University of California
Santa Barbara

Reachability and Real-Time Actuation Strategies for the Active SLIP Model

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Mechanical Engineering

by

Giulia Piovan

June 2015

The Dissertation of Giulia Piovan is approved.

Professor João Hespanha

Professor Jeff Moehlis

Professor Brad Paden

Professor Katie Byl, Committee Chair

May 2015

Reachability and Real-Time Actuation Strategies for the Active SLIP Model

Acknowledgements

Several people helped and supported me in this journey toward a Ph.D.

First of all, I want to thank my advisor, Dr. Katie Byl, for her help, support and contagious enthusiasm. As a young faculty, working towards a big goal herself (becoming a tenured professor) while managing a family and a young child, she has been a great source of inspiration.

Thanks to the members of my committee, Dr. Joao Hespanha, Dr. Jeff Moehlis, and Dr. Brad Paden, for having influenced my work both with their comments and suggestions. I would also like to acknowledge several faculty members both of the department of Mechanical Engineering and Electrical Engineering for the high quality classes they taught and that I had the luck to attend.

Thanks to the fellow students and members, past and present, of the Robotics Laboratory at UCSB. Min-Yi, Marco, Hosein, Paul, Pat, Brian, Oguz, Sebastian, Tom, Chelsea, Chris, Marty, Jason, and all the other local and international students that spent some time with us.

Things would have been much harder these past years without the support of all my friends, both here and far away. It is a true blessing to know there are wonderful people ready to listen, encourage and make you laugh.

Of course, thanks to my family. There are no words to describe how much love I have for them.

Last but not least, thanks to John for being an amazing husband, for supporting and encouraging me throughout this whole journey.

This thesis includes the following articles:

- ©2012 IEEE. Reprinted, with permission, from Giulia Piovan and Katie Byl, *Enforced symmetry of the stance phase for the Spring-Loaded Inverted Pendulum,* In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), May 2012

- ©2013 IEEE. Reprinted, with permission, from Giulia Piovan and Katie Byl, *Two-element control for the active SLIP model,* In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), May 2013

- G. Piovan and K. Byl. *Partial Feedback Linearization and Control of the SLIP Model Via Two-Element Leg Actuation Strategy.* Submitted to the IEEE Transactions on Robotics, 2015

- G. Piovan and K. Byl. *Reachability-based Control for the Active SLIP Model.* The International Journal of Robotics Research, DOI: 10.1177/0278364914552112, 2015

Part of this work has been funded by the Defense Advanced Research Projects Agency (DARPA) as part of the Maximum Mobility and Manipulation (M3) program.

Curriculum Vitæ
Giulia Piovan

Education

2015	Ph.D. in Mechanical Engineering (Expected), University of California, Santa Barbara.
2010	M.Sc. in Mechanical Engineering, University of California, Santa Barbara.
2008	Laurea Specialistica in Ingegneria dell'Automazione, Università degli Studi di Padova, Italy
2005	Laurea Triennale in Ingegneria dell'Informazione, Università degli Studi di Padova, Italy

Publications

(i) G. Piovan and K. Byl. *Partial Feedback Linearization and Control of the SLIP Model Via Two-Element Leg Actuation Strategy.* Submitted to the IEEE Transactions on Robotics, 2015

(ii) G. Piovan and K. Byl. *Reachability-based Control for the Active SLIP Model.* The International Journal of Robotics Research, DOI: 10.1177/0278364914552112, 2015

(iii) G. Piovan and K. Byl. *Two-Element Control for the Active SLIP Model.* In Proc. IEEE Int. Conf. Robotics and Automation (ICRA), 5636-5642, 2013

(iv) G. Piovan, I. Shames, B. Fidan, F. Bullo, and B. D. O. Anderson. *On Frame and Orientation Localization for Relative Sensing Networks.* Automatica, 49(1):206-213, 2013

(v) G. Piovan and K. Byl. *Enforced symmetry of the stance phase for the spring-loaded inverted pendulum.* In Proc. IEEE Int. Conf. Robotics and Automation (ICRA). 1908-1914, 2012.

(vi) G. Piovan and F. Bullo. *On Coordinate-Free Rotation Decomposition: Euler Angles about Arbitrary Axes.* IEEE Transactions on Robotics, 28(3):728-733, 2012.

(vii) Katie Byl, Marten Byl, Martin Rutschmann, Brian Satzinger, Louis van Blarigan, Giulia Piovan, and Jason Cortell. *Series-elastic actuation prototype for rough terrain hopping.* In Proc. IEEE International Conference on Technologies for Practical Robot Applications (TePRA). 103-110, 2012.

(viii) G. Piovan, I. Shames, B. Fidan, F. Bullo B.D.O. Anderson. *On Frame and Orientation Localization for Relative Sensing Networks.* in *IEEE Conference on Decision and Control*, 2008, pp. 2326-2331

Abstract

Reachability and Real-Time Actuation Strategies for the Active SLIP Model

by

Giulia Piovan

Running and hopping follow similar patterns for different animals, independent of the number of legs employed. An aerial phase alternates with a ground contact phase, during which the center of mass moves as if a spring were compressed and then extended to recover stored elastic energy. Hence, consisting of a point mass mounted on a massless spring leg, the Spring Loaded Inverted Pendulum (SLIP) is a prevalent model for analyzing running and hopping. In this work we consider an actuated version of the SLIP model, with a series elastic actuator added to the leg, serving the purposes of adding/removing energy to/from the system and of modifying dynamics during stance, toward achieving non-steady locomotion on varying terrain. While the SLIP model has been a topic of research in legged locomotion for several decades, studies on the effect of actuation on the system's behavior are still not complete. The goal of this thesis is to explore how a series elastic actuator applied to the SLIP model's leg can change the system's dynamics. This, in turn, enables a variety of long-term planning strategies for using limited footholds and design non-steady gaits while simultaneously recovering from unexpected perturbations, both sensorial and due to a limited knowledge of the terrain profile. We principally investigate how, through actuation, we can solve partially or completely the system's equations of motion, to enforce a desired trajectory and reach a desired state. We also determine the reachable state space of the model using several different actuation strategies, investigating the variation of the reachable set with respect to particular actuator motions and providing relationships between local actuator displacements throughout stance and location of the reached apex state. We then propose a control strategy based on graphical and numerical studies of the reachability space to drive the system to a desired state, with the ability to reduce the effects of sensing errors and disturbances happening at landing as well as during ground contact.

Contents

Chapter 1

Introduction and Motivation

Thanks to consolidated knowledge in modelling, manufacturing and control, wheeled vehicles are still the most popular choice in terms of locomotion. Their efficiency on continuous surfaces, however, is paired with a lack of versatility and adaptability that poses a limitation in many real-world applications, particularly those instances that see the presence of obstacles in the terrains. These same terrains can easily be reached by most animals and humans. Hence, legged locomotion has been a topic of study in robotics and has seen remarkable advances in the last forty years. The advantages of legged systems over wheeled or track-based systems are multifold. Firstly, the continuous path of support needed by wheeled vehicles is substituted by a set of isolated footholds in the case of legged systems and, as a consequence, not all the terrain needs to be specified and to be accessible. This implies an ability to adapt to surfaces that are uneven or with big variations. Additionally, the movement is determined by the robot's dynamics more than by the terrain surface due to an active suspension of the body. Finally, by studying legged locomotion, we hope to gain insights into human mobility as well, which can be applied to different areas of medicine, for example prosthetics.

Based on our daily experience, walking on legs seems to be a trivial task, and yet the capabilities of legged animals are far superior to those of any existing robot in that animals can negotiate a variety of different terrains in an extremely efficient way. However, since the mechanisms of locomotion are yet not fully understood, and are coupled with mechanical and sensorial limitations ([1]), legged locomotion is still very much an open problem. First of all, it is governed by a much more complicated dynamics than wheeled systems, which presents a challenge from the controller's point of view. Furthermore, legged systems have higher peak energy and torque requirements than wheeled vehicles, and they experience energy loss at each impact and due to friction at the joints. On hardware platforms, autonomous legged robots need to carry all their weight (motors, actuators, batteries) on their legs, which have to be robust.

1.1 Literature Review

The two basic gaits available to legged systems are walking and running. Loosely speaking, the distinction between the two is dependent on the presence or absence of an aerial phase: while running, there is at most one leg in contact with the ground at any given time, followed by a state with no ground contact, Fig. 1.1.

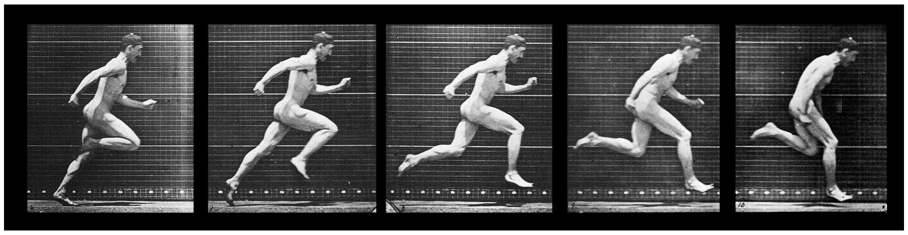


Figure 1.1: Man running at full speed. From *Muybridge's Human Figure in Motion* (1872-1885), by Eadweard Muybridge. Courtesy of Dover Publications, Inc.

Legs acting in unison in some respects can be considered as equivalent to a virtual leg located at the center of the pair. For this reason, numerous researchers interested in running have focused their attention to single-legged systems.

The conventional template used for analyzing running and hopping is the so-called Spring Loaded Inverted Pendulum (SLIP) model. Consisting of a point mass mounted on a massless spring leg, this 2 degrees of freedom (DOF) system is energetically conservative. The 2-DOF SLIP model can be extended to its 3-DOF counterpart by substituting the point mass with a body that is free to rotate via an applied torque at the hip. The spring-mass system mimics animals that hop at a particular preferred frequency below which the motion requires more energy. Furthermore, it can be seen that hopping follows similar patterns for different animals, e.g., the ratio of contact time versus aerial time, which are independent on the number of legs. During contact with the ground, the center of mass moves as if a spring were compressed and then extended recovering its stored elastic energy. Hence, in combination with biological data fitting, this simple model can predict the center of mass dynamics of running insects and animals, as shown in [2], in [3], as well in [4], [5], and [6]. These works stress the capability of the SLIP model of describing the interdependency between physical and morphological parameters characterizing running and hopping, such as frequency, stiffness and stride time. In general, animals' anatomy is far more complex than what the SLIP model captures, due to the presence of joints, ankles, knees, hips, leg mass and inertia, and muscles that act as nonlinear springs. However, as Full and Koditschek explain in [1], the SLIP model is a valid template which removes the non essential complexity of the animal's anatomy but still preserves the behavior that serves as a guide for the development of control strategies for locomotion. These strategies can then be applied to an anchor, a more complex and realistic model to mimic the animal's morphology and physiology.

Studies on guinea fowls ([7]), and on insects ([8], and [9]), show that animals and insects use their legs' natural actuation to stabilize their body. This suggests that the passive SLIP is not fully able to model running and hopping gaits. Since it is clearly desirable to enable legged locomotion in situations with variability in both terrain height and forward velocity, requiring accompanying modification of net energy, this justifies the introduction of an actuated version of the SLIP. There are several methods to add actuation to the SLIP model. The most common is the use of a telescopic leg with elastic actuator acting during the stance phase to modify the net energy of the system ([10], [11], [12],[13], [14], [15], [16]). Authors of [17] added a clock-based torque at the hip of the system, while [18] propose instead to modify the leg length during the flight phase. In [19], authors consider the effect of modifying both leg length and spring stiffness during stance, limiting their analysis to the vertical hopping case. These added actuations have been used to reach different energy levels, meaning changing the velocity or height of the body, toward controlling step size.

1.1.1 Control Strategies

The passive 2-DOF SLIP model has been employed as a starting point in studies focusing on control strategies to stabilize the system in the presence of disturbances due to uneven terrain or unexpected forces. Because of the completely passive nature of the system, its main tool in terms of balance is the placement of the swing leg. In general, to each forward speed corresponds a ground contact leg angle for steady-state motion. Variations of this angle have the effect of decelerating or accelerating the system. A proper adjustment of spring stiffness and fixed leg angle serves as a control strategy for stabilization [20], and in [21] it is shown how choosing a fixed leg angle at touch-down results in asymptotically stable periodic gaits.

By observing that running animals retract their legs right before contact to the ground, Seyfarth et al. ([22]) propose a swing-leg retraction strategy for steady-state running. Instead of being kept fixed during the flight phase, the leg angle moves as a linear function of time. Due to the constant rotational speed at flight, the leg angle at ground contact varies as a function of the apex state (i.e., the highest point reached during flight), and the relative speed of the foot with respect to the ground is reduced. The system is stabilized at steady-state faster than when using a fixed angle of attack, and it is stable with respect to variations on the spring stiffness, the forward speed and the leg angle. The optimal retraction rates for maximum disturbance rejection, for minimal energy loss at impact, and for minimal slipping, are studied by Karssen et al. in [23]. Return maps for both fixed contact angle and swing-leg retraction are computed in [24]. This stability analysis determines necessary parametric condition for asymptotic stability, and it is used to quantify sensory costs.

Another parameter that could be varied is the leg stiffness. Stance time, and consequently leg compression, is a function of leg stiffness, and therefore a careful choice can maximize the passive stability of the system. For example, in [25], Ernst and colleagues propose a feedforward controller that continuously adapts spring stiffness and leg angle during flight (following a precomputed curve stored in a look-up table) regardless of when the foot touches the ground, thus allowing the system to hop on uneven terrain. Riese and Seyfarth [19] investigate stability of vertical hopping while varying spring stiffness and leg length.

For the active SLIP model, the most famous control strategy is proposed by Raibert [10]: a constant thrust is applied during ground contact at the point of maximum spring compression in order to change the net energy of the system. Theoretical work in [12], integrated by experimental work in [26], considers a 3-DOF SLIP model with a series elastic actuator at the leg, and a rotational actuator and a spring at the hip. The control action proposed exploits the system's passive dynamics motion in the hip to minimize energy consumption and reach a steady state gait at a target forward speed.

Other studies, either with the passive or active SLIP model, focus on negotiating uneven terrains: e.g., [27] propose three strategies for step-length adjustment that involve changing either one of forward speed, stance time or flight time, while keeping the other two constant, [25] propose a control action to keep the running speed constant. Koepl and Hurst [28] apply the leg controller proposed in [25] to the leg actuated 3-DOF SLIP model, and incorporate it with a controller of the impulses applied by leg and hip actuators during ground contact to match the impulse profile of the passive SLIP model, in order to reject unexpected ground disturbances (of damping, stiffness and surface height). Schmitt and Clark [13] propose a leg-length actuation strategy inspired by muscle activations seen in cockroaches running over uneven terrain. The actuator is moved following a sinusoidal function of time, so that energy is removed from the system during leg compression and added during leg extension. This latter control strategy is extended by Andrews et al. [14], who compare its stability to Raibert's fixed thrust controller and to the swing-leg retraction technique, both in simulation and experimentally.

The authors of [29] propose an algorithm for trajectory planning, robust to model uncertainty and measurement noise.

Work in [15] combines an actuator displacement strategy with a model predictive control approach to plan a trajectory based on a fixed set of desired footholds.

1.1.2 Approximation of the system's dynamics

While the SLIP model is a very simple system, there is not an analytic solution for its dynamics during ground contact [30]. Trajectories have to be pre-computed and stored in look-up tables, which can be timely and computationally expensive to generate, store and access. Furthermore, motions not pre-computed need to be interpolated from existing trajectories. Additionally, the lack of closed-form solution represents a limitation on our ability to understand the SLIP dynamics and to design and implement real-time controllers, as well as when generating return maps to investigate the system's stability.

To overcome these issues, several approximations to the closed-form solution have been given throughout the years. Schwind and Koditschek [31] apply the mean value theorem to the integral of the SLIP dynamics, and find a closed-form solution for the case where there is conservation of momentum. Then, the effects of gravity are iteratively added to equations so that under certain assumptions the solution will converge to the exact SLIP dynamics. Geyer et al. [32] have proposed a solution based upon conservation of momentum and the hypothesis of small angle span. Conservation of momentum is a consequence of ignoring the effect of gravity along the leg; hence, the accuracy of this solution degrades the more the trajectory deviates from the symmetric case.

Locomotion on uneven terrain may imply non-symmetric strides, where the effect of gravity cannot be neglected. Following this observation, works by Arslan et al. [33] and Yu et al. [34] propose approximations which include gravity corrections, improving the model performance. More recently, [35] propose an approximation of the stance phase that yields linear equations, thus allowing to write the entire SLIP dynamics as a piecewise linear function.

1.1.3 SLIP-based robots

In the 80s, Mark Raibert [10] presented the first and best-known planar monopod hopping robot, the so-called Raibert hopper, Fig. 1.2a. The Raibert hopper consists of a body and a compliant elastic leg, both with mass and inertia, connected by a joint at the hip. An actuator at the hip can provide torque between the body and the leg, while a series elastic actuator at the leg compresses or extends the spring. The *three-part control strategy* considers hopping height, speed and body attitude as three separate problems: forward velocity is controlled by the positioning of the leg; hopping height is regulated by the amount of thrust given by the linear actuator at the leg; and the body attitude is controlled by applying torque at the hip. A state machine permits switching between states. Subsequently, Raibert extended the same mechanism and controls to a 3D version of the monopod, Fig. 1.2b, and to biped and quadruped robots with prismatic legs.

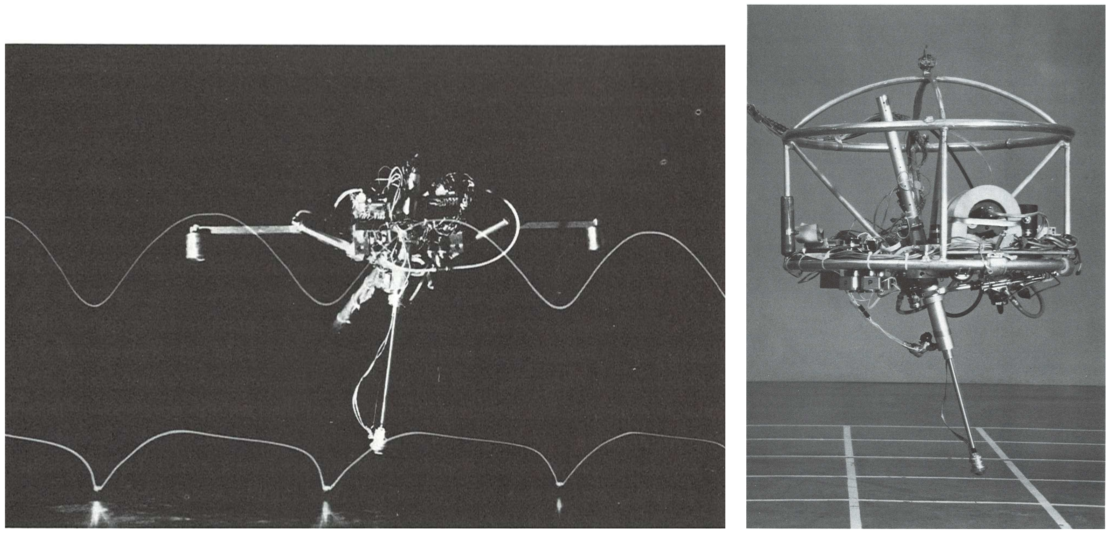

(a) (b)

Figure 1.2: Raibert planar hopper (a) and 3-dimensional hopper (b). From M. Raibert, *Legged Robots That Balance*, MIT Press, 1986

The Raibert hopper opened the door to a long series of SLIP-based robots. Zeglin's Bow Leg hopper [11] is composed of a leg controlled with strings that store energy during flight but are loose during ground contact, so that it behaves like a spring-mass system. A planar hopper mounted on a boom, the Bow Leg hopper, has been also extended to a 3D version ([36] and [37]). An autonomous and untethered hexapod robot, RHex, is proposed by Saranli et al. [38]. With its 6 compliant legs, this robot is able to undergo uneven terrains with obstacles exceeding its body clearance, with limited energy consumption. The ARL-Monopod II [26] (and a previous version [39]) is a hopping robot with a prismatic leg attached to the body via a torsional joint at the hip, hopping on a treadmill. Jonathan Hurst's BiMASC [40] and the most recent ATRIAS robot have been mechanically designed to match the dynamics of the active SLIP model, with particular focus on energy efficiency (see, for example [41]).

1.2 Motivation and Contributions

With so much pre-existing research on the SLIP model, one can ask “What is still left to be done?”. Surprisingly, while most SLIP-like robots are equipped with compliant legs, there is little investigation on the effect that actuation has on the system’s dynamics. As shown in the previous section, actuation is mostly applied in a “blind” way (i.e., via feedback) or by mimicking animals’ muscles. Besides inputting energy into the system to compensate from losses at impact or due to friction, actuation is employed to reach a certain state in terms of running speed/height or stride length. What is missing is an understanding of how a different amount of actuation applied at different times during stride can modify the state reached by the robot. In particular, is it possible to use actuation to simplify the system’s dynamics? What is the reachable state-space of the system, and what actuation policy enables the system to reach the most states?

The goal of this thesis is to explore how a series elastic actuator applied to the SLIP model’s leg can change the system’s dynamics. We principally investigate how, through actuation, we can solve partially or completely the system’s equations of motion, and enforce a desired trajectory. We also determine the reachable state-space of the model using several different actuation strategies, and we investigate the variation of the reachable state with respect to the particular actuator motion, providing a relationship between direction of actuator displacement and location of the reached state. Since the SLIP model can be seen as a template for more complex systems, it is realistic to believe that studies on the effect of actuation can be applicable to more realistic dynamics.

This thesis is structured as follows. Chapter 2 introduces the passive and active SLIP models, characterizes their dynamics, underlines the differences between the two, and contains symbols and definitions that will be used throughout the document. Chapter 3 investigates the shape of the reachable space for the active SLIP model for different actuator motions. Chapter 4 proposes an actuation strategy to enforce a symmetric trajectory for the stance phase, and, at the same time, to allow us to compute an analytical solution for the system's dynamics. In Chapter 5 we propose a partial feedback linearization action for actuator displacement to analytically solve part of its dynamics, thereby reducing computational time and increasing the practicality of performing online control actions. This is then paired with a two-part control action to add/remove energy to/from the system and modify the upcoming apex state to span an open set within the reachable apex states. In addition, we develop two control strategies for online computation of actuator displacement and leg positioning: one to drive the system to a desired state, even in the presence of terrain perturbation; the other to control the system to hop on a desired set of terrain footholds. Furthermore, we propose an adaptive control technique for steady-state locomotion on flat terrain to reduce computation errors by the use of an approximation of the leg-angle dynamics during the stance phase, and we demonstrate the proposed strategy on a more dynamically sophisticated planar hopper model. In Chapter 6 we investigate how different actuator motions can affect the system's state, and we propose a control strategy, based on graphical and numerical studies of the reachability space, and updated throughout the stance phase, to drive the system to a desired state. We quantify its performance benefits, particularly in serving as an error-recovering method. The objective of our control strategy is not to replace any leg-placement approach proposed by other works, but rather to be paired with any other leg-placement or path planning method. Its main advantage is the ability to reduce the effects of sensing errors and disturbances happening at landing as well as during the stance phase. Finally, Chapter 7 contains conclusions and future work.

Chapter 2

The SLIP Model: Preliminaries

In this chapter we review the structure of the passive SLIP model and its dynamics. We also introduce its actuated version that will be used throughout the thesis, the so-called active SLIP model, explaining in detail its dynamics and the modifications made to the original model to incorporate energy variations.

2.1 Passive SLIP Model

The passive SLIP is modelled as a point mass, M, attached to a massless spring leg, with length ℓ and spring stiffness constant k, as shown in Fig. 2.1. The terminal part of the leg is referred to as the foot. Running dynamics for the SLIP model occur on the sagittal plane, and constitute an hybrid system. In fact, they consist of two phases (see Fig. 2.2a): the *flight phase*, where the body is in the air and follows a ballistic trajectory; and the *stance phase*, where the foot is in contact with the ground, and the compression/extension of the spring completely defines the mass dynamics. We will call *touch-down (TD)* the instant that marks the transition from flight to stance; and *take-off (TO)* the instant that marks the transition between stance and flight. During the flight phase gravity is the only force acting on the system. Defining as x and y respectively the forward and vertical coordinates of the mass, the equations of motion during flight can be written as:

$$\ddot{x}(t) = 0,$$
$$\ddot{y}(t) = -g,$$

where, as customary, g is the gravitational acceleration. The highest point reached by the mass during flight is called the *apex state*, and it is defined by zero vertical velocity, i.e., $\dot{y}_a = 0$ [m/s]. Therefore, the apex state is completely defined by a three-dimensional vector $\boldsymbol{s} = \{x_a,\, y_a,\, \dot{x}_a\}$.

The stance phase starts with the leg hitting the ground with a touch-down angle θ_{TD}. While the body moves forward, the spring compresses until it reaches its minimum compression point, then it starts expanding (see Fig. 2.2b). When the spring reaches its equilibrium position (i.e., when the forces in the spring are back to zero), the system leaves the ground with a certain take-off angle θ_{TO}. As shown in Fig. 2.1, we define $\ell(t)$ as the leg-length as a function of time, and $\theta(t)$ as the leg-angle measured counterclockwise with respect to the positive horizontal axis, while ℓ_k is the spring length, and $\ell_{k,0}$ is the spring length at equilibrium. The state of the mass can be easily converted from Cartesian coordinates into polar coordinates:

$$x(t) = \ell(t)\cos\theta(t), \quad y(t) = \ell(t)\sin\theta(t).$$

Then, the Lagrangian for the system during stance phase is derived as follows:

$$L = \frac{M}{2}(\ell^2\dot{\theta}^2 + \dot{\ell}^2) - Mg\ell\sin\theta - \frac{k}{2}(\ell_k - \ell_{k,0})^2,$$

and the equations of motion in polar coordinates for the stance phase can be written as:

$$\ddot{\ell} = -\frac{k}{M}(\ell_k - \ell_{k,0}) - g\sin\theta + \ell\dot{\theta}^2, \tag{2.1}$$

$$\ddot{\theta} = -2\frac{\dot{\ell}}{\ell}\dot{\theta} - \frac{g}{\ell}\cos\theta. \tag{2.2}$$

Note that, despite their simplicity, equations (2.1)-(2.2) are not analytically solvable.

Let us introduce the commonly used non-dimensional *relative spring stiffness*, γ, that will be used throughout this thesis, defined as:

$$\gamma = \frac{k\ell_0}{Mg}. \tag{2.3}$$

The introduction of the relative spring stiffness is motivated by the fact that, if we focus on biped and quadruped robots where each leg functions and behaves the same way as the other(s), we can study their performance by considering the relative spring stiffness γ per leg (or net per multiple legs in contact).

In this work, we will call *jump* the transition from one apex state to the next. A jump is *successful* if the forward velocity during the entire trajectory only takes positive values. Furthermore, to allow the leg to swing forward in flight, we require the distance between y_a and the terrain height to be bigger than the leg length at equilibrium ℓ_0.

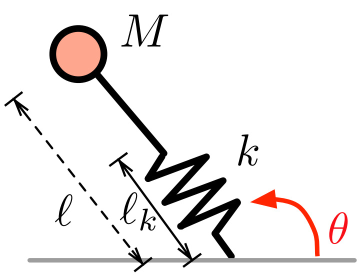
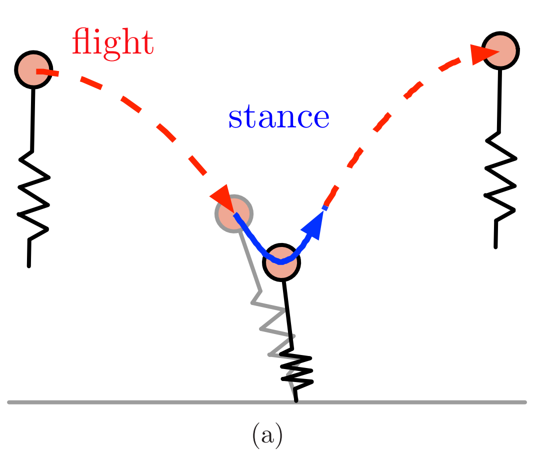


Figure 2.1: Classic SLIP model (passive SLIP)

For each forward velocity (or apex height, equivalently, since energy is conserved)



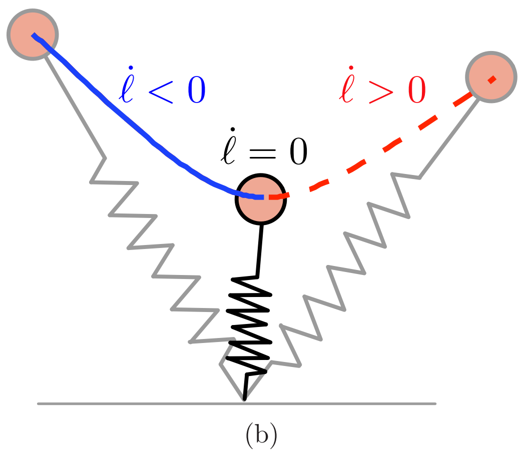


Figure 2.2: (a) Scheme that illustrates the phases of the SLIP model trajectory. (b) Stance phase: the point of maximum compression corresponds to $\dot{\ell} = 0$.

there is a unique touch-down angle the gives a symmetric step, i.e., the apex state is preserved due to zero net acceleration during stance. The corresponding point on the ground is called the *neutral point*. Variations from the neutral point have the effect of increasing or decreasing the next forward velocity. For forward hopping, if the foot is positioned before the neutral point, than there is a net acceleration, while if the foot is positioned after the neutral point there is a net deceleration.

2.2 Active SLIP model

The classic SLIP model is energetically conservative: the repositioning of the leg to the desired touch-down angle does not require any energy, and no energy is lost during impact with the ground as well. However, various studies on legged locomotion, e.g., [8] and [42], suggest that legs store and dissipate energy during motion. Based on this evidence, we modify the passive SLIP by adding to the leg a piston-like actuator in series with the spring, as shown in Fig. 2.3. We will refer to the actuated SLIP model as the *active SLIP*.

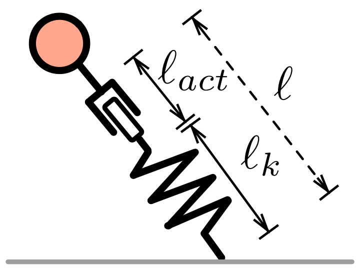


Figure 2.3: Active SLIP model (active SLIP)

Define $\ell_{act}(t)$ to be the actuator length, and $\ell_{act,0}$ be its nominal position at rest. Throughout the stance phase, the actuator can continuously extend ($\ell_{act} > \ell_{act,0}$) or retract ($\ell_{act} < \ell_{act,0}$) from its nominal position within a certain displacement range. Then, the equation that describes the evolution of the leg length (2.1) can be re-written as:

$$\ddot{\ell}(t) = -\frac{k}{M}(\ell(t) - \ell_0 - \ell_{act}(t)) - g\sin\theta + \ell\dot{\theta}^2, \tag{2.4}$$

where $\ell_0 = \ell_{act,0} + \ell_{k,0}$. The actuator manages absorption and production of energy during the stance phase by compressing and decompressing the spring, with the main advantage of allowing the energy at the beginning and at the end of the stance phase to be different. Positive and negative values of ℓ_{act} correspond to a compression and extension of the spring, respectively.

Chapter 3

Reachable Space

Given an inital apex state $\boldsymbol{s}_0 = \{x_a,\, y_a,\, \dot{x}_a\}$, let us define the following function as the transition from one apex to the next:

$$\mathcal{X}_{pass}(\boldsymbol{s}_0,\, \theta_{TD}) = \boldsymbol{s}_1, \tag{3.1}$$

in the case of the passive SLIP, or, for the active SLIP:

$$\mathcal{X}_{act}(\boldsymbol{s}_0,\, \theta_{TD}, \ell_{act}(t)) = \boldsymbol{s}_1, \tag{3.2}$$

where $\boldsymbol{s}_1$ is the apex state reached in one jump from the initial apex state $\boldsymbol{s}_0$, with touch-down angle θ_{TD} and actuator value function during stance phase $\ell_{act}(t)$ (if applicable).

Then, we call the *reachable space* of $\boldsymbol{s}_0$ the set of all apex states that can be reached from $\boldsymbol{s}_0$ in one jump by positioning the leg and, in the case of the active SLIP, by moving the actuator:

$$\mathcal{R}_{pass}(\boldsymbol{s}_0) := \{\boldsymbol{s}_1 \mid \exists\, \theta_{TD} : \mathcal{X}_{pass}(\boldsymbol{s}_0, \theta_{TD}, \ell_a(t)) = \boldsymbol{s}_1\}, \tag{3.3}$$

and

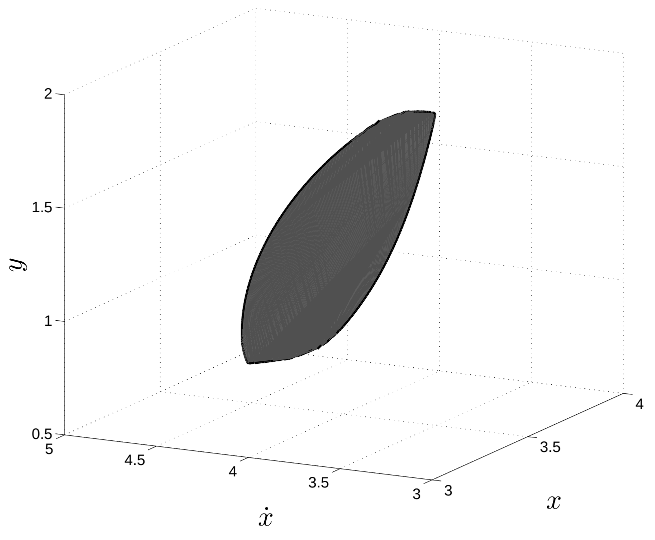
$$\mathcal{R}_{act}(\boldsymbol{s}_0) := \{\boldsymbol{s}_1 \mid \exists\, \theta_{TD},\, \ell_a(t) : \mathcal{X}_{act}(\boldsymbol{s}_0, \theta_{TD}) = \boldsymbol{s}_1\}, \tag{3.4}$$

where $\theta_{TD} \in [\pi/2, \pi]$, and $\ell_{act}(t)$ is a continuous function bounded by the physical limitations of the actuator, such as limited travel and velocity and torque limits.

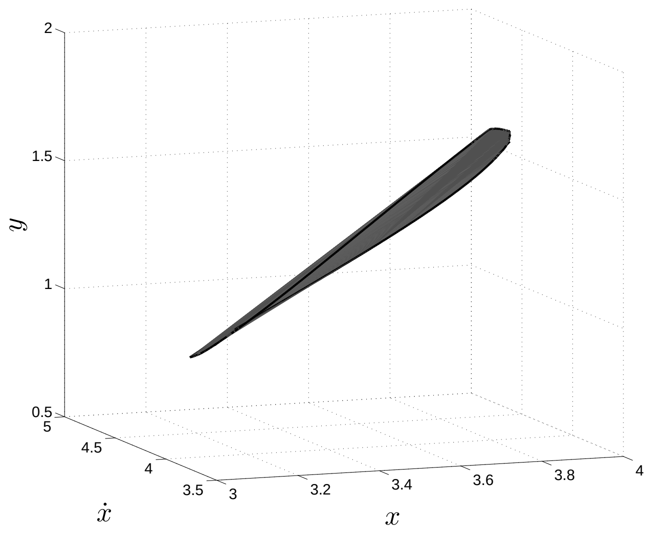
For the passive SLIP model, the reachable space is a line in the $\{x,\, y,\, \dot{x}\}-$space, or, when fixing the touch-down angle, a point in the $\{x,\, y,\, \dot{x}\}-$space. Adding the series actuator has the effect of extending the reachable set to a 3-dimensional surface, whose shape is determined by the "shape" of the function $\ell_{act}(t)$ during the stance phase, and is a function of the actuator's characteristics, such as maximum and minimum compression and extension, and maximum velocity and acceleration allowed. To take into account the limitations of a real-life actuator, we model the actuator dynamics as:

$$\ell_{act}(t) = \ell_{act}(t_i) + \dot{\ell}_{act}(t_i)(t - t_i) + k_{acc}(t - t_i)^2, \tag{3.5}$$

where k_{acc} is the acceleration constant, and t_i is the time at which movement first occurs. When the actuator reaches its maximum velocity allowed, v_{max}, it stops accelerating and proceeds with a constant velocity motion. Additionally, the maximum actuator displacement is assumed to be 10% of the leg length at equilibrium, ℓ_0, and the actuator is assumed to start its motion with zero velocity and displacement. By fixing the touch-down angle, the reachable space for any actuator movement consists of a 2-dimensional manifold, as shown in Fig. 3.1a and 3.1b. The apex states $\{x,\, y,\, \dot{x}\}$ reachable in one step from a fixed touch-down angle form a 2-dimensional manifold, suggesting that the three coordinates are not independent from each other. In Fig. 3.2, the complete reachable space for different values of the actuator's acceleration constant, k_{acc} are computed. As one might expect, the higher the acceleration, the bigger the reachability space. In



(a)



(b)

Figure 3.1: Subplots (a) and (b) show two different views of the 3-dimensional reachable space in one step for a fixed touch-down angle. As we can see, the set is a 2-dimensional manifold.

these particular examples, initial conditions were chosen to achieve a passive symmetric jump, i.e., $\chi_{act}(\boldsymbol{s}_0,\, \theta_{TD},\, 0) = \chi_{pass}(\boldsymbol{s}_0,\, \theta_{TD})\boldsymbol{s}_0$ with no actuator movement. Note that, due to the lack of closed-form solution for the stance dynamics of the SLIP model, the reachable spaces were numerically computed. This operation generally requires significant computational effort; hence, it is not normally feasible to compute in real time the reachable set for any particular initial condition.

