the boards arrived. Each of the two control boards were tested in controlling their subsystem separately, before any full system tests were done to set up the syncing behavior between the two as would be done in hopping experiments.

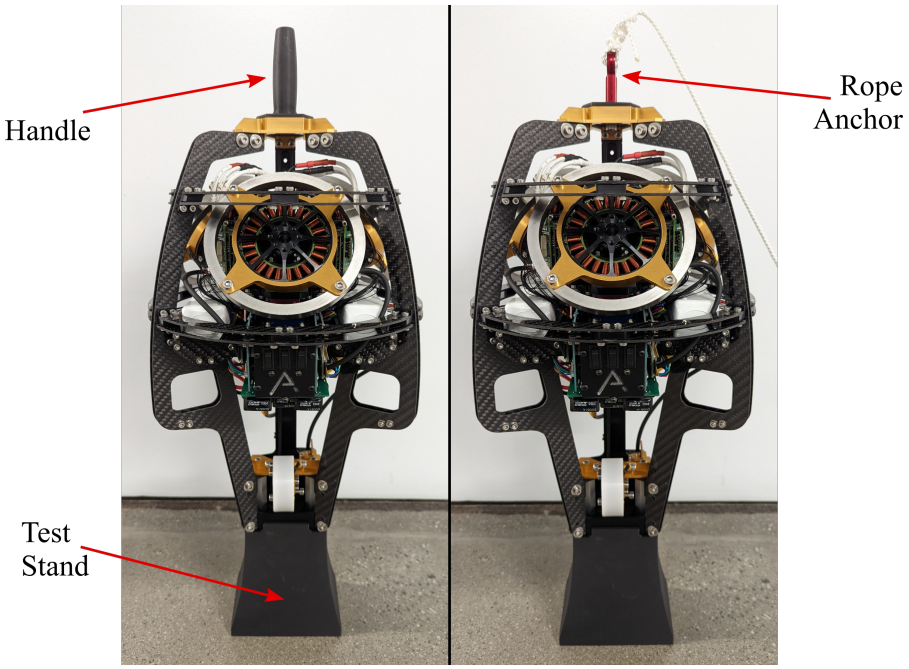


Figure 6.29: Photo of ARCHER on its test stand with labels showing the experimental handle and anchor point for the safety rope.

To prepare for later testing and experiments, a few extra parts were 3D-printed.

The first of these was a stand for the robot when not in use, since the foot of the robot comes to a point and the robot does not have a good surface to rest on otherwise. Fig. 6.29 shows this stand along with the other 3D-printed parts. The second of these was a handle bolted to the top of the robot, which would serve as a way of holding/supporting the robot during tests, as well as for the location of perturbations being given to the robot during balance experiments. The last of these parts was an anchor point for a rope that would be used during the initial hopping experiments. To make sure the robot would be safe from catastrophic falls during hopping experiments, a rope-pulley system was put in place. The rope would anchor to the top of ARCHER, then pass up and over a pulley in the ceiling, then back down to the user. This would allow the user to lift and lower the robot during testing without having to reach near the flywheels.

Balancing

Before moving straight to hopping, a large amount of time was devoted to making sure the balancing controller was dialed in. This was essentially done via the robot running its normal balancing controller, while the leg controller was inactive. For these tests, the robot was placed on the ground and held near vertical until the control went active, then letting go. The motor controllers here send commands of current rather than torque, so the gains had to be placed higher by a factor of roughly 12 as a starting point compared to the simulations.

For the first few months of testing, the controller was not able to successfully balance the robot. Rather than converging on something near vertical, it would reach a sort of orbit instead. Many attempts at changing gains took place, leading to a discovery that the robot would vibrate when the flywheels reach high velocity. An unforeseen aspect of the design, was that the resonance frequency was low enough to get triggered by the flywheel motors. This was apparent in the angular rate data from the IMU, once the flywheels got above roughly 250 rad/s. It even became apparent on a visible and auditory scale once the flywheel velocity reached around 500 rad/s. The gains were lowered to limit how quickly the flywheels would spin up, since raising the gains wasn't solving the problem anyway.

The level of filtering on the angular rate data was increased in an attempt to solve the oscillation/orbiting behavior, but this also didn't change the issue. After months of trying to debug why it would be tracking something other than vertical, it was discovered that the yaw and roll rate data were swapped at one section of the

controller which was the source of the orbiting issues. In testing it had appeared that the pitch angle was also orbiting, but it was just a side-effect of the yaw issue, since any yaw rate causes an outward force trying to push the robot away from vertical.

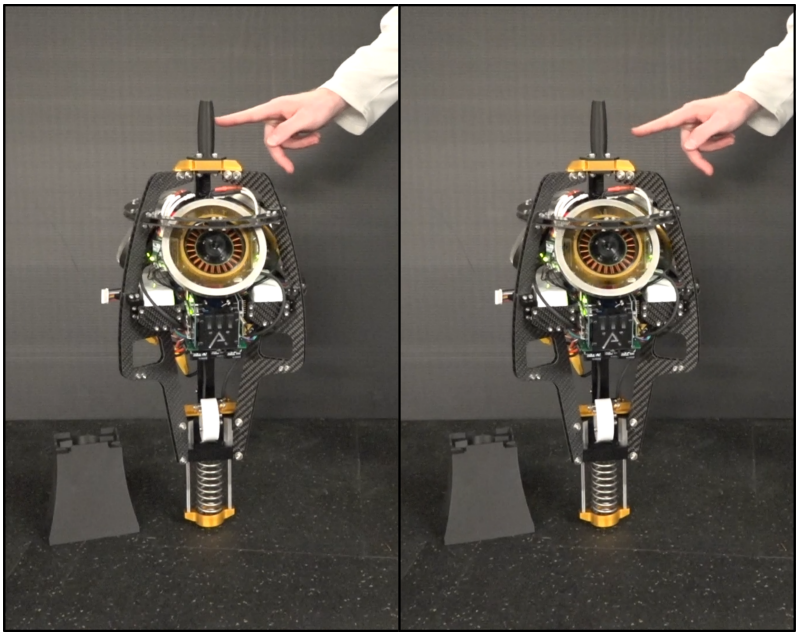


Figure 6.30: Video frames from one of the push recovery balance tests.

