Beyond Basins of Attraction: Quantifying Robustness of Natural Dynamics

Steve Heim and Alexander Spröwitz

***Abstract*—Properly designing a system to exhibit favorable natural dynamics can greatly simplify designing or learning the control policy. However, it is still unclear what constitutes favorable natural dynamics and how to quantify its effect. Most studies of simple walking and running models have focused on the basins of attraction of passive limit cycles and the notion of self-stability. We instead emphasize the importance of stepping beyond basins of attraction. In this paper, we show an approach based on viability theory to quantify robust sets in state-action space. These sets are valid for the family of all robust control policies, which allows us to quantify the robustness inherent to the natural dynamics before designing the control policy or specifying a control objective. We illustrate our formulation using spring-mass models, simple low-dimensional models of running systems. We then show an example application by optimizing robustness of a simulated planar monoped, using a gradient-free optimization scheme. Both case studies result in a nonlinear effective stiffness providing more robustness.**

***Index Terms*—Legged locomotion, nonlinear dynamical systems, viability, robust control, robot learning, robot control, physical design.**

I. Introduction

ANIMALS are not only agile and efficient, but also remarkably adaptable and robust [1], [2], with arguably simple control and morphology [3]–[5]. Reproducing this performance in legged robots has been difficult. Most robots use sophisticated algorithms [6]–[9], which rely on accurate models and state estimation at a substantial computational cost. This reliance tends to make model-based approaches brittle.

Recently, there have been attempts to combine these approaches with machine learning to improve robustness and adaptability [10]–[12]; however, it is notoriously difficult to apply learning directly in hardware. We are motivated by the question "how should a legged robot be designed, such that it is easier to apply model-free learning directly in hardware?" A key part of the answer is the *inherent robustness of the natural dynamics* of the system.

Indeed, designing a system with favorable natural dynamics can simplify the control problem [13]–[17] and enable quick learning directly in hardware [18], [19]. It is, however, still unclear how to quantify and evaluate the effects of design choices on the control problem, especially in terms of robustness and ease of designing or learning the control policy. After a robot is deployed successfully, it is difficult to distinguish what is due to the mechanical design, controller design, implementation, or other factors. Designers must instead rely on experience and intuition.

Many studies of natural dynamics focus on the concept of self-stability and the basins of attraction of passively stable limit cycles [20]–[23] or open-loop stable limit cycles [24]. In this study, we advocate the importance of stepping away from thinking in terms of limit cycles and their basins of attraction. We present a formulation grounded in viability theory, which allows us to quantify the inherent robustness of the natural dynamics, prior to specifying the control policy parameterization or control objective.

A. Natural Dynamics and Spring Mass Models

Perhaps the clearest example of natural dynamics is Tad McGeer's passive dynamic walker [25]: this purely mechanical system with no sensors or actuators, and hence no control, exhibits passively stable limit cycles for downhill walking. This idea has been extended in several robots, adding a little actuation and control to allow walking on level ground [26], [27] and to increase the basin of attraction of the passively stable limit cycle. A key concept is to *exploit the natural dynamics*. The intuition behind this concept is that the control can be "lazy": if a perturbation pushes the system out of the basin of attraction, the control should guide it back in. Once the state is inside the basin of attraction, the control can allow the system to naturally evolve to the attracting limit cycle.

Simulation studies of idealized walking models such as the rimless wheel [28] and compass walker [29] have provided more understanding of McGeer's empirical results. These models also have passively stable limit cycles albeit with rather small basins of attraction.

For running, we turn to a different idealized model: the spring-mass model. This simple model was initially developed by the biomechanics community to study running [30], where the spring abstracts the natural compliance of the muscle-tendon system in the leg. While the effective leg stiffness depends on many factors including muscle activation, it is modeled as a constant parameter, and thus the model has no control inputs. Thus, at the level of abstraction of the model, the natural dynamics

Manuscript received December 15, 2018; accepted March 31, 2019. Date of publication May 9, 2019; date of current version August 1, 2019. This work was made possible thanks to a Max Planck Group Leader grant awarded to A. Sprwitz by the Max Planck Society. This paper was recommended for publication by Associate Editor R. D. Gregg and Editor T. Murphey upon evaluation of the reviewers' comments. *(Corresponding author: Steve Heim.)*

The authors are with the Dynamic Locomotion Group, Max Planck Institute for Intelligent Systems, 70569 Stuttgart, Germany (e-mail: heim.steve@gmail.com; sprowitz@is.mpg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2019.2910739

seem passive even though the system may have active control embedded in it.

This simple model, also called a *template*, accurately predicts the overall behavior of many seemingly very different systems, called *anchors* [31]. Indeed, by proper parameter tuning, the spring-mass model can be used to accurately model diverse running systems, from humans [32] to cockroaches [33], bipedal [14] to hexapedal [34] robots.