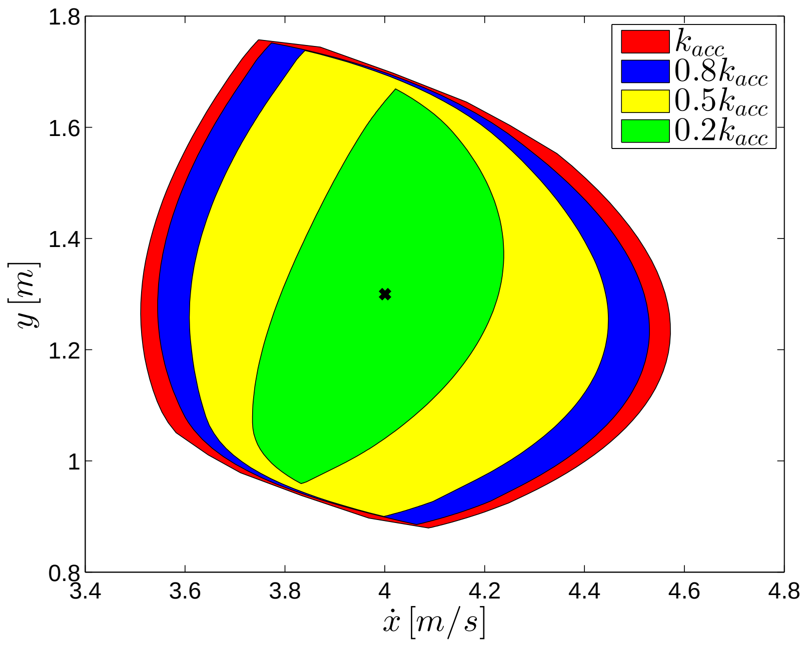
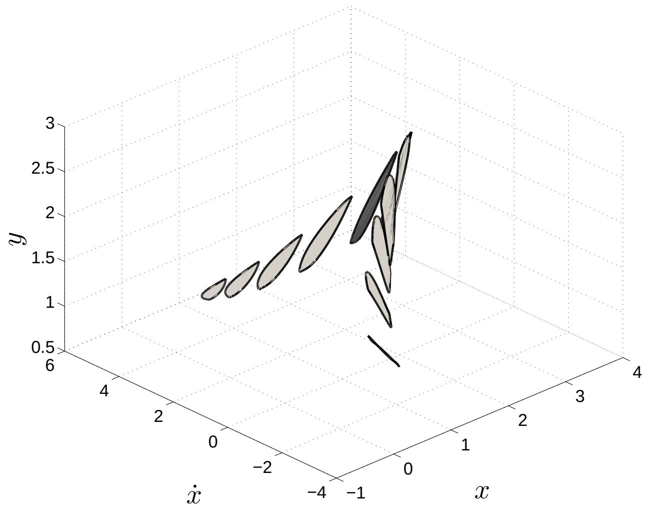
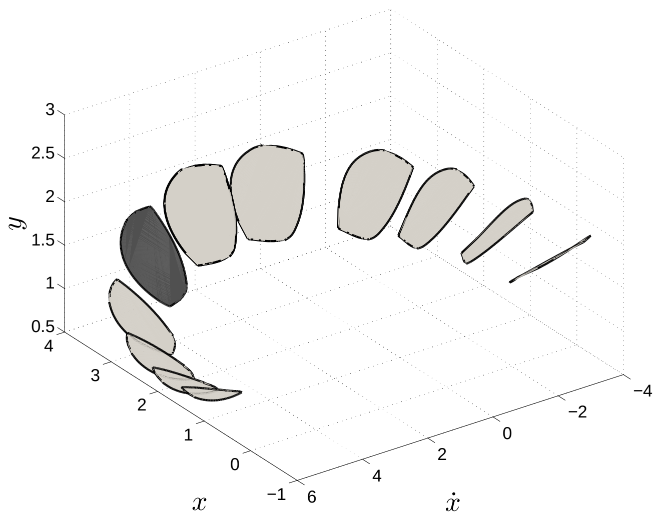


Figure 3.2: Reachability space on the $\{y,\,\dot{x}\}-$space for a passively symmetric jump with actuator moving at different acceleration values, given a maximum allowed velocity of $v_{max} = 0.5$ [m/s], and maximum acceleration constant of $k_{acc} = 10$ [m/s^2]. The "corner" points are the points reached with maximum actuator movement in the positive (upper corner) or negative (lower corner) direction. The black x mark represents the apex state passively reachable (i.e., without any actuation). From this graph we can see that the higher the maximum acceleration allowed, the bigger the reachability space is, in terms of area. Initial conditions chosen: $\gamma = 10$, initial apex state $\boldsymbol{s}_0 = \{0,\, 1.3\,[\text{m}],\, 4\,[\text{m/s}]\}$, touch-down angle $\theta_{TD} = 122.14$ [deg].

In general, for varying touch-down angle, the reachable space is a set of 2-dimensional manifolds, one for every touchdown angle employed, forming a fan-shaped 3-dimensional volume, as shown in Fig. 3.3a and 3.3a.



(a)



(b)

Figure 3.3: Subplots (a) and (b) show two different views of the 3-dimensional reachable space in one step for varying touch-down angles. The dark grey manifold corresponds to the reachability space in Fig. 3.1a and 3.1b.

Chapter 4

Enforced Symmetry of the Stance Phase

Let us assume we want to control the system dynamics to follow a certain trajectory during stance. From equation (2.4), the actuator displacement necessary to maintain a desired feasible trajectory $\theta_{des}(t)$ and $\ell_{des}(t)$ during the stance phase is given by:

$$\ell_{act}(t) = \frac{M}{k}(\ddot{\ell}_{des}(t) + g\sin\theta_{des}(t) - \ell_{des}(t)\dot{\theta}^2_{des}(t)) + \ell_{des}(t) - \ell_0. \tag{4.1}$$

For the control action to succeed, the commanded trajectory needs to be *feasible*: the trajectory we want to enforce needs to be consistent with the dynamics of the system and its limitations. Since the dynamics of $\theta(t)$ and $\ell(t)$ are strongly coupled, it is not possible to control them separately by applying actuation at the spring. Furthermore, to compute (4.1), the desired trajectory needs to be fully known. Given we can observe the initial conditions at touch-down, one starting point in planning is to exploit the fact that it is often possible to reach the same conditions at take-off. When this is the case, the straightforward answer is to find a *symmetric* trajectory which gives a closed-form solution for the stance phase.

4.1 Enforced symmetric gait

The control action required to enforce a symmetric gait to the SLIP model is not unique. However, not all possible symmetric laws can be solved in closed form. In what follows, we start by observing that the passive natural dynamics of the angular velocity with respect to the angular position has a shape that is similar to a part of a sine wave (see Fig. 4.1c). Therefore, a logical approach is to enforce the following dynamics to the angular velocity:

$$\dot{\theta}(t) = A \sin \theta(t) + c_1, \tag{4.2}$$

where the parameters A and c_1 are uniquely determined such that position and velocity of the mass at the start of the stance phase match the ones at the touch-down state. From (2.2) and the time derivative of (4.2), we have:

$$\begin{aligned} A &= -\frac{2\dot{\ell}_{TD}\dot{\theta}_{TD} + g\cos\theta_{TD}}{\ell_{TD}\dot{\theta}_{TD}\cos\theta_{TD}}, \\ c_1 &= \dot{\theta}_{TD} - A\sin\theta_{TD}, \end{aligned}$$

where θ_{TD}, $\dot{\theta}_{TD}$, ℓ_{TD}, $\dot{\ell}_{TD}$ are position and velocity at touch-down of leg angle and leg length, respectively. We can then solve (2.1) to find a closed-form solution for the leg length dynamics:

$$\ell(t) = \frac{g}{A\dot{\theta}(t)} + \frac{c_2}{\sqrt{\dot{\theta}(t)}}, \tag{4.3}$$

with

$$c_2 = \ell_{TD}\sqrt{\dot{\theta}_{TD}} - \frac{g}{A\sqrt{\dot{\theta}_{TD}}}.$$

It is important to note that the leg length $\ell(t)$ can never take negative values. Since the minimum value of $\ell(t)$ corresponds to the case $\theta(t) = \pi/2$, then

$$\frac{g}{A(A+c_1)} + \frac{c_2}{\sqrt{A+c_1}} > 0.$$

For the leg angle, we can solve (4.2) to find a closed-form solution for the angle dynamics during stance:

$$\theta(t) = -2\tan^{-1}\Big(\frac{A + c_4 \tanh(c_4(t+c_3)/2)}{c_1}\Big), \tag{4.4}$$

for $t \in [0, t_{TO}]$, with

$$c_3 = -\frac{2}{c_4}\tanh^{-1}\Big(\frac{A + c_1\tan(\theta_{TD}/2)}{c_4}\Big),$$

and

$$c_4 = \sqrt{A^2 - c_1^2}. \tag{4.5}$$

The take-off time, t_{TO}, is given by:

$$t_{TO} = -c_3 - \frac{2}{c_4}\tanh^{-1}\Big(\frac{A + c_1\tan(\theta_{TO}/2)}{c_4}\Big),$$

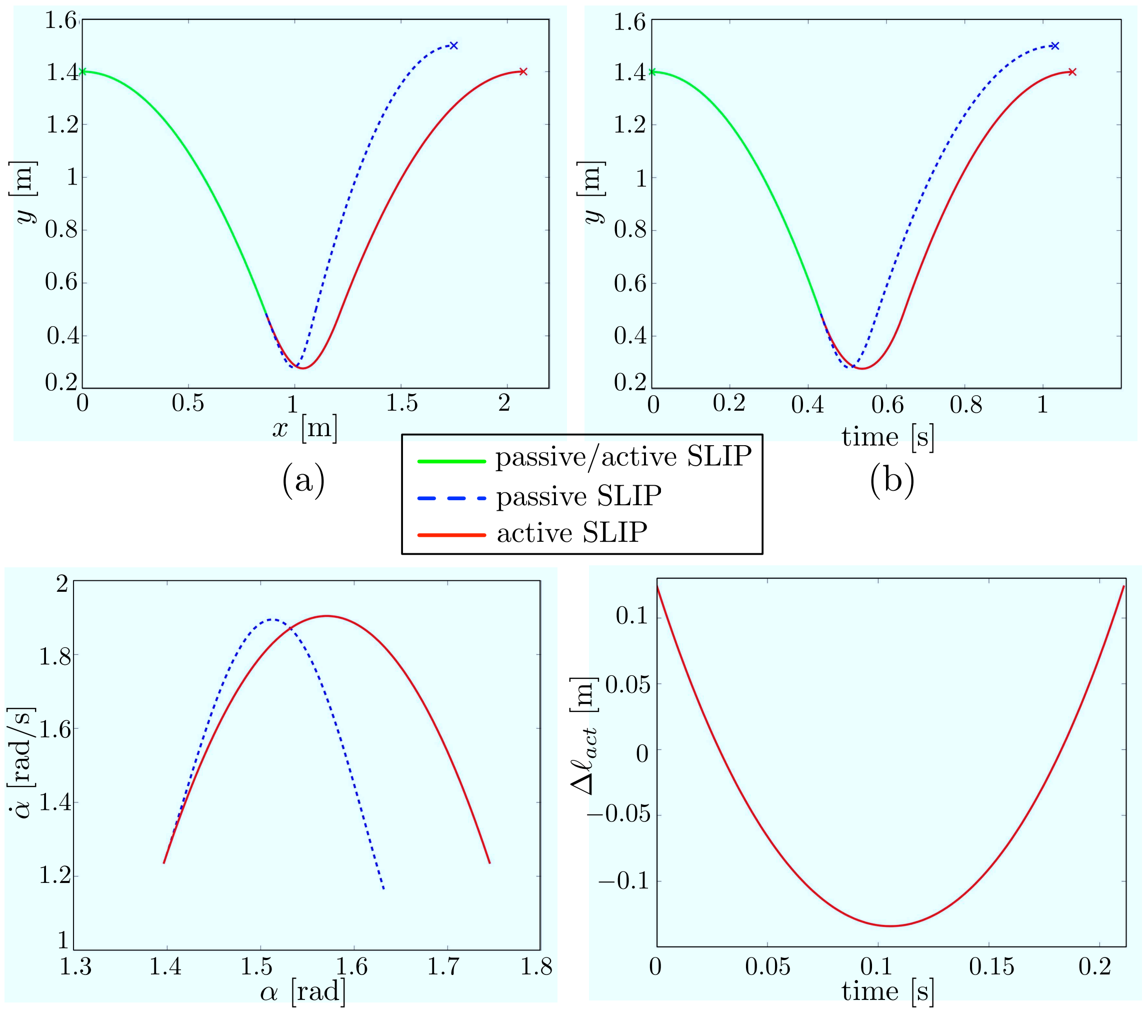
where θ_{TO} is the take-off angle. In order for the above equations to be feasible, we require the following constraint:

$$|A + c_1\tan(\theta_{TD}/2)| \leq |\sqrt{A^2 - c_1^2}|,$$

and the same constraint can be enforced for all $\theta(t)$, obtaining the following:

$$\theta(t) \in [2\tan^{-1}\Big(\frac{-c_4 - A}{c_1}\Big),\ 2\tan^{-1}\Big(\frac{c_4 - A}{c_1}\Big)]. \tag{4.6}$$

Note that the parameter c_4, defined in (4.5), can be a real or imaginary number. All equations from (4.4) to (4.6) still hold.



(c) (d)

Figure 4.1: In these figures, the blue dotted line refers to the passive SLIP, while the solid red line refers to the active SLIP. Figs. (a) and (b) show the mass trajectory for passive and active SLIP with respect to the horizontal position and with respect to time, respectively. Fig. (c) shows the evolution of $\alpha = \pi - \theta$ and $\dot{\alpha}$ during stance for the passive and active SLIP, and Fig. (d) represents the required actuator displacement $\Delta\ell_{act}$. The parameters used are $k/M = 500[s^{-2}]$, $\ell_0 = 1[m]$ and $\ell_{k,0} = 0.5$[m]. The initial apex state is $\boldsymbol{s}_a = \{0\,[\text{m}],\ 1.4\,[\text{m}],\ 2\,[\text{m/s}]\}$, the touch-down angle is $\theta_{TD} = 100$ [deg], and terrain height is 0 [m].

Plots in Fig. 4.1 show an example of the mass trajectory and the angular acceleration of passive and active SLIP for the same initial conditions, and the required actuator displacement for the active case.

4.2 Control actions

4.2.1 Fixed leg length at take-off

When the trajectory is symmetric, we have that $\theta_{TO} = \pi - \theta_{TD}$, and the apex velocity components, $\dot{x}_a$ and $\dot{y}_a$, and the mass apex height y_a are preserved from one apex state to the next. The only control action available is the touch-down angle θ_{TD}, and its choice affects the forward position x_a at the next apex state. If we start from the same initial apex state and we use the same set of possible touch-down angle θ_{TD}, it is clear that enforcing a symmetric trajectory changes the set of reachable apex states with respect to the set reachable by the passive SLIP. As an example, Fig. 4.2 shows the set of reachable apex states for the active and passive SLIP. We can see that, despite the different spatial direction of the two curves, the span of reachable points is comparable in size. This suggests that, even if the reachable apex states are not the same, enforcing a symmetric gate does not generally represent a limitation with respect to the passive SLIP, but it rather changes the direction of the reachable space.

4.2.2 Variable leg length at take-off

There are some cases when enforcing a symmetric gait can pose a limit on the capability of successfully undergoing a particular terrain. For example, on terrains with tall obstacles, or with stairs, it may be necessary to add (or remove) energy to the system to achieve a successful jump. Therefore, we allow the leg to leave the ground either before or after the state which mirrors the touch-down state has been achieved. This

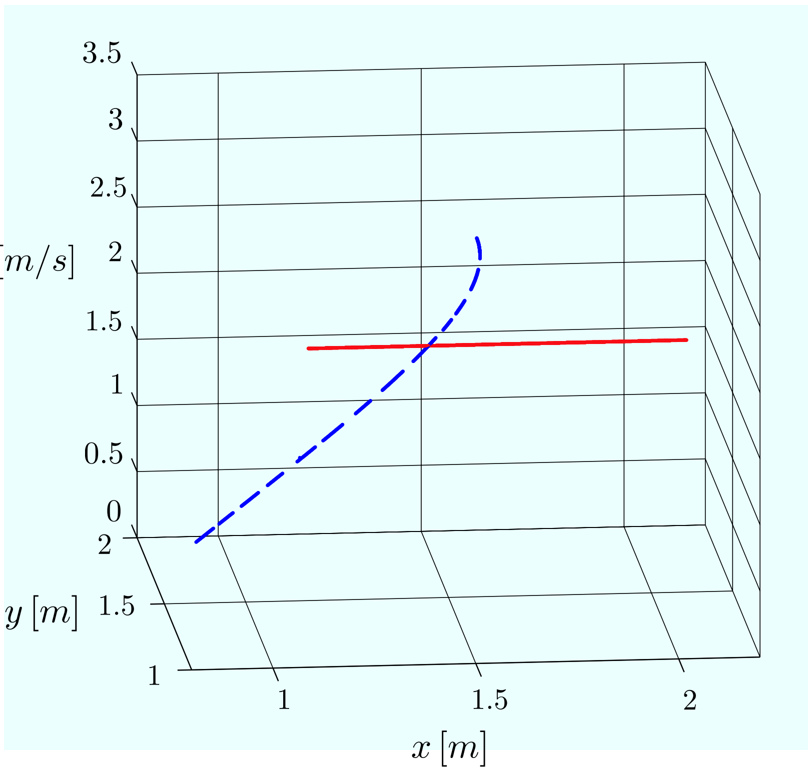
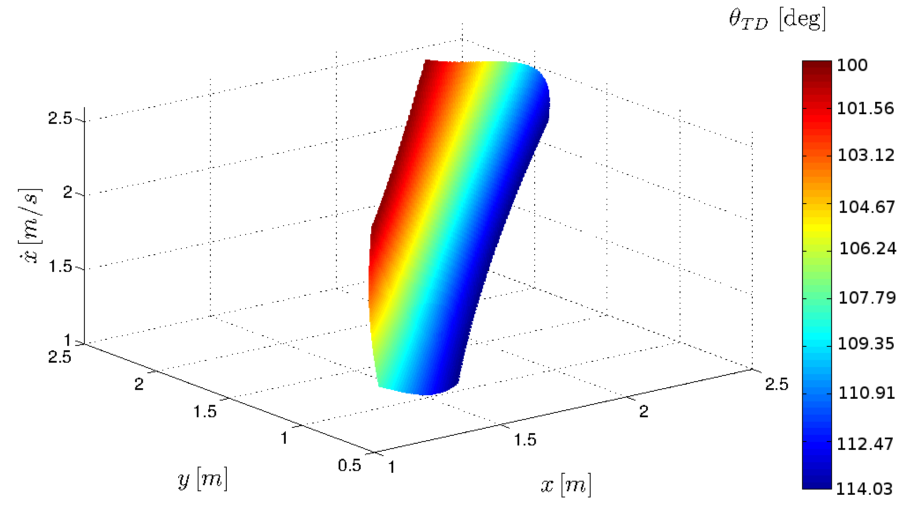
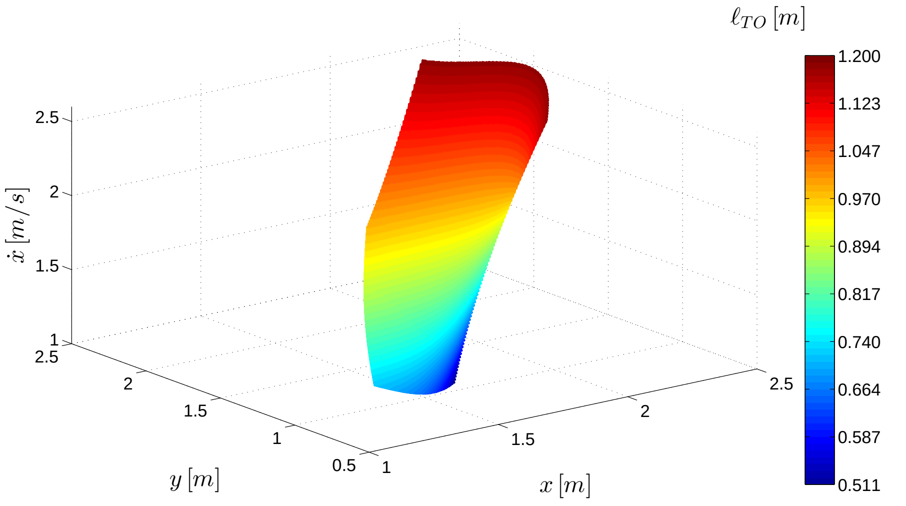


Figure 4.2: The dotted blue line and the solid red line represent the reachable points for passive and active SLIP, respectively, with touch-down angle $\theta_{TD} \in [10,\, 90]\,[deg]$. The system parameters are $k/M = 500[\mathrm{s}^{-2}]$, $\ell_0 = 1$ [m], initial apex state $\boldsymbol{s}_a = \{0\,[\mathrm{m}],\, 1.4\,[\mathrm{m}],\, 2\,[\mathrm{m/s}]\}$, and terrain height 0 [m]. Note that both lines lie on a surface of constant energy.

simply requires setting ℓ_{act} such that the leg spring is fully unloaded, i.e., such that $\ell_{act} = \ell - \ell_0 + \ell_{act,0}$. It is then possible to choose the leg length at take-off, or equivalently to choose the take-off angle θ_{TO} to be different than $\pi - \theta_{TD}$. Hence, the mass will leave the ground at a different take-off state, $(x_{TO},\, y_{TO},\, \dot{x}_{TO},\, \dot{y}_{TO})$, and will reach a different new apex state than otherwise reached. In particular, the velocity components at take-off, $\dot{x}_{TO}$ and $\dot{y}_{TO}$, will be different than the velocity components at touch-down, $\dot{x}_{TD}$ and $\dot{y}_{TD}$. Fig. 4.3 shows the set of all the apex states reachable from an initial nominal apex state. The coloring scheme reflects the direction of the next apex in state space with respect to controlling the touch-down angle θ_{TD}, Fig. 4.3a, or the leg length at take-off ℓ_{TO}, Fig. 4.3b.



(a)



(b)

Figure 4.3: These plots represent the set of apex states reachable from the initial apex state $\boldsymbol{s}_a = \{0\,[m],\,1.4\,[m],\,2\,[m/s]\}$ in the actuated case with leg length at take-off $\ell_{TO} \in [0.5,\,1.2]\,[\mathrm{m}]$ and $\theta_{TD} \in [100,\,117]$ [deg]. The system parameters are $k/M = 500[\mathrm{s}^{-2}]$, $\ell_0 = 1$ [m], and the terrain height is 0 [m].

The main advantages of such a procedure are twofold. First, the capability of adding/subtracting energy to/from the system allows us to reach apex states otherwise unreachable, which translates into the ability of operating in a wider variety of terrains. Second, while the choice of θ_{TD} acts as a control action that happens *before* the impact with the terrain, the choice of θ_{TO} is done *after* the impact. This represents a big advantage in case of noisy measurements (e.g., on the terrain, on the position of the leg, etc.), since it is now possible to partially act against the effect of the noise during the actuated stance phase.

4.3 Applications

This section contains two applications of the proposed symmetric enforcement strategy. Note that the applications are manifold, and here below we just show two useful examples. In the first application, we assume the terrain is divided into allowed and not allowed foothold sets, and we want to determine a path from an initial foothold to an end foothold. In the second application, we aim to reach a desired constant forward velocity or constant apex height during motion on unknown rough terrain. We will assume that the actuator can be extended/contracted only up to a certain percentage of its equilibrium length, and define with ℓ_{act}^{max} the maximum feasible actuator length. This will pose a constraint on the velocity/height reachable in one step.

4.3.1 Foothold placement

One of the main applications of the closed-form solution is the ability to plan a trajectory given a set of feasible footholds: for example, trajectory planning on a terrain with forbidden areas, such as holes or water.

For any initial apex state $\{x_a,\, y_a,\, \dot{x}_a\}$ and terrain height h, we can identify a minimum and a maximum touch-down angle, θ_{min} and θ_{max}, that allow the system to complete a successful jump. Let us call *step length* the distance between two subsequent landing points (Fig. 4.4), and let us define the two quantities d_m and d_M as

$$\begin{aligned} d_m &= \dot{x}_a \sqrt{\frac{2}{g}(y_a - h - \ell_0 \sin\theta_{max})} - \ell_0 \cos\theta_{max}, \\ d_M &= \dot{x}_a \sqrt{\frac{2}{g}(y_a - h - \ell_0 \sin\theta_{min})} - \ell_0 \cos\theta_{min}. \end{aligned} \tag{4.7}$$

When the gait is symmetric, the minimum and maximum step lengths at each apex state $\{x_a, y_a, \dot{x}_a\}$ depend on the length of the previous step as follows: the minimum step length is $s_m = d_0 + d_m$, and the maximum is $s_M = d_0 + d_M$, where d_0 is the distance between x_a and the previous landing point. It follows that the distance between two

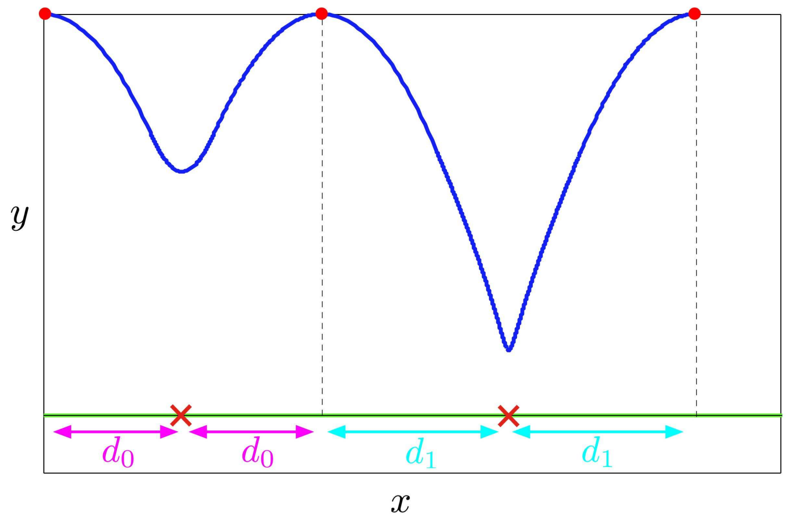


Figure 4.4: The figure shows the distance between touch-down locations for two consecutive steps.

feasible footholds has to take values between $[d0 + m, d0 + M]$. It is straightforward to show that d_0 is bounded: $d_0 \in [d_m, d_M]$. Therefore, a necessary but not sufficient condition for foothold placement is that the distance between two feasible footholds has to be between $[2d_m, 2d_M]$. It is clear that the required distance between footholds is not an absolute measure, but it depends on the previous step length and, in the case of rough terrain, on the terrain height. In fact, values (4.7) depend on the terrain height at the landing point, h. For simplicity, we will assume the terrain to be flat; however, what follows can be easily adjusted in case of rough terrain.

Assume now that the terrain is divided into allowed and not allowed foothold sets, and that we want to determine a path from one foothold v_0 to another one v_{end} (or to a set of footholds, $\mathcal{V}_{end}$), given an initial apex state and initial d_0. We also assume that, at each step, we have the capability to see the terrain up to a certain distance D from our current location. Let us create a graph where each vertex represents a foothold. Starting from an initial foothold/vertex v_0, and initial length d_0, if $D \geq d_0{+}m$ we add an edge from v_0 to all the footholds v_i whose forward distance from v_0 is $d(v_0,\, v_i) \in [d_0{+}d_m,\, \min\{D,\, d_0{+}d_M\}]$, and we associate to the edge connecting v_0 and v_i the length $d_i = d(v_0,\, v_i) - d_0$. (Note that, if $D < d_0{+}d_m$, there is no feasible foothold for that specific initial condition.) Then, for each vertex v_i, we repeat the same process, until all footholds have been explored. As output, we obtain a graph for each feasible path from node v_0 to node $v \in \mathcal{V}_{end}$. The optimal path can then be chosen based on the specific definition of cost function: e.g., we can associate a cost to each edge based on the length covered, or based on the probability that the foothold location is correctly sensed to be feasible.

Algorithm 1 compute feasible paths

```
 1: $\mathcal{V} = \{f_0, ... f_n\}$ set of all footholds
 2: $\mathcal{V}_{end} \subseteq \mathcal{V}$ set of desired path end
 3: Initial state $v = f_0$, initial length $d_0$
 4: graph $G = (V, E, W)$, with $V = \{v\}$, $E = \emptyset$, $W = \emptyset$.
 5: for $v_i \in \mathcal{V}$ s.t. $d(v_i,\, v) \in \min\{D,\, [d_0 + d_m,\, d_0 + d_M]\}$ do
 6:     create edge $e_{v,v_i}$ with weight $d_i = d(v,\, v_i) - d_0$
 7:     cost $C_i$
 8:     $V_i := \{V,\, v_i\}$, $E_i := \{E,\, e_{v,v_i}\}$, $W_i := \{W,\, d_0\}$
 9:     $G_i := \{V_i,\, E_i\, W_i\}$
10:     Repeat steps 5–10 for $v = v_i$, $d_0 = d_i$, and $G = G_i$ until no more possible forward
        jump is feasible
11: end for
12: return  all graphs $G$ that end in $\mathcal{V}_{end}$
```

The complexity of Algorithm 1 grows with the amount of knowledge of the terrain available and the number of feasible footholds. However, we can intuitively suspect that, when the feasible regions of the terrain are high in number and equally spaced, a complete knowledge of the terrain is not required. Such an algorithm is practical if jump transitions can be computed quickly, therefore having an analytic solution is a necessary condition to keep the computation complexity low.

4.3.2 Constant forward velocity/constant apex height

Here, we consider two applications for our variable take-off leg length strategy, concerning the problems of reaching a desired running speed and of reaching a desired apex height with respect to the terrain. In particular, the problem of reaching a desired running speed has already been studied in the literature with different assumptions on the model (see [25] and [43]).

First, let us consider the problem of reaching a desired running speed at the next apex state, $\dot{x}_d$, on rough terrain. Given the initial touch-down conditions (θ_{TD} and ℓ_{TD}), we want to compute the take off angle that allows us to obtain the desired forward velocity, i.e., we need to solve the following equation:

$$\dot{x}_d = \dot{\ell}_{TO} \cos\theta_{TO} - \dot{\theta}_{TO}\ell_{TO} \sin\theta_{TO}. \tag{4.8}$$

Define the variable $z = \sqrt{A \sin\theta_{TO} + c_1}$. After some tedious calculations, one can show that solving equation (4.8) is equivalent to solving for z the following equation:

$$Ac_2 z^5 + (2c_1 g + 2A^2\dot{x}_d)z^2 + (A^3 c_2 - Ac_1^2 c_2)z + (2A^2 g - 2c_1^2 g) = 0,$$

which cannot be solved analytically but can be easily solved numerically. In order to obtain $\dot{x}_d$, the leg needs to leave the ground at an angle $\theta_{TO} = \pi - \sin^{-1}(\frac{z^2-c_1}{A})$, corresponding to a certain take-off leg length.

Since the actuator ability to contract/extend is limited to ℓ_{act}^{max}, then $\ell_{max} = \ell_{act}^{max} + \ell_k$. From equation (4.3) we obtain

$$\frac{g}{A\dot{\theta}_{TO}} + \frac{c_2}{\sqrt{\dot{\theta}_{TO}}} \leq \ell_{max},$$

and therefore

$$z \geq \frac{c_2 + \sqrt{c_2^2 + 4g\ell_{max}/A}}{2\ell_{max}}. \tag{4.9}$$

By defining

$$\hat{z} = \frac{c_2 + \sqrt{c_2^2 + 4g\ell_{max}/A}}{2\ell_{max}}, \quad c_5 = \frac{\hat{z}^2 - c_1}{A},$$

we can compute the maximum forward velocity obtainable in one step as:

$$\begin{aligned} \dot{x}_{max} &= \ldots \\ &= (\frac{g}{A\hat{z}^2} + \frac{c_2}{\hat{z}})(-\frac{A}{2} + \frac{Ac_5^2}{2} - \hat{z}^2 c_5) + \frac{g}{2\hat{z}^2}(c_5^2 + 1). \end{aligned} \tag{4.10}$$

The minimum forward velocity can be easily computed as the value the velocity takes at the time $\tilde{t}$ when $\theta(\tilde{t}) = \pi/2$.

Now, let us consider the problem of maintaining a constant apex height with respect to the ground. This application is particularly useful in situations where the terrain height is progressively increasing and/or decreasing. In fact, it is crucial to maintain a certain minimum distance from the terrain to avoid collision of the leg with the ground. At the same time, it is important to keep the relative apex height limited to avoid gaining excessive velocity during the ballistic descent, to avoid “bottoming out” the passive spring, with the resulting impossibility to perform a successful jump.

Let us assume no prior knowledge of the terrain, in which case our strategy focuses on keeping the height of the next apex constant with respect to the ground at the previous landing point, which is known a posteriori. Given the initial touch-down conditions (θ_{TD} and ℓ_{TD}) and the terrain height at the previous foothold, y_t, we want to compute the take-off angle that results in obtaining the desired apex height Δy_d with respect to y_t. Note that $\Delta y_d = y_a = y_t$, where y_a is the next apex height. Solving this problem is equivalent to solving for θ_{TO} the following equation:

$$\Delta y_d = \ell_{TO} \sin\theta_{TO} + \frac{1}{2g}\Big(\dot{\ell}_{TO}\sin\theta_{TO} + \ell_{TO}\dot{\theta}_{TO}\cos\theta_{TO}\Big)^2. \tag{4.11}$$

As for the constant velocity case, this is equivalent to solve for $z = \sqrt{A\sin\theta_{TO} + c_1}$ a 10^{th}-order polynomial equation in z (not shown here because of its excessive length). Because of limitation on the actuator length, we obtain again constraint (4.9) on z. Correspondingly, we can compute the maximum and minimum apex height obtainable in one step.

4.4 Simulations

In what follows we illustrate through simulations the performance of our example control strategies as discussed in 4.3.1 and 4.3.2. The system parameters used are defined in the following table:

k/M [s^{-2}]	g [m/s^2]	$\ell_{act,0}$ [m]	$\ell_{k,0}$ [m]
100	9.81	0.5	0.5

4.4.1 Foothold placement

The following simulations refer to 4.3.1. Algorithm 1 has been tested on a rough terrain of length 50 [m]. Knowledge of the terrain ahead has been assumed to be $D = 10$

[m]. The terrain height has been built as a normal distribution with 0 mean and standard deviation $\sigma = .1$. The terrain step length has been chosen to be 0.6 [m]. The apex state is $\boldsymbol{s}_a = \{1.3\,[\mathrm{m}],\, 1.4\,[\mathrm{m}],\, 3\,[\mathrm{m/s}]\}$ and $d_0 = 1.3$ [m]. In order to use the algorithm, we discretized the terrain in intervals of size 0.1 [m], and we considered to be unfeasible all the points x_t with height $y_t(x_t)$ such that $|y_t(x_t)| \geq 0.2$ [m], and all the parts close to a change of height, in order to avoid leg collision with the terrain during stance. The cost function has been chosen to be the final covered distance. The output is shown in Fig. 4.5. The green line and red line represent respectively the feasible and unfeasible parts of the terrain, the blue dotted line represents the mass trajectory, and the black x marks are the landing points.