Once this bug was fixed, the balancing was immediately improved. After a little bit of tuning, the robot was able to balance well enough for the final hopping experiments. Some data was collected from these later successful balancing tests along with videos. These balancing experiments involved the robot running its attitude controller while the user pushed on the handle to provide disturbances in both roll and pitch directions. The robot was able to successfully reject these disturbances, and stay within a small window of vertical throughout. Fig. 6.30 shows some video frames from one of these trials both during and after a disturbance. Each push caused the robot's pitch/roll angle to deviate by about 1°, then recover back towards its zero position. The max motor current commanded by the controller during these pushes was recorded at just under 7 A, which is less than a third of the current limit set in the motor controllers showing that the robot is capable of much larger corrections if needed. Fig. 6.31 shows the pitch and roll data from this same balance test. Each of the moments of disturbances are marked by a red dot. Some of the pushes from the user were aimed in the direction of roll, and others in pitch, so the red markers are separated in the plots accordingly. Given the success from these balance tests, experiments moved on to full hopping. These experiments are

detailed in the following subsection.

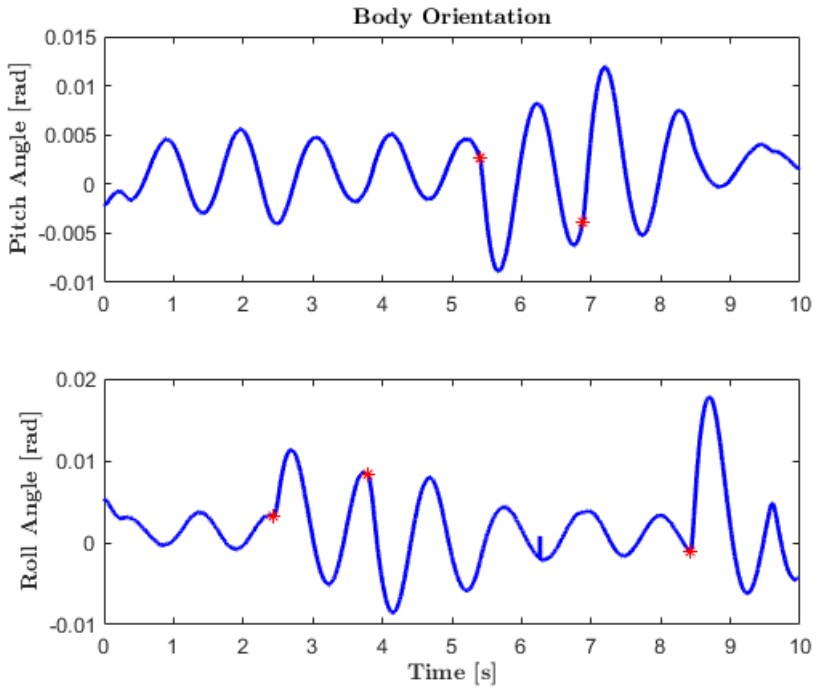


Figure 6.31: Plot of the body pitch and roll data from one of the balancing experiments, with disturbance points marked by red dots.

Hopping

For these first round of hopping experiments, there was no foot placement algorithm in place. The on-board IMU and OptiTrak cameras in the lab could help track the necessary global position and linear velocity of ARCHER, but this was left off until later experiments, time permitting. To start the robot would attempt to balance in place without foot placement's help. This required dropping the robot with near vertical motion to give the robot a starting condition within its limited region of stability. Initial tests were given a fixed number of hops to attempt before shutting off to make sure the hopping controllers could succeed, before moving up to longer duration tests.

To begin, the experiments were limited to 3 hops. It was found that using the safety rope system to drop the robot at the start of each test worked well to give the robot manageable initial conditions, and ARCHER was able to pass these tests. The experiments were then modified to go up until 10 hops, and at this point the deviations from perfect initial conditions became more apparent. The rope easily became twisted during repeated use and torsion built up along its length, so when the

robot was being dropped it was sometimes given an initial yaw rate that led to failure in fewer than 10 hops. With taking care to hold the robot until just before it was dropped, this was able to be mitigated, leading to successful runs of 10 continuous hops.

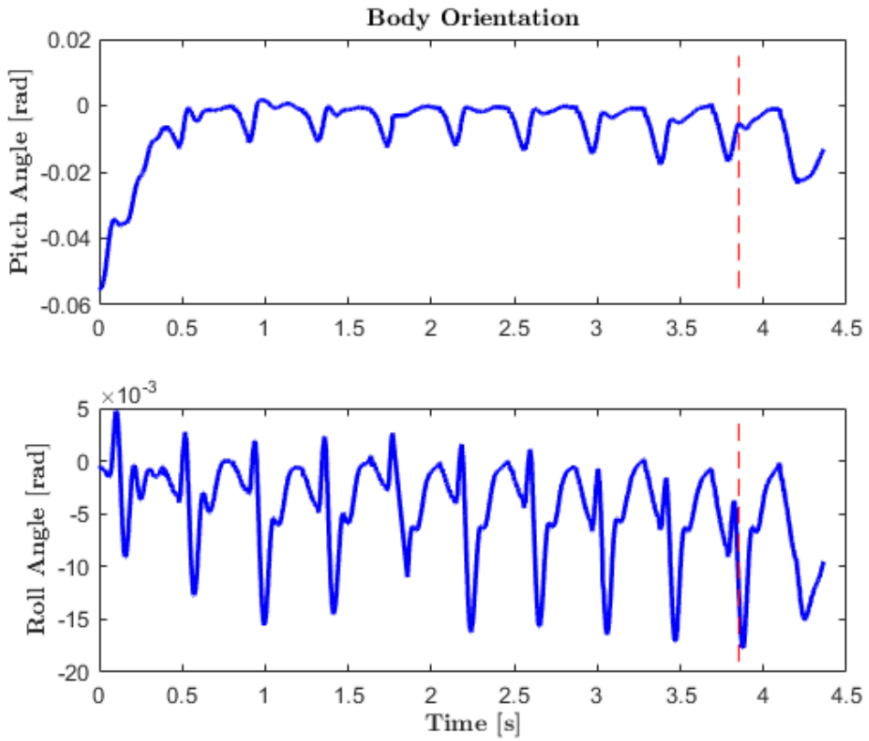


Figure 6.32: Body pitch and roll data from one of the 10-hop hopping experiments. The dashed red line represents the moment when the leg controller shuts off.

Fig. 6.32 shows the pitch and roll data from one of these experiments. The robot began with an initial error in pitch of about 0.055 rad, then successfully corrected itself over the first two hops. Over the course of the test, both pitch and roll angles drift a little. This aligns with what was observed in real-time, where the robot was beginning to hop sideways by the end. Meanwhile, the vertical control tracked very well throughout all the hops. Fig. 6.33 shows the spring deflection data from this same experiment. The black curve in the plot is the actual measurements taken from the foot encoder, while the red lines represent the desired position that the compression actuator is attempting to track. When the red line drops down to 0, that means the robot is on the ground and the compression actuator is driving the pulley back to its zero position while the spring is actually undergoing its natural compression cycle for ground contact.