B. Templates, Anchors, and Hierarchical Control

Spring-mass model templates are often used for understanding hierarchical control [31] since the template and anchor division offers a natural split in hierarchy. A high-level control policy can be designed based on a template in a low-dimensional space, while a low-level control policy based on the anchor is designed in the high-dimensional space. Thus, as long as the low-level controller enforces a template-like behavior on the system, the high-level controller design can be greatly simplified [35]–[37].

In this hierarchical context, the term natural dynamics is always relative to the level of abstraction being considered. Indeed, to a high-level control policy, there is no distinction between which part of the system behavior is truly "passive" and which has been influenced by the low-level controller.[1]

The template and anchor approach to hierarchical control has been used to develop various discrete-time high-level controllers: for example, the spring stiffness or landing angle of attack might be chosen once per step, but the continuous-time dynamics in between are left "passive" [38]–[41]. One result with this approach is that choosing an open-loop trajectory of landing angles of attack during flight can achieve deadbeat control without active control during stance [24], [42].

While these results are impressive, they generally suffer from the curse of dimensionality: they are only tractable on the low-dimensional template models. Therefore, the high-level control relies on the overall system behaving as a simpler, lower dimensional system. This is usually achieved through a combination of appropriate mechanical design, and a low-level controller that exposes a simpler dynamical behavior to the high-level controller.

There are two common approaches to low-level controller design. On the one hand, a low-level control policy can enforce the dynamical behavior of a specific template model [43]–[46]. While this approach offers more rigorous guarantees on the behavior of the high-level system, it is also generally more difficult to implement in practice.

On the other hand, the low-level control policy can be designed to produce a lower dimensional behavior without enforcing the specific template dynamics [47]–[50]. This approach requires further tuning of the high-level control policy, since it explicitly allows for a mismatch between the high-level model and the actual system behavior.

Robustness is a key indicator of how accurate a model needs to be, regardless of the approach taken: a policy that is robust will suffer less from model inaccuracies. Our main contribution is a means to quantify the robustness of the natural dynamics, prior to designing the high-level control policy, or even specifying its objective. We first illustrate the quantification on template models in a rigorous manner. We then show an example application using gradient-free optimization to find robust parameters of a low-level controller, without enforcing a specific template model. We are thus able to quantify robustness without relying on low-dimensional template models.

[1]This is equivalent to the split between agent and environment in reinforcement learning.

C. Computation of Viability

Our quantification relies on the concept of viability: a state is said to be viable if there exists a set of control actions that keeps it inside the viability kernel for all time [51]. In other words, a state that starts outside the viability kernel will fail within a finite time, regardless of the control actions applied.

There has been much interest recently in computing viable sets and its dual, back-reachable sets [52], for safe control verification and design [53]–[56], and more recently safe learning of control [57], [58]. Our contribution complements prior work by using a viability formulation to quantify robustness of the system design prior to control policy design.

Viability-based approaches share a common challenge: computing viability kernels relies on gridding the search space, making the general case intractable [53], [59].

For particular classes of systems, more efficient algorithms have been developed to find either inner or outer approximations of viable sets, which can generally be scaled to 6–10 dimensions [59]. Thus, it is often beneficial to use approximations that fit these classes and dimension restrictions. Computation of viable sets is then performed on the low-dimensional approximation, which can be tracked using a hierarchical control strategy [60], [61].

This matches well with the existing template and anchor paradigm commonly used in legged robotics. We will show an example application, in which we optimize the parameters of the low-level control policy to exhibit robust natural dynamics to a high-level control policy.

D. Notes on Terminology

We use terminology common to the reinforcement learning community, such as actions instead of control inputs and control policies instead of controllers. We will speak of control policies *sampling* an action, or the system sampling a state-action pair, to indicate the policy can be stochastic.

Much of the mathematics in the paper revolves around sets in different spaces. Capital letters such as S denote spaces (in this case the state space). Capital letters with a subscript such as S_F denote a set in the corresponding space, the meaning of the subscript being explained in the text (in this case the set of failure states).

E. Structure

In Section II, we cover the details of the two spring-mass models we examine, their dynamics, and a typical bifurcation diagram for the spring-loaded inverted pendulum (SLIP) model.