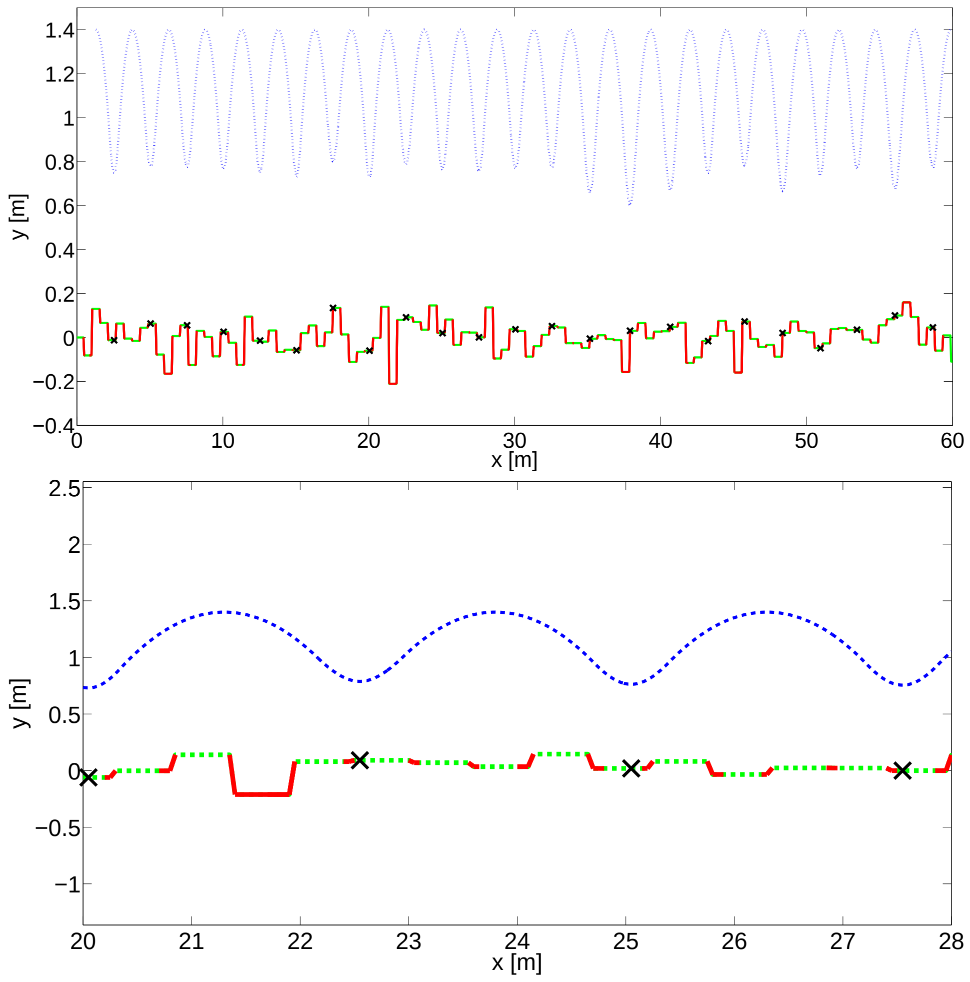


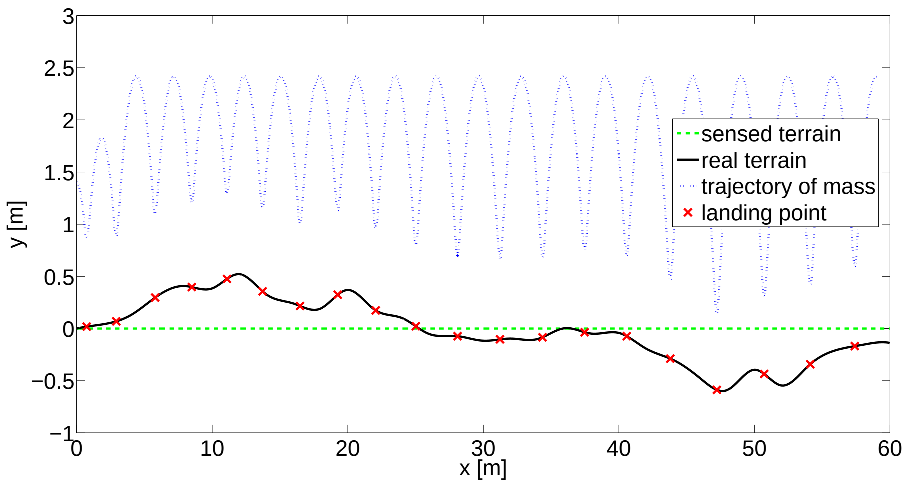
Figure 4.5: The figure shows the path computed by algorithm 1 where the cost function is defined to be the final covered distance. The green and red line represent respectively the feasible and unfeasible parts of the terrain, the blue line represents the mass trajectory, and the black x marks are the landing points.

4.4.2 Constant forward velocity/constant apex height

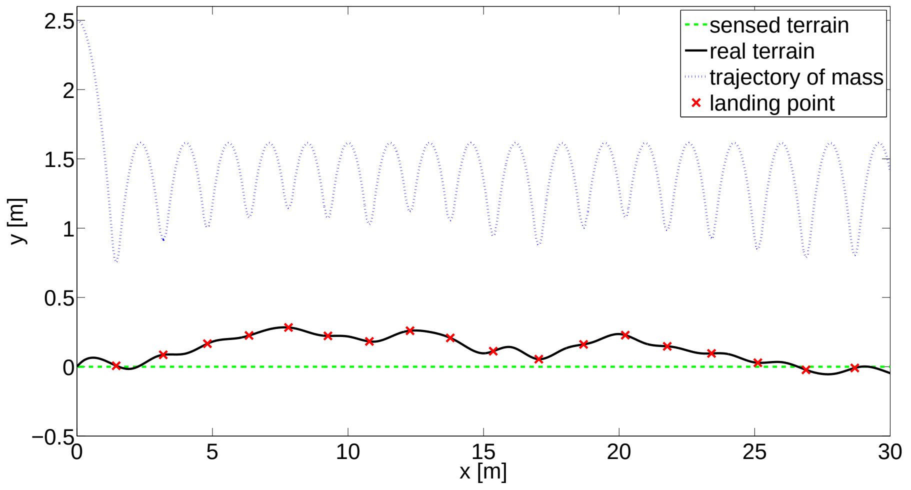
We now demonstrate through simulations the performance typical of the control strategies introduced in 4.3.2.

First, let us examine the problem of reaching a desired forward speed. Consider the active SLIP model, and assume the actuator on the leg can extend/contract up to 20% of the leg equilibrium length. Let us assume the active SLIP travels on a rough terrain, but that it is not provided with any knowledge of it. We will then treat the terrain as if it were completely flat. We also fix the touch-down angle at $\theta_{TD} = 100$ [deg]. During stance, we compute the take-off angle to reach the desired velocity by solving (4.8) as follows. If the maximum forward velocity obtainable in one step, $\dot{x}_{max}$ computed with (4.10), is $\dot{x}_{max} \geq \dot{x}_d$, then the take-off angle θ_{TO} will be computed using (4.8). If $\dot{x}_{max} < \dot{x}_d$, then θ_{TO} will be chosen to reach at the next apex forward velocity $\dot{x}_{max}$, and therefore the number of steps necessary to reach the desired steady-state will be greater than one. Fig. 4.6a shows a numerical example, with initial apex state $s_a = \{0\,[\text{m}],\, 1.3\,[\text{m}],\, 2\,[\text{m/s}]\}$ and desired forward velocity $\dot{x}_d = 2.5$ [m/s]. As we can see, the system reaches the desired forward velocity in 2 steps, and it is able to successfully run on a rough terrain with error that often exceeds 50% of the leg length. Fig. 4.6b shows a numerical example, with initial apex state $s_a = \{0\,[\text{m}],\, 2.5\,[\text{m}],\, 2.3\,[\text{m/s}]\}$ and desired forward velocity $\dot{x}_d = 2[m/s]$, smaller than the inital apex velocity. As we can see, the system reaches the desired forward velocity in 1 step.

Let us now examine the problem of reaching a desired height from the terrain, assuming the actuator can extend/contract up to 20% of the leg equilibrium length. Let us assume the active SLIP is not provided with any knowledge of the terrain it travels on. We fix the touch-down angle at $\theta_{TD} = 100$ [deg]. During stance, we compute the take-off angle to reach the desired terrain height with respect to the previous land-



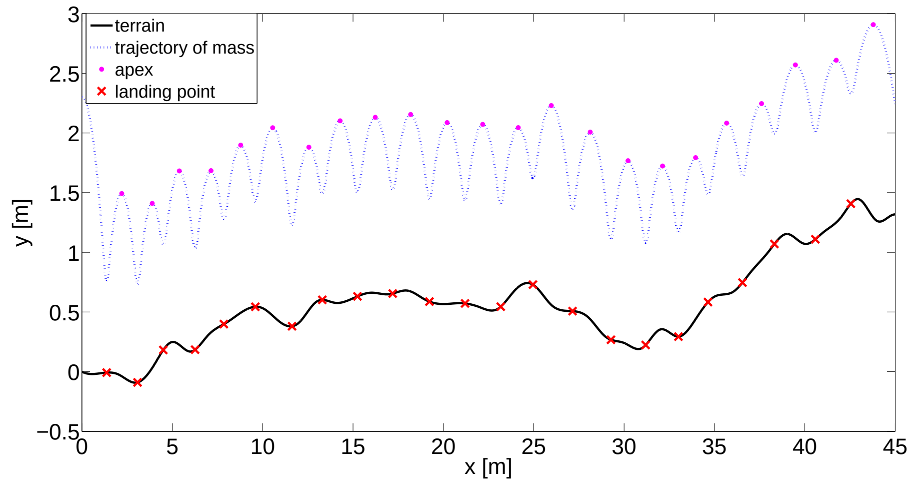
(a)



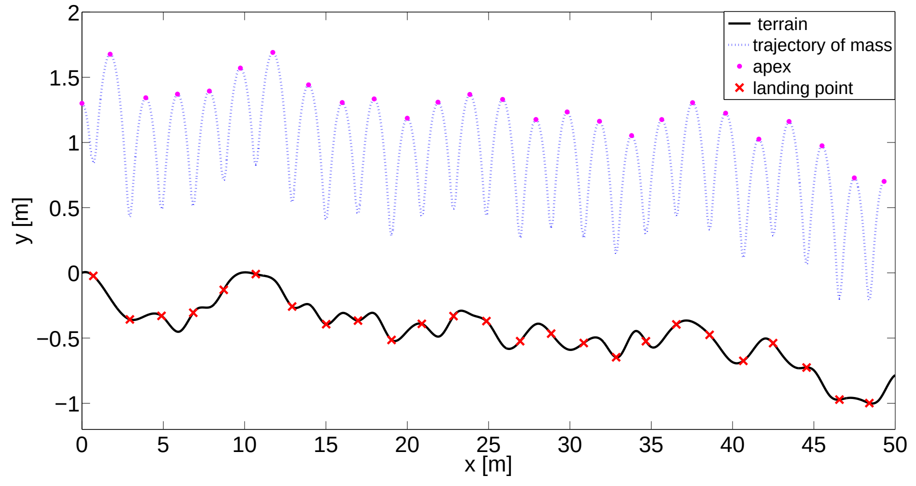
(b)

Figure 4.6: The figure shows two numerical examples of the forward velocity control strategy proposed in 4.3.2. The black line represents the real terrain, and the dotted green flat line the sensed terrain. The red x marks are the landing points and the dotted blue line is the mass trajectory.

ing height by solving (4.11). Fig. 4.7a shows a numerical example, with initial apex state $s_a = \{0\,[\mathrm{m}],\, 2.3\,[\mathrm{m}],\, 2.3\,[\mathrm{m/s}]\}$ and desired relative height $\Delta y_d = 1.5$ [m], whereas Fig. 4.7b shows a numerical example, with initial apex state $s_a = \{0\,[\mathrm{m}],\, 1.3\,[\mathrm{m}],\, 2\,[\mathrm{m/s}]\}$ and desired relative height $\Delta y_d = 1.7$ [m].



(a)



(b)

Figure 4.7: The figure shows two numerical examples of the relative height control strategy proposed in 4.3.2. The black line represents the real terrain, the dotted blue line is the mass trajectory, the red x marks are the landing points and the magenta dots are the apex states.

4.5 Conclusions

In this chapter, we developed a control law for actuator displacement in order to enforce a symmetric stance phase to the model. For the specific control law we enforced, we were able to compute a closed-form solution for the trajectory of the mass during the stance phase. We then exploited the possibility of the leg leaving the ground either before or after reaching the take-off state that is symmetric to the touch-down state, and as a consequence we were able to add and remove energy from the system. We also provided two examples of useful applications for our method (trajectory planning on fixed footholds, and running on rough terrain at constant forward velocity and at a constant relative height), and we validated them through simulations.

Chapter 5

Approximation and two-element control

We have seen in Chapter 4 that the series elastic actuator can be used to enforce a trajectory for the stance phase. In particular, the trajectory was chosen to be symmetric to be able to analytically solve the equations of motion. There are undeniable benefits in having access to a closed-form solution for the system's dynamics, both on a control and on a computational point of view. Unfortunately, it is not always practical to enforce a symmetric stance phase. Especially if the ability of locking the leg at take-off is precluded, it is not possible to inject/remove energy to/from the system.

In this chapter we investigate how it is possible to use actuation to analytically solve part of the stance phase dynamics. Subsequently, we pair this solution with an approximation for the remaining equation, and we propose a control strategy to drive the system to a desired apex state.

5.1 Approximating the stance phase dynamics through PFL

Here, we propose a strategy for the actuator displacement $\ell_{act}(t)$ to solve part of the stance dynamics, and we propose an approximation of the remaining equation.

5.1.1 Exact solution via partial feedback linearization

Let us write the total actuator displacement as: $\ell_{act}(t) = \ell_{nl}(t) + \ell_c(t)$, with total velocity $v_{act}(t) = v_{nl}(t) + v_c(t)$. The first term, $\ell_{nl}(t)$, performs a partial feedback linearization (PFL): it has the purpose of cancelling the nonlinear terms in (2.4):

$$\ell_{nl}(t) = \frac{M}{k}\big(g \sin\theta(t) - \ell(t)\dot{\theta}(t)^2\big). \tag{5.1}$$

Note that, at touch-down, this becomes:

$$\ell_{nl,TD} = \frac{M}{k}\big(g \sin\theta_{TD} - \ell_0\dot{\theta}_{TD}^2\big),$$

which is in general not zero, and therefore the spring may need to be pre-compressed or pre-extended during flight. Substituting (5.1) in (2.4), we obtain:

$$\ddot{\ell}(t) = -\frac{k}{M}(\ell(t) - \ell_0 - \ell_c(t)),$$

where $\ell_c(t)$ is chosen as follows.

We drive the second term, $\ell_c(t)$, to a particular value $\bar{\ell}_c$. However, since the actuator does not move instantaneously, we assume that, after cancelling the nonlinearity, the actuator moves with a constant velocity v_c from its initial position, $\ell_c(t_i)$, until it reaches the desired value $\bar{\ell}_c$:

$$\ell_{act}(t) = \begin{cases} \ell_{nl}(t) + \ell_c(t_i) + v_c t, & \text{if } |v_c t + \ell_c(t_i)| < |\bar{\ell}_c| \\ \ell_{nl}(t) + \bar{\ell}_c, & \text{otherwise,} \end{cases}$$

as illustrated in Fig. 5.1.

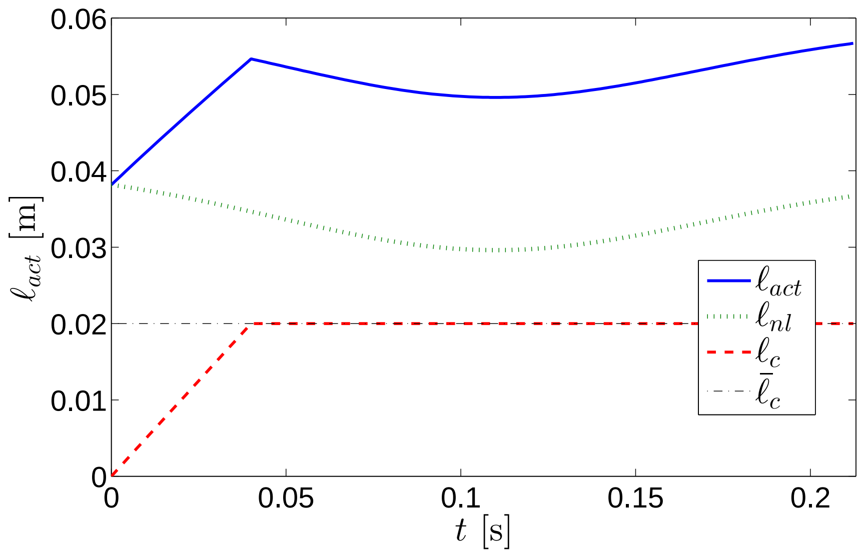


Figure 5.1: Example of the total actuator value $\ell_{act}(t)$ during stance phase, and its two components $\ell_{nl}(t)$ and $\ell_c(t)$.

Note: in general, $v_{nl}(t)$ is not a constant value. Since v_c is set to be a constant, the total actuator velocity required $v_{act}(t)$ is a time-varying function.

During the transition time required to reach the desired actuator value $\bar{\ell}_c$, the equation that describes the leg-length dynamics can be analytically solved as

$$\ell(t) = r + v_c t + a \cos(\omega t + \beta), \tag{5.2}$$

where[1]

$$\beta = \text{atan}_2\Big(-\frac{\dot{\ell}(t_i) - v_c}{\omega},\ \ell(t_i) - \ell_0 - \ell_c(t_i)\Big),$$
$$a = \frac{\ell(t_i) - \ell_0 - \ell_c(t_i)}{\cos\beta}, \quad r = \ell_0 + \ell_c(t_i),$$

t_i is the initial time, and $\omega = \sqrt{k/M}$.

Once the actuator reaches the desired final value $\bar{\ell}_c$, the leg-length dynamics are described by

$$\ell(t) = r + a\cos(\omega t + \beta), \tag{5.3}$$

where

$$\beta = \text{atan}_2(-\frac{\dot{\ell}(t_c)}{\omega},\ \ell(t_c) - \ell_0 - \bar{\ell}_c),$$
$$a = \frac{\ell(t_c) - \ell_0 - \bar{\ell}_c}{\cos(\beta)}, \quad r = \ell_0 + \bar{\ell}_c,$$

and t_c is the time at which $v_c t_c = \bar{\ell}_c$.

5.1.2 Approximation of the leg angle dynamics

The equation of motion for the angular displacement over time, $\theta(t)$ (2.2), is still not analytically solvable. However, we notice that Equation (5.3) has the same form as the approximation of the leg-length dynamics in [34], with two main differences: (*i*) [34] considers the passive SLIP model only, and (*ii*) the equation for the leg-length dynamics provided in [34] is an approximation, while (5.3) and (5.2) are exact solutions. We then chose to follow the same initial steps of the procedure to approximate $\theta(t)$ proposed in [34], modifying and extending the results to adapt them to our actuated case.

[1]For any point (x, y) in the xy-plane minus the origin, $\text{arctan}_2(y, x)$ is defined to be the angle between the horizontal positive axis and the point (x, y) measured counterclockwise.

When the actuator reaches the final desired value ℓ_c, the dynamics of the leg length during stance are described by Equation (5.3). Let us define $\alpha(t) = \theta(t) - \pi/2$, and assume that both $\alpha(t)$ and the angular span $\Delta\alpha = \alpha_{TD} - \alpha_{TO}$ are sufficiently small to be simplified as $\sin(\alpha) \approx \alpha$. Then, equation (2.2) can be simplified and re-written in terms of α, becoming:

$$\ddot{\alpha}(t) \approx -2\frac{\dot{\ell}(t)\dot{\alpha}(t)}{\ell(t)} + \frac{g}{\ell(t)}\alpha(t). \tag{5.4}$$

Toward solving for a corresponding analytic approximation for $\alpha(t)$, we introduce the variables $u(t)$ and $p(t)$ such as $\alpha(t) = p(t)u(t)$. Substituting $\alpha(t)$ in (5.4), we obtain

$$\ddot{u}p + \dot{u}(2\dot{p} + 2\frac{\dot{\ell}p}{\ell}) + u(\ddot{p} + 2\frac{\dot{\ell}\dot{p}}{\ell} - g\frac{p}{\ell}) = 0.$$

Setting the coefficients associated to $\dot{u}$ to be equal to zero yields

$$\ell\dot{p} + \dot{\ell}p = 0,$$

whose solution is $p = 1/\ell$. Then, equation (5.4) becomes:

$$\ddot{u} - u(\frac{\ddot{\ell} + g}{\ell}) = 0. \tag{5.5}$$

When the actuator reaches the final desired value ℓ_c, the equation that describes the dynamics of the leg length during stance is Equation (5.3). Then, (5.5) becomes

$$\ddot{u} - u\left(-\omega^2 + \frac{\omega^2 + g/r}{1 + \epsilon\cos(\omega t + \beta)}\right) = 0,$$

with $r = \ell(t_c) + \ell_c$, and $\epsilon = \frac{a}{r}$. We can write

$$\epsilon = \frac{z}{\cos\beta},$$

where

$$z = \frac{\ell(t_c)}{\ell_0 + \bar{\ell}_c}, \quad \text{or} \quad z = \frac{\ell(t_i)}{\ell_0 + \ell_c(t_i)}.$$

Let us assume small leg compression and small actuation, i.e., $(\ell(t) - \ell_0 - \ell_c)/(\ell_0 + \ell_c) \ll 1$, then we have that

$$\lim_{z \to 0} |\epsilon| = |\frac{\dot{\ell}(t_c)}{\omega(\ell_0 + \bar{\ell}_c)}|,$$

which, for typical values of w and leg length velocity, has magnitude less than 1. Then, for small values of ϵ, we have that

$$\frac{1}{1 + \epsilon\cos(\omega t + \beta)} \approx 1 - \epsilon\cos(\omega t + \beta) + \epsilon^2\cos^2(\omega t + \beta) - \cdots,$$

and thereby obtain the Mathieu equation

$$\ddot{u} - u\left(\lambda^2 - \epsilon\delta\cos(\omega t + \beta)\right), \tag{5.6}$$

with

$$\delta = \omega^2 + g/r, \quad \lambda^2 = g/r.$$

Equation (5.6) can be developed as

$$u(t) = u_0(t) + \epsilon u_1(t) + \epsilon^2 u_2(t) + \cdots.$$

We then obtain

$$\alpha(t) \approx \frac{1}{\ell(t)}(u_0(t) + \epsilon u_1(t)),$$

where $u_0(t)$ and $u_1(t)$ are solutions of

$$\begin{aligned}&\frac{d^2}{dt^2}u_0 - \lambda^2 u_0 = 0,\\ &\frac{d^2}{dt^2}u_1 - \lambda^2 u_1 = -\delta u_0 \cos\omega t + \beta.\end{aligned}$$

We want now to modify the procedure to approximate the leg angle dynamics proposed in [34] and performed above, to adapt it to (5.2). During the transition time required to reach the desired value ℓ_c, the equation that describes the leg length dynamics can be analytically solved as (5.2). Then, equation (5.5) becomes

$$\ddot{u}(t) - u(t)\frac{-c_2\omega^2 \cos(\omega t + \beta) + g}{c_1 + c_2 \cos(\omega t + \beta) + v_c t} = 0.$$

Defining $\psi = \omega t + \beta$, we have $\frac{d}{dt}u(t) = \omega u'(\psi)$ and $\ddot{u}(t) = \omega^2 u''(\psi)$, where $(\cdot)'$ represents the derivative with respect to ψ. Then, the previous equation becomes:

$$u''(\psi) - u(\psi)\Big(-1 + \frac{1+\zeta+\xi\psi}{1+\kappa\cos\psi+\xi\psi}\Big) = 0, \tag{5.7}$$

with

$$\xi = \frac{v_c}{c_1\omega - \beta v_c}, \quad \kappa = \frac{\xi c_2 \omega}{v_c}, \quad \zeta = \frac{\xi g}{\omega v_c}.$$

We can expand the fractional term in (5.7) around $\cos\psi = 0$, obtaining

$$\begin{aligned}&-1 + \frac{1+\zeta+\xi\psi}{1+\kappa\cos\psi+\xi\psi} \backsimeq\\ &\tilde{\lambda}^2 - \tilde{\delta}(\tilde{\varepsilon}\cos\psi - \tilde{\varepsilon}^2\cos^2\psi + \tilde{\varepsilon}^3\cos^3\psi - \cdots),\end{aligned}$$

where

$$\tilde{\varepsilon} = \frac{2(\kappa - \xi)}{2 + \pi\xi}, \quad \tilde{\lambda}^2 = \frac{2\zeta}{2 + \pi\xi}, \quad \tilde{\delta} = \frac{\kappa}{(\kappa - \xi)} + \frac{2(1 + \zeta)}{2 + \pi\xi}.$$

Then

$$\alpha(t) \approx \frac{1}{\ell(t)}(\tilde{u}_0(t) + \tilde{\varepsilon}\tilde{u}_1(t)),$$

where $\tilde{u}_0(t)$ and $\tilde{u}_1(t)$ are solutions of

$$\begin{aligned} \tilde{u}_0'' - \tilde{\lambda}^2 \tilde{u}_0 &= 0, \\ \tilde{u}_1'' - \tilde{\lambda}^2 \tilde{u}_1 &= -\tilde{\delta}\tilde{u}_0 \cos\psi. \end{aligned}$$

5.2 Choice of the actuator constant value ℓ_c: two-element strategy

We now propose a strategy for the choice of the actuator constant value ℓ_c.

Let us divide the stance phase in two parts, separated by the point of maximal leg compression: a first part, where $\dot{\ell}(t) \leq 0$, and a second part, where $\dot{\ell}(t) \geq 0$. Our main control action consists in choosing two constant values for ℓ_c: one for the first part, ℓ_{c1}, and one for the second part ℓ_{c2}, of the stance phase, as shown in Fig. 5.2.

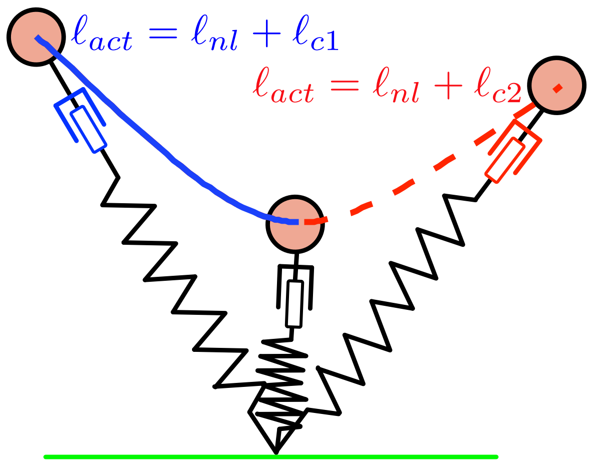

$\ell_{act} = \ell_{nl} + \ell_{c1}$
$\ell_{act} = \ell_{nl} + \ell_{c2}$

Figure 5.2: Proposed two-element actuator motion during stance phase

Dissimilarly from Raibert's famous single-legged hopper, which adjusts the energy via actuation once at mid-stance, we propose to adjust the actuator motion twice during stance. This approach allows us not only to regulate the system's energy, but, more particularly, its two components of forward velocity and apex height. As Fig. 5.2 illustrates, actuation during the entire stance phase allows the system to reach a wider variety of apex states than would otherwise be reached with mid-stance actuation only. Specifically, using a single (1-dimension) thrust parameterization will clearly map to only a 1-dimensional set of apex states, while the full reachable set of apex states is an approximately 2-dimensional surface ([44], [15]).

Furthermore, the decision of limiting our control to a two-parameter (ℓ_{c1}, ℓ_{c2}) choice for the actuator displacement as opposed to a time dependent function (as, for example, in [13], [44], and [15]) has been dictated by the purpose of keeping the system as simple as possible, without much loss on performance. Fig. 5.3a and 5.3b provide an example of how, by setting only two actuator values, it is possible to reach in one jump a wide range of apex states, influencing both apex height, apex velocity and apex forward position, in all reachable directions.

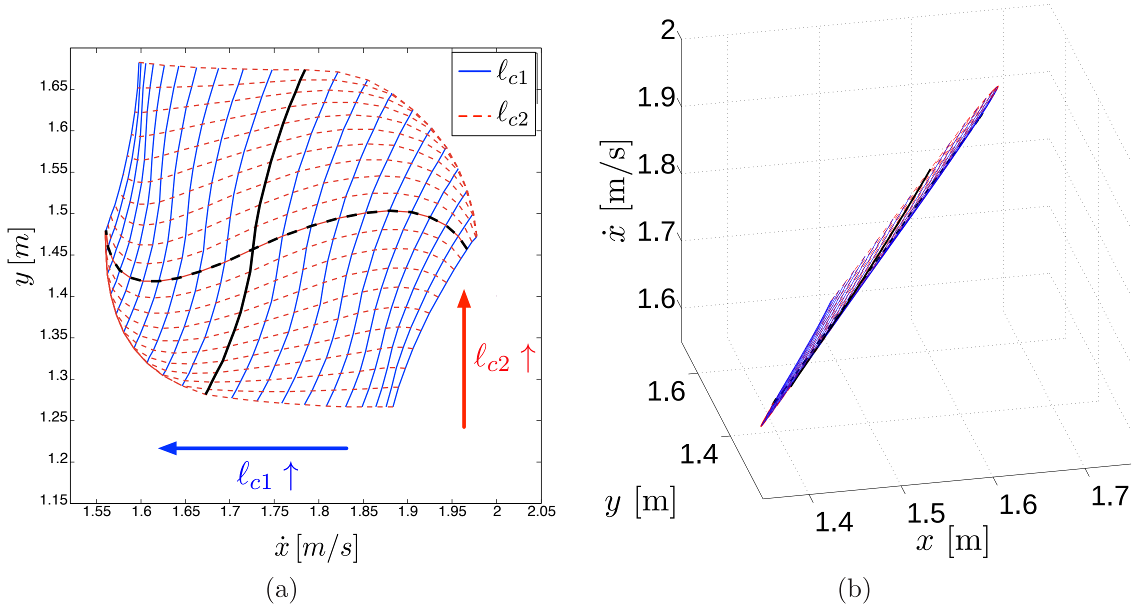


Figure 5.3: Apex states $\{y, \dot{x}\}$ (a), and $\{x, y, \dot{x}\}$ (b), reachable in one jump, with $M = 10$ [kg], $k = 1962$ [N/m], $\ell_0 = 1$ [m]. Initial apex: $x_{apex} = 0$ [m], $y_{apex} = 1.4$ [m], $\dot{x}_{apex} = 2$ [m/s], $\theta_{TD} = 103$ [deg]. Maximum/minimum actuator length: ± 0.05 [m]. Actuator moves with velocity $v_c = 0.5$ [m/s]. Each solid blue line corresponds to a different actuator value during the first half of the stance phase, ℓ_{c1}, while each dotted red line corresponds to a different actuator value during the second half of the stance phase, ℓ_{c2}. The black lines correspond to the case of either $\ell_{c1} = 0$ or $\ell_{c2} = 0$.

Because the reachable $\{x, y, \dot{x}\}$−space obtained by regulating the actuator value (Fig. 5.3b) takes the form of a 2-dimensional surface, increasing the dimension of the reachable state-space requires that we add another degree of freedom. This can be done by varying the touch-down angle θ_{TD}. Then, the reachable space becomes a non-convex set in the 3-dimensional space, as shown in Fig. 5.4.

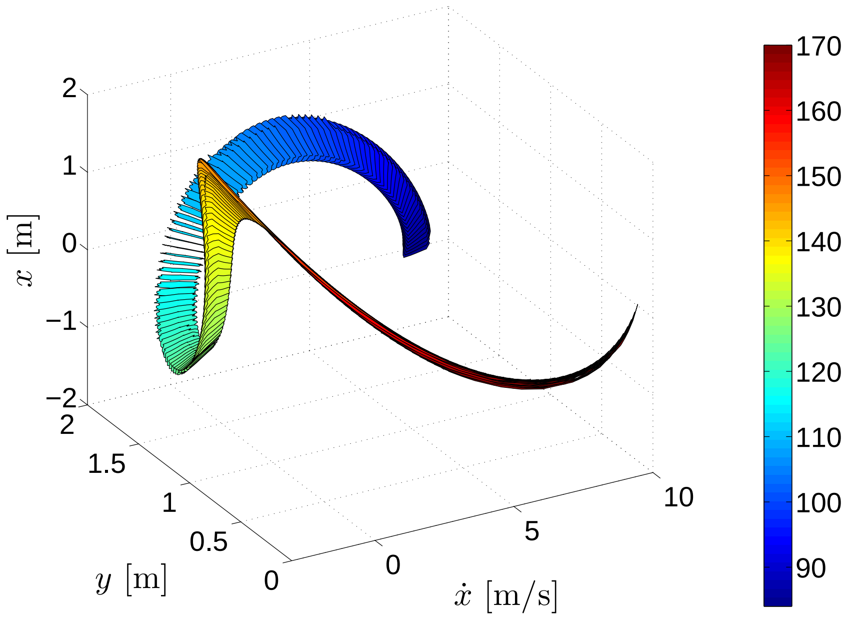


Figure 5.4: Apex states $\{x,\, y,\, \dot{x}\}$ reachable by our control strategy in one jump from the initial apex state $\{0, 1.4\,[\mathrm{m}], 2\,[\mathrm{m/s}]\}$, for varying values of $\theta_{TD} \in [85,\ 170]$ [deg] (colorbar), ℓ_{c1} and $\ell_{c2} \in [-.05,\ 0.5]$. Actuator moves with velocity $v_c = 0.5$ [m/s]. Relative spring stiffness $\gamma = 20$, and leg length $\ell_0 = 1$ [m].

As stated earlier, moving the actuator results in a change of the system's energy. The energy at apex is defined as

$$E_a = Mgy_a + \frac{1}{2}M\dot{x}_a^2. \tag{5.8}$$

Fig. 5.5 shows the change in energy from one initial apex state to the next for different values for the couple (ℓ_{c1}, ℓ_{c2}). As a general rule of thumb, the maximum energy increase is obtained by extending the spring in the first half of the stance phase (i.e., $\ell_{c1} < 0$) and compressing it during the second half (i.e., $\ell_{c2} > 0$), while compressing first, and then extending (i.e., $\ell_{c1} > 0$ and $\ell_{c2} < 0$) results in a maximum energy decrease.

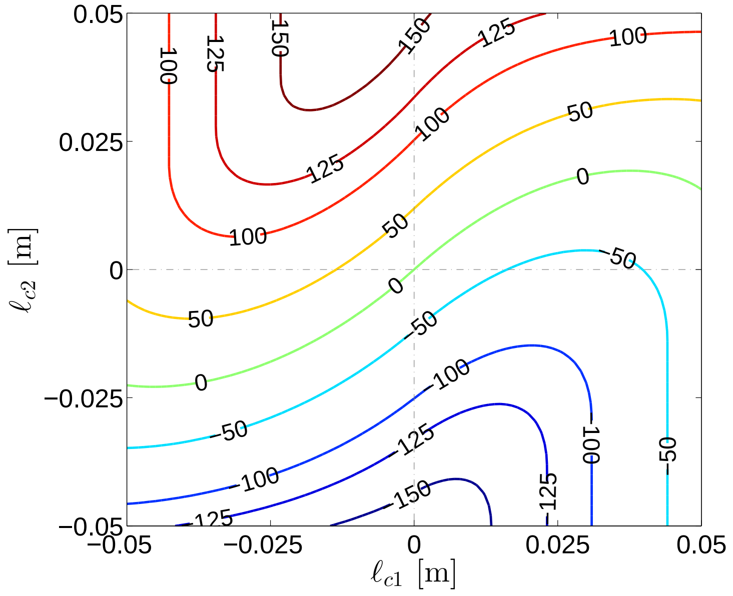


Figure 5.5: The contour plot represents the energy change values $\Delta E = E_{a,2} - E_{a,1}$. $E_{a,1}$ is the energy (5.8) at the initial apex state $y_a = 1.4$ [m] and $\dot{x}_a = 2$ [m/s]. $E_{a,2}$ is the energy (5.8) at each next apex state computed for $\theta_{TD} = 103$ [deg] and varying values of the couple (ℓ_{c1}, ℓ_{c2}).