The experiments were then shifted to a less controlled setup, where the robot would be dropped by hand and there would be no safety rope to catch it. The duration of

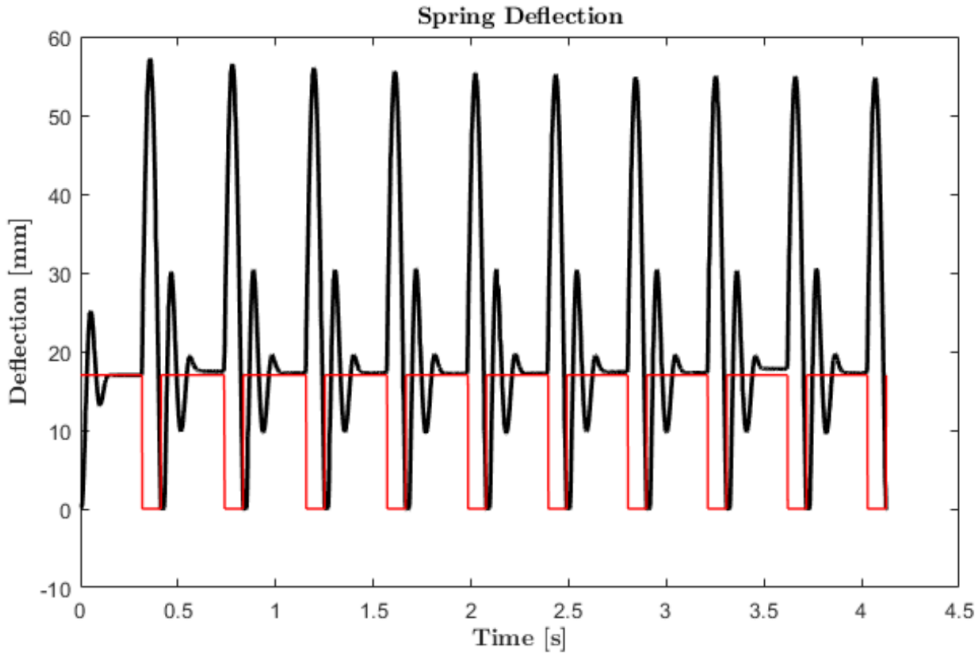


Figure 6.33: Spring deflection data from the same 10-hop hopping experiment. The red line represents the desired effort of the compression actuator.

the hops were also allowed to go beyond 10 hops, in the attempt to push the limit of how many hops could be achieved without a foot placement algorithm. Dropping the robot without giving initial lateral velocity to the body was a challenge, and took a lot of trial and error to learn a good method. At this point it was discovered that not all locations in the lab were created equal with respect to levelness. Some areas had sloped ground as high as a few degrees, which was causing the robot to drift to the side regardless of initial velocity. Once a reasonably level area was found, the final experiments commenced.

Fig. 6.34 shows the pitch and roll data from an experiment that achieved 15 consecutive hops. The run-away nature of the lateral velocity was very noticeable here. On roughly the 11th hop, the attitude control was no longer able to overcome the y-velocity, leading to the immediate up slope in the roll data from hop to hop. The red line in this plot marks the last successful hop before the user grabbed the robot. Despite the issues in the attitude on control and over all leaning of the robot that was occurring, the leg controller was still tracking very well as seen in Fig. 6.35. On this trial, it would seem that the robot was dropped from a height lower than its goal height. This can be seen by both the increase in spring deflection during the first few hops, and by comparing to the spring deflection of the previous 10-hop experiment shown in Fig. 6.33.

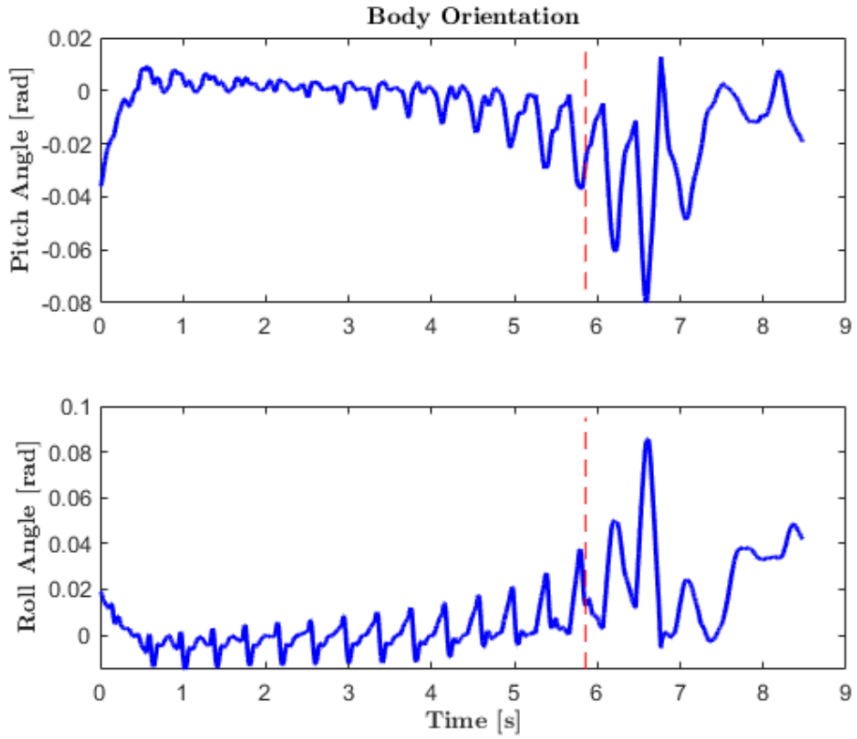


Figure 6.34: Body pitch and roll data from a hopping experiment reaching 15 hops. The dashed red line represents the end of the 15th hop.