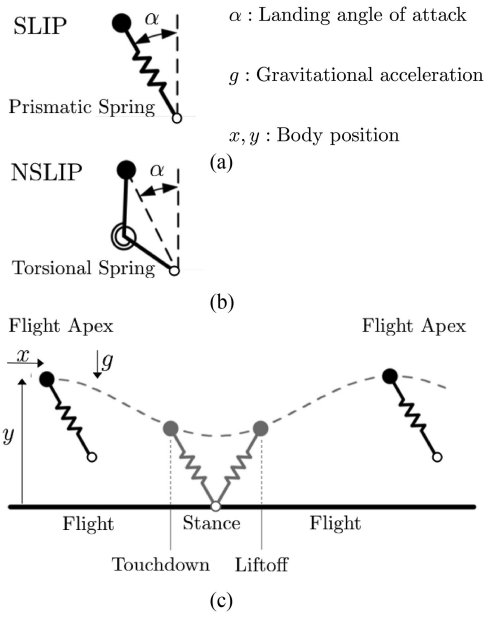

Fig. 1. We focus on two spring-mass models: (a) the SLIP model with a linear prismatic spring, and (b) the NSLIP model, with a segmented leg and a linear torsional spring. (c) A qualitative trajectory over one cycle, starting and terminating with a flight apex event.

In Section III, we compute the viability kernel as well as the transition map in state-action space. We illustrate how this encompasses the bifurcation diagram, and why bifurcation diagrams are limiting once we introduce control.

In Section IV, we introduce our definitions of robustness, and how to use this to evaluate two different designs of leg compliance prior to designing a control policy.

In Section V, we show an example application, in which the quantification developed is used as the fitness function to perform gradient-free optimization of a simulated planar monopedal robot.

In Section VI, we summarize our key contributions, open questions, and outlook.

II. Spring-Mass Models

We use two well-studied spring-mass models to illustrate our concepts: the SLIP model and a nonlinear spring mass (NSLIP) model as first studied by Rummel and Seyfarth [23] (see Fig. 1). Both models have hybrid dynamics with the governing equations of motion switching between flight and stance phases.

During the flight phase, the body follows a ballistic trajectory, whereas during the stance phase it follows a spring-mass motion, which depends on the modeled spring. The details of the equations of motion have been derived in [23] and [30], and can be found in the appendix. For convenient comparison, we use the same parameters as in [23], which are similar to human averages.

A. Discrete Analysis via Poincaré Sections

The continuous motion of the point-mass body is fully described in planar Cartesian coordinates by the state vector $[x, y, \dot{x}, \dot{y}]^\intercal$. We simplify analysis by only evaluating the state on a Poincaré section at flight apex, a common approach for cyclic motion. At flight apex, potential and kinetic energy are conveniently contained in the vertical position and forward velocity, respectively. Thus, the continuous state vector of $[x, y, \dot{x}, \dot{y}]^\intercal$ can be reduced to $[y,\ \dot{x}]^\intercal$. Taking advantage of the constant energy constraint, we can further reduce the system to a single state, the normalized apex height s, which defines our state space

$$s = \frac{E_{\text{pot}}}{E_{\text{pot}} + E_{\text{kin}}} = \frac{g\, y}{\frac{\dot{x}^2}{2} + g\, y}$$

$$\text{State Space: } s \in S = [0, 1]$$

where E_{pot} and E_{kin} are potential and kinetic energy, respectively, and g is the gravitational acceleration.

Starting from any state at apex s, we can numerically integrate the continuous time dynamics until the system either transitions to a second apex height or to a failure state. We thus obtain the Poincaré map, also called a transition map, for our discrete dynamics

$$s_{k+1} = P(s_k, \alpha)$$

where the landing angle of attack α is a model parameter of interest. We will use this as our control action in Section III.

We will consider as failures all states in which the body hits the ground with $y = 0$, as well as when the system reverses direction with $\dot{x} < 0$. More formally

$$\text{Failure Set } S_F := \{s : y = 0 \text{ or } \dot{x} < 0\}.$$

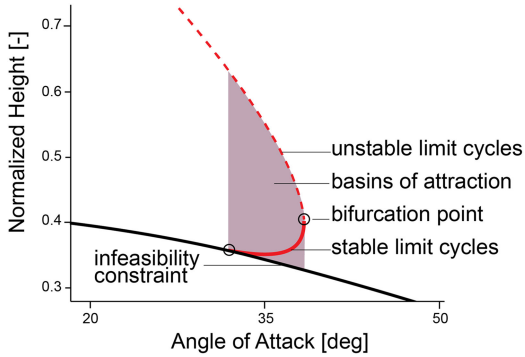

Fig. 2. Bifurcation diagram of the passive SLIP model. This diagram highlights the small range of parameters for which stable limit-cycles exist. The basins of attraction are bounded by infeasibility and unstable limit cycles. Beyond these basins of attraction, however, is a lot of structure that can be exploited through control.

B. Bifurcation Diagram of the SLIP Model

A bifurcation diagram allows the study of the existence and stability of fixed points and limit cycles, as a dependence of model parameters.