The lack of body inertia in this simplified model prevents the use of body attitude as additional control action to reach a desired state. However, it is not unreasonable to believe that the strategy we propose can be easily paired with other leg placement and body attitude control strategies. For example, the leg actuator proportional controller proposed in [12] could be replaced by a two-part thrust actuation strategy such as ours, preserving the energy efficient hip actuator controller based on the hip passive oscillations.

5.3 Performance

We study the performance of our monoped hopper in terms of the relative spring stiffness γ as defined in (2.3). Simulations are conducted for $\gamma \in [10, 200]$, using constant values for ℓ_0 and M, and varying k. The initial apex height and velocity have been chosen as a function of the leg length and the time scale $\tau = 1$ [s] to be $y_a \in [\ell_0, 2.5\ell_0]$ and $\dot{x}_a \in [0.5\frac{\ell_0}{\tau}, 3\frac{\ell_0}{\tau}]$, while touch-down angle has been chosen as $\theta_{TD} \in [90, 150]$ [deg]. The spring length at equilibrium has been assumed to be $\ell_{k,0} = 0.5\ell_0$, with a maximum compression of $\ell_{k,min} = 0.05\ell_0$.

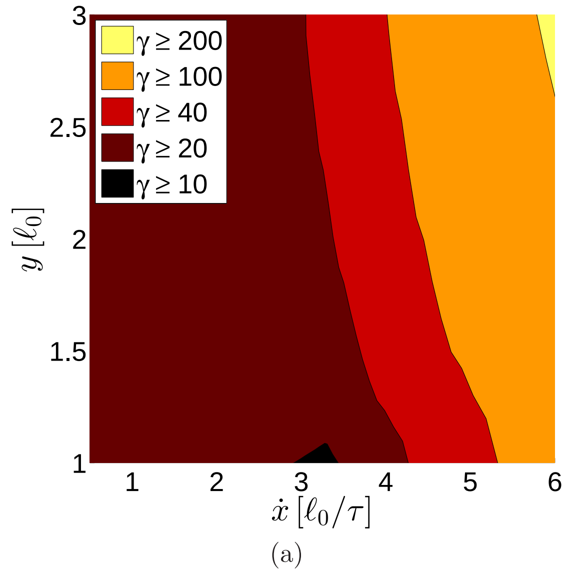
5.3.1 Feasibility

It is important to point out that, depending on the system's parameters and its initial conditions, the actuator displacement required to cancel the nonlinear terms, $\ell_{nl}(t)$, could exceed the maximum actuator displacement and velocity allowed, or could bottom-out the spring. To include such feasibility constraints in our work, we assume the total actuator displacement $\ell_{act}(t) = \ell_c(t) + \ell_{nl}(t)$ must not exceed 10% of the leg length ℓ_0, the maximum velocity $v_{act} = v_c + v_{nl}$ not exceed ℓ_0/τ, and that $\ell_{act}(t)$ at any given time must not bottom-out the spring, i.e., $\ell_k(t) \geq \ell_{k,min}$. The allowable amount of displacement and velocity for the nonlinear part (ℓ_{nl}) and the constant part (ℓ_c) can be allocated in an infinite number of ways. For example, Fig. 5.6a shows the initial apex states that require $\ell_{nl} \leq 0.5\ell_0$ and $v_{nl} \leq 0.5\ell_0/\tau$ to perform a symmetric jump, while Fig. 5.6b considers apex states that require $\ell_{nl} \leq 0.4\ell_0$ and $v_{nl} \leq 0.3\ell_0/\tau$. The simulations have been computed for several values of γ.

Our strategy works well for values of $\gamma \geq 100$ for both allocations considered. For $\gamma \leq 10$ the size of the set of feasible initial conditions is small, posing a heavy limit to the application of our controlling strategy.

5.3.2 Error reduction

Now, we want to test the benefits (in terms of approximation error) of our proposed strategy for cancelling the nonlinear terms via our active SLIP control. We introduce the



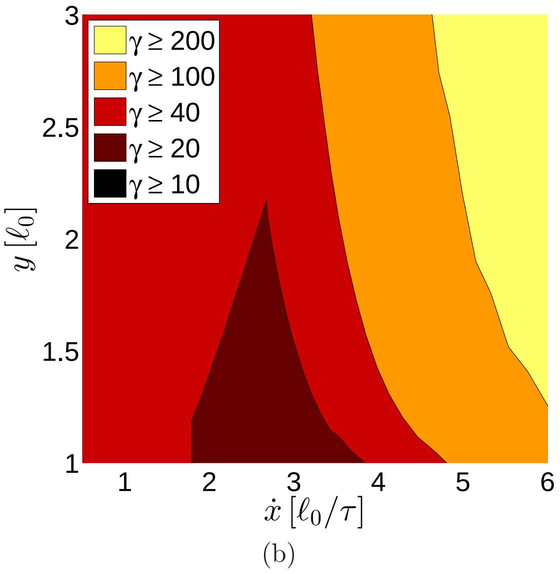


Figure 5.6: Subplot (a) shows the initial apex states that require $\ell_{nl} \leq 0.5\ell_0$ and $v_{nl} \leq 0.5\ell_0/\tau$ to perform a symmetric jump, while subplot (b) considers apex states that require $\ell_{nl} \leq 0.4\ell_0$ and $v_{nl} \leq 0.3\ell_0/\tau$. Different shades refer to different intervals of γ: $\gamma \geq 10$ (black), $\gamma \geq 20$, $\gamma \geq 40$, $\gamma \geq 100$, and $\gamma \geq 200$ (light yellow).

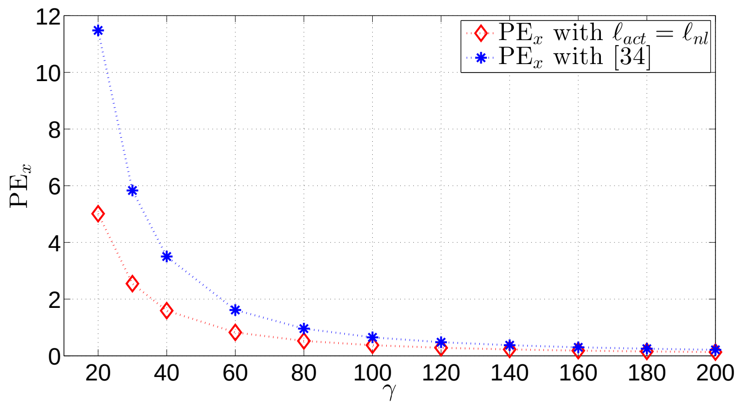
non-dimensional percentage errors of variables x, y and $\dot{x}$, respectively, as:

$$\begin{aligned} PE_x &= 100\frac{\| x - \tilde{x} \|_2}{\ell_0}, \quad PE_y = 100\frac{\| y - \tilde{y} \|_2}{\ell_0}, \\ PE_{\dot{x}} &= 100\frac{\| \dot{x} - \tilde{\dot{x}} \|_2}{\ell_0}\tau, \end{aligned} \tag{5.9}$$

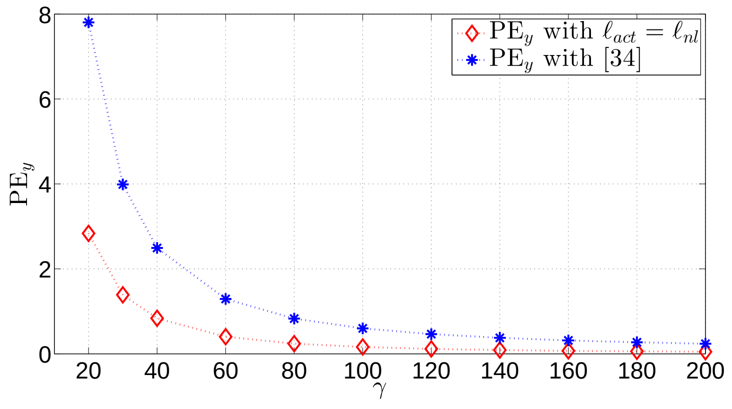
where $\tilde{y}$ and $\tilde{\dot{x}}$ are height and velocity at apex computed via approximation, while y and $\dot{x}$ are the actual apex height and velocity computed using Matlab numerical solver *ode45*, with absolute and relative tolerances set at 10^{-8}. The time constant τ has been chosen to be equal to $\tau = 1$ [s].

First of all, why is it useful to cancel the nonlinear terms, i.e., what is the benefit of having an exact solution for $\ell(t)$? We answer this question by comparing the percentage errors (5.9) for the approximation proposed in [34] versus our approximation with nonlinearity cancellation (5.1). Since the approximation in [34] does not consider actuation, the comparison is performed with respect to our approximation computed with $\ell_{act}(t) = \ell_{nl}(t)$, i.e., $\ell_c(t) = 0$. Fig. 5.7a, 5.7b and 5.7c show the mean percentage errors PE_x, PE_y and $PE_{\dot{x}}$ for symmetric and non-symmetric trajectories, with values of $\gamma \in [20, 200]$ (values of $\gamma < 20$ have not been considered due to their limitation, as shown in Subsection 5.3.1). We can see that our proposed strategy significantly reduces the percentage errors, especially for lower values of γ. This can serve as a starting point for the choice of γ while building a hardware prototype.

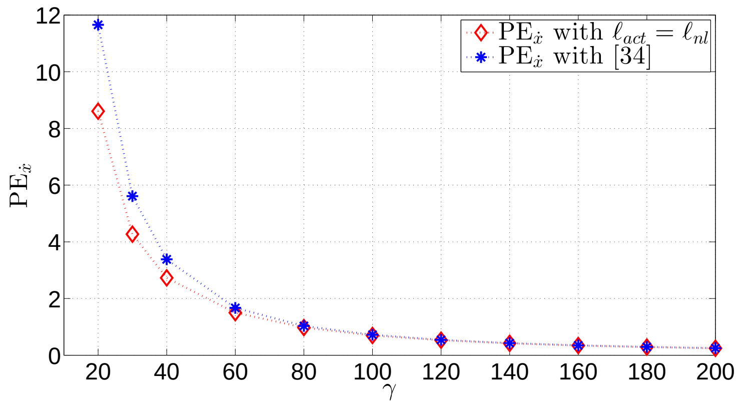
We now compute the percentage errors (5.9) for the actuated SLIP model, with our proposed actuator displacement strategy $\ell_{act}(t)$ as in (5.1.1). Fig. 5.8a, 5.8b and 5.8c show the mean and standard deviation of, respectively, the percentage errors PE_x, PE_y and $PE_{\dot{x}}$, computed for several values of $\gamma \in [20, 200]$ and a set of 60,000 initial conditions.



(a)

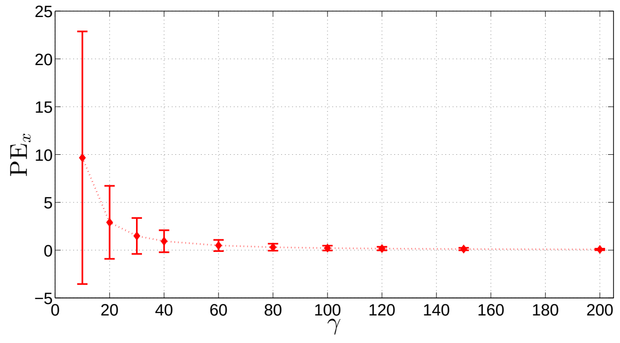


(b)

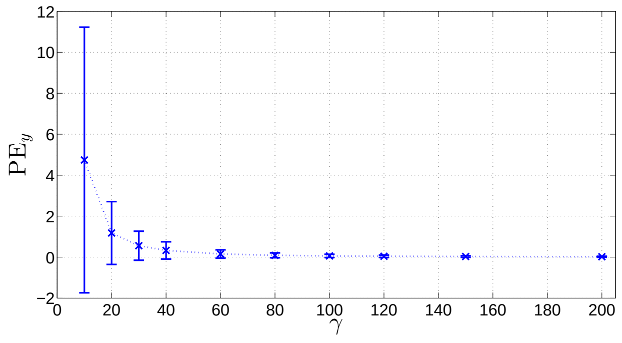


(c)

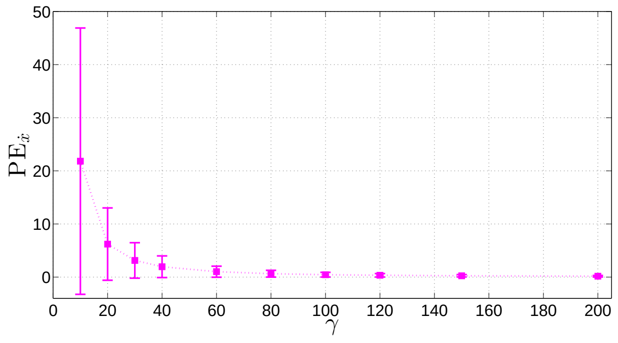
Figure 5.7: The blue stars represent the PEs computed with respect to the stance phase approximation proposed in [34]. The red diamonds represent the PEs computed using our proposed approximation via nonlinearity cancellation. Initial apex conditions have been chosen to be $y \in [\ell_0, 2.5\ell_0]$, $\dot{x} \in [0.3\frac{\ell_0}{\tau}, 3\frac{\ell_0}{\tau}]$, and $\theta_{TD} \in [85, 150]$ [deg]. Maximum actuator displacement and velocity are $max(|\ell_{act}|) = 0.05\ell_0$ and $max(|v_{act}|) = 0.5\ell_0/\tau$.



(a)



(b)

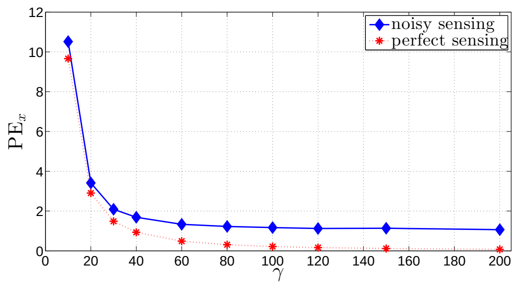


(c)

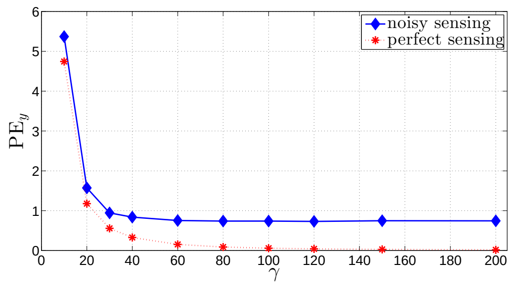
Figure 5.8: Percentage errors PE_x (a), PE_y (b) and $PE_{\dot{x}}$ (c) for our proposed approximation with actuator displacement (5.1.1). The colored symbols represent the mean values of the percentage errors, and the vertical bars the respective standard deviations, computed for a pool of 60,000 initial conditions. The initial apex conditions have been chosen to be $y \in [\ell_0, 1.8\ell_0]$, $\dot{x} \in [0.5\frac{\ell_0}{\tau}, 3\frac{\ell_0}{\tau}]$, and $\theta_{TD} \in [90, 135]$ [deg]. Maximum actuator displacement and velocity are $max(|\ell_{act}|) = 0.1\ell_0$ and $max(|v_{act}|) = 1\ell_0/\tau$, with $\ell_c \in [-0.05\ell_0, 0.05\ell_0]$, and $v_c \in [0.5\ell_0/\tau, 0.5\ell_0/\tau]$. If either the total actuator displacement or velocity exceeds the maximum values allowed, the actuator is assumed to saturate during numerical computation of the stance phase trajectory.

5.3.3 Robustness to sensor noise

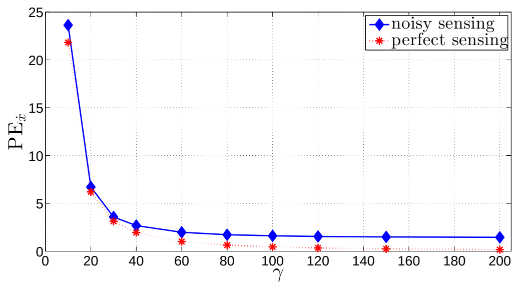
The exact solution of the leg length dynamics via partial feedback linearization and, more generally, the approximation of the overall stance dynamics, depend on the accuracy of the sensors' measurements available. We test the percentage errors (5.9) for the actuated SLIP model when sensor noise is present. In particular, we consider errors in the detection of the initial apex height, position and forward velocity, which translates into errors of the stance phase's initial conditions, and noise on the partial feedback linearization term (5.1). Fig. 5.9a, 5.9b, and 5.9c show the mean percentage errors when sensor noise is present, and compare it with the noiseless case. Sensor noise on the initial apex state is taken from a uniform distribution within $\pm 1\%$ the actual values. Furthermore, we assume the controller is updated at 1000 Hz, and the partial feedback linearization term (5.1) is affected by measurement noise modelled as a Gaussian distribution with zero mean and standard deviations $\sigma = 0.1$ for θ and $\dot{\theta}$, and $\sigma = 0.01$ for ℓ. While the presence of noise degrades the accuracy of the approximation of the system's dynamics, the noise considered still results in a good level of accuracy.



(a)



(b)



(c)

Figure 5.9: Percentage errors PE_x (a), PE_y (b) and $PE_{\dot{x}}$ (c) for our proposed approximation with actuator displacement (5.1.1), with and without sensor noise. The blue diamond dotted lines represent the mean values of the percentage errors with noisy sensors, while the red star solid lines represent the mean values of the percentage errors with perfect sensing, for varying values of γ. PEs are computed for a pool of 60,000 initial conditions, chosen to be $y \in [\ell_0, 1.8\ell_0]$, $\dot{x} \in [0.5\frac{\ell_0}{\tau}, 3\frac{\ell_0}{\tau}]$, and $\theta_{TD} \in [90, 135]$ [deg]. Maximum actuator displacement and velocity are $max(|\ell_{act}|) = 0.1\ell_0$ and $max(|v_{act}|) = 1\ell_0/\tau$, with $\ell_c \in [-0.05\ell_0,\, 0.05\ell_0]$, and $v_c \in [0.5\ell_0/\tau,\, 0.5\ell_0/\tau]$. Sensor noise is modelled as follows. Noise on the initial apex states are taken from a uniform distribution within $\pm1\%$ of the actual values, while the partial feedback linearization term (5.1) noise is modelled as a Gaussian distribution with zero mean and standard deviations $\sigma = 0.1$ for θ and $\dot{\theta}$, and $\sigma = 0.01$ for ℓ.

5.3.4 Cost of Transport

At each time t during the stance phase, it is possible to compute the total work done by the spring as:

$$W_{spring}(t) = \int_{t_{TD}}^{t} k(\ell_k(t) - \ell_{k,0})\dot{\ell}(t)dt. \tag{5.10}$$

Note that $\dot{\ell}(t) = \dot{\ell}_k(t) + \dot{\ell}_{act}(t)$, where $\dot{\ell}_{act}(t)$ is the actuator's velocity.

The energy stored in the spring is

$$E_k(t) = \frac{k}{2}(\ell_k(t) - \ell_{k,0})^2.$$

If there is no actuation, the energy stored in the spring is equivalent to the work done by the spring, i.e., $E_k(t) = W_{spring}(t)$. However, when actuation is present, it is responsible for part of the energy stored in the spring: $E_k(t) = W_{spring}(t) + W_{act}(t)$, where $W_{act}(t)$ is the work done by the actuator. Then

$$\begin{aligned} W_{act}(t) &= E_k(t) - W_{spring}(t) \\ &= \frac{k}{2}(\ell_k(t) - \ell_{k,0})^2 - \int_{t_{TD}}^{t} k(\ell_k(t) - \ell_{k,0})\dot{\ell}(t)dt \\ &= \dots \\ &= \int_{t_{TD}}^{t} P_{act}(t)dt + \frac{k}{2}(\ell_k(t_{TD}) - \ell_{k,0})^2, \end{aligned}$$

and $P_{act}(t) = -k(\ell_k(t) - \ell_{k,0})\dot{\ell}_{act}(t)$ is the power required by the actuator.

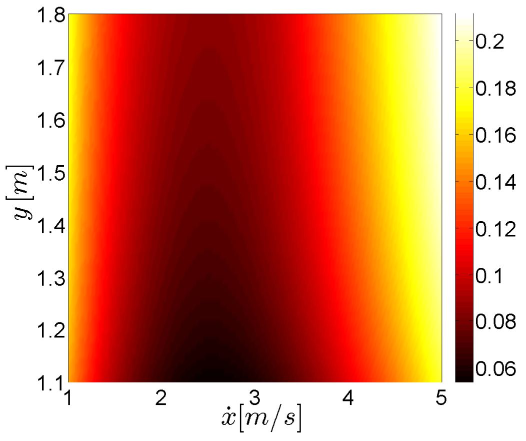
The dimensionless specific cost of transport is defined as:

$$\cot \triangleq \frac{W_{act,tot}}{Mgd}, \tag{5.11}$$

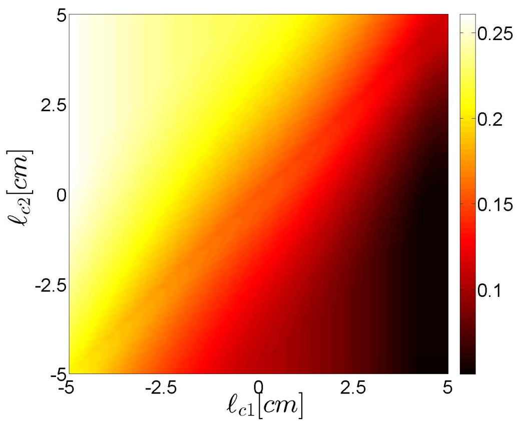
where d is the total distance travelled by the system, and $W_{act,tot}$ is the total work done by the actuator for the entire stance phase. To avoid power regeneration, it is appropriate to compute $W_{act,tot}$ as the integral of the unsigned power:

$$W_{act,tot} = \int_{t_{TD}}^{t_{TO}} |P_{act}(t)|dt + \frac{k}{2}(\ell_k(t_0) - \ell_{k,0})^2.$$

In our case, the total cost of transport can be computed for one jump (apex-to-apex), and therefore the distance travelled d is equal to the distance travelled from one initial apex state to the next. The work done by the actuator is a combination of the work done to cancel the nonlinearity as in (5.1), and the work required to move the actuator at the desired constant values $\ell_{c,1}$ and $\ell_{c,2}$. Fig. 5.10a shows the cost of transport computed for symmetric jumps starting at different initial conditions, where the only actuation considered is the one due to ℓ_{nl}, while $\ell_{c,1}$ and $\ell_{c,2}$ are set to zero. For varying initial conditions, cancelling the non-linear terms in equation implies a cost of transport cot < 0.21. One should note that the considered jumps are, in general, passively non-symmetric, and therefore the cost of transport computed includes the cost to drive many non-symmetric jumps to be symmetric. Fig. 5.10b shows instead the cost of transport for an assigned initial conditions, considering both the effect of ℓ_{nl} and of varying values of ℓ_{c1} and ℓ_{c2}. Also in this case the cost of transport is cot < 0.26.



(a)



(b)

Figure 5.10: Fig. 5.10a illustrates the cost of transport computed for symmetric jumps starting at different initial conditions, where the only actuation considered is the one due to ℓ_{nl}, while $\ell_{c,1}$ and $\ell_{c,2}$ are set to zero. Fig. 5.10b shows instead the cost of transport for an assigned initial apex state, $y_a = 1.3$ [m], $\dot{x}_a = 2$ [m/s], and $\theta_{TD} = 78$ [deg]. Both the effect of ℓ_{nl} and of varying values of ℓ_{c1} and ℓ_{c2} are considered.

5.4 Control Actions: Online Computation of Optimal Parameters

We start by noting that, in order to increase the dimension of the reachable state-space to a 3-dimensional space, we need to add another degree of freedom: the touch-down angle θ_{TD}, as illustrated in Fig. 5.4. In the formulation of our strategies, we will then consider three control parameters: the actuator values ℓ_{c1} and ℓ_{c2}, and the touch-down angle θ_{TD}.

The main advantage of using an approximation of the stance phase versus its numerical solution is a reduction of computational time, which increases the practicality of performing online control actions. To give an example, on a representative pool of 60,000 initial conditions (apex state and touch-down angle), we computed the average time to simulate the stance phase using Matlab's function *ode45* versus an analytical approximation. The calculations were performed on a Microsoft Windows based computer (Intel Core i7 eight core processor CPU, 2.80 GHz) using Matlab version R2012a. While the average time for *ode45* was 0.0259 [s], the average time for an approximate solution was $9.4243 * 10^{-5}$ [s]: a decrease in computation time of over 250 times.

We start from this preliminary remark to introduce our proposed control actions.

5.4.1 Controlling the $\{y, \dot{x}\}-$state space

In this subsection we will focus our attention on controlling the height and velocity at apex only, disregarding the forward position. Therefore, the apex state becomes a two-dimensional vector $\boldsymbol{s} = \{y_a, \dot{x}_a\}$. We use a modified version of the Matlab function *fminsearch* (which optimizes constrained problems using the Nelder-Mead algorithm).

At any current apex state $\boldsymbol{s}_n = \{y_n, \dot{x}_n\}$, we compute the values for the touch-down angle θ_{TD} and the two actuator values ℓ_{c1} and ℓ_{c2} that minimize in one jump the distance to a desired apex $\boldsymbol{s}_{des} = \{y_{des}, \dot{x}_{des}\}$. The optimization problem is defined to be constrained due to the bounds on the values taken by the touch-down angle and the actuator displacement.

At each step, n, the cost function to be minimized, $J(n)$, is defined as:

$$J(n) = 100\sqrt{\frac{(y_{n+1} - y_{des})^2}{\ell_0^2} + \frac{\tau^2(\dot{x}_{n+1} - \dot{x}_{des})^2}{\ell_0^2}}, \tag{5.12}$$

which expresses the percentage distance from the next apex state $\boldsymbol{s}_{n+1} = \{y_{n+1}, \dot{x}_{n+1}\}$ to the desired one, $\boldsymbol{s}_{des}$.

Note that the average time to solve the optimization problem has been computed to be 0.05 [s], which is much faster than the average ballistic apex-to-apex time. In particular, the shortest apex-to-apex time corresponds to a take-off and touch-down angle $\theta_{TO} = 90$ [deg] and $\theta_{TD} = 90$ [deg]. In order to guarantee the apex-to-apex time to be smaller than 0.05 [s], it is required for the apex height to be $y_{ap} \geq 1.0123\ell_0$.

5.4.2 Adaptive control for steady-state locomotion

Due to errors in the approximation (see Fig. 5.8a, 5.8b and 5.8c), the touch-down angle and actuator values that minimize the cost function in Equation (5.12) may drive the system to an apex state that differs from the desired one by a certain amount. Therefore, the system will converge to an apex state that is not the desired one. In order to address this problem, we propose here a strategy to reduce such steady-state error, driving the system closer to the desired apex state over time. The strategy is summarized as follows.

Let us start from an initial apex state, $\{y(0), \dot{x}(0)\}$, and let us assume we want to ultimately reach the value $\{y_{des,0}, \dot{x}_{des,0}\}$. At each step n, we define the error between the actual and the approximate state as

$$\Delta y_n = y_n - y_{des,0}, \tag{5.13}$$

$$\Delta \dot{x}_n = \dot{x}_n - \dot{x}_{des,0}, \tag{5.14}$$

where y and $\dot{x}$ are the actual apex height and velocity of the system. At the $n-th$ step, we update the desired value for the next step $y_{des,n+1}$ and $\dot{x}_{des,n+1}$ to be

$$y_{des,n+1} = y_{des,n} - \sigma_1 \Delta y_n$$

$$\dot{x}_{des,n+1} = \dot{x}_{des,n} - \sigma_2 \Delta \dot{x}_n.$$

The proportional gains σ_1 and σ_2 are chosen to be $0 < \sigma_1 < 1$, $0 < \sigma_2 < 1$. The desired apex state is updated at each step, until the errors $\Delta y = 0$ and $\Delta \dot{x} = 0$, and the system reaches an equilibrium. Fig. 5.11a, 5.11b, 5.11c and 5.11d show, respectively, the percentage distance $J(n)$ from the desired apex state after $n = 1$, 3, 6 and 9 jumps, on flat terrain, for $\gamma = 20$ and $\sigma_1 = \sigma_2 = 0.8$. Our proposed controller reduces the percentage distance J after 9 jumps from a maximum of about 70% to a maximum of about 0.3%. Fig. 5.12a, 5.12b, and 5.12c show an example of error reduction for 3 different initial conditions: the error converges to zero.

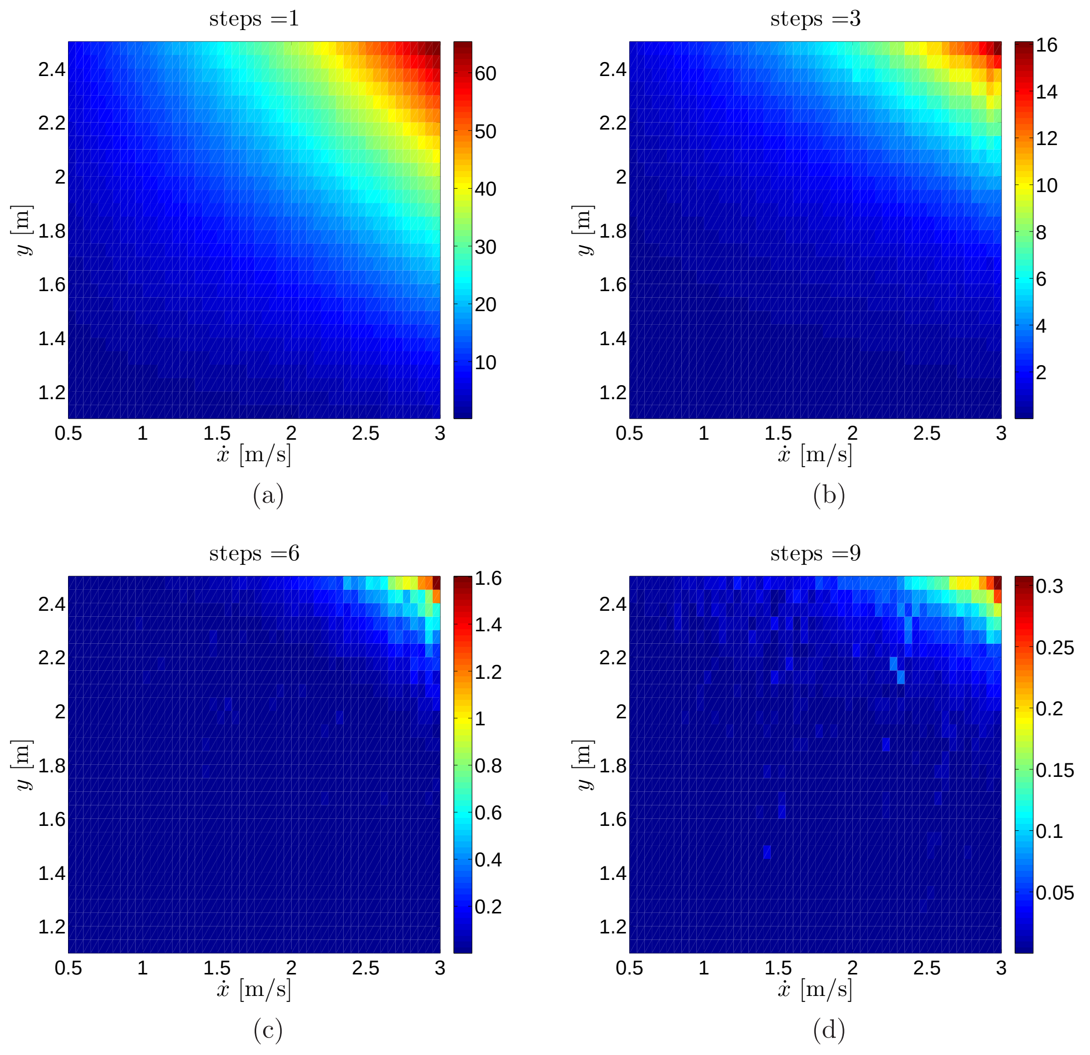
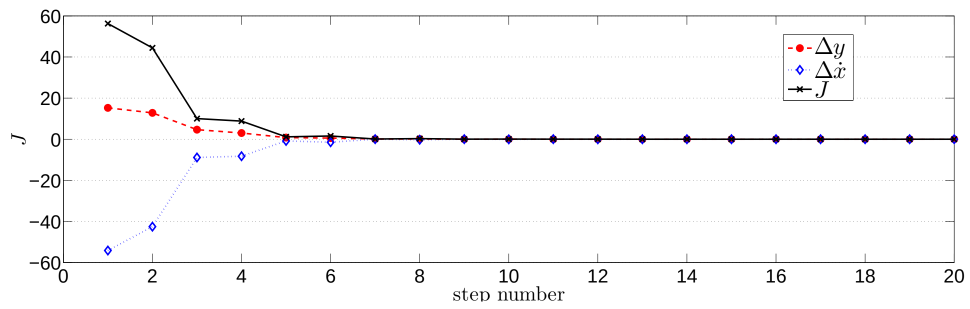
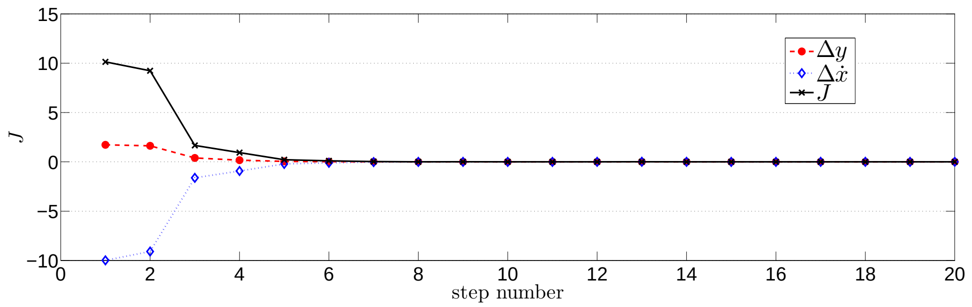


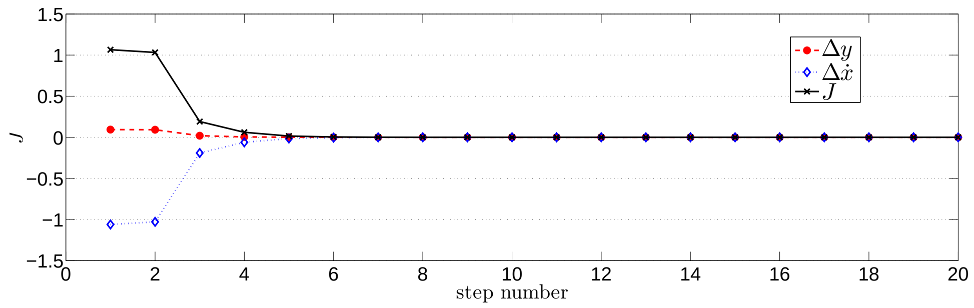
Figure 5.11: These plots show the percentage error J after 1 (a), 3 (b), 6 (c) and 9 (d) jumps, for $\gamma = 20$. The x-axis and y-axis represent the apex velocity and the apex height, respectively. Parameters σ_1 and σ_2 have been chosen to be equal to 0.8.



(a)



(b)



(c)

Figure 5.12: Evolution at each jump of the apex errors Δy (5.13) and $\Delta \dot{x}$ (5.14), and the cost function J (5.12) for inital and desired apex state $s = \{y_a,\, \dot{x}_a\} = \{2.4,\, 2.8\}$ (a), $\{1.8,\, 1.6\}$ (b), and $\{1.3,\, 0.8\}$ (c). As we can see, the errors converge to zero.