From here, it was clear that some sort of foot placement method was going to be required to improve the endurance of the hopping. It was determined that getting the OptiTrak system set up to work with the robot smoothly would take a long time, so instead an approximate foot placement strategy was used. This method included looking at the angle of launch at each take-off moment and then attempting to track

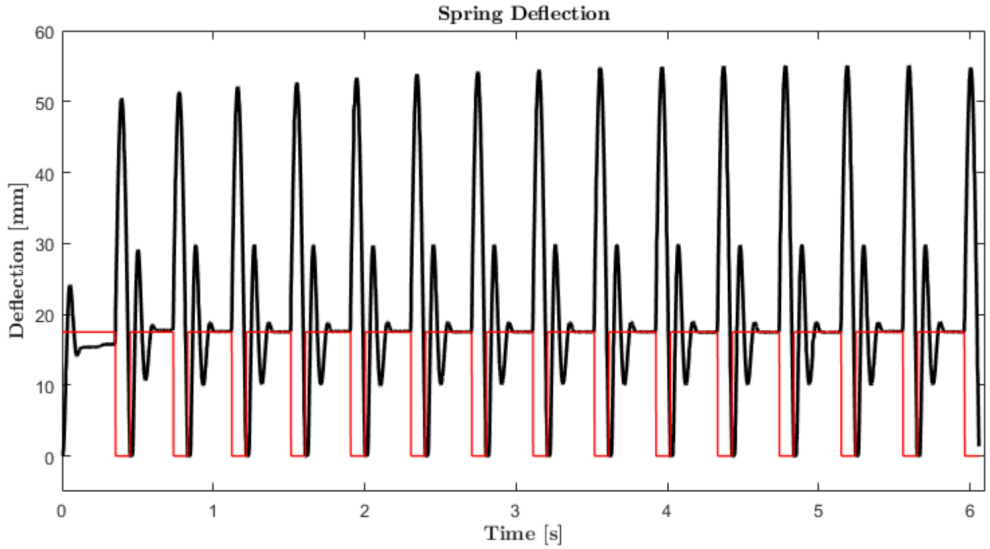


Figure 6.35: Spring deflection data from the same 15-hop hopping experiment, showing that the vertical controller was largely unaffected by the lateral velocity.

complementary roll and pitch angles throughout the next hop. There was a gain put on this term of the planner, so the actual desired angle had a magnitude equal to half the launch angle and with the opposite sign. For example, if the robot left the ground with a pitch angle of 2deg, than it would attempt to land with an angle of -1deg. This was a very simplistic replacement for the foot placement methods described previously which would require the global position and velocity information of the robot's base coordinate frame.

Using this planner, the robot was able to hop for longer periods of time with much less lateral movement or build up of orientation error. Fig. 6.36 show the results of one of the final experiments where the robot was able to hop for 20 consecutive hops without issue. In this experiment the robot was stopped prematurely by the user, but as can be seen in the data and collected video, the robot could have continued hopping longer. The orientation angle of roll and pitch were both able to stabilize within 1deg and remain there through all of the remaining hops. The moment the user bumps the handle of the robot is pretty clear from the sudden change of the periodic data after the 20th hop. A video of this experiment can be found here (E. Ambrose, 2022) for your reference.

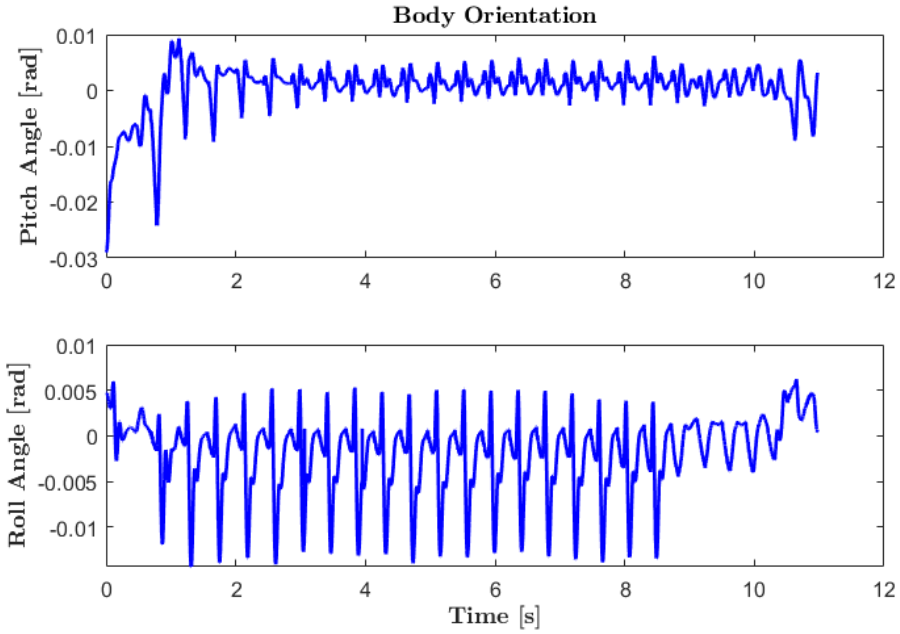


Figure 6.36: Body pitch and roll data from a final hopping experiment reaching 20 hops. The last portion of the data represents the time after user intervention.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The work presented in this thesis covers the design and testing of 8 different robotic platforms, taking place over the course of 9 years. To begin, the AMBER-3 humanoid was introduced showing the benefit of a modular and consistent robot for testing out new behaviors, controllers, and other methods. AMBER-3M is still in the AMBER today, operating with its original mechanical hardware, but via a new computation system allowing for more involved computations such as used in gait generation and neural networks. Next, the transfemoral prosthesis platform AMPRO-3 was presented, along with a detailed description of its mechanical hardware and novel additions from previous generations of the device. The final iteration of the prosthesis, AMPRO-4, was also presented along with its use by the EPIC at Georgia Tech in a clinical setting, tested by amputees.

The last few chapters dove into the process of assessing the stability and feasibility of different hopping mechanisms that fit into the scheme of safe-actuation, making sure that the robot will minimize its impulses on its environment. These mechanisms were then used in later generations of hopping robots to plan methods of improving both energetic efficiency and stability in systems with uncertainty. A technique of performing design-in-the-loop optimization was developed, with the goal of designing custom non-linear springs that could further improve efficiency. Finally, a 3D hopper design was introduced, along with simulations of balancing, foot placement methods, and control for a flywheel-based attitude system. Each of the hopping robots were experimentally validated over the course of a few years, culminating in the experiments on this 3D robot, ARCHER.

7.2 Future Work and Ideas

This section will delve into a few of the paths forward each of the robotic platforms that were presented in this work could take.

AMBER

Given that AMBER-3M has been used in tests ranging over roughly 6 years, there has of course been some wear and tear seen by the joints and actuators. The most

obvious issue on the robot currently is the backlash in the joints, which is caused by worn out bushings. These bushings could be replaced by bearings and metal shafts which would provide a tighter and longer lasting fit, and subsequently remove almost all of the backlash from the joints.