5.4.3 Foothold Placement Control

One of the key problems in legged locomotion on different kinds of terrain is to determine whether the foothold, i.e., the point at which the foot comes in contact with the terrain, is considered safe. For example, we can imagine the case of a terrain where only a specific set of N footholds, $x^f_{des,i}$, $i = \{1, \dots N\}$, is allowed, and everything else has to be avoided, as studied in [15]. We can then plan a trajectory to follow based on the knowledge we have of the terrain. In particular, at each jump i we want to find a sequence of control actions θ_{TD}, ℓ_{c1}, and ℓ_{c2}, to minimize the distance between the desired footholds, $x^f_{des,i}$ and the actual landing of the foot, x^f:

$$d_i = \parallel x^f_i - x^f_{des,i} \parallel_2,$$

where the position of the foot at touch-down is computed as:

$$x^f = x_a + \dot{x}_a \sqrt{\frac{2}{g}(y_a + \ell_0 \cos\theta_{TD})} + \ell_0 \sin\theta_{TD}.$$

The foothold error is dependent on the planning *horizon*, i.e., the number of jumps we can pre-compute. For a planning horizon of length N, we can write the following cost function to minimize:

$$J^f_N = \sum_{i=1}^{N} d_i^2.$$

However, to ensure that the hopper maintains a certain desired height, $y_{des,i}$, with respect to the terrain, we can modify the above cost to be:

$$J^f_N = \sum_{i=1}^{N} d_i^2 + w_y \sum_{i=1}^{N} (y_i - y_{des,i})^2,$$

where $w_y \in [0,\ 1]$ is the associated weight. Furthermore, to ensure that the optimal solution at the N-th step does not result in a jump with negative forward velocity, another element can be added to the cost function, and we obtain:

$$J_N^f = \sum_{i=1}^{N} d_i^2 + w \sum_{i=1}^{N} (y_i - y_{des,i})^2 + w_{\dot{x}} (\dot{x}_N - \dot{x}_{N-1})^2, \tag{5.15}$$

with $w_{\dot{x}} \in [0,\ 1]$.

Ideally, one would want an infinite planning horizon: $N = \infty$. In fact, the longer the horizon, the better the performance of the optimized problem (e.g., see Fig. 5.13). However, the horizon length affects the computation time required to plan the desired

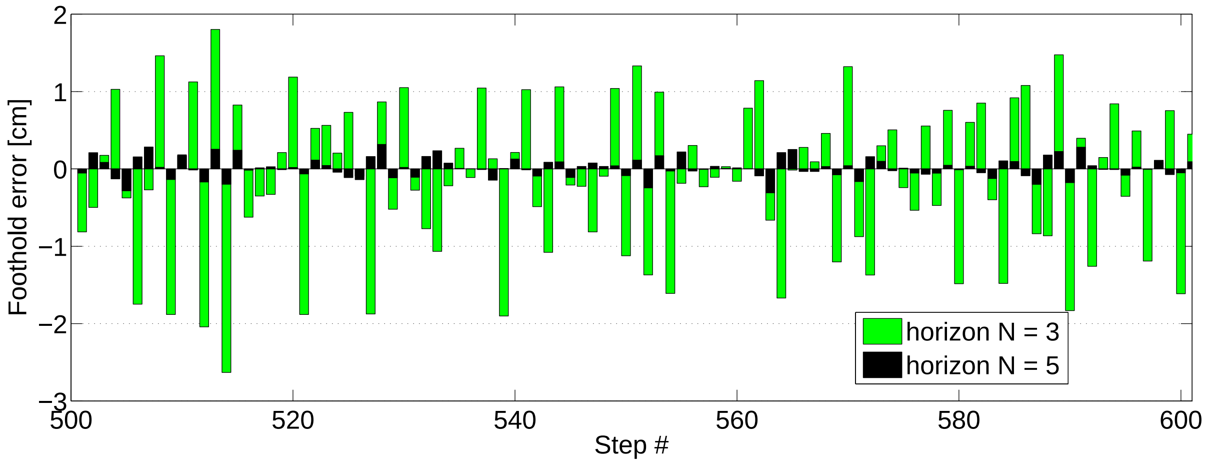
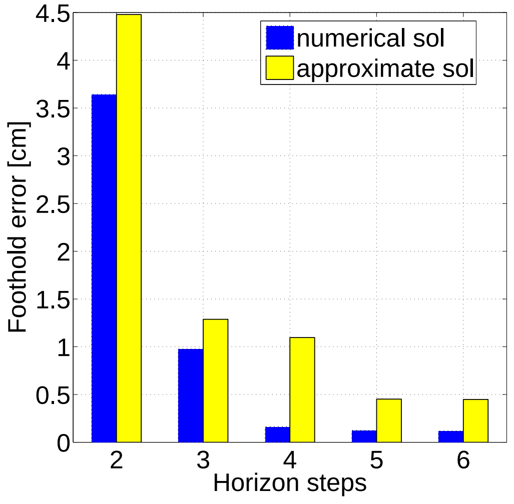
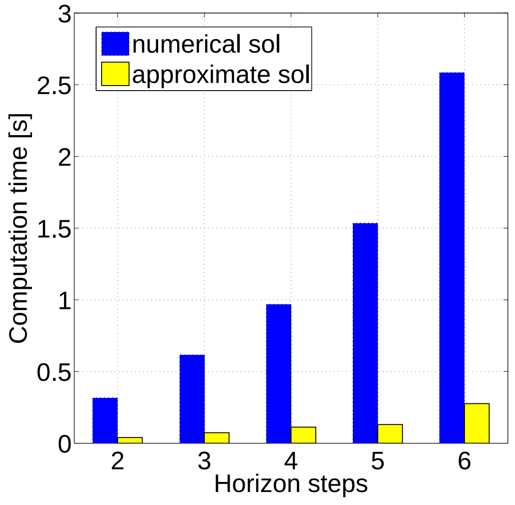


Figure 5.13: Example of foothold error for 100 jumps for the case with horizon $N = 3$ (green bars) and $N = 5$ (black bars). As expected, a higher planning horizon gives better performances in terms of error.

trajectory online. This is where having an approximation for the stance phase becomes highly beneficial: in fact, being able to compute the optimal path via approximation significantly reduces the computational time, and as a consequence, it is possible to extend the planning horizon. However, one should keep in mind that the approximation, as such, carries an error: there is then a trade-off between horizon length/computation time, and foothold error. Fig. 5.14a and 5.14b show an example of trajectory planning on flat terrain, comparing a case in which the optimization problem has been solved using the approximate solution for the stance phase, and a case in which the numerical solution has been used. As expected, the minimization via numerical solution has a smaller error but a higher computational time compared to the minimization via approximation (on the same horizon length N). To be able to perform an online computation of the optimal control parameters, the computation time needs to be much smaller than the average time during flight: $t \ll 0.5$ [s]. Then, the only viable option for the numerical solution case is $N = 2$, with average foothold error over $10,000$ jumps of 3.64 [cm]. For the approximate solution case, instead, one can use $N = 6$, and the mean foothold error is only 0.45 [cm], i.e., 8 times smaller than the numerical case.



(a)



(b)

Figure 5.14: These plots show the mean foothold error (a) and the mean computation time (b) over 10,000 jumps, for trajectory planning, for different horizon lengths. The terrain has been chosen to be flat, with footholds drawn from a uniform distribution on the open interval $(0.3\ell_0, \ell_0)$. The blue dotted bars refer to the optimization using the numerically computed solution, while the yellow bars refer to the optimization computed using the approximate solution.

5.5 Applications

In this section we show the performance of our controller to perturbations in terrain. In particular, we test recovery for perturbations on the terrain height, and we show an example of running on rough terrain.

The parameters and initial conditions used in our simulations are defined in Table 6.1, and are based on biological data for a typical human. In [3], the relative spring stiffness γ was found to be very similar for different gaits, such as running and hopping, of various animals. In particular, values were computed to be between 7.1 and 14.6 for runners, and 7.7 and 13.6 for hoppers. Because of the limitations shown in Fig. 5.6a and 5.6b, for our simulations we use $\gamma = 20$, which is slightly higher than the average biological value, but it also matches a hardware prototype currently under development in our lab. During simulations, if either the total actuator displacement $\ell_{act} = \ell_{nl} + \ell_c$ or the total actuator velocity $v_{act} = v_{nl} + v_c$ required were exceeding the maximum values allowed, the actuator was assumed to saturate its maximum allowed value for ℓ_{act} or v_{act}, respectively. Furthermore, to acknowledge the time to solve the optimization problem, we limit our simulations to initial apex heights and touch down angles that corresponds to a time during flight $t_f \geq 0.15$ seconds.

5.5.1 Recovery from perturbations

We consider the set of initial conditions in Table 6.1. We test the recovery capabilities of our controller when the active SLIP encounters an unexpected (positive or negative) perturbation on the terrain height of up to 50% of the leg length ℓ_0.

At each apex state, we use the strategy in 5.4.1 to compute the optimal values for ℓ_{c1}, ℓ_{c2} and θ_{TD} for flat terrain. Once the leg touches the ground with the computed touch-down angle and the desired ℓ_{c1}, the strategy in 5.4.1 is simulated again during

Simulation Parameters	
$g =$	9.81 m/s^2
$M =$	80 kg
$\ell_0 =$	1 m
$\ell_{k,0} =$	0.5 m
$\ell_{k,min} =$	0.05 m
$\ell_{act} \in$	$[-0.1, 0.1]$ m
$\ell_c \in$	$[-0.05, 0.05]$ m
$v_{act} \leq$	1 m/s
$v_c =$	0.5 m/s
$y \in$	$[1, 2.5]$ m
$\dot{x} \in$	$[0.5, 3]$ m/s
$\gamma =$	20

Table 5.1:

the first half of the stance phase, this time to compute only ℓ_{c2} to take into account the encountered perturbation on the terrain height. Note that the search of this second value takes on average 0.01 [s], which is much smaller than the average time required for the first half of the stance phase, and therefore it can be realistically implemented. Fig. 5.15a, 5.15b, 5.15c, and 5.15d show the number of jumps necessary for the system in order to return within 1% of the initial apex state, in the case of positive or negative perturbations. Our controller is robust to perturbations of varying magnitudes, with ability to recover in up to 8 jumps. Clearly, these results are not only due to the control strategy, but are partly affected by actuator limits. One can expect the number of jumps to recover to increase or decrease if the actuator limits are more or less stringent.

Fig. 5.16a and 5.16b show an example of apex recovery for an unforeseen drop of magnitude 50% of the leg length.

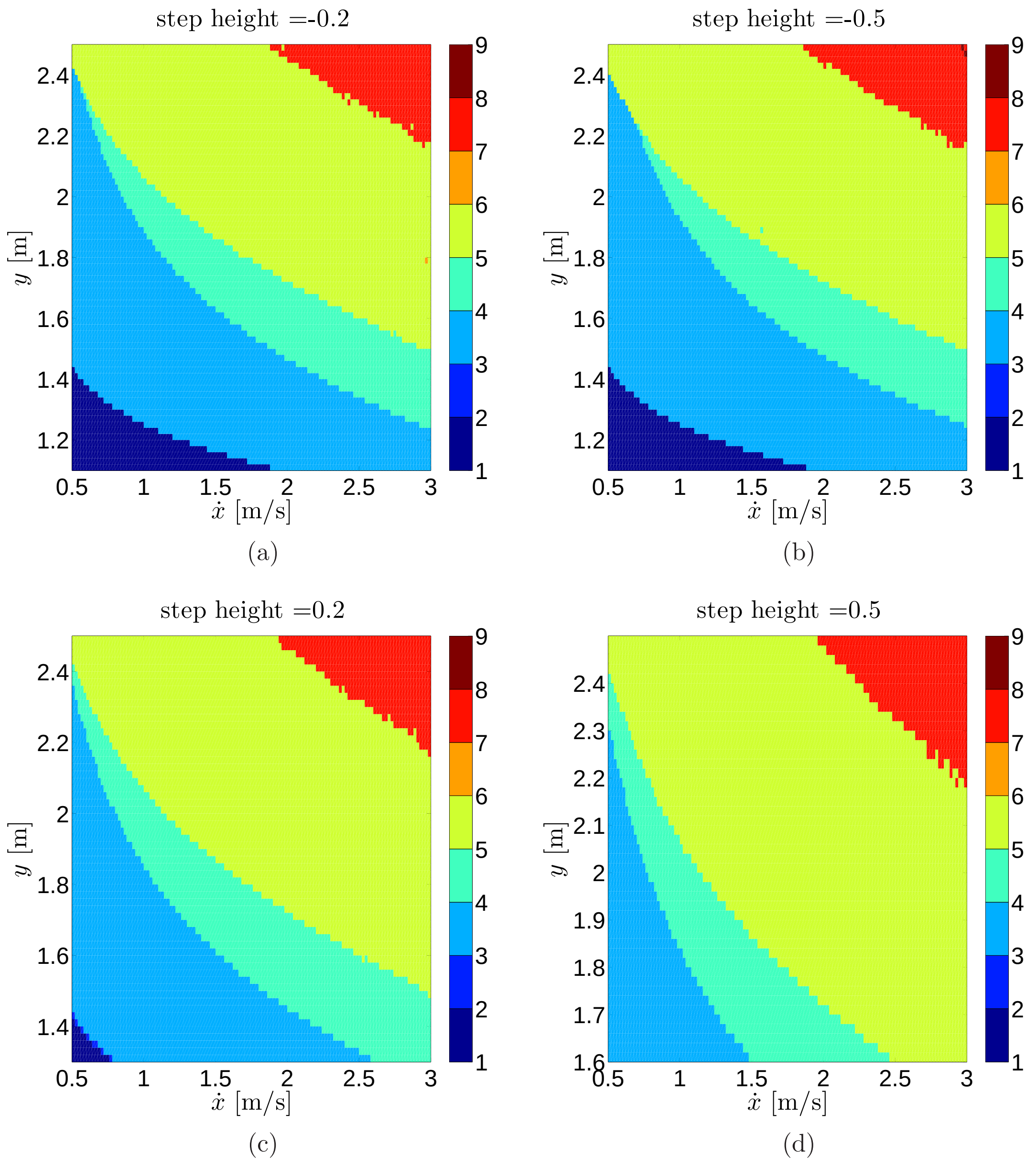
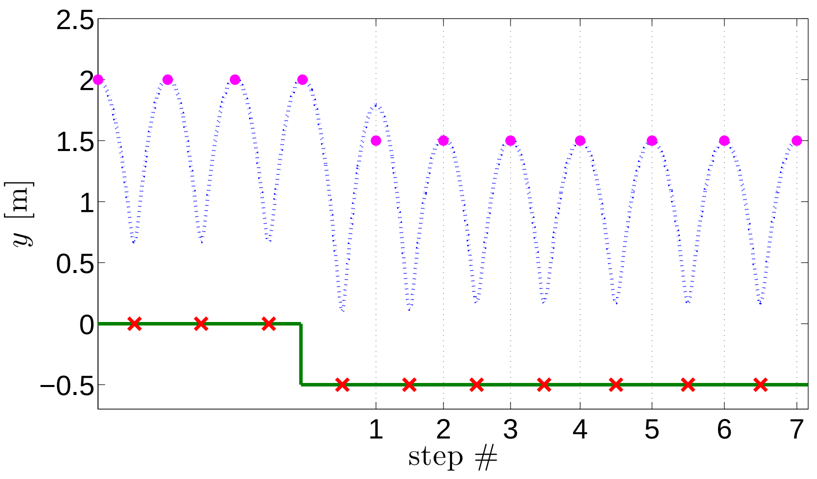
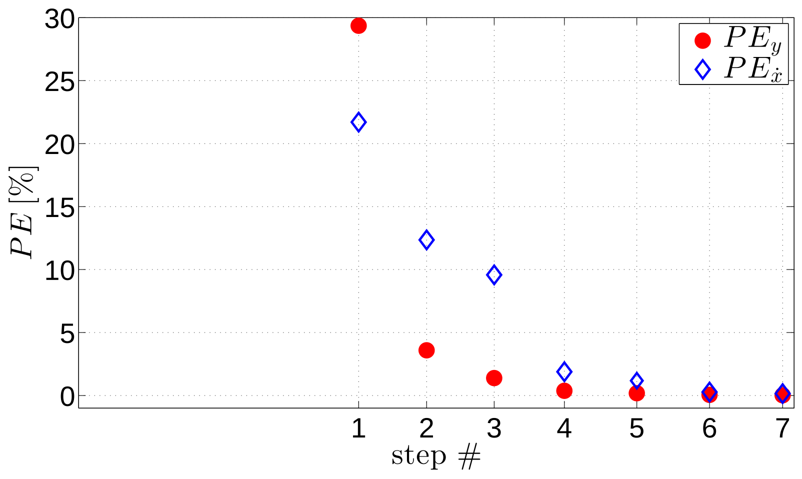


Figure 5.15: Number of jumps (colorbar) to reach 1% of desired value, for terrain perturbation of magnitude (a) -0.2 [m], (b) -0.5 [m], (c) 0.2 [m], (d) 0.5 [m]. Simulation parameters are chosen as per Table 6.1. Note that in case of a positive terrain perturbation (subplots (c) and (d)), the apex height y has been chosen to leave enough room to the leg to swing during flight without colliding with the terrain.



(a)



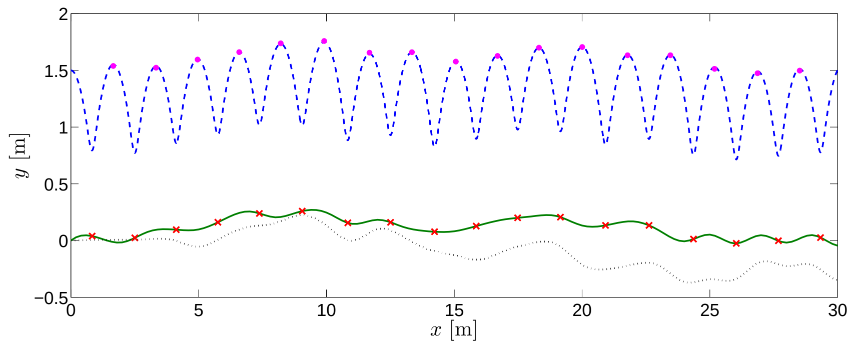
(b)

Figure 5.16: Plot (a) shows the trajectory of the active SLIP for initial apex state $y = 2$ [m], $\dot{x} = 1.3$ [m/s], and negative terrain perturbation of magnitude 0.6 [m]. The blue dotted line represent the trajectory of the mass, the purple circles the desired apex height. The green solid line is the terrain height and the red x marks are the landing points of the foot. Plot (b) shows the percentage error of apex height (red circle) and velocity (blue diamond) at each jump after the drop. We can see that the controller reaches and remains within 1% of the original apex state in 5 jumps.

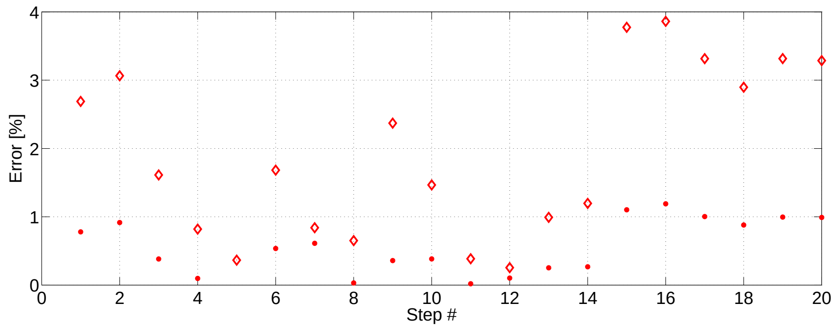
5.5.2 Hopping on rough terrain

In this section we show an example of the active SLIP model hopping on rough terrain. Additionally, we assume that the estimates of upcoming terrain height are faulty, and we want to maintain the same forward velocity and the same distance from the terrain with respect to the last jump. As we can see from Fig. 5.17a, 5.17b, and 5.17c, the system is able to successfully hop on a rough terrain with a maximum magnitude landing height of 0.26 [m] and a maximum perturbation of 0.46 [m], i.e., 46% of the leg length. Consistent with what is in Fig. 5.8b and 5.8c, the error on $\dot{x}$ is on average higher than the error on y, and in this example they both do not exceed 4%. Note that, since the terrain height varies continuously, the energy of the system varies at each jump.

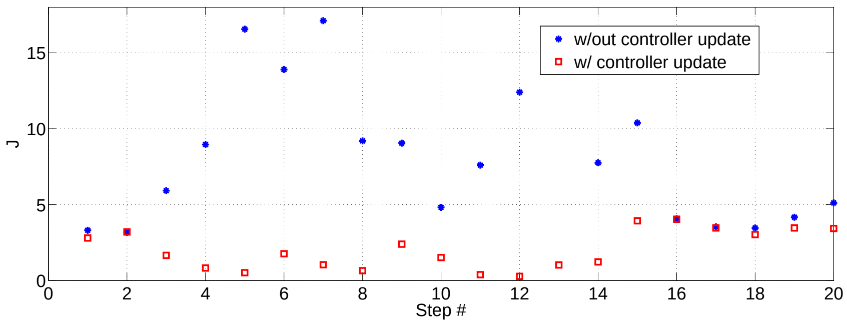
The strategy of recomputing ℓ_{c2} during the first portion of the stance phase after encountering a perturbation can dramatically improve the performance of our controller. Indeed, performance on the same terrain has been studied using the computation during flight only versus with controller update via recomputing of ℓ_{c2} during the first half of stance. Fig. 5.17c shows that the cost function J (computed as in (5.12)) is significantly smaller than the case without controller update.



(a)



(b)



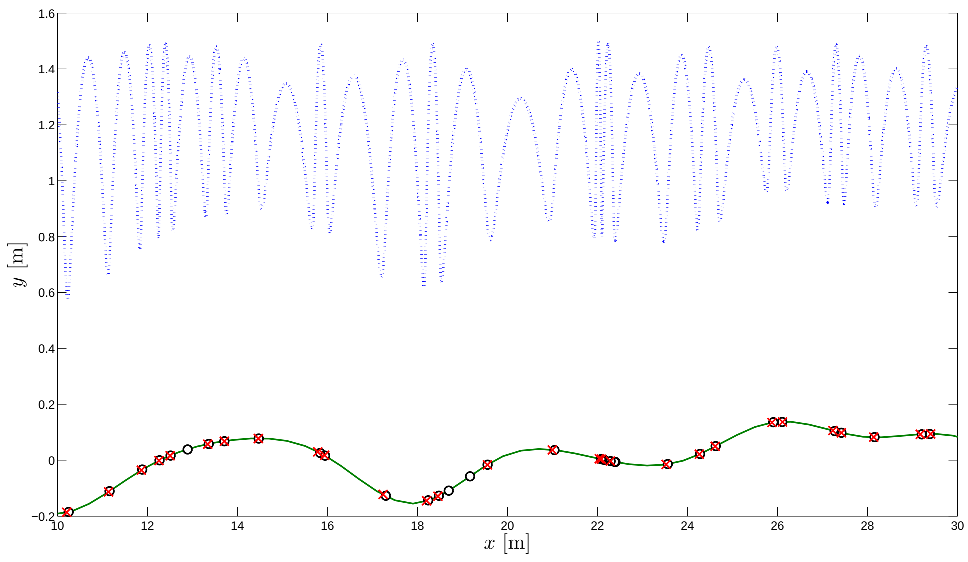
(c)

Figure 5.17: Plot (a) shows the trajectory of the active SLIP hopping on a random–generated rough terrain. The controller acts to maintain a constant apex height with respect to the terrain of $y = 1.5$ [m], and a constant forward velocity of $\dot{x} = 2$ [m/s]. The blue dotted line represent the trajectory of the mass, the purple circles the desired apex height. The black dotted line is the expected terrain height, while the green solid line is the actual terrain height and the red x marks are the landing points of the foot. Plot (b) shows the percentage error of apex height PE_y (circle) and velocity $PE_{\dot{x}}$ (diamond) at each jump. Plot (c) shows the cost J as in (5.12) for the case without (blue star) and with (red square) controller update during the first part of the stance phase.

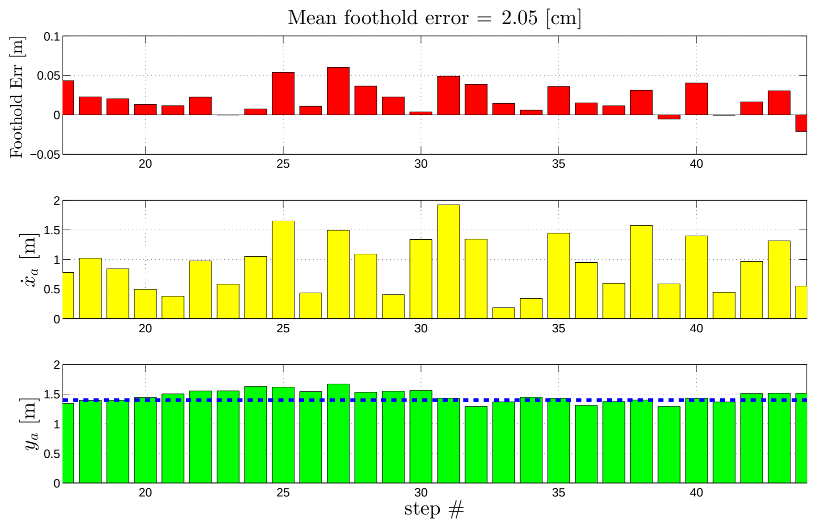
5.5.3 Landing on feasible footholds

We show here an example of hopping on a set of predefined footholds.

The set of footholds on an uneven terrain have been generated taking points from a uniform distribution between 0.6 [m] and 1.2 [m]. The cost function the be minimized is defined in Eq. (5.15), with weights $w_y = 0.8$ and $w_{\dot{x}} = 1$. We used a least-square algorithm to find the optimal solution for a planning horizon of $N = 4$ steps, for a total of 50 steps. Note that, according to the specifics of our minimization algorithm, the hopper was allowed to skip some footholds if pertinent. Fig. 5.18a shows a close-up of the output trajectory, while Fig. 5.18b shows at each step the foothold error (i.e., the distance from the desired to the actual foothold), the forward velocity and the apex height. The desired apex height with respect to the terrain was set as $y_{des} = 1.4$ [m]. As we can see, over 50 steps, the mean foothold error was 2.05 [cm].



(a)



(b)

Figure 5.18: Plot (a) shows the trajectory of the point mass (blue dotted line) on a terrain (green line). The white circles are the desired footholds, while the red x marks are the actual placements of the foot. As noted in the text, our minimization algorithm allows for some footholds to be skipped. Plot (b) from top to bottom shows the foothold error (red bars), the forward velocity (yellow bars) and the apex height with respect to the terrain height (green bars). The dotted blue line represents the desired apex height.

5.5.4 Extension to general planar hopper models

The actuated SLIP model, while encompassing the general behavior of hopping spring-mass systems, is a highly idealized model. It is then logical to ask how the strategy presented in this work would fare when applied to models that match more accurately real hardware prototypes. To answer this question, the two-step strategy has been tested on a more realistic hopper model, shown in Fig. 5.19, similar to the hopper studied in [10]. The leg has mass m_l and moment of inertia, J_l, while the body has mass and inertia M and J. Body and leg are connected by a joint at the hip, where an actuator can apply a torque, τ_{hip}. Equations of motion for this system can be easily computed, for example via

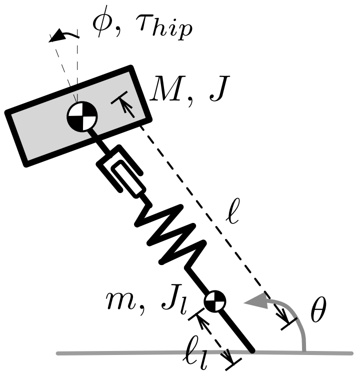
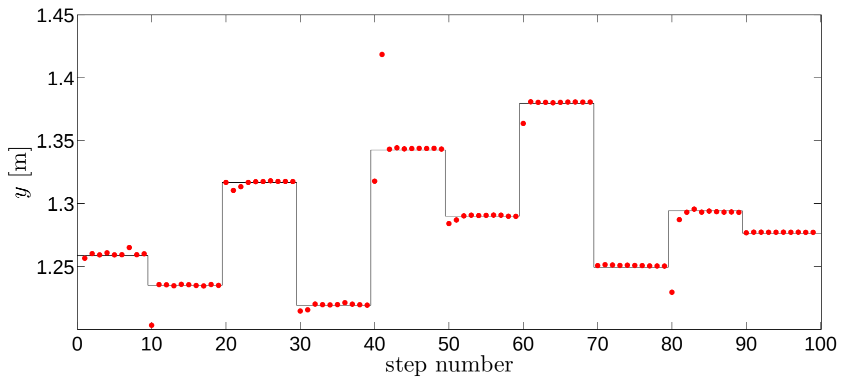


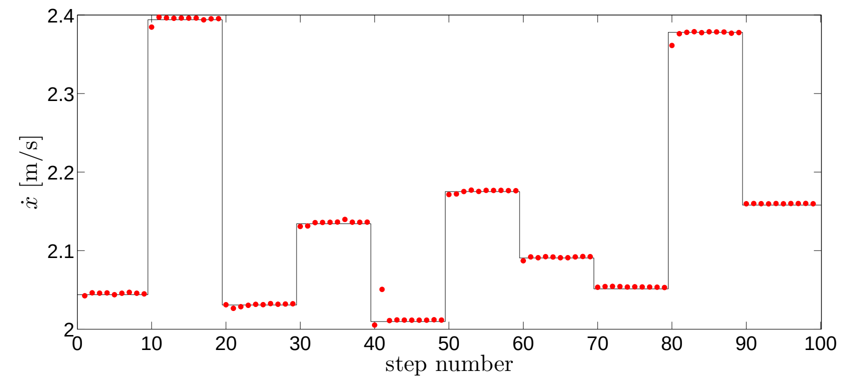
Figure 5.19: Planar hopper model. The body has mass M and moment of inertia J, and center of mass at the hip. ϕ is the body angle. The leg has unsprung mass m_l and moment of inertia J_l, and its center of mass is located at distance ℓ_l from the foot. ℓ is the total leg length, and θ the angle that the leg forms with respect to the ground during contact. The spring has stiffness k. ℓ_{act} denotes the length of the series elastic actuator. An actuator at the hip can apply a torque τ_{hip} between the body and the leg.

the Lagrangian method. During flight, a PD controller is applied to the the hip motor to position the leg at the desired angle, and the center of mass of the system follows a ballistic trajectory. The effect of the energy loss at impact on the velocities of the leg length and leg angle is approximated to determine the initial (post-impact) conditions of the stance phase. During stance, the rotation of the body affects the dynamics of the leg angle, while the dynamics of the leg length is not affected and can be solved as in Section 5.1. To keep a body attitude, i.e., to prevent the body from tipping forward or backward during stance, a torque is applied at the hip. Therefore, an approximation of the equation of motion for the angular displacement over time, $\theta(t)$, is given by the approximation computed in Section 5.1, with the addition of a term that approximates the effect of the motion of $\phi(t)$. Note that this is still an approximation, and we exploit the actuator at the hip by applying an additional torque to control the system to follow the approximated trajectory. The series elastic actuator is modelled as a function of input current, with a damping and friction term. Details of the approximation, controllers and parameter values are illustrated in Appendix A.

The control strategy has been tested on a flat terrain. Every 10 jumps the target apex height and velocity have been changed to random values taken form a uniform distribution between ± 0.1 [m] and ± 0.3 [m/s] from the previous target state, respectively. Fig. 5.20a and 5.20b show the results for 100 jumps. The system is controlled to reach simultaneously the apex height and forward velocity desired.



(a)



(b)

Figure 5.20: Data showing the apex state reached for a set of 100 jumps. Plot (a) and (b) show, respectively, the apex height and forward velocity reached (red dots) and the desired values (black line).

5.6 Conclusions

In this chapter we proposed a strategy for actuator displacement of the active SLIP model. On one hand, it allows us to analytically solve the equation that describes the leg-length dynamics during stance via partial feedback linearization. On the other, by dividing the stance phase in two parts defined by the point of maximum leg compression and setting two different actuation values for each part, it allows us to add or remove energy and to modify the upcoming apex state to span an open set within the reachable apex state set. Our strategy was tested to deal with terrain perturbations, and, for a set of system parameters, we quantitatively define the number of jumps necessary for full recovery. We additionally extend the proposed approximation to the case of a more realistic hopper model with leg mass and body inertia.

Chapter 6

Reachability-based control

As seen in the previous chapters, actuating the leg has the effect of modifying the apex state that would otherwise be reached passively. In particular, each specific actuation policy differently modifies the reachable space of the active SLIP model. In this chapter, we want to answer the following questions. "Is it possible to characterize the effect that the actuation has on the following apex state? How does applying different amounts of actuation at various instances during the stance phase affect the apex state? Furthermore, is there an actuation strategy that maximizes the reachability space?"

Displacing the actuator during the stance phase affects all three dimensions of the next apex state reached. However, any leg actuation strategy applied to its motion can only control two of the three dimensions simultaneously, as shown in Chapter 3. For this reason, here we choose to consider the apex height and forward velocity only, disregarding the forward position x, which is useful mostly in path planning scenarios. Hence, we will define the reduced apex state as $S = \{y_a, \dot{x}_a\}$. In this work, we are interested in determining how the actuator's displacement affects the reachable space. Our study does not aim to propose a leg-placement strategy, but rather to understand the effect that the actuation has during the stance phase. Therefore, we limit our analysis to a reduced reachable space, restricted to the case where the touch-down angle, θ_{TD}, is fixed. The reachable space defined in (3.4) reduces to:

$$\hat{\mathcal{R}}(S_0, \theta_{TD}) := \{S_1 \mid \exists\, \ell_{act}(t) : \mathcal{X}(S_0, \theta_{TD}, \ell_a(t)) = S_1\}.$$

Obviously, $\hat{\mathcal{R}}(S_0, \theta_{TD}) \subseteq \mathcal{R}(\boldsymbol{s}_0)$.

As mentioned in Chapter 3, adding the series actuator has the effect of extending the reachable set to a 2-d surface, whose shape is determined by the shape of the function $\ell_{act}(t)$ during the stance phase. A few leg-actuation functions have been proposed and can be found in the literature. Among them, we choose two functions that we consider relevant because they provide an insightful representation of how different leg-actuation motions can affect the locus of all the points that can be reached in one step. Indeed, we want here to compare their reachability, to determine the best course of action when choosing an actuation strategy.

In [13], J. Schmitt and J. Clark propose a leg-length actuation inspired by studies on the net energy production of muscles groups in humans:

$$\ell_{act}(t) = \ell_{act,0} - \ell_{dev} \sin(\omega t), \tag{6.1}$$

where $\ell_{act,0}$ is the initial actuator length, ℓ_{dev} is the amplitude of the variation of the actuator length from its nominal length, and ω is the frequency of the actuator's motion. t is the time from the beginning of the stance phase, and $t = 0$ corresponds to the touch-down time. Let us use $\hat{\mathcal{R}}_s(S_0,\, \theta_{TD})$ to indicate the reachable space computed with the actuator movement described in (6.1).

In Chapter 5, the stance phase is divided in two parts marked by the point of maximal spring compression (which is equivalent to $\dot{\ell}(t) = 0$). The actuator is then moved from its initial position to a desired value $\ell_{c,1}$ during the first half of the stance phase, and a desired value $\ell_{c,2}$ during the second half:

$$\ell_{act}(t) = \begin{cases} \ell_{c,1}, & \forall t : \dot{\ell}(t) \leq 0 \\ \ell_{c,2}, & \forall t : \dot{\ell}(t) \geq 0. \end{cases} \tag{6.2}$$

According to the finite response of a physical actuator, the actuator does not move instantaneously, but it is assumed to move with its maximum allowable velocity and acceleration. Let $\hat{\mathcal{R}}_p(S_0,\, \theta_{TD})$ be the reachable space computed with the actuator movement (6.2), as described in Chapter 5.

How do the reachable spaces $\hat{\mathcal{R}}_s(S_0, \theta_{TD})$ and $\hat{\mathcal{R}}_p(S_0, \theta_{TD})$ compare? And, how do they compare with respect to all the possible apex states reachable in one step for other actuator's motions? To answer to these questions, we compare the two reachable sets with the set of all reachable states obtained by moving the actuator at any time during the stance phase in any direction. Let us call this space $\hat{\mathcal{R}}_{tot}(S_0,\, \theta_{TD})$. Fig. 6.1 shows the reachability space of the three methods considered, starting from the same initial conditions at touch-down, and with the same actuator's limitation. The actuator has been modelled as in Equation (3.5). As we can see, the area of $\hat{\mathcal{R}}_{tot}$ is bigger than the areas of $\hat{\mathcal{R}}_s$ and $\hat{\mathcal{R}}_p$. Furthermore, the convex ellipsoid-like shape of $\hat{\mathcal{R}}_{tot}$ presents the advantage of reaching any point close to a desired state. This suggests that, in order to reach a wider range of apex states, it is beneficial to consider a strategy that involves updating the actuator movement throughout the entire stance phase, instead of choosing a priori some fixed parameters. Additionally, it was shown in Fig. 3.2 that the complete reachable space is a function of the actuator's characteristics. As a result, in the strategy proposed in this work, we will focus on moving the actuator with its maximum allowed parameters.

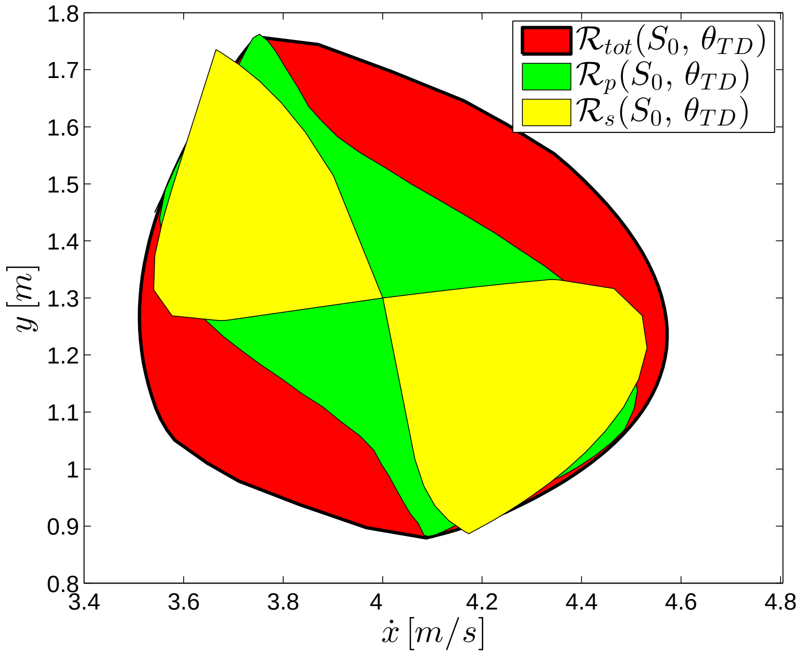


Figure 6.1: Reachable space of a passively symmetric jump $\hat{\mathcal{R}}(S_0, \theta_{TD})$ computed for $S_0 = (1.3\,[\mathrm{m}], 4\,[\mathrm{m/s}])$, $\theta_{TD} = 122.14$ [deg], leg length $\ell_0 = 1$ [m], and $\gamma = 10$. Maximum actuator velocity allowed $v_{max} = .5$ [m/s] and maximum acceleration constant $k_{acc} = 10$ [m/s^2], respectively. Defining A_{tot}, A_p and A_s the areas of, respectively, $\hat{\mathcal{R}}_{tot}$, $\hat{\mathcal{R}}_p$, and $\hat{\mathcal{R}}_s$, we have that $A_p \cong 0.619 A_{tot}$, and $A_s \cong 0.445 A_{tot}$. Note that, while $\hat{\mathcal{R}}_p \subset \hat{\mathcal{R}}_{tot}$, this does not necessarily hold true for $\hat{\mathcal{R}}_s$, since its actuator dynamics (6.1) start with nonzero velocity.

6.1 Actuator movement

In this section, we aim to understand and characterize the effects that actuation motion has on the next apex state reached. We will show the relationship between moving the actuator at different times during the stance phase and the variation of the apex state thereby reached.

Our objective is to find an actuator movement strategy to (*i*) move the system to a desired apex and (*ii*) counteract the effects of perturbations (on terrain height or on initial position at apex). As previously stated in Section 2.2, compressing or extending the actuator during the stance phase corresponds to an increase or decrease of the system's energy: the additional potential energy stored in the spring through actuation will then be transformed into additional kinetic energy throughout the stance phase, until take-off, and vice versa. However, the system's energy variation is a function of the values of the states of the system when actuation is performed. Therefore, to drive the system from an initial state $S_0 = \{y_0, \dot{x}_0\}$ to a desired apex state, $S_{des} = \{y_{des}, \dot{x}_{des}\}$, it is crucial to understand the effects that the particular time during stance when actuation happens has on the evolution of the states, and, therefore, on the position of the reached apex.

The lack of closed-form solution for the system's dynamics constrains us to perform a numerical study. Toward generality in this particular analysis, we consider a step-type actuation, i.e., we assume that the actuator can be instantaneously moved to reach a specified value, $\ell_{act,des}$:

$$\ell_{act}(t) = \begin{cases} 0, & \forall t < t_s \\ \ell_{act,des}, & \forall t \geq t_s \end{cases} \tag{6.3}$$

where t_s is the time at which the step actuation occurs. From an inital apex state and touch-down angle, we computed the passively-reached state, i.e., the state reached without actuation $S_{pass} = \{y_{pass}, \dot{x}_{pass}\}$. Given a positive and negative step actuation, we computed the apex reached as a function of the time at which the positive or negative actuation was applied: $S_r(\ell_{act,des}, t_s) = \{y(\ell_{act,des}, , t_s),\, \dot{x}(\ell_{act,des},\, t_s)\}$. Fig. 6.2 shows an example of the variation of the two components of the reached apex with respect to the passively-reached apex, as a function of the time t_s at which the step actuation is applied:

$$\Delta y_r(\ell_{act,des}, t_s) = (y_r(\ell_{act,des}, t_s) - y_{pass})/\ell_0,$$
$$\Delta \dot{x}_r(\ell_{act,des}, t_s) = (\dot{x}_r(\ell_{act,des}, t_s) - \dot{x}_{pass})/\sqrt{g\ell_0},$$

where the states are normalized to be dimensionless for comparison purposes. As we can see, the variation of y_r follows the sign of the step input: any positive actuation will result in an increase of the next apex height, whereas any negative actuation will result in its decrease, regardless of the time at which the actuation is performed. Dissimilarly, the evolution of $\dot{x}_r$ initially increases or decreases according to the sign of the actuation, and

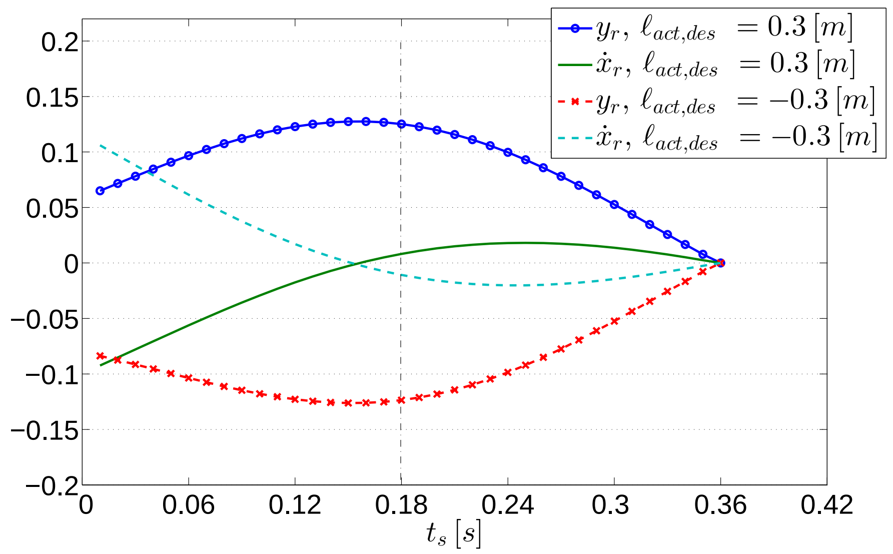


Figure 6.2: This figure shows the variation of Δy_r and $\Delta \dot{x}_r$ as a function of the time t_s at which the step actuation is applied to the system. The blue lines refer to a positive actuation $\ell_{act,des} = 0.3$ [m], while the red lines refer to a negative actuation $\ell_{act,des} = -0.3[m]$, for a model with $\ell_0 = 1$ [m] and $\gamma = 20$. The vertical dotted line signals the time at which the spring reaches its maximum compression.

then increases of decreases opposite of the sign of the actuation. By lacking analytical solution for the stance phase, it has not been possible to pinpoint the exact time at which the variation of apex state happens. However, we can see that the change in direction for $\Delta\dot{x}_r$ happens when the actuator moves close to the time corresponding to the point of half stance with respect to the spring movement, i.e., when the spring reaches its maximum compression. We can then summarize the above observations as:

- First half:
  - $\dot{\ell}_{act} > 0$: $y\uparrow$, and $\dot{x}\downarrow$,
  - $\dot{\ell}_{act} < 0$: $y\downarrow$, and $\dot{x}\uparrow$,
- Second half:
  - $\dot{\ell}_{act} > 0$: $y\uparrow$, and $\dot{x}\uparrow$,
  - $\dot{\ell}_{act} < 0$: $y\downarrow$, and $\dot{x}\downarrow$.

Another important insight we can gather from Fig. 6.2 concerns the different rates at which y and $\dot{x}$ vary throughout stance. While providing actuation during the first half of stance has the biggest effect on the variation of the $\dot{x}$ component of the next reached apex state, the second half has little effect on it. On the contrary, the y component is influenced throughout the entire stance phase, with maximum variation during the second part. This is an indicator and a confirmation of our idea that varying the actuator throughout stance gives the greatest control authority to reach a desired state, compared to choosing any fixed time during stance at which to activate the actuation. This, for example, also explains and is confirmed by the bowtie-like shape of the reachability set $\hat{\mathcal{R}}_s$ in Fig. 6.1: in order to modify y without much variation of the otherwise passively reached value of $\dot{x}$, the actuation should mostly be applied during the first half of the stance phase. However, the choice of the sinusoidal function (6.1) with constant amplitude and frequency, distributes the actuation almost evenly throughout the entire stance phase, precluding the possibility of favoring one part of stance with respect to another.

These observations consider only one-step actuator movement during stance, but can be generalized to the case of subsequent actuation movements, with more realistic dynamics, as for example (3.5). We accomplish this by updating the actuator value every few time-steps during the stance phase. Let us define t as the time elapsed from the beginning of the stance phase (where $t_0 = 0$ is the touch-down time), and divide the stance phase in time intervals of length δt. Let $\tilde{S}_n$ be the apex state that would be reached if the actuator were to be moved as a certain function $f_n(t)$ from touch-down to time $n\delta t$, and then stopped at the current value for the rest of the stance phase:

$$\tilde{S}_n \longrightarrow \ell_{act} = \begin{cases} f_n(t), & \text{if } t \in [t_0,\, n\delta t] \\ f_n(n\delta t), & \text{if } t > n\delta t \end{cases}$$

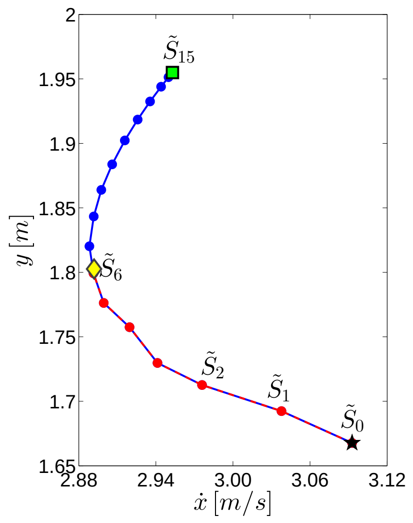
Then, $\tilde{S}_{n+1}$ will be the apex state reached if the actuator were to be moved as:

$$\tilde{S}_{n+1} \longrightarrow \ell_{act} = \begin{cases} f_{n+1}(t), & \text{if } t \in [t_0,\, (n+1)\delta t] \\ f_{n+1}((n+1)\delta t), & \text{if } t > (n+1)\delta t \end{cases}$$

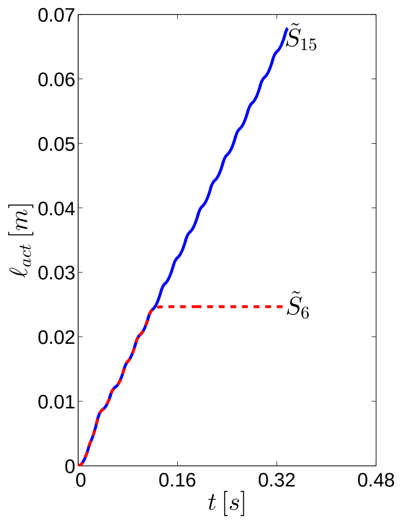
where $f_{n+1}(t) = f_n(t)$, for $t \in [0,\, n\delta t]$.

Fig. 6.3a and 6.3c show the position of the next apex, $\tilde{S}_n$, when moving the actuator as respectively in Fig. 6.3b and 6.3d for values of $n = 1, 2, \dots$ until take-off. A time interval of length $\delta t = 0.02$ [s] has been chosen. As we can see, and according to earlier observations, at each time step n, compressing the spring results in an increase of y_n, while extending the spring results in a decrease of y_n with respect to its value at the

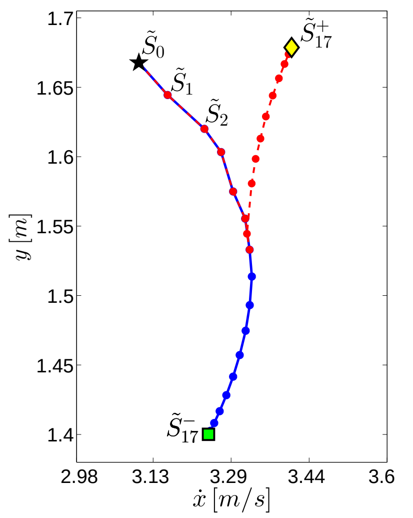
$(n-1)-$th step. Conversely, any spring compression translates in a decrease or increase of $\dot{x}_n$ if this happens, respectively, during the first or second half of the stance phase. Vice versa, an extension of the spring applied during the first or second half of the stance phase results in an increase or decrease, respectively, of $\dot{x}_n$. This is summarized graphically in Fig. 6.4.



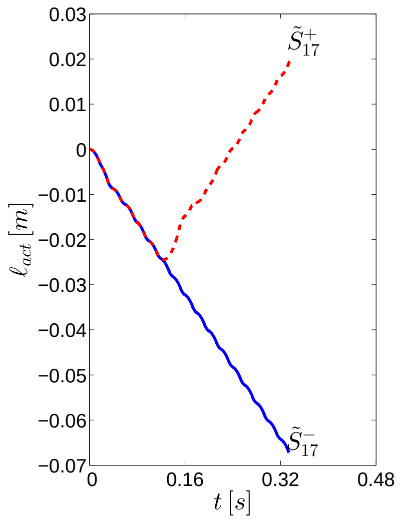
(a)



(b)



(c)



(d)

Figure 6.3: Subfig. 6.3a and Subfig. 6.3c show the next apex states $\tilde{S}_n$, $n = 1, 2, \ldots$ until take-off, for the different actuation motions shown in Subfig. 6.3b and 6.3d. The blue line and the dotted red line in Subfig. 6.3a and 6.3c are the apex states reachable if the actuator motion follows the solid blue and dotted red trajectories in Subfig. 6.3b and 6.3d, respectively. The black star symbol represents $\tilde{S}_0$, i.e., the passively reached state. The yellow diamonds represent the apex states reached following the dotted red actuator trajectory in Subfig. 6.3b and 6.3d, while the green squares represent the apex states reached if the actuator follows the solid blue trajectory in Subfig. 6.3b and 6.3d.

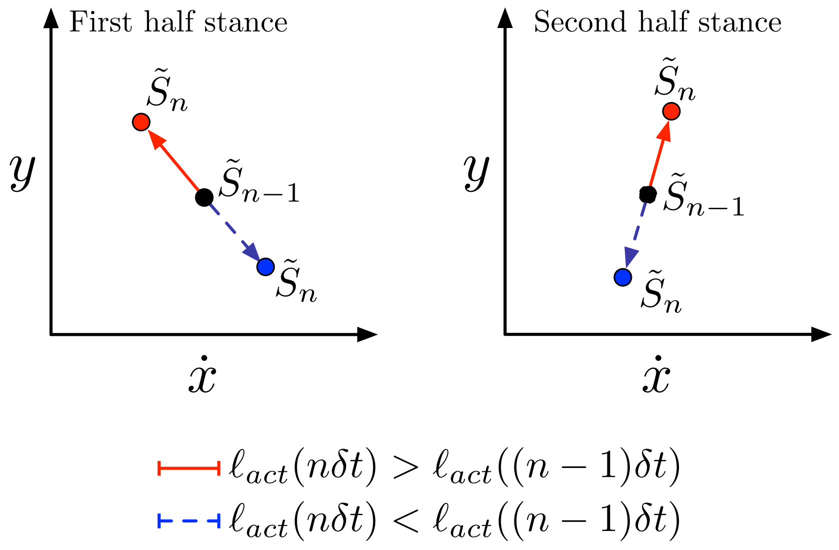

First half stance
Second half stance
y
$\dot{x}$
$\tilde{S}_n$
$\tilde{S}_{n-1}$
$\ell_{act}(n\delta t) > \ell_{act}((n-1)\delta t)$
$\ell_{act}(n\delta t) < \ell_{act}((n-1)\delta t)$

Figure 6.4: Position of the apex state that would be reached providing a positive or negative actuation during the following interval $[(n-1)\delta t,\, n\delta t]$. Note that the slope of the directional curve from $\tilde{S}_{n-1}$ to $\tilde{S}_n$ is a function of the particular actuation motion, the system's parameters and the initial condition $\tilde{S}_{n-1}$.

6.2 Control Strategy

We introduce here our control strategy to reduce the error between the reached and a desired apex state, first giving an example, and then explaining in detail our proposed algorithm.

The objective of our control strategy can be summarized as follows.

Let $S_0 = \{y_0, \dot{x}_0\}$ be the initial apex state, and θ_{TD} the angle of the leg at touch-down. Let us define $S_{des} = \{y_{des}, \dot{x}_{des}\}$ as the desired apex state, and $S_r = \{y_r, \dot{x}_r\}$ the apex state reached with a certain actuation motion. Then, we want to find an actuation motion strategy to reduce the Euclidean distance between the desired apex and the actual apex reached. In order to compute the Euclidean distance over variables with different measurement units (apex height and velocity), we reduce the states to their dimensionless counterpart. By defining the dimensionless time $\tau = \sqrt{g/\ell_0}t$, we can write the dimensionless apex height as $\hat{y} = y/\ell_0$, and the dimensionless apex forward velocity as $\hat{\dot{x}} = \dot{x}/\sqrt{g\ell_0}$. The dimensionless apex state is $\hat{S} = \{\hat{y}, \hat{\dot{x}}\}$. The error is then computed as:

$$Err = \| \hat{S}_{des} - \hat{S}_r \|_2 \,. \tag{6.4}$$

We explain here our proposed strategy to minimize error during stance.

During the stance phase, let us define each time interval of size δt as $\Delta t_n = [n\delta t, (n+1)\delta t]$, $n = 1, 2, \ldots$ until take-off. Our control strategy for actuator movement is to update the actuator's value every time-step δt of the stance phase, according to predictions of next apex state based on the current state. This means that at each time interval Δt_n, a new value for the actuator displacement at the next time interval, $\ell_{act}(\Delta t_{n+1})$, is computed.

At each interval Δt_n, we can compute the apex state, S_n, that the SLIP model would reach if the actuator remains at the current value (computed at Δt_{n-1}):

$$\ell_{act}(t) = \ell_{act}((n{-}1)\delta t), \quad \forall t \geq n\delta t,$$

i.e., the apex state reachable without further actuator movement. This is motivated by the observations highlighted in Section 6.1, i.e., that activating the actuator during the first part of the stance phase has the effect of increasing or decreasing the value of the forward height at the next apex, y, while at the same time decreasing or increasing, respectively, the value of the next apex's forward velocity, $\dot{x}$. By contrast, moving the actuator during the second half of the stance phase has the effect of increasing or decreasing the values of both y and $\dot{x}$ at the next apex state. Hence, to determine how to control the actuator's displacement at the next time step, it is imperative to determine the position of S_n with respect to the position of the desired apex state S_{des}. At the same time, the choice of the next value for the actuator, $\ell_{act}(\Delta t_{n+1})$, is computed predicting the position of S_{n+1} when moving the actuator for one step Δt to one direction or the other, i.e., compressing or extending the spring.

For example, let us assume that the relative position of S_{des} with respect to S_{pass} is given in Fig. 6.5 *I*. From what is graphically summarized in Fig. 6.2 and Fig. 6.4, we can already expect that the actuator motion necessary to reach the desired state will consist of either an initial spring extension or compression, followed by a compression during the second half of the stance phase. What we aim to accomplish with our algorithm, is to control the amount of the desired compression/extension, by updating its value throughout the entire stance phase. At each time-step during the first half of the stance phase, we will compute the next apex reached if the actuator were to be compressed or extended during the next time-step, Fig. 6.5 *II*, kept at the reached value for the rest of the first half of the stance phase, and subsequently compressed during the entire second half of the stance phase, as in Fig. 6.5 III. By computing and averaging the distance of the resulting apex states to the desired state, we can compute how much to move the actuator at the next step, Fig. 6.5 IV. Once the dynamics enter into the second half of the stance phase, Fig. 6.6 I, we can start compressing the actuator: we compute the states the system would reach if the actuator were to move at its maximum and at half its capabilities (e.g., actuator acceleration) at the next time step, and then kept at the current value for the rest of the stance phase, Fig. 6.6 II. Based on an average of the distance of these two predicted states with respect to the desired apex, the actuator's action to implement at the following time step can be computed, Fig. 6.6 III. Final result of our algorithm is shown in Fig. 6.6 IV.

Since at each step the algorithm predicts where the next apex state would be when displacing the actuator in the following time step and in the second half of the stance, the algorithm is essentially creating approximate "grid lines" (see Fig. 6.5 III) that are based on limited numerical simulations of the stance phase. Computing more intermediate states, i.e., higher order fits for the dynamics of the system subjected to actuation, would certainly result in a better estimate, but would require more computational time. For applicability, we want to ensure that the algorithm finishes well within the time δt between actuation updates. As a result, an error is introduced in the algorithm. Such error is the greatest during the first half of the stance phase, when we have more uncertainty about the direction the motion will go, and results in the bouncing motion of the running estimate of the next apex state, seen in Fig. 6.6 IV. Fortunately, the first half of the stance is also characterized by a higher time-to-apex, allowing for adequate time to significantly reduce the approximation error.

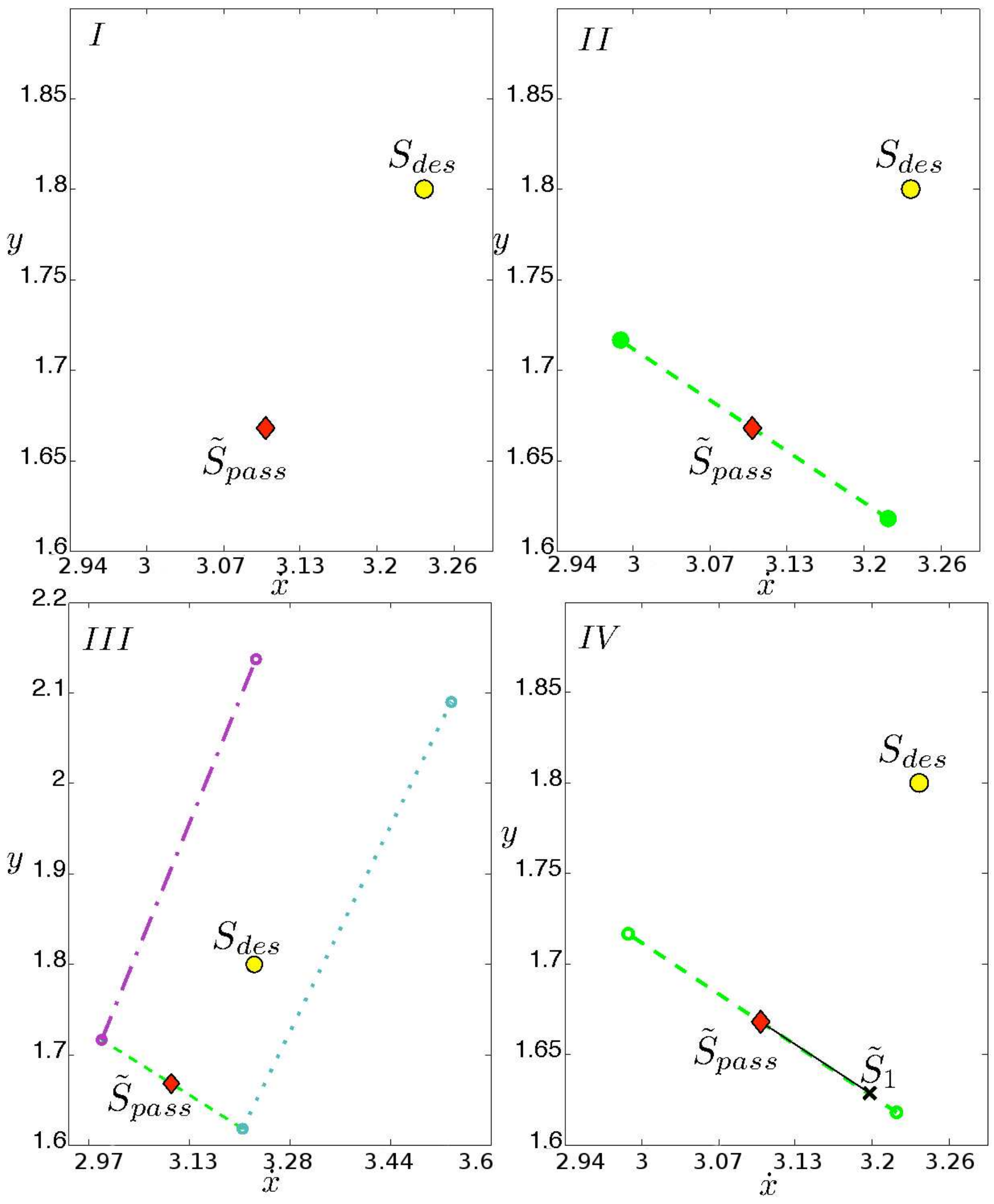


Figure 6.5: These plots show an example of the steps implemented during the first half of the stance phase in order to approach the desired apex state.

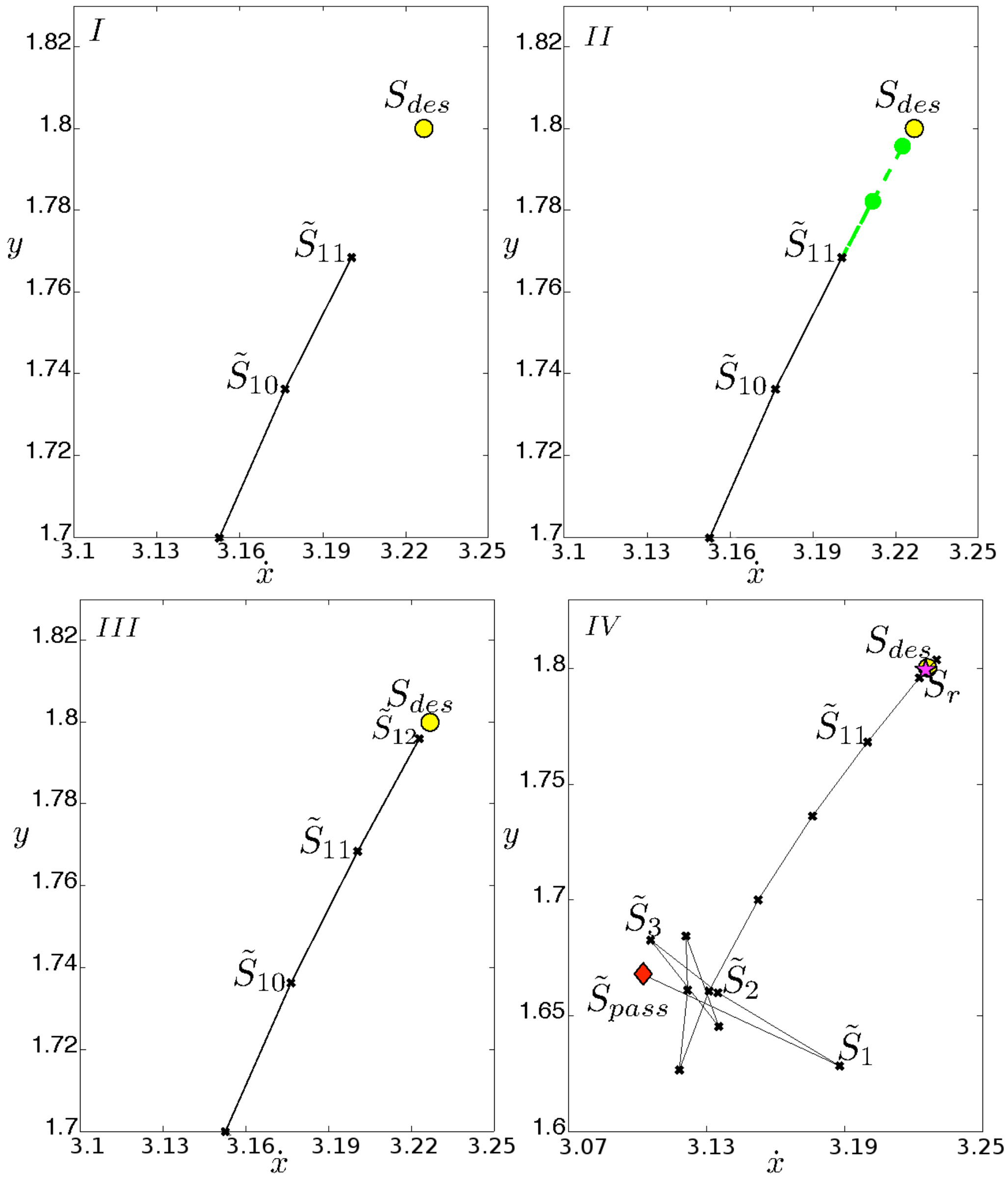


Figure 6.6: Plots I to III show an example of the steps implemented during the second half of the stance phase in order to approach the desired apex state. Plot IV provides a summary of the steps implemented during the entire stance phase.

6.2.1 Algorithm

Let us call $\ell_{act}^{m}(\Delta t_i)$ and $-\ell_{act}^{m}(\Delta t_i)$ the maximum actuator movements possible in the time interval Δt_i with respect to the actuator's initial position and its specifications, respectively in the positive (spring compression) and negative (spring extension) direction. Let $\ell_{act}^{end}(\Delta t_i)$ be the value that the actuator assumes at the end of Δt_i.

At each time interval Δt_n, with $n = 0, 1, \ldots,$ until the end of the stance phase:

(i) Compute the apex state, $\tilde{S}_{n+1}$, that the SLIP model would reach if keeping the actuator at the current value :

$$\ell_{act}(t) = \ell_{act}^{end}(\Delta t_n), \quad \forall t \geq (n+1)\delta t,$$

i.e., the apex state reachable without further actuator movements.

- If $\dot{\ell}(t) \leq 0$ (first half of stance phase):

  (iiA) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at its maximum negative or positive motion for a time interval Δt, with no further actuation movement occurring for the rest of the stance phase.

$$\ell_{act}(t) = \begin{cases} \pm\ell_{act}^{m}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell_{act}^{end}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

  Let us call these points S_{n+1}^{+} and S_{n+1}^{-}.

  (iiiA) Compute the apex state that the SLIP model would reach if we were to move the actuator with its maximum negative or positive motion for a time interval

Δt and then stop until the beginning of the second half of the stance phase. Then, the actuator moves at its maximum negative or positive motion until the end of the stance phase.

$$\ell_{act}(t) = \begin{cases} \pm\ell_{act}^{m}(\Delta t_{n+1}), & t \in \Delta t_{n+1} \\ \ell_{act}^{end}(\Delta t_{n+1}), & \forall t : \dot{\ell}(t) \leq 0 \\ \xi\ell_{act}^{m}(\Delta t_{n+1}), & \forall t : \dot{\ell}(t) > 0, \end{cases}$$

The direction of movement during the second half ($\dot{\ell}(t) > 0$) is dictated by the position of S_{n+1} with respect of S_{des}, as shown in Fig. 6.7a. In fact, during the second half of the stance phase, a positive actuation ($+\ell_{act}^{m}$) would increase both y and $\dot{x}$ values at the next apex, while a negative actuation ($-\ell_{act}^{m}$) will result in a decrease of y and $\dot{x}$. Therefore, if S_n is in zone I (see Fig. 6.7a), then $\xi = 1$; else, $\xi = -1$. Let us call these points P_{n+1}^{+} and P_{n+1}^{-}.

(ivA) Consider the line segments $s^{+} = (S_{n+1}^{+}, P_{n+1}^{+})$ and $s^{-} = (S_{n+1}^{-}, P_{n+1}^{-})$, and compute the vector distance between the segments and the desired apex state S_{des}:

$$d^{+} := \text{vector distance}(s^{+}, S_{des}),$$
$$d^{-} := \text{vector distance}(s^{-}, S_{des}).$$

(vA) Now, Compute the next actuator value desired, $\ell_{act}(\Delta t_{n+1})$, as follows.

$$\ell_{act}(\Delta t_{n+1}) = \\ \begin{cases} \mu \ell^m_{act}(\Delta t_{n+1}), & \text{if } |d^+| \leq |d^-| \\ -\mu \ell^m_{act}(\Delta t_{n+1}), & \text{if } |d^-| \leq |d^+| \end{cases}$$

where the parameter μ is a weight $\in$ [0, 1]. In particular, if the two vectors d^+ and d^- have a positive dot product, then $\mu = 1$, as shown in Fig. 6.7c and 6.7d. Otherwise, the desired apex point is "between" the two line segments, and the required actuator movement will be a fraction of the previous predictions, as shown in Fig. 6.7b:

$$\mu = \left| \frac{|d^-| - |d^+|}{|d^-| + |d^+|} \right| .$$

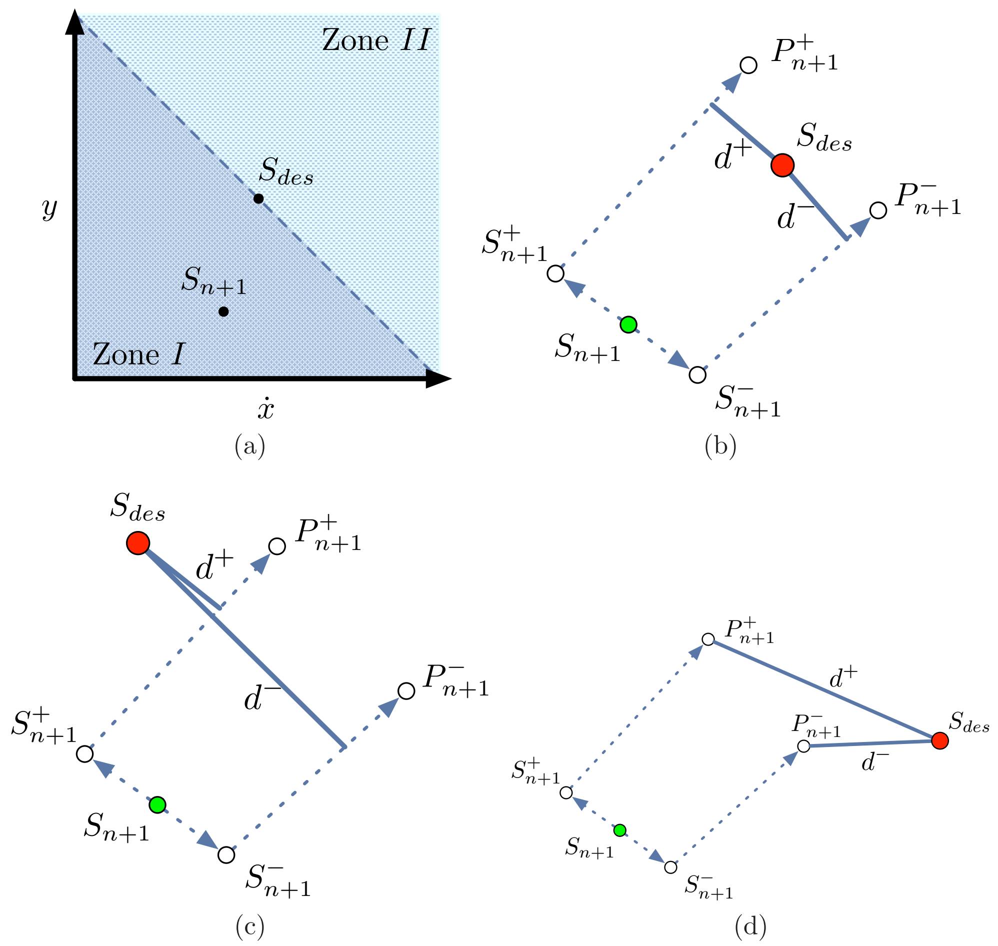


Figure 6.7: Subfig. 6.7a shows the two dividing zones where S_{n+1} can be with respect to the position of S_{des}. If S_{n+1} is in zone I, during the second half of stance the actuator displacement needs to increase; else, it needs to decrease. The algorithm creates approximate grid lines (dotted blue line segments) based on limited numerical simulations of the stance phase: only the states S^+_{n+1}, S^-_{n+1}, P^+_{n+1}, and P^-_{n+1} are computed. The actuator motion during the first half of stance is chosen based on the distance and location of the line segments with respect to the desired state: Subfig. 6.7b, 6.7c, and 6.7d demonstrate possible location of vectors d^+ and d^- with respect to S_{des}. Computing more states would result in a better fit for the direction along which the dynamics of the system move when subjected to actuation, which is in general not a straight line, and as a consequence it would result in a more accurate choice of actuator motion. However, this would be at the expense of the computational time, and could preclude the use of the algorithm in real time.