A second pair of feet were created for AMBER back at Georgia Tech, which were specifically designed to improve the robots ability to walk with a multi-contact behavior, which has been widely shown as a method of lowering the cost of transport for both humans and our robotic counterparts. Replacing the current feet on the robot with these would help with all experiments that wish to move beyond strict flat-footed walking.

AMPRO

Typically, prosthetic devices are tested on humans, both amputees and non-amputees. However, humans and other animals naturally adapt to our situation to improve comfort and save energy. This often times will lead to the user walking in ways that are actually not optimal, or just not working efficiently with the device. If the goal is to make a device that will conform to the user, and hopefully save them energy, then it would actually be best to test on someone that does not adapt. AMPRO could be connected up to AMBER as an alternative leg configuration, which would allow for such assessment, given that AMBER is not inherently able to adapt to anything.

ARCHER

Obviously, there are a lot of directions left which could be pursued with ARCHER. The experiments can be expanded to utilize the cameras and allow for true foot placement and other motion planning methods to be used. Moving onto experiments of higher hops and moving around the lab can show the full capability of this hardware platform.

Additionally, it would be great to finally get a pair of custom designed springs for this robot made from E-glass or S-glass. These would allow for a huge increase in the maximum hop height that the hardware would be capable of, as well as move the design towards the route of putting this mechanism into a ball-like enclosure.

BIBLIOGRAPHY

(N.d.). 2D Walking Experiments. https://youtu.be/sHA2Bwjij7Y.

A. Hereid, S. Kolathaya, and A. D. Ames (2016). "Online Hybrid Zero Dynamics Optimal Gait Generation Using Legendre Pseudospectral Optimization". In: *to appear in IEEE Conference on Decision and Control*.

Ackerman, Evan (2022). "A Robot for the Worst Job in the Warehouse: Boston Dynamics' Stretch can move 800 heavy boxes per hour". In: *Ieee Spectrum* 59.1, pp. 50–51.

Aguilar, J. and D. I. Goldman (2016). "Robophysical study of jumping dynamics on granular media". In: *Nature Physics* 12.3, p. 278.

Ambrose, E. (2019). *Video of the MMH hopping experiments*. https://youtu.be/lKLuz2ugMVw.

– (2020). *Video of the Double-Spring hopper experiments*. https://www.youtube.com/watch?v=Ua_n970IPDQ.

– (2021). *Video of the Curved-Spring hopper experiments*. https://youtu.be/S81eUVXF4d4.

– (2022). *Video of ARCHER hopping in the lab for its final experiments*. https://youtu.be/Kkp3ZY2f2D0.

Ambrose, E. and A. D. Ames (2020). "Improved Performance on Moving-Mass Hopping Robots with Parallel Elasticity". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2457–2463. DOI: 10.1109/ICRA40945.2020.9197070.

Ambrose, E., N. Csomay-Shanklin, et al. (2019). "Design and Comparative Analysis of 1D Hopping Robots". In: *Intelligent Robots and systems (IROS), IEEE International Conference on*. DOI: 10.1109/IROS40897.2019.8967692.

Ambrose, Eric, Wen-Loong Ma, and Aaron D Ames (2021). "Towards the Unification of System Design and Motion Synthesis for High-Performance Hopping Robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7073–7078. DOI: 10.1109/ICRA48506.2021.9561322.

Ambrose, Eric, Wen-Loong Ma, Christian Hubicki, et al. (2017). "Toward benchmarking locomotion economy across design configurations on the modular robot: AMBER-3M". In: *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, pp. 1270–1276. DOI: 10.1109/CCTA.2017.8062633.

Ames, A. D. (2013). "Human-inspired control of bipedal robots via control lyapunov functions and quadratic programs". In: *16th International conference on Hybrid systems: computation and control*. ACM, pp. 31–32.

Ames, A. D. (2014). “Human-inspired control of bipedal walking robots”. In: *Automatic Control, IEEE Transactions on* 59.5, pp. 1115–1130.

Ames, Aaron D, Kevin Galloway, et al. (2014). “Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics”. In: *IEEE Transactions on Automatic Control* 59.4, pp. 876–891.

Ames, Aaron D and Matthew Powell (2013). “Towards the unification of locomotion and manipulation through control Lyapunov functions and quadratic programs”. In: *Control of Cyber-Physical Systems*. Springer, Heidelberg, pp. 219–240.

Ames, Aaron D. et al. (2015). “First Steps Toward Formal Controller Synthesis for Bipedal Robots”. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC ’15. Seattle, Washington: ACM, pp. 209–218.

Antonelli, Gianluca (2013). “Interconnected dynamic systems: An overview on distributed control”. In: *IEEE Control Systems Magazine* 33.1, pp. 76–88.

Azimi, Vahid, Tony Shu, Huihua Zhao, Eric Ambrose, et al. (2017). “Robust control of a powered transfemoral prosthesis device with experimental verification”. In: *American Control Conference (ACC), 2017*. IEEE, pp. 517–522. DOI: 10.23919/ACC.2017.7963005.

Azimi, Vahid, Tony Shu, Huihua Zhao, Rachel Gehlhar, et al. (2019). “Model-Based Adaptive Control of Transfemoral Prostheses: Theory, Simulation, and Experiments”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Barber, James R (2010). *Intermediate mechanics of materials*. Vol. 175. Springer Science & Business Media.

Bastien, G.J. et al. (2005). “Effect of load and speed on the energetic cost of human walking”. In: *Journal of Applied Physiology* 94.1, pp. 76–83.

Bauby, C. E. and A. D. Kuo (2000). “Active control of lateral balance in human walking”. In: *Journal of Biomechanics* 33.11, pp. 1433–1440.

Betts, John T (1998). “Survey of numerical methods for trajectory optimization”. In: *Journal of guidance, control, and dynamics* 21.2, pp. 193–207.

Bhakta, Krishan, Jonathan Camargo, William Compton, et al. (2021). “Evaluation of Continuous Walking Speed Determination Algorithms and Embedded Sensors for a Powered Knee amp; Ankle Prosthesis”. In: *IEEE Robotics and Automation Letters* 6.3, pp. 4820–4826.

Bhakta, Krishan, Jonathan Camargo, Luke Donovan, et al. (2020). “Machine Learning Model Comparisons of User Independent amp; Dependent Intent Recognition Systems for Powered Prostheses”. In: *IEEE Robotics and Automation Letters* 5.4, pp. 5393–5400.