- If $\dot{\ell}(t) > 0$ (second half of the stance phase):

  During the second part of the stance phase, both $\dot{x}$ and y functions are increasing or decreasing, as seen in Fig. 6.2. Therefore, the choice of whether moving the actuator in the positive or negative direction is again done based on the position between S_{n+1} and S_{des}, and it is the same as the value of ξ computed in (iii) .

(iiB) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at its maximum velocity for the time interval Δt_{n+1}. Then, no further actuation movement occurs for the rest of the stance phase.

$$\ell_{act}(t) = \begin{cases} \xi \ell_{act}^{m}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell_{act}^{end}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

Let us call this point P_{n+1}^{+}.

(iiiB) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at half of its maximum (positive or negative) displacement possible for a time interval Δt_{n+1}. Then, no further actuation movement for the rest of the stance phase.

$$\ell_{act}(t) = \begin{cases} \xi \frac{1}{2} \ell_{act}^{m}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell_{act}^{end}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

Let us call this point P_{n+1}^{-}.

(ivB) Consider the line segments $s^{+} = (S_{n+1}, P_{n+1}^{+})$ and $s^{-} = (S_{n+1}, P_{n+1}^{-})$. Compute the vector distance between the segments and the desired apex state

S_{des}:

$$v^+ := \text{vector distance}(s^+, S_{des}),$$

$$v^- := \text{vector distance}(s^-, S_{des}).$$

(vB) Now, compute the next actuator value desired, $\ell_{act}(\Delta t_{n+1})$, as follows.

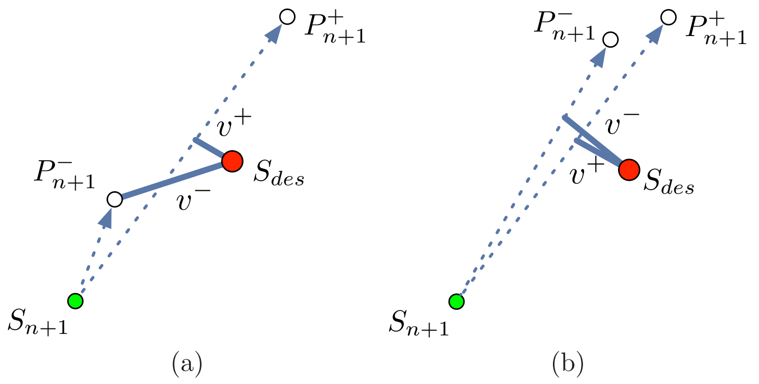
$$\ell_{act}(\Delta t_{n+1}) = \begin{cases} \mu \ell_{act}^m(\Delta t_{n+1}), & \text{if } |v^+| \leq |v^-| \\ \mu \frac{1}{2} \ell_{act}^m(\Delta t_{n+1}), & \text{if } |v^-| \leq |v^+| \end{cases}$$

where the parameter μ is a weight $\in$ [0, 1]. In particular, if both or neither vectors are orthogonal to the respective line segment s^+ or s^-, then $\mu = 1$, as shown in Fig. 6.8b and 6.8c. Else, the desired apex point is "between" the two line segments, and the required actuator movement will be a fraction of the previous predictions, as in Fig. 6.8a:

$$\mu = \left| \frac{|v^-| - |v^+|}{|v^-| + |v^+|} \right|.$$

(vi) repeat (i) to (v) until the end of the stance phase.

Note that the choice of moving the actuator with its maximum velocity and acceleration allowed results from the fact that a higher velocity and acceleration correspond to a bigger reachability space (e.g., see Fig. 3.2 where reachability sets computed for different values of maximum velocities are compared). The algorithm can of course be implemented with any other velocity and acceleration values, provided they are in accordance with the limitations of the specific actuator used.



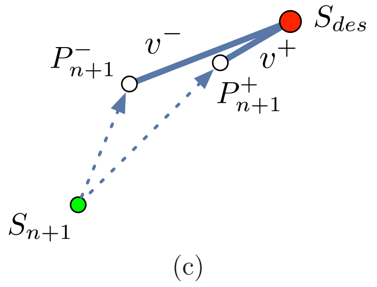


Figure 6.8: These figures demonstrate possible locations of the vectors v^+ and v^- with respect of S_{des}.

6.3 Performance

We show here the performances of our algorithm. In particular, we start with considerations on the required time delay for the algorithm implementation and the time interval used, and we continue with simulation results showing how effective the algorithm is in reducing the distance to the desire state.

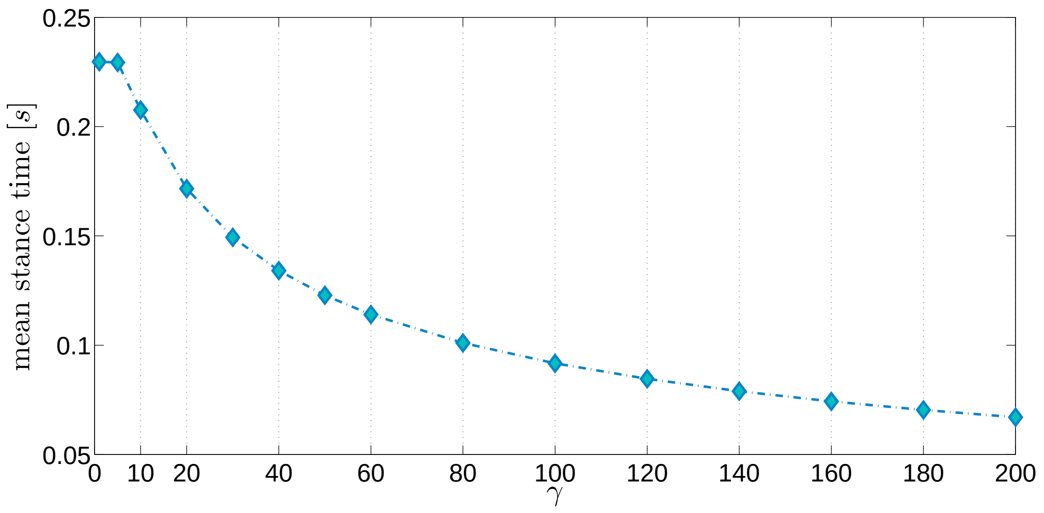
6.3.1 Time interval and delay at first computational instance

The choice of the time interval δt determines the frequency at which the algorithm is updated. Any control action applied in real-time during the stance phase needs to be executed in finite time, which corresponds to the duration of the stance phase itself. The execution of one update of the algorithm is only a function of the processor used, and therefore the number of iterations of the algorithm depends on the duration of the stance phase, which in turn is a function of the relative spring stiffness γ: higher values of γ correspond to a stiffer spring, and therefore a shorter stance phase; conversely, lower values of γ correspond to a longer stance phase caused by a softer spring. Let us choose initial conditions and parameters based on biological data for a typical human, as in Table 6.1. Then, the average stance time is shown in Fig. 6.9a as a function of the

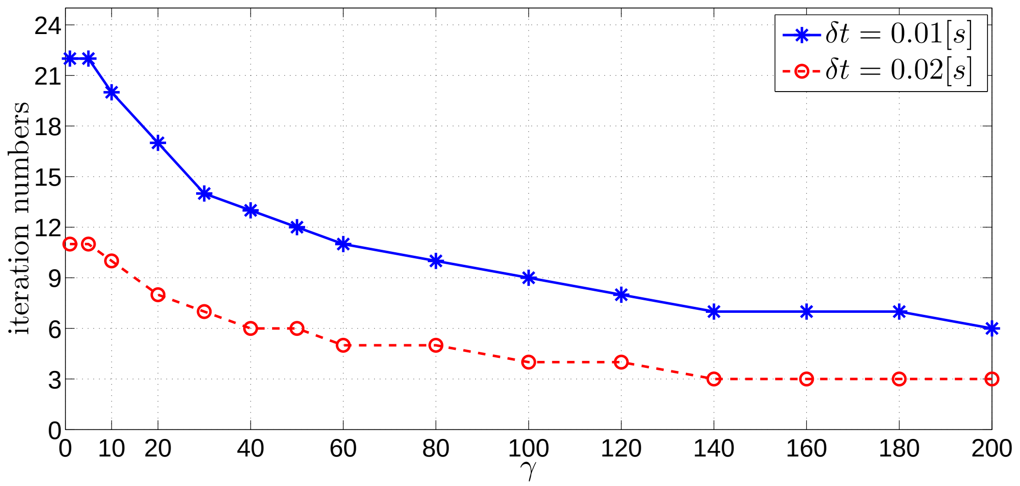
Parameters	
$g =$	9.81 m/s^2
$M =$	80 kg
$\ell_0 =$	1 m
$\ell_{k,0} =$	0.5 m
$\ell_{act} \in$	$[-0.1, 0.1]$ m
$y \in$	$[1, 2.5]$ m
$\dot{x} \in$	$[0.4, 5]$ m/s

Table 6.1: Parameters based on biological data for a typical human.

relative spring stiffness. Considering that on an average computer (Intel Core i7 eight core processor CPU, 2.8GHz) running Matlab R2012a, the average time to perform all the calculations required for one update of the algorithm is $\approx 0.01, 0.02$ [s], one can ask what values of γ for a real robot are a good trade-off between number of algorithm iterations and considerations on biological inspiration. Fig. 6.9b shows how many algorithm updates are possible on average during stance, for $\delta t = 0.01$ [s] and $\delta t = 0.02$ [s]. As expected, the smaller the value of γ, the more algorithm iterations can be performed. A study by [3] shows that the relative spring stiffness assumes very similar values for various gaits in different animals (both quadrupeds and bipeds). For runners, values were found between $\gamma \in [7.1, 14.6]$, while $\gamma \in [7.7, 13.6]$ in hoppers. It is then reasonable to use values of $\gamma \in [8, 13]$, which correspond to a high number of algorithm iterations, validating the applicability of our control strategy.



(a)



(b)

Figure 6.9: Subfig. 6.9a shows the average stance time as a function of γ for a set of initial apex states and system's parameters as in Table 6.1. Subfig. 6.9b shows instead the corresponding number of algorithm iterations.

When implementing a control action, the time delay of such controller is usually a concern that requires considerations. In the case of the algorithm here proposed, each time interval Δt_n is dedicated to plan the desired actuation value at the next interval, $\ell_{act}(\Delta t_{n+1})$. This implies that at the initial time interval Δt_1 there will be no actuation; namely, the actuator will begin its motion from the second time-step. We want here to investigate how this will affect the performances of our algorithm. As one would expect, increasing time delay reduces the reachable space (Fig. 6.10) by limiting the maximum achievable actuator movement.

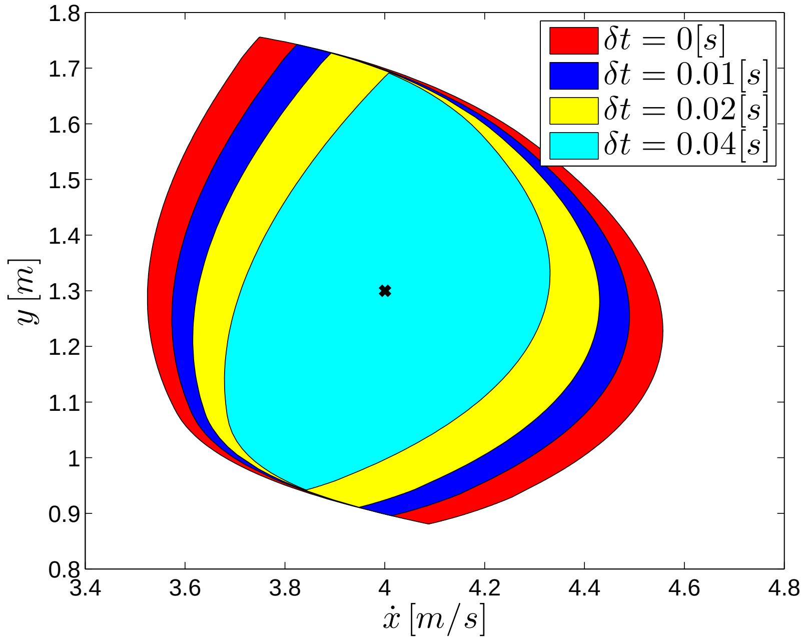


Figure 6.10: Reachability sets computed for different delay values. As expected, increasing δt reduces the reachable space.

The delay depends on the time interval chosen to perform our algorithm, δt, which in turn depends on the computational velocity of our computer. While the delay may seem a limitation, it is important to point out the following. On an average computer, the required time step is $\delta t \simeq 0.01$ [s]. In these circumstances, we can compare the reachable space with delay with the reachable spaces obtained with other control strategies, as shown in Fig. 6.11. As we can see, the reachability set with delay is still bigger (area-wise) than the case $\hat{\mathcal{R}}_p$ and $\hat{\mathcal{R}}_s$. Thus we conclude that even with a time delay, the proposed algorithm offers significant advantages to previous control strategies.

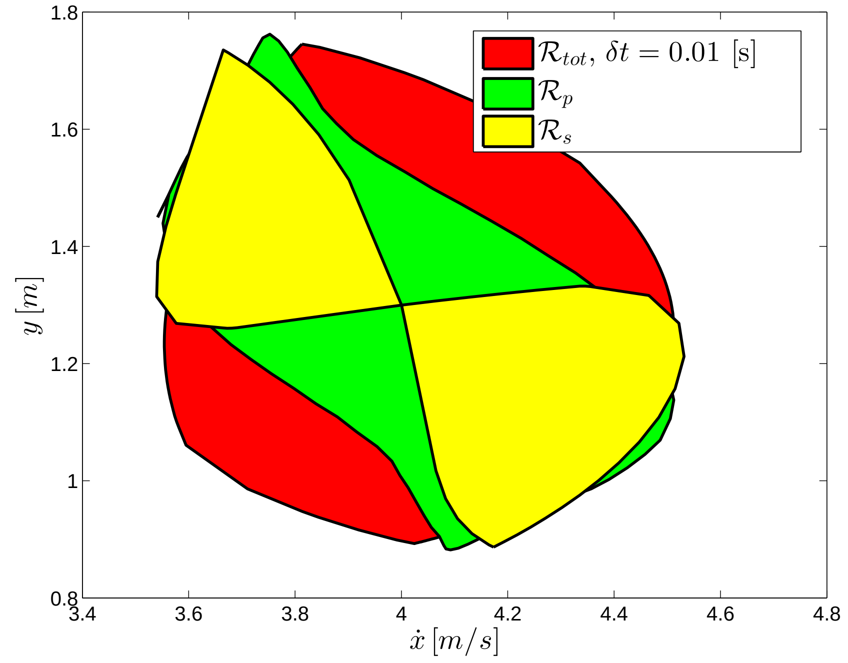


Figure 6.11: Reachability sets with $\delta t = 0.01$ [s] delay, compared to $\hat{\mathcal{R}}_p$ and $\hat{\mathcal{R}}_s$.

This performance study considers the ideal SLIP model, where the body is a point mass, and the leg is massless. Nonetheless, we expect to be able to utilize the same control strategy on a real platform with SLIP-like dynamics. Because of the nature of its dynamics, the real system will behave similarly to the ideal model when subjected to a piston-like actuation, e.g., it is reasonable to believe that a negative actuation will result in a decrease of the apex height. Additionally, at each time step throughout the stance phase, computations of the subsequent reachable state are performed numerically using equations of motion for a model of the system. Thus, the performance of the algorithm is strongly tied to how accurately the model that we construct of the physical system captures its actual dynamics.

6.3.2 Percentage change

We will evaluate our controller using equation (6.4). To quantify the reduction in Err, for an initial apex we introduce the percentage change, PC, as:

$$PC = 100 \frac{\| \hat{S}_{pass} - \hat{S}_{des} \|_2 - \| \hat{S}_r - \hat{S}_{des} \|_2}{\| \hat{S}_{pass} - \hat{S}_{des} \|_2}, \tag{6.5}$$

where $\hat{S}_{des}$ is the dimensionless desired apex state, $\hat{S}_{pass}$ is the dimensionless state reached without actuation, and $\hat{S}_r$ is the dimensionless apex state reached with our control strategy. Using percentage change as a performance index allows us to compare the system performance with and without control action over a broad range of desired states.

The performance of the controller has been tested as follows: an initial apex state S_0 and touch-down angle θ_{TD} are chosen, and a set of $10,000$ points are generated inside the corresponding reachability set without delay, $\hat{\mathcal{R}}_{tot}(S_0, \theta_{TD})$. S_{pass} is defined as the passive apex state, i.e., the apex state that is reached without actuation movement. Then, for each point chosen inside the reachability set, the controller is tested by setting the desired apex state, S_{des}, to be one of those points, with a controller update time step of $\delta t = 0.01$ [s]. As we can see in Fig. 6.12 and Fig. 6.13, the effect of the controller is to dramatically reduce the error.

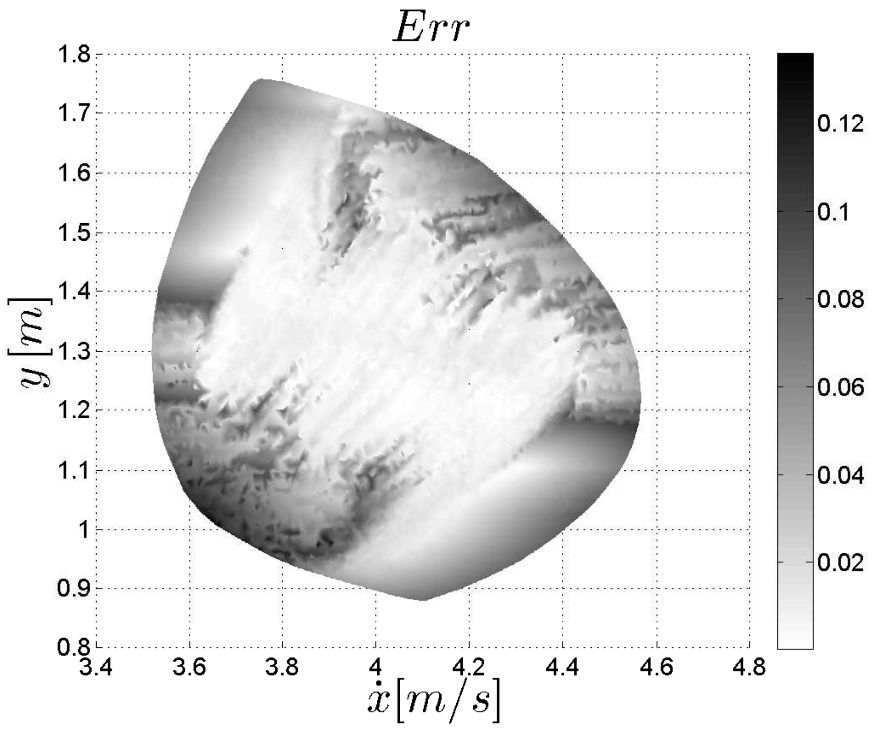


Figure 6.12: Error from desired point, $Err = \parallel \hat{S}_r - \hat{S}_{des} \parallel_2$, computed for 10,000 points randomly taken inside the reachable set $\hat{\mathcal{R}}(S_0,\, \theta_{TD}\}$, with $S_0 = \{1.3\,[\mathrm{m}],\, 4\,[\mathrm{m/s}]\}$ and $\theta_{TD} = 121.15$ [deg]. The update time has been chosen to be $\delta t = 0.01$ [s].

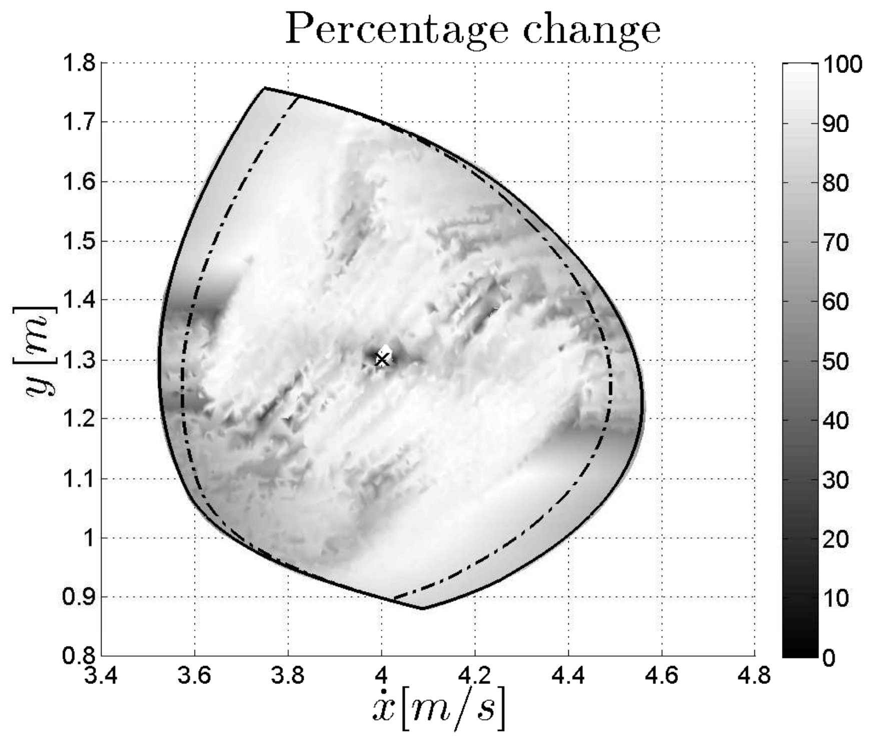


Figure 6.13: Percentage change graph computed for 10,000 points randomly taken inside the reachable set $\hat{\mathcal{R}} = \{S_0, \theta_{TD}\}$, with $S_0 = \{1.3\,[\text{m}], 4\,[\text{m/s}]\}$ and $\theta_{TD} = 121.15$ [deg]. The update time is $\delta t = 0.01$ [s], and the inner dotted black line marks the border of the reachability space computed with a δt gap. The black x mark represents S_{pss}, the apex state reached without any actuator motion. As expected, the percent change of the subset $\hat{\mathcal{R}} - \hat{\mathcal{R}}_{gap}$ is on average smaller than the one on $\hat{\mathcal{R}}$ alone, since points in $\hat{\mathcal{R}} - \hat{\mathcal{R}}_{gap}$ are not directly reachable with our proposed strategy. Note that the dark (lower improvement) and empty areas around S_{pass} are mainly due to the fact that since the desired apex points are so close to the passively reached apex, the margin for improvement is very slim.

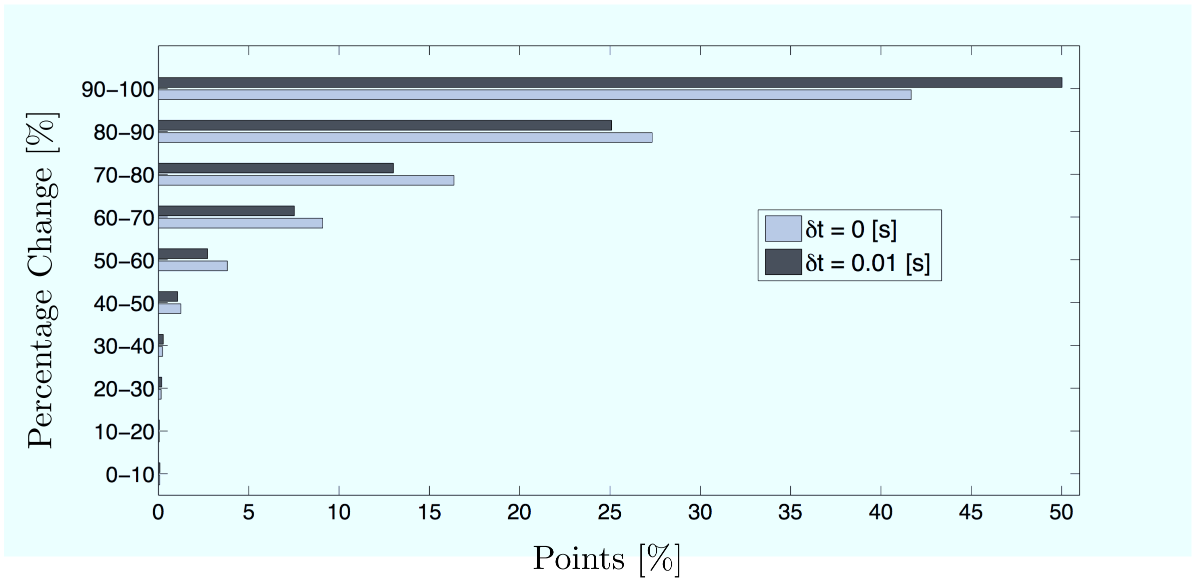


Figure 6.14: Percentage distribution of PC. The bars represent the percentage of points with a PC between [0, 10], (10, 20],... (90, 100] [%] in the reachability set without delay (light grey bars) and in the reachability set with delay $\delta t = 0.01[s]$ (dark bars).

Fig. 6.14 shows how many point inside the reachability set have a PC value between [0, 10], (10, 20],... (90, 100] [%], both with and without time delay ($\delta t = 0.01$ [s]), for $10,000$ random desired states. If we consider only the points inside the time-delay reachability set, we can see that $\sim 50\%$ of the points show a percentage change $PC \in$ (90, 100], meaning that the actuator motion has reduced the error by a factor between 90% and 100%. However, if we consider instead the set of all points reachable (i.e., without delay), the percentage of points with $PC \in$ (90, 100] is instead $\sim$ 42%, as expected smaller than its subset with delay. Indeed, due to the delay in the execution of the controller, points outside the subset $\hat{\mathcal{R}} - \hat{\mathcal{R}}_{gap}$ are not reachable anymore. In both cases, though, more than two thirds of the points have $PC \in$ (80, 100], highlighting the performance of our proposed controller, even if it requires a delay.

Fig. 6.15, and Fig. 6.16a and 6.16b show an example of the performance of our controller. An initial apex state has been chosen to be $S_0 = \{1.3\,[\text{m}],\, 4\,[\text{m/s}]\}$, and

$\theta_{TD} = 121.15$ [deg], $\gamma = 10$. The desired apex state is $S_{des} = \{1.5\,[\mathrm{m}], 3.8\,[\mathrm{m/s}]\}$. Maximum velocity and acceleration constant were chosen to be $v_{max} = 0.5$ [m/s] and $k_{acc} = 10$ [m/s^2]. After running the controller with $\delta t = 0.01$ [s], the reached state is $S_r = \{1.505\,[\mathrm{m}], 3.805\,[\mathrm{m/s}]\}$. Fig. 6.15 shows the values of S_n computed at each time interval Δt, while Fig. 6.16a and 6.16b show, respectively, the errors $\| S_{des} - S_n \|_2$ and its $\dot{x}$ and y component at each time interval, and the actuator value $\ell_{act}(t)$ over time.

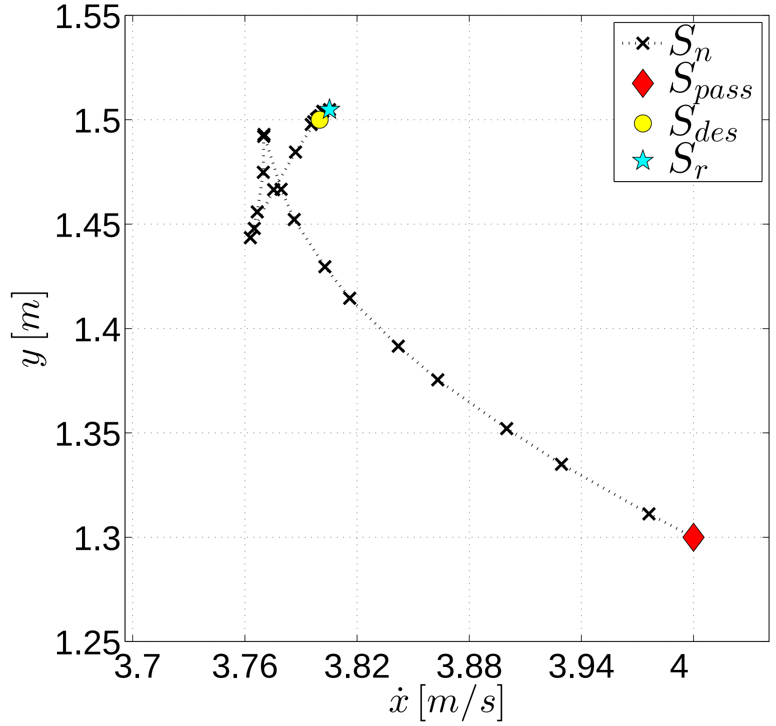
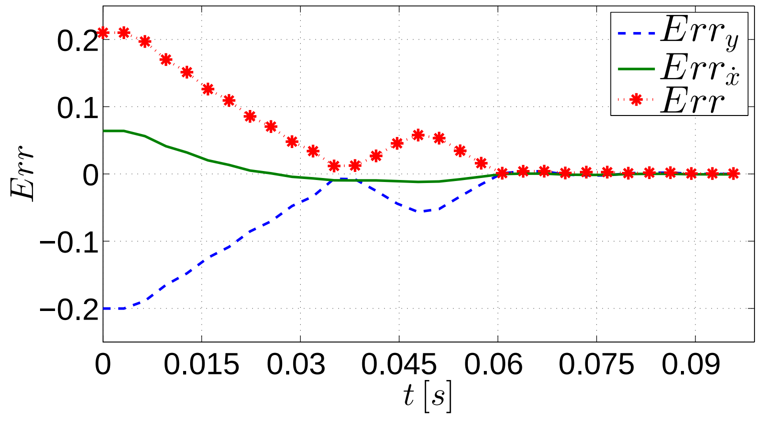
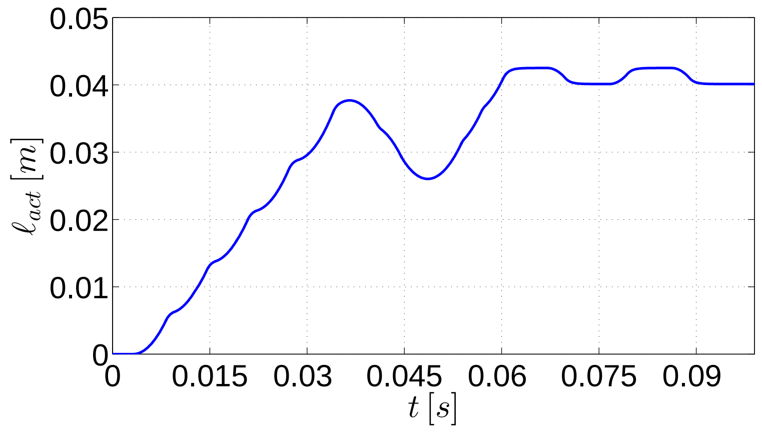


Figure 6.15: Example of the performance of the proposed control algorithm. The red diamond is the passive state, i.e., the state reached without any actuation. The x-marked line represents the apex states S_n, $n = 1, 2, \ldots$ computed by the algorithm, while the yellow circle and the cyan star represent respectively the desired state, S_{des}, and the reached state, S_r.



(a)



(b)

Figure 6.16: Subfig. 6.16a and 6.16b show respectively the evolution of the error (6.4) and the evolution of its component in $\dot{x}$ and y, and the computed actuation for the example in Fig. 6.15.

6.3.3 Percentage change over a set of consecutive steps

Section 6.3.2 presents the performance of our control strategy for one specific set of initial conditions: starting from one apex state and the touch-down angle necessary to perform a symmetric hop, we showed the performance of the algorithm when trying to reach $10,000$ states inside its reachability set. To show that the presented results still hold for arbitrary initial conditions and touch-down angles, we consider a path of $1,000$ subsequent hops. At each hop, the touch-down angle is chosen randomly from the set of angles that allow a successful jump, not necessarily symmetric, where a successful jump is defined as any apex-to-apex hop characterized by a positive velocity and an apex height exceeding the leg length to avoid collision between the leg and the ground. Then, the desired apex state is set by adding to each dimension of the passively reached state an offset picked from a uniform distribution. Fig. 6.17 shows all the apex states reached by the system during the $1,000$ subsequent hops. As each one of these states is the initial condition of the following hop, our control algorithm has been tested for a wide variety of initial conditions which encompass the typical operating range for the SLIP model. Furthermore, generating desired apex states by adding uniformly distributed noise to the passive apex state ensures that our controller has been tested for a wide range of initial error. Fig. 6.18 shows the resulting percentage change for $1,000$ hops. As we can see, over two thirds of the states have a percentage change $PC \in [90, 100]$, meaning that the controller is able to reduce the distance to the desired state of a factor between 90% and 100%. Overall, over 80% of the states have a $PC \geq 70$. More in detail, Fig. 6.19 shows the output error $\| \hat{S}_r - \hat{S}_{des} \|_2$ as a function of the magnitude of the initial error $\| \hat{S}_{pass} - \hat{S}_{des} \|_2$. For each initial error interval of size 0.025, the mean value (grey circle) and standard deviation (black line) of the output error has been computed: as expected, while the output error is higher for higher initial error, there is on average an improvement of over 70%.

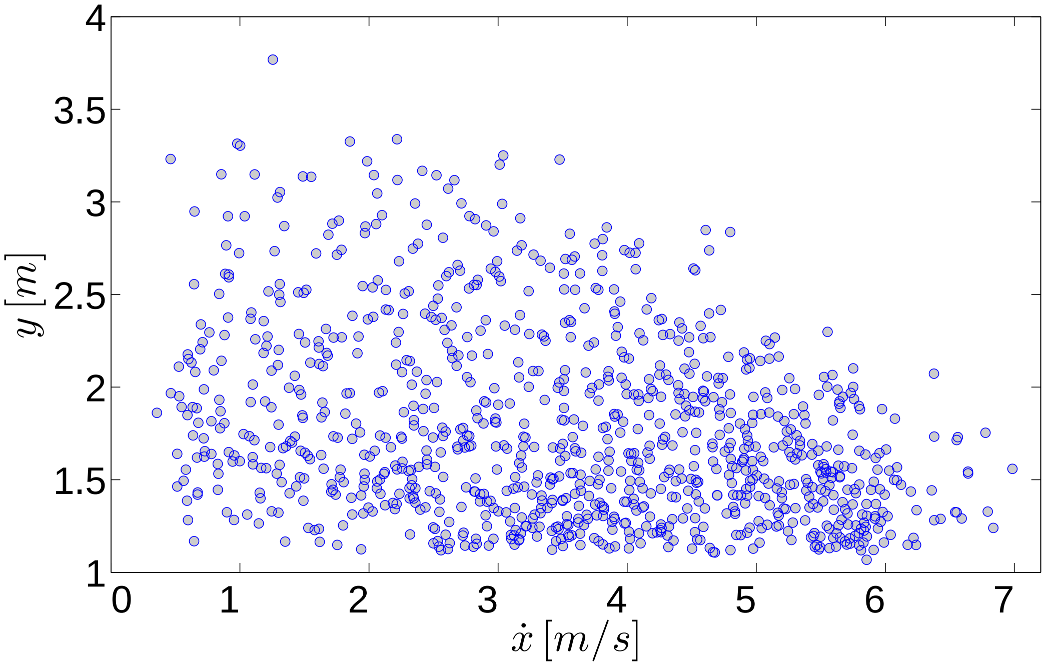


Figure 6.17: This plot shows the $1,000$ hops (initial conditions) used to test the controller. The initial apex state has been chosen as $S_0 = \{1.3\,[\mathrm{m}],\, 4\,[\mathrm{m/s}]\}$. At each hop, the desired apex state has been chosen by adding to the passively reached state an offset drawn from a uniform distribution between $[-.3, , .3]$ for y and $[-0.46,\, 0.46]$ for $\dot{x}$. These bounds have been chosen based on the reachability set previously computed for the initial condition S_0. However, note that each hop drives the system to a new apex state, hence a new reachability set, whose computation is expensive and not feasible in real time. It is therefore possible at any given hop that the desired apex is not in the corresponding reachability set.