Bhakta, Krishan, Jonathan Camargo, Pratik Kunapuli, et al. (2019). "Impedance Control Strategies for Enhancing Sloped and Level Walking Capabilities for Individuals with Transfemoral Amputation Using a Powered Multi-Joint Prosthesis". In: *Military Medicine* 185.1, pp. 490–499. ISSN: 0026-4075.

Control and Experimental Validation of a Powered Knee and Ankle Prosthetic Device (2018). Vol. 1. Dynamic Systems and Control Conference.

Bhounsule, P.A. et al. (2014). "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge". In: *IJRR* 33.10, pp. 1305–1321.

Borders, J. (2009). *Planetary Geartrain Analysis*. www.bordersengineering.com/tech_ref/planetary/planetary_analysis.pdf.

BostonDynamics (2012). *Sand Flea Jumping Robot*. https://www.youtube.com/watch?v=6b4ZZQkcNEo&ab_channel=BostonDynamics.

Brown, H. B. and G. Zeglin (1998). "The Bow Leg Hopping Robot". In: *Robotics and Automation (ICRA), IEEE International Conference*.

Brown, Travis L and James P Schmiedeler (2016). "Reaction wheel actuation for improving planar biped walking efficiency". In: *IEEE Transactions on Robotics* 32.5, pp. 1290–1297.

Buchfuhrer, David and Christopher Umans (2011). "The complexity of Boolean formula minimization". In: *Journal of Computer and System Sciences* 77.1. Celebrating Karp's Kyoto Prize, pp. 142–153. ISSN: 0022-0000.

Chadwick, Michael et al. (Aug. 2020). "Vitruvio: An Open-source Leg Design Optimization Toolbox for Walking Robots". In: *IEEE Robotics and Automation Letters* PP, pp. 1–1.

Chang, Alexander H et al. (2019). "Every Hop is an Opportunity: Quickly Classifying and Adapting to Terrain During Targeted Hopping". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3188–3194.

Collins, S. et al. (2005). "Efficient Bipedal Robots Based on Passive-Dynamic Walkers". In: *Science* 307, pp. 1082–1085.

Csomay-Shanklin, Noel, Ryan K Cosner, et al. (2021). "Episodic Learning for Safe Bipedal Locomotion with Control Barrier Functions and Projection-to-State Safety". In: *Learning for Dynamics and Control*. PMLR, pp. 1041–1053.

Csomay-Shanklin, Noel, Maegan Tucker, et al. (2021). *Learning Controller Gains on Bipedal Walking Robots via User Preferences*.

DisneyResearchHub (2016). *Untethered One-Legged Hopping in 3D Using Linear Elastic Actuator in Parallel (LEAP)*. https://www.youtube.com/watch?v=M0ZXmGRCuts.

ELMO-MotionControl (2021). *Gold Solo Twitter*. https://www.elmomc.com/product/gold-solo-twitter/.

Eslamy, M., M. Grimmer, and A. Seyfarth (2012). "Effects of unidirectional parallel springs on required peak power and energy in powered prosthetic ankles: Comparison between different active actuation concepts". In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, pp. 2406–2412.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Kluwer international series in engineering and computer science: Robotics. Springer.

Ficanha, E., M Rastgaar, and K. Kaufman (2014). "A two-axis cable-driven ankle-foot mechanism". In: *Robotics and Biomimetics* 1.17.

Fiorini, P. and J. Burdick (2003). "The Development of Hopping Capabilities for Small Robots". In: *Autonomous Robots* 14.2, pp. 239–254.

Galliker, Manuel Y et al. (2022). "Bipedal Locomotion with Nonlinear Model Predictive Control: Online Gait Generation using Whole-Body Dynamics". In: *arXiv preprint arXiv:2203.07429*.

Gao, Chang et al. (2020). "Recurrent Neural Network Control of a Hybrid Dynamical Transfemoral Prosthesis with EdgeDRNN Accelerator". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5460–5466.

Gehlhar, Rachel and Aaron D Ames (2021). "Model-dependent prosthesis control with interaction force estimation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3226–3232.

Gehlhar, Rachel, Je-han Yang, and Aaron D Ames (2021). "Model-Dependent Prosthesis Control with Real-Time Force Sensing". In: *arXiv preprint arXiv: 2105.11561*.

Georgiev, Nikola and Joel Burdick (2017). "Design and analysis of planar rotary springs". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4777–4784.

– (2018). "Optimization-based Design and Analysis of Planar Rotary Springs". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 927–934.

Gottlieb, D. and S. Orszag (1977). *Numerical Analysis of Spectral Methods*. Society for Industrial and Applied Mathematics.

Grizzle, J. W., C. Chevallereau, A. D. Ames, et al. (2010). "3D bipedal robotic walking: models, feedback control, and open problems". In: *IFAC Symposium on Nonlinear Control Systems*. Bologna.

Grizzle, J. W., C. Chevallereau, R. W. Sinnet, et al. (2014). "Models, feedback control, and open problems of 3D bipedal robotic walking". In: *Automatica* 50.8, pp. 1955–1988. ISSN: 0005-1098.

Haftka, Raphael T. and Zafer Gürdal (1992). "Elements of Structural Optimization". In: Springer Netherlands. ISBN: 978-94-011-2550-5.

Haldane, D. W., J. K. Yim, and R. S. Fearing (2017). "Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3345–3351.

Hereid, A., E. A. Cousineau, et al. (2016). "3D Dynamic Walking with Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics". In: *To appear in the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Hereid, A., C. M. Hubicki, et al. (2018). "Dynamic Humanoid Locomotion: A Scalable Formulation for HZD Gait Optimization". In: *IEEE Transactions on Robotics*, pp. 1–18. ISSN: 1552-3098.

Hereid, Ayonga and Aaron D Ames (2017). "FROST: Fast Robot Optimization and Simulation Toolkit". In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, pp. 4552–4559.

Hoffmann, Charles et al. (2016). "Feasibility Test of the MedaCube". In.

Hoyt, D.F. and C.R. Taylor (1981). "Gait and the energetics of locomotion in horses". In: *Nature* 292, pp. 239–240.

Hubicki, C. M. et al. (2016). "Tractable terrain-aware motion planning on granular media: An impulsive jumping study". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3887–3892.

Hwangbo, J. et al. (2018). "Cable-driven actuation for highly dynamic robotic systems". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 8543–8550.

Ihrke, C. A. et al. (2010). "Planar Torsion Spring". Patent US 20100145510.