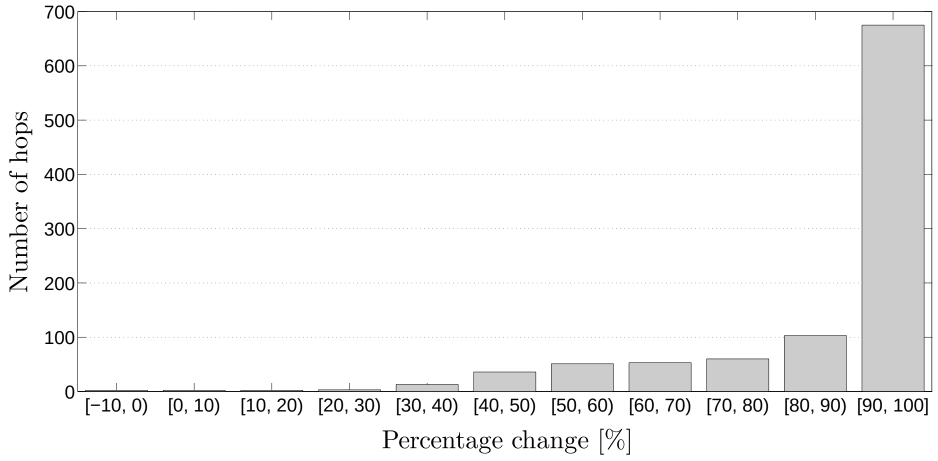


Figure 6.18: In this figure the resulting percentage change values are shown: as we can see, 675 points have a $PC \geq 90$, i.e., the initial distance to the desired apex is reduced of more than 90%. Overall, 832 states over $1,000$ have an improvement above 70%. Note that two states have negative percentage change, meaning that in these two cases our algorithm failed at improving the initial error, due to numerical approximation introduced by the algorithm.

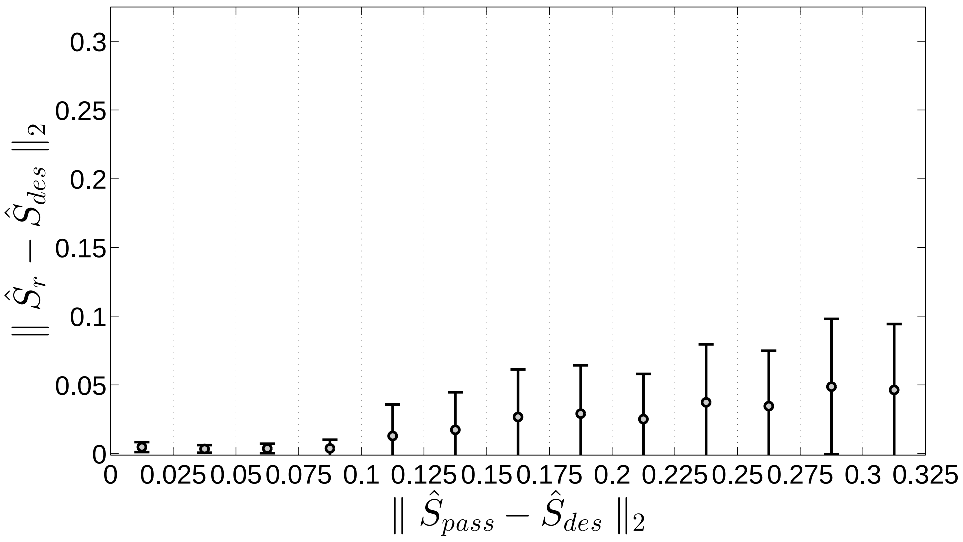


Figure 6.19: Plot of the initial error versus the final error. Each grey circle represents the mean value of the final error for all the initial errors in the corresponding interval, i.e., between $[0, 0.025)$, $[0.025, 0.05)$, etc. The black vertical lines are the corresponding standard deviations.

6.4 Conclusions

In this chapter, we have proposed a control algorithm for the actuated SLIP model to reach a desired apex state. Its main application is to correct errors happening at touch-down and/or during the stance phase, independent of any foot placement strategy employed. Most of the control strategies developed in the past for the SLIP model (whether passive or active) are aimed at controlling the touch-down position of the leg, θ_{TD}, and/or the leg length at touch-down, in order to reach a stable gait even on unknown or partially known terrain, or in presence of system noise. For the actuated case, this choice was paired with actuation displacement during the stance phase, and the desired actuation was precomputed during flight based on the knowledge on the terrain. Conversely, our strategy focuses on reaching a desired apex state without any insight on the choice of the touch-down angle: the actuation is computed online during the stance phase, based on the particular state of the system at each time interval. The main advantage of this procedure is not only the ability to counteract sensing errors at landing (e.g., ground sensing noise), which is a goal taken into consideration by other works as well, but especially the ability to reduce errors and disturbances that happen during stance. For example, a slipping of the foot, or an external force, such as a strong wind. Furthermore, we want to point out that our strategy does not aim to replace any leg-placement strategy: on the contrary, it is reasonable to think that our controller can easily be paired with any other leg-placement or path planning method.

Another contribution of this chapter is a qualitative characterization of the effects of the actuator motion on the reachable apex state. Throughout the stance phase, computing the spatial relationship between the current reachable state without further actuation and the desired state gives a strong indication of how to provide additional actuation to the system.

Finally, by delineating the reachability space for a given apex state and touch-down angle, this chapter reveals the advantage of utilizing the actuation at its maximum capabilities. In fact, we show that updating the actuation values throughout the stance phase allows the system to reach a wider range of evenly distributed apex states with respect to the ones reached by pre-computing some fixed actuator motions.

Chapter 7

Conclusions and future work

In this thesis we considered the SLIP model, used as a template for modelling the dynamics typical of legged systems. In particular, we considered an actuated version of the SLIP, with an added series elastic actuator to the leg, serving the purpose of adding/removing energy to/from the system. While the SLIP model has been a topic of research in legged locomotion for several decades, studies on the effect of actuation on the system's behavior are still not complete. Additionally, most of the proposed control strategies focus on steady-state gaits and are tuned over a set of initial conditions or terrain profiles. The purpose of this thesis is to fill the gap, and give a characterization of the reachable space for the active SLIP, paired with real-time actuation strategies to maximize the reachable space and drive the system to a desired state, with the ability to change stride at each step.

Chapter 3 characterized the reachability space of the actuated SLIP model by activating the series elastic actuator at any possible time during the stance phase. Starting from the same initial conditions at apex, for each possible touch-down angle the reachable space is shown to be a set of 2-dimensional manifolds, one per touch-down angle, forming a fan-shaped 3-dimensional volume.

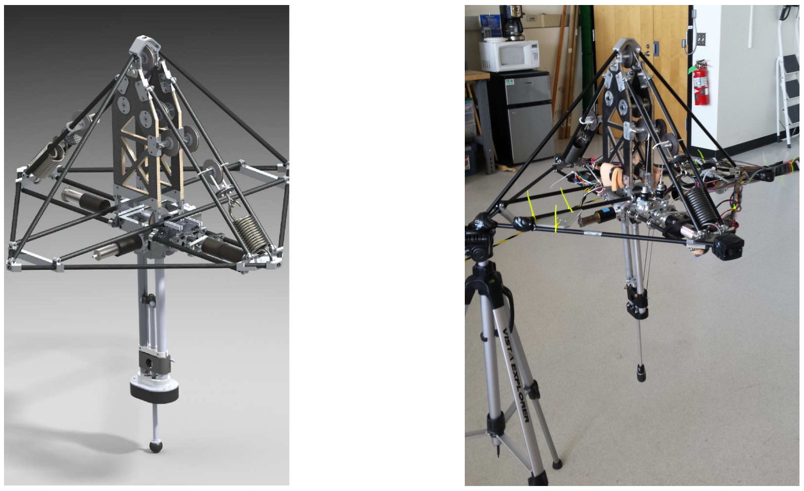
In Chapter 4, we investigated the possibility to use actuation to solve in closed-form the system's dynamics. This has been done by enforcing a desired trajectory for the dynamics during stance phase. The actuator only acts along the leg of the system, and therefore it is not possible to independently enforce a trajectory for both degrees of freedom (leg angle and leg length). The desired trajectory has then been chosen to be consistent with the system dynamics. In particular, the desired trajectory has been chosen to achieve a symmetric motion, i.e., apex height and forward velocity are preserved from one jump to the next. Enforcing a symmetric trajectory limits the reachable space, which becomes a point in the 3-dimensional space. Therefore we proposed to lock the spring during stance to reach an asymmetric take-off state, thus combining the ability to having a closed-form solution of the stance phase with the possibility to reach several points in the state space.

Chapter 5 aimed to extend the concept of utilizing the series elastic actuator to solve the system's dynamics. Via partial feedback linearization we were able to fully solve the dynamics of the leg-length, $\ell(t)$, and we then proposed an approximation of the leg-angle equation that took into account the actuator's dynamics. The main benefit of having an approximation for the stance phase dynamics is the reduced computational time to predict the system's behavior and the effect that diverse actuation strategies have on the next apex reached. We then proposed a two-element control strategy to apply two actuation references throughout the stance phase, thus driving the system to a desired apex state. The advantage of proposing a two-element control strategy with respect to other strategies proposed in the past (for example, Raibert's thrust at mid stance) lays in the fact that it is possible to reach a wider number of apex states in one step. Also, by having an approximation of the stance phase, we can pre-compute optimal trajectories.

Finally, in Chapter 6 we studied the effect that applying actuation at different times during stance has on the overall reachable region, and on independently controlling each of the two apex dimensions (height and forward velocity). Based on an understanding of how to change the next apex through actuation, we developed an algorithm to iteratively re-direct the dynamics throughout the entire stance phase to achieve a desired apex state, if reachable, or decrease the distance to it. We also showed how this strategy maximizes the reachable set with respect to other actuation strategies, and since it is iteratively computed throughout the stance phase, it can deal with errors or adverse events happening during motion.

Overall, the main contribution of this thesis is to understand the implications that adding a series elastic actuator have on the system's dynamics, and particularly on the system's reachable space, and to determine closed-form solutions for the stance phase. Based on this understanding, we proposed real-time control strategies for actuator's motion that drive the system to a desired state, and that can be applied to a variety of situations, from running on rough terrain to running on a set of desired footholds.

Future work can take several directions. The most natural, and what we are currently working on, is the application of our proposed strategies to a hardware implementation of the SLIP model. As clearly explained in [1], the ideal SLIP model is a template that can then be "anchored" to systems with more complex dynamics. The prototype presently under development and study at the UCSB Robotics Laboratory can be seen in Fig. 7.1a and 7.1b. The UCSB hopper consists of a body of mass $M = 7.6$ [kg] mounted on a compliant leg of natural length $\ell_0 = 0.54$ [m] and mass $m_{leg} = 0.55$ [kg] that can move both in the x and z direction. Actuation on the leg is applied through a series of pulleys activating a motor at the hip that compresses/extends the spring. As many other hopping/walking robots, the UCSB hopper is currently mounted on a boom to reduce lateral stability issues, and can therefore only move in the 2-dimensional space. However, it will eventually be unmounted from the boom with the goal to control it in



(a) (b)

Figure 7.1: UCSB hopper, CAD design (a), and hardware prototype (b)

the 3-dimensional space. The body is free to rotate around the hip, where a motor can apply a torque to control leg positioning and body attitude. However, to resemble the point-mass of the classical SLIP model, the body is currently locked via a mechanical stop at the base of the boom. Encoders at the leg, actuators and boom allow us to measure leg's and actuators' positions. A gyroscope on board, coupled with two distance sensors at the two corners of the body, gives us a measurement of the body tilt.

There are several challenges in applying our proposed strategy to the UCSB hopper. First of all, some parameters such as the body inertia are still undetermined and need to be computed via system identification. Also, the mechanical lock is applied at the boom, thus locking the body on the boom side. However, on the hopper side the body is still free to vibrate a few degrees, both due to impact and robot movements as well as to torsional dynamics of the boom. Sensor noise is another issue, coupled with computing the exact instances at which touch-down and take-off happen. The control strategies proposed and the reachable space then have to be adapted to take into account the additional complexity of this system and the actuator's dynamics versus the ideal SLIP model.

Other directions for this work involve extending our control strategies to optimize energy expenditure, as well as expanding our strategies to include optimal leg placement at touch-down. It would be interesting as future work to quantify the range of terrain where the SLIP model successfully works (in terms of metastability and/or energetic requirements). For example, there can be terrains where a hopping gait is optimal, but on other terrain profiles, two legs and a walking gait would be more efficient. We can then foresee the possibility and the benefits of combining the hopping SLIP model with a compass gait like robot, in order to be able to transition between walking and running/hopping and negotiate virtually any terrain profile.

Appendix A

Appendix

A.1 Planar one-legged hopper

The approximation for the stance phase dynamics discussed in Section 5.1 can be extended and used to predict the dynamics of more realistic planar one-legged hoppers. In particular, we show here how our approximation can be applied to the system shown in Fig. 5.19 and described in Section 5.5.4.

The equations of motion for this one-legged hopper can be easily computed from its Lagrangian, with non conservative forces given by the torque at the hip, τ_{hip}. During flight, the center of mass of the system follows a ballistic trajectory, and the servo at the hip moves the body to drive the leg to a desired touch-down angle. The initial conditions for the leg-angle and leg-length dynamics during the stance phase depend on the state of the body at touch-down, $x_{body}(t_{TD})$, $y_{body}(t_{TD})$, and their velocities. These values can be approximated from the touch-down state of the center of mass as:

$$\begin{aligned}
x_{body}(t_{TD}) &\approx x_{com}(t_{TD}) + \frac{m(\ell_0 - \ell_l)}{M+m}\sin\theta_{TD},\\
y_{body}(t_{TD}) &\approx y_{com}(t_{TD}) + \frac{m(\ell_0 - \ell_l)}{M+m}\cos\theta_{TD},\\
\dot{x}_{body}(t_{TD}) &\approx \dot{x}_{com}(t_{TD}),\\
\dot{y}_{body}(t_{TD}) &\approx \dot{y}_{com}(t_{TD}).
\end{aligned}$$

The energy loss at impact is taken into account at take-off as follows:

$$\begin{aligned}
\dot{x}_{com}(t_{TO}) &\approx \frac{M}{M+m}\dot{x}_{body}(t_{TO}) + \frac{m\ell_l}{M+m}\sin(\theta_{TO})\dot{\theta}_{TO},\\
\dot{y}_{com}(t_{TO}) &\approx \frac{M}{M+m}\dot{y}_{body}(t_{TO}) + \frac{m\ell_l}{M+m}\cos(\theta_{TO})\dot{\theta}_{TO}.
\end{aligned}$$

During stance, the partial feedback linearization (5.1) is applied to the system; hence, the closed form solution for the leg length ℓ derived in section 5.1 is still valid. The series elastic actuator as a function of input current $u_{curr}(t)$ is modelled as:

$$\ddot{\ell}_{act} = \frac{1}{m_{eff}}(-b\dot{\ell}_{act} - \kappa u(t)) - f\text{sign}(\dot{\ell}_{act}), \tag{A.1}$$

where m_{eff} is the effective mass seen by the actuator, b is the damping constant, f is the friction constant, and κ is the motor constant.

In general, the non-zero momentum of the body affects the dynamics of the leg-angle, $\theta(t)$. Then, θ can be approximated with the solution computed in section 5.1, θ_{appr}, with the addition of the term that includes the effect of the body motion over the leg angle, θ_ϕ:

$$\theta(t) = \theta_{appr}(t) + \theta_\phi(t), \tag{A.2}$$

where

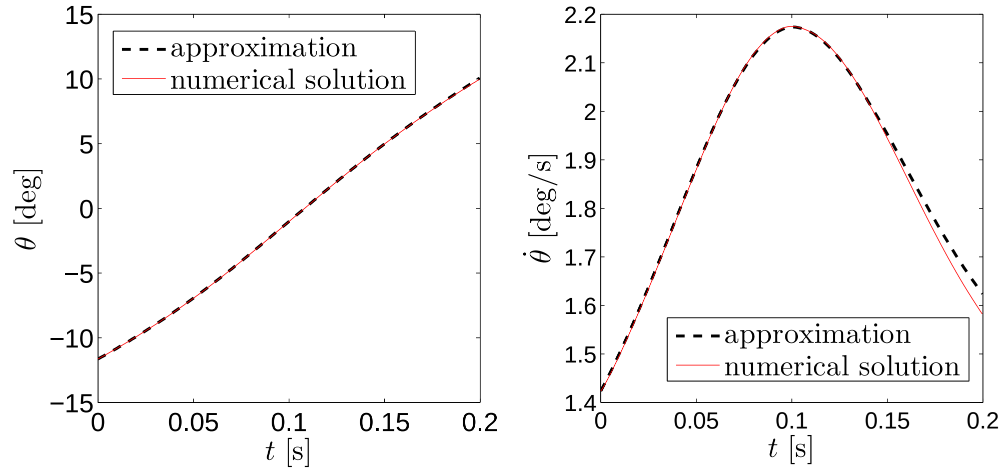
$$\theta_\phi = \frac{J(M+m)}{J_l(M+m) + Mm(\ell - \ell_1)^2}\ddot{\phi}.$$

From the equations of motion, we have that $J\ddot{\phi} = \tau_{hip}$; hence, by choosing an appropriate torque at the hip, θ_ϕ can be analytically solved. In particular, to avoid excessive tipping of the body, τ_{hip} has been chosen as the constant value necessary to keep the body upright. The solution for $\theta(t)$ computed in (A.2) is an approximation. During stance, a high level PID controller for τ_{hip} is applied to control the system to follow the target dynamics (A.2).

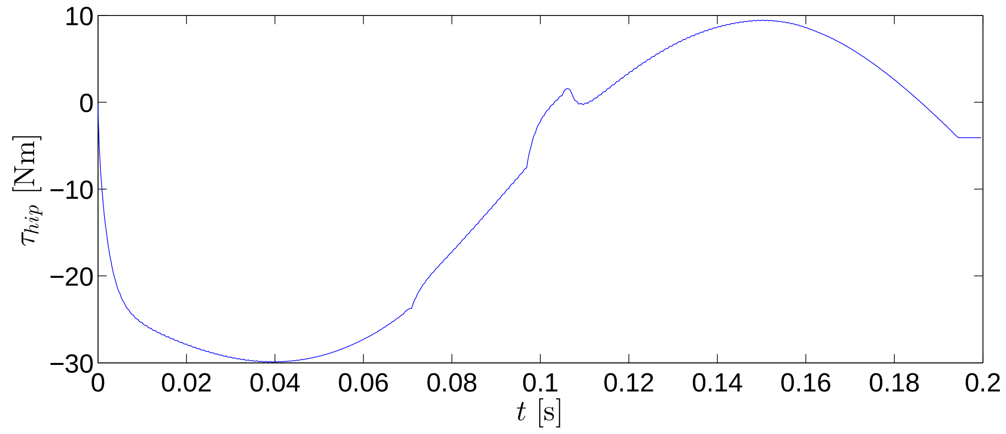
Simulation Parameters	
$M =$	10 kg
$m =$	1 kg
$J =$	10 kg $\cdot$ m^2
$J_l =$	1 kg $\cdot$ m^2
$\ell_0 =$	1 m
$\ell_l =$	0.5 m
$\ell_{act} \in$	$[-0.1, 0.1]$ m
$\ell_c \in$	$[-0.05, 0.05]$ m
$v_{act} \leq$	1 m/s
$v_c =$	0.5 m/s
$\gamma =$	20
$u_{curr} \in$	$[-20, 20]$ A
$\tau_{hip} \in$	$[-50, 50]$ N $\cdot$ m
$b =$	50
$f =$	5
$\kappa =$	68.8
$m_{eff} =$	10 kg

Table A.1:

This approximation has been tested on a path of 100 jumps with varying target apex states, as described in 5.5.4, with parameters from Table A.1. Fig. A.1a show an example of the approximated dynamics of the leg angle during stance compared to their numerical solution, and A.1b the hip torque required. Fig. A.2 shows the series elastic actuator motion, ℓ_{act} and its two components: ℓ_{nl} and ℓ_c, and the current required u_{curr}.



(a)



(b)

Figure A.1: (a) Data showing the approximated solution of the dynamics of $\theta(t)$ and $\dot{\theta}(t)$ compared to the numerically computed solution. Fig. (b) shows the required torque at the hip.

In Fig. A.3a we can see the approximation of the trajectory of the body mass compared to the numerical computed solution, while Fig. A.3b show the approximation and numerical solution of the forward velocity of the center of mass, for a few jumps.

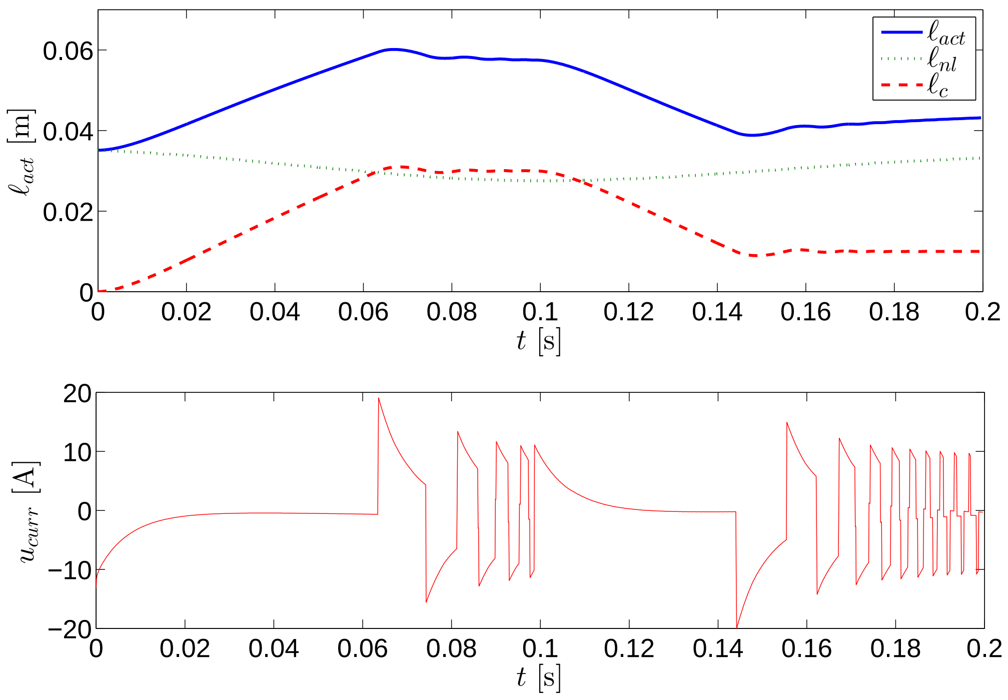
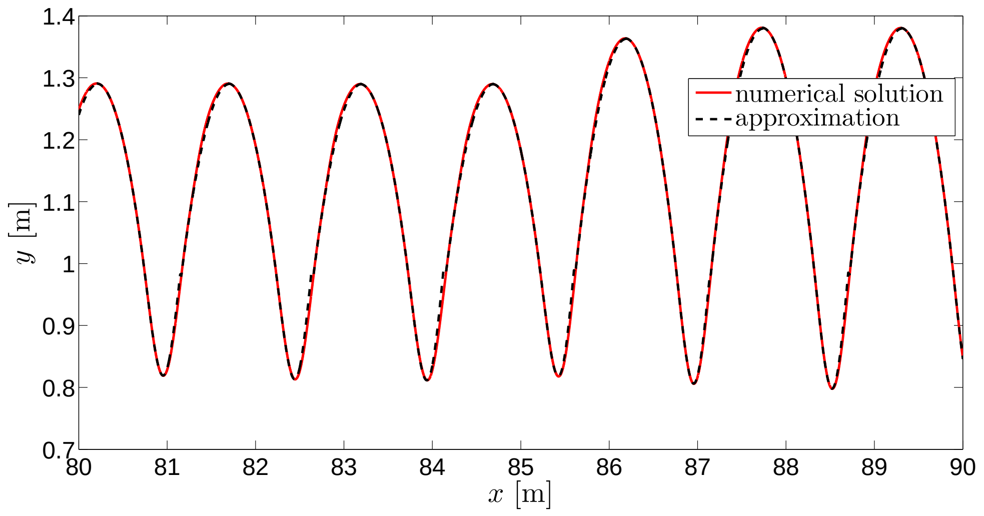
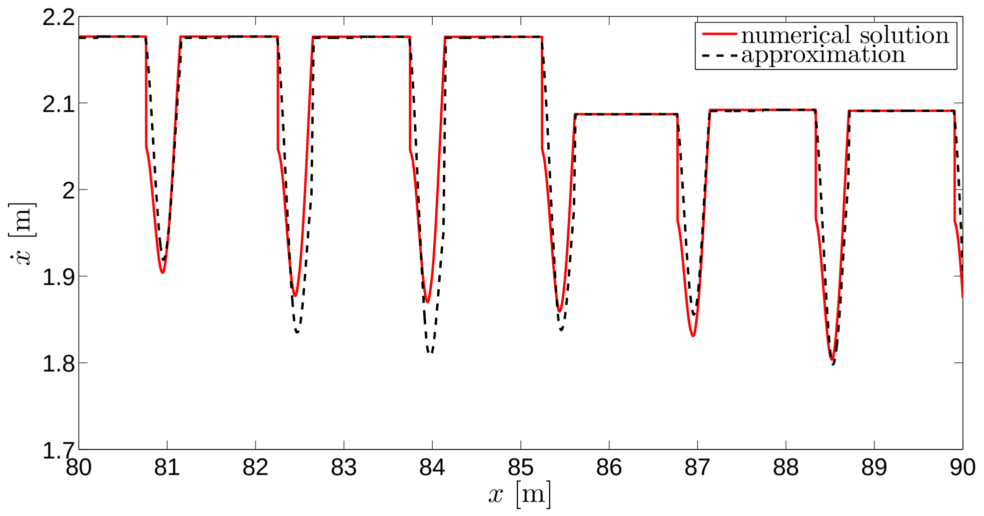


Figure A.2: Series elastic actuator motion $\ell_{act}(t) = \ell_{nl} + \ell_c$ and current u_{curr}. A PID controller drives the actuator dynamics (A.1) to the target values. The constant values desired during the first and second half of the stance phase are, respectively, $\ell_{c1} = 0.03$, and $\ell_{c2} = 0.01$.



(a)



(b)

Figure A.3: Approximation of the trajectory (a) and forward velocity (b) of the mass compared to the numerically computed solution. The apparent mismatch during the stance phase in $\dot{x}$ is due to the fact that the energy loss at impact is modelled at take-off in the approximated solution.

Bibliography

[1] R. J. Full and D. E. Koditschek, *Templates and anchors: Neuromechanical hypotheses of legged locomotion*, *Journal of Experimental Biology* **202** (1999) 3325–3332.

[2] R. Blickhan, *The spring-mass model for running and hopping*, *Journal of Biomechanics* **22** (1989), no. 11/12 1217–1227.

[3] R. Blickhan and R. J. Full, *Similarity in multilegged locomotion: Bouncing like a monopode*, *Journal of Comparative Physiology A: Neurothology, Sensory, Neural, and Behavioral Physiology* **173** (Nov., 1993) 509–517.

[4] J. Schmitt and P. Holmes, *Mechanical models for insect locomotion: dynamics and stability in the horizontal plane i. theory*, *Biological Cybernetics* **83** (2000), no. 6 501–515.

[5] J. Schmitt and P. Holmes, *Mechanical models for insect locomotion: dynamics and stability in the horizontal plane ii. application*, *Biological Cybernetics* **83** (2000), no. 6 517–527.

[6] R. Full, C. Farley, and J. Winters, *Musculoskeletal dynamics in rhythmic systems: A comparative approach to legged locomotion*, in *Biomechanics and Neural Control of Posture and Movement* (J. Winters and P. Crago, eds.), pp. 192–205. Springer New York, 2000.

[7] M. A. Daley, G. Felix, and A. A. Biewener, *Running stability is enhanced by a proximo-distal gradient in jointy neuromechanical control*, *Journal of Experimental Biology* **210** (2007) 383–394.

[8] D. Dudek and R. J. Full, *Passive mechanical properties of legs from running insects*, *Journal of Experimental Biology* **209** (2006) 1502–1515.

[9] S. Sponberg and R. J. Full, *Neuromechanical response of musculo-skeletal structures in cockroaches during rapid running on rough terrain*, *Journal of Experimental Biology* **211** (2008) 433–446.

[10] M. H. Raibert, *Legged Robots that Balance.* MIT Press, 1986.

[11] G. Zeglin, *The Bow Leg Hopping Robot*. PhD thesis, Carnegie Mellon University, Pittburgh, PA, USA, Oct., 1999.

[12] M. Ahmadi and M. Buehler, *Stable control of a simulated one-legged running robot with a hip and leg compliance*, *IEEE Transactions on Robotics* **13** (1997), no. 1.

[13] J. Schmitt and J. Clark, *Modeling posture-dependent leg actuation in sagittal plane locomotion*, *Bioinspiration and Biomimetics* **4** (2009) 1–17.

[14] B. Andrews, B. Miller, J. Schmitt, and J. E. Clark, *Running over unknown rough terrain with a one-legged planar robot*, *Bioinspiration & Biomimetics* **6** (2011), no. 2 026009.

[15] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl, *Nonlinear model predictive control for rough terrain hopping*, in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (Villamoura, Algarve, Portugal), pp. 1859–1864, Oct., 2012.

[16] K. Byl, M. Byl, M. Rutschmann, B. Satzinger, L. van Blarigan, G. Piovan, and J. Cortell, *Series-elastic actuation prototype for rough terrain hopping*, in *IEEE International Conference on Technologies for Practical Robot Applications*, pp. 103–110, 2012.

[17] J. Seipel and P. Holmes, *A simple model for clock-actuated legged locomotion*, *Regular and Chaotic Dynamics* **12** (2007), no. 5 502–520.

[18] H. R. Vejdani and J. W. Hurst, *Swing leg control for actuated spring-mass robots*, in *Int. Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, (Baltimore, MD, USA), pp. 536–542, 2012.

[19] S. Riese and A. Seyfarth, *Stance leg control: variation of leg parameters supports stable hopping*, *Bioinspiration & Biomimetics* **7** (2012), no. 1 016006.

[20] A. Seyfarth, H. Geyer, M. Günther, and R. Blickhan, *A movement criterion for running*, *Journal of Biomechanics* **35** (2002) 649–655.

[21] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, *A simply stabilized running model*, *SIAM Rev.* **47** (Mar., 2005).

[22] A. Seyfarth, H. Geyer, and H. Herr, *Swing-leg retraction: a simple control model for stable running*, *Journal of Experimental Biology* **206** (2003) 2547–2555.

[23] J. G. D. Karssen, M. Haberland, M. Wisse, and S. Kim, *The optimal swing-leg retraction rate for running*, in *IEEE Int. Conf. on Robotics and Automation*, (Shangai, China), pp. 4000–4006, May, 2011.

[24] R. Altendorfer, D. E. Koditschek, and P. Holmes, *Stability analysis of legged locomotion models by symmetry-factored return maps*, *International Journal of Robotics Research* **23** (2004), no. 10–11 979–999.

[25] M. Ernst, H. Geyer, and R. Blickhan, *Spring-legged locomotion on uneven ground: a control approach to keep the running speed constant*, in *12th International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 639–644, 2009.

[26] M. Ahmadi and M. Buehler, *Controlled passive dynamic running experiments with the ARL-Monopod II*, *IEEE Transactions on Robotics* **22** (2006), no. 5.

[27] J. K. Hodgins and M. H. Raibert, *Adjusting step length for rough terrain locomotion*, *IEEE Transactions on Robotics and Automation* **7** (1991), no. 3.

[28] D. Koepl and J. Hurst, *Impulse control for planar spring-mass running*, *Journal of Intelligent and Robotic Systems* **74** (2014) 589–603.

[29] Ö. Arslan and U. Saranlı, *Reactive planning and control of planar spring-mass running on rough terrain*, *IEEE Transactions on Robotics* **28** (2012), no. 3 567–579.

[30] E. T. Whittacker, *A treatise on the analytical dynamics of particles and rigid bodies.* Cambridge University Press, New York, fourth ed., 1904.

[31] W. J. Schwind and D. E. Koditschek, *Approximating the stance map of a 2 DOF monoped runner*, *Journal of Nonlinear Science* **10** (2000), no. 5 533–588.

[32] H. Geyer, A. Seyfarth, and R. Blickhan, *Spring-mass running: simple approximate solution and application to gait stability*, *Journal of Theoretical Biology* **232** (Feb., 2005) 315–328.

[33] Ö. Arslan, U. Saranlı, and Ö. Morgül, *Approximate stance map of the spring mass hopper with gravity Correction for Nonsymmetric Locomotions*, in *IEEE Int. Conf. on Robotics and Automation*, (Kobe, Japan), pp. 2388–2393, May, 2009.

[34] H. Yu, M. Li, and H. Cai, *Approximating the stance map of the SLIP runner based on perturbation approach*, in *IEEE Int. Conf. on Robotics and Automation*, (Saint Paul, MN, USA), pp. 4197–4203, May, 2012.

[35] Z. Shen and J. Seipel, *A piecewise-linear approximation of the canonical spring-loaded-inverted-pendulum model of legged locomotion*, *Journal of Computational and Nonlinear Dynamics* (January 2015).

[36] J.-C. Zufferey, *First jumps of the 3d bow leg hopper*, Master's thesis, E'cole Polytechnique Fe'de'rale de Lausanne, and Carnegie Mellon University, 2001.

[37] G. Zeglin and H. B. Jr., *First hops of the 3d bow leg*, in *Int. Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 357–364, 2002.

[38] U. Saranli, M. Buehler, and D. E. Koditschek, *RHex: A simple and highly mobile hexapod robot*, *International Journal of Robotics Research* **20** (2001) 616–631.

[39] P. Gregorio, M. Ahmadi, and M. Buehler, *Design, control, and energetics of an electrically actuated legged robot*, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **27** (Aug, 1997) 626–634.

[40] J. W. Hurst, J. Chestnutt, and A. Rizzi, *Design and philosophy of the BiMASC, a highly dynamic biped*, in *IEEE Int. Conf. on Robotics and Automation*, (Roma, Italy), Apr., 2007.

[41] B. Dadashzadeh, H. R. Vejdani, and J. Hurst, *From template to anchor: A novel control strategy for spring-mass running of bipedal robots*, in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (Chicago, IL, USA), pp. 2566–2571, Sept., 2014.

[42] T. Lejeune, P. Willems, and N. Heglund, *Mechanics and energetics of human locomotion on sand*, *Journal of Experimental Biology* **201** (1998) 2071–2078.

[43] Ö. Arslan, U. Saranlı, and Ö. Morgül, *Reactive footstep planning for a planar spring mass hopper*, in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (St. Louis, USA), Oct., 2009.

[44] G. Piovan and K. Byl, *Enforced symmetry of the stance phase for the spring-loaded inverted pendulum*, in *IEEE Int. Conf. on Robotics and Automation*, (Saint Paul, MN, USA), pp. 1908–1914, May, 2012.