Jameson, Antony, Luigi Martinelli, and Niles A Pierce (1998). "Optimum aerodynamic design using the Navier–Stokes equations". In: *Theoretical and computational fluid dynamics* 10.1, pp. 213–237.

Kelly, Matthew P (2017). "Transcription Methods for Trajectory Optimization: a beginners tutorial". In: *arXiv preprint arXiv:1707.00284*.

Kooi, BW (1994). "The design of the bow". In: *Proc. Kon. Ned. Akad. v. Wetensch* 97.3, pp. 1–27.

Kuo, A.D. (2007). "Choosing your steps carefully". In: *IEEE Robotics and Automation Magazine* 14.2, pp. 18–29.

Lens, T. and O. Von Stryk (2012). "Investigation of safety in human-robot-interaction for a series elastic, tendon-driven robot arm". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4309–4314.

Li, Kejun et al. (2021). "Natural Multicontact Walking for Robotic Assistive Devices via Musculoskeletal Models and Hybrid Zero Dynamics". In: *arXiv preprint arXiv:2109.05113*.

Lohn, Jason D., Gregory S. Hornby, and Derek S. Linden (2005). "An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission". In: *Genetic Programming Theory and Practice II*. Boston, MA: Springer US, pp. 301–315. ISBN: 978-0-387-23254-6.

Lyu, Zhoujie, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins (2015). "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark". In: *AIAA Journal* 53.4, pp. 968–985.

M'Closkey, R. T. and J. W. Burdick (1993). "Periodic motions of a hopping robot with vertical and forward motion". In: *The International journal of robotics research* 12.3, pp. 197–218.

Ma, W et al. (2014). "Human-inspired walking via unified PD and impedance control". In: *IEEE International Conference on Robotics and Automation*.

Ma, W., A. Hereid, et al. (2016). "Efficient HZD Gait Generation for Three-Dimensional Underactuated Humanoid Running". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Ma, W., S. Kolathaya, et al. (2017). "Bipedal Robotic Running with DURUS-2D: Bridging the Gap between Theory and Experiment". In: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, pp. 265–274. DOI: 10.1145/3049797.3049823.

Ma, W.-L., K. A. Hamed, and A. D. Ames (2019). "First Steps Towards Full Model Based Motion Planning and Control of Quadrupeds: A Hybrid Zero Dynamics Approach". In: *Intelligent Robots and systems (IROS), IEEE International Conference on*.

Ma, Wen-Loong, Noel Csomay-Shanklin, and Aaron D. Ames (2020). "Coupled Control Systems: Periodic Orbit Generation with Application to Quadrupedal Locomotion". In: *IEEE Control Systems Letters*. arXiv: 2003.08507 [cs.RO].

Ma, Wen-Loong, Yizhar Or, and Aaron D Ames (2019). "Dynamic walking on slippery surfaces: Demonstrating stable bipedal gaits with planned ground slippage". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3705–3711.

Ma, Wen-Loong. et al. (2017). "Bipedal Robotic Running with DURUS-2D: Bridging the Gap between Theory and Experiment". In: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. HSCC '17. Pittsburgh, Pennsylvania: ACM. ISBN: 978-1-4503-3433-4. DOI: 10.1145/3049797.3049823.

Manchester, I.R. et al. (2011). "Stable dynamic walking over uneven terrain". In: *IJRR* 30.3, pp. 265–279.

Martin, A.E., D.C. Post, and J.P. Schmiedeler (2014). "The effects of foot geometric properties on the gait of planar bipeds walking under HZD-based control". In: *The International Journal of Robotics Research* 33.12, pp. 1530–1543.

Mettin, U. et al. (2010). “Parallel Elastic Actuators as a Control Tool for Preplanned Trajectories of Underactuated Mechanical Systems”. In: *The International Journal of Robotics Research* 29.9, pp. 1186–1198.

Murray, R. M., Z. Li, and S. S. Sastry (1994). *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC Press.

Paluska, D. and H. Herr (2006). “The effect of series elasticity on actuator power and work output: Implications for robotic and prosthetic joint design”. In: *Robotics and Autonomous Systems* 54.8, pp. 667–673.

Park, J. et al. (2014). “Raptor: Fast bipedal running and active tail stabilization”. In: *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 215–215.

Patterson, M. A. and A. V. Rao (2014). “GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming”. In: *ACM Trans. Math. Softw.* 41.1, pp. 1–37.

PJRC (2021). *Teensy 4.1 Development Board*. https://www.pjrc.com/store/teensy41.htmlm.

Pontzer, H. (2007). “Effective limb length and the scaling of locomotor cost in terrestrial animals”. In: *Journal of Experimental Biology* 210, pp. 1752–1761.

Powell, Matthew J et al. (2016). “Mechanics-based design of underactuated robotic walking gaits: Initial experimental realization”. In: *Humanoid Robots, 2016 IEEE-RAS 16th International Conference on*. IEEE, pp. 981–986.

Pulse, Nippon (2019). *Linear Servomotors*. http://www.nipponpulse.com/products/browse/linear-shaft-servomotors.

Raibert, M. (1986). *Legged robots that balance*. Cambridge, MA: MIT Press.

Raibert, M. H. (1984). “Hopping in Legged Systems - Modeling and Simulation for the Two-dimensional One-Legged Case”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 14.3, pp. 451–463.

Raibert, M. H., H. B. Brown, and M. Chepponis (1984). “Experiments in Balance with a 3D One-Legged Hopping Machine”. In: *IJRR* 3.2, pp. 75–92.

Reher, J., E. A. Cousineau, et al. (2016). “Realizing Dynamic and Efficient Bipedal Locomotion on the Humanoid Robot DURUS”. In: *International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1794–1801.

– (2016). “Realizing dynamic and efficient bipedal locomotion on the humanoid robot DURUS”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1794–1801.

Reher, Jake, Wen-Loong Ma, and Aaron D. Ames (2019). “Dynamic Walking with Compliance on a Cassie Bipedal Robot”. In: *2019 18th European Control Conference (ECC)*, pp. 2589–2595.

Renishaw (2021). *LM13 Magnetic Incremental Encoder*. https://www.rls.si/eng/lm13-magnetic-linear-and-rotary-encoder-system.

Rodriguez, Ivan Dario Jimenez et al. (2022). "Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies". In: *arXiv preprint arXiv:2204.08120*.

Saar, K. A., F. Giardina, and F. Iida (2018). "Model-Free Design Optimization of a Hopping Robot and Its Comparison With a Human Designer". In: *IEEE Robotics and Automation Letters* 3.2, pp. 1245–1251.

Salton, J. R. et al. (2010). "Urban Hopper". In: *SPIE* 7692, pp. 7692 - 7692 –9.

Schmit, Nicolas and Masafumi Okada (2013). "Optimal design of nonlinear springs in robot mechanism: simultaneous design of trajectory and spring force profiles". In: *Advanced Robotics* 27.1, pp. 33–46.

Seok, S. et al. (2013). "Design Principles for Highly Efficient Quadrupeds and Implementation on the MIT Cheetah Robot". In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3307–3312.

Sharbafi, M. A. et al. (2019). "Parallel Compliance Design for Increasing Robustness and Efficiency in Legged Locomotion—Proof of Concept". In: *IEEE/ASME Transactions on Mechatronics* 24.4, pp. 1541–1552.

Sinnet, R. W. et al. (2011). "A human-inspired hybrid control approach to bipedal robotic walking". In: *IFAC Proceedings Volumes* 44.1, pp. 6904–6911.

Smoot, L. S. et al. (2018). *Robot bouncing ball*. US Patent #: 10,092,850.

Sobajima, M. et al. (2013). "Bipedal Walking Control of Humanoid Robots by Arm-Swing". In: *International conference on nstrumentation, Control, Information Technology and System Integration (SICE)*. Nagoya.

Sreenath, K., HW Park, and J.W. Grizzle (2014). "Embedding Active Force Control within the Compliant Hybrid Zero Dynamics to Achieve Stable, Fast Running on MABEL". In: *IJRR* 33, pp. 988–1005.

Sreenath, K., HW Park, I. Poulakakis, et al. (2011). "A Compliant Hybrid Zero Dynamics Controller for Stable, Efficient and Fast Bipedal Walking on MABEL". In: *IJRR* 30.9, pp. 1170–1193.

Srinivasan, M. and A. Ruina (2006). "Computer optimization of a minimal biped model discovers walking and running". In: *Nature* 439, pp. 72–75.

T-Motor (2021a). *Anti-Gravity MN7005 Motor*. https://store.tmotor.com/goods.php?id=461.

– (2021b). *U10 Motor*. https://store.tmotor.com/goods.php?id=360.

Tabuada, Paulo et al. (2017). "Data-driven control for feedback linearizable single-input systems". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 6265–6270.

Tran, Thi-Trang and CheolKeun Ha (2018). “Non-contact gap and flush measurement using monocular structured multi-line light vision for vehicle assembly”. In: *International Journal of Control, Automation and Systems* 16.5, pp. 2432–2445.

Tucker, Maegan et al. (2021). “Preference-based learning for user-guided HZD gait generation on bipedal walking robots”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2804–2810.

VectorNav (2021). *VN-100 Inertial Measurement Unit and Attitude Heading Reference System*. https://www.vectornav.com/products/detail/vn-100.

Westervelt, E. R., J. W. Grizzle, and D. E. Koditschek (2003). “Hybrid zero dynamics of planar biped walkers”. In: *IEEE Transactions on Automatic Control* 48.1, pp. 42–56.

Wickler, S.J. et al. (2000). “Preferred speed and cost of transport: the effect of incline”. In: *Journal of Experimental Biology* 203, pp. 2195–2200.

Xi, W., Y. Yesilevskiy, and C. David Remy (2015). “Selecting gaits for economical locomotion of legged robots”. In: *IJRR* 35.9, pp. 1140–1154.

Xiong, Xiaobin and Aaron Ames (2020). “Sequential Motion Planning for Bipedal Somersault via Flywheel SLIP and Momentum Transmission with Task Space Control”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. URL: http://ames.caltech.edu/xiong2020sequential.pdf.

Yadukumar, Shishir Nadubettu, Murali Pasupuleti, and Aaron D Ames (2012). “Human-inspired underactuated bipedal robotic walking with amber on flat-ground, up-slope and uneven terrain”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 2478–2483.

Yang, Cheng et al. (2019). “A vascular interventional surgical robot based on surgeon’s operating skills”. In: *Medical & biological engineering & computing* 57.9, pp. 1999–2010.

Yim, J. K., E. K. Wang, and R. S. Fearing (2019). “Drift-free Roll and Pitch Estimation for High-acceleration Hopping”. In: *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8986–8992.

Yoon, S.-S. et al. (2005). “Safe arm design with MR-based passive compliant joints and visco—elastic covering for service robot applications”. In: *Journal of Mechanical Science and Technology* 19.10, pp. 1835–1845. ISSN: 1738-494X.

Young, Aaron and Bhakta Krishan (2019). *User-Independent Intent Recognition on a Powered Transfermoral Prosthesis*. Tech. rep. Georgia Institute of Technology Atlanta United States.

Yu, You et al. (2020). “Biofuel-powered soft electronic skin with multiplexed and wireless sensing for human-machine interfaces”. In: *Science Robotics* 5.41.

Zeglin, G. (1999). “The Bow Leg Hopping Robot”. In: *Diss. Carnegie Mellon University*.

Zhao, H., J. Horn, et al. (2016a). “First steps toward translating robotic walking to prostheses: a nonlinear optimization based control approach”. In: *Autonomous Robots*, pp. 1–18.

– (2016b). “Multicontact Locomotion on Transfemoral Prostheses via Hybrid System Models and Optimization-Based Control”. In: *IEEE Transactions on Automation Science and Engineering* 13.2, pp. 502–513.

Zhao, H., W.-L Ma, et al. (2014). “Human-inspired multi-contact locomotion with AMBER2”. In: *Cyber-Physical Systems (ICCPS), International Conference on*, pp. 199–210.

Zhao, H., J. Reher, et al. (2015). “Realization of Nonlinear Real-Time Optimization Based Controllers on Self-Contained Transfemoral Prosthesis”. In: *6th International Conference on Cyber Physics System*. Seattle, WA, pp. 130–138.

Zhao, Hui-Hua et al. (2014). “Human-inspired multi-contact locomotion with AMBER2”. In: *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*. IEEE, pp. 199–210.

Zhao, Huihua, Eric Ambrose, and Aaron D. Ames (2017). “Preliminary results on energy efficient 3D prosthetic walking with a powered compliant transfemoral prosthesis”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1140–1147. DOI: 10.1109/ICRA.2017.7989136.

Zhao, Huihua, Ayonga Hereid, et al. (2016). “3D multi-contact gait design for prostheses: Hybrid system models, virtual constraints and two-step direct collocation”. In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, pp. 3668–3674. DOI: 10.1109/CDC.2016.7798821.