The bifurcation diagram of the SLIP model with respect to the angle of attack α is shown in Fig. 2. Similar bifurcation diagrams for spring-mass models can be found in [62], and bifurcation diagrams for spring stiffness can be found in [23] and [38].

We only evaluate period-1 limit cycles, that is when $s_{k+1} = s_k$, and do not consider orbits which require multiple

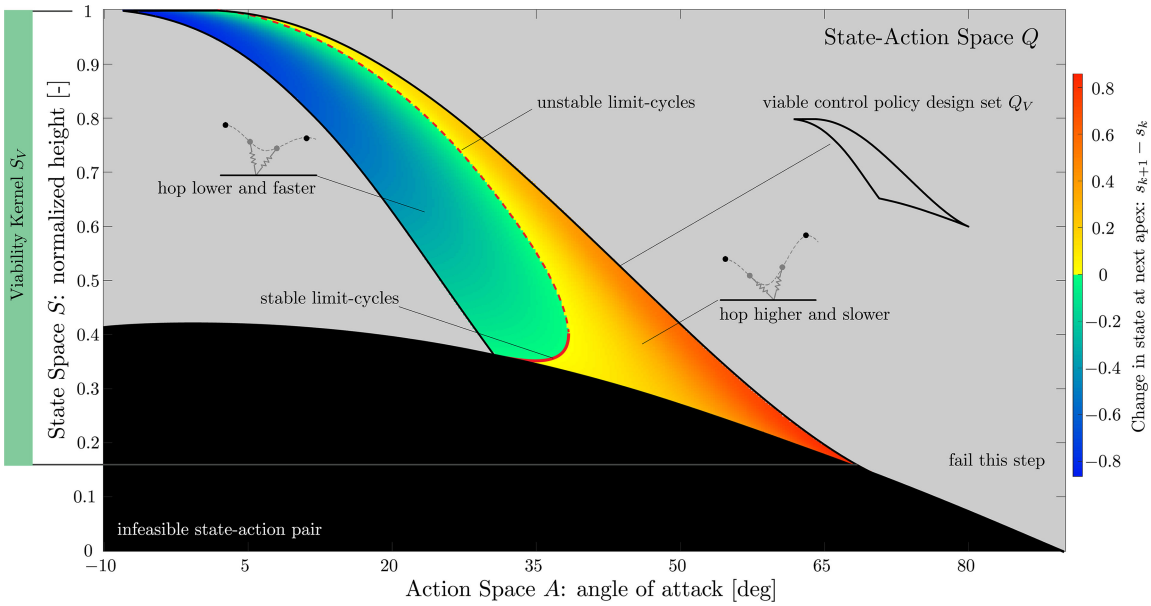

Fig. 3. Lookup table of the SLIP model's transition map. This map shows combinations of state (height at apex) and action (landing angle of attack), and their transition to either a second apex or a failure. State-action pairs in the gray region result in failure. State actions in the warm and cool colored regions result in hopping higher and lower, respectively, with the color indicating the change in state (vertical axis) at the next apex. Also marked are passively stable (solid red) and unstable (dashed red) limit cycles.

iterations to return to periodicity. Stable limit cycles are marked with a solid red line and unstable limit cycles with a dashed red line. The basins of attraction of the stable limit cycles are highlighted by the shaded area.

These basins of attraction are bounded from below by an infeasibility constraint: below this line, the foot would begin underground. The unstable limit cycles bound the basins of attraction from above: being perturbed onto an unstable limit cycle will keep the system at that new state; beyond this threshold, it will diverge until the system fails.

Since either infeasibility or unstable limit cycles bound the basins of attraction, many previous studies have been limited to identifying these bounds. The relevant range of parameters and states for studying basins of attraction tends to be narrow, as illustrated in Fig. 2. We will show in the following section that there is a lot of structure outside the basins of attraction of these passively stable limit cycles. Once we allow parameters such as the angle of attack α to be actively chosen as a control decision, the relevant bounds are no longer the bounds of the basins of attraction, but those of failure and viability.

III. Natural Dynamics and Viable Control

We begin the section by introducing control, then evaluate the effect the natural dynamics have on the set of possible control policies. A key concept is the link between the viability kernel, a set within the state space, and the set of viable state-action pairs.

A. Control Policies and State-Action Space

We now allow the landing angle of attack α to be freely chosen at each apex. This defines our action space A

$$a = \alpha$$

$$\text{Action Space: } a \in A = [-180^\circ,\ 180^\circ]$$

where a is any action in A. In the figures, we only show the relevant range, excluding the range which contains only failures or infeasible state-action pairs.

A control policy π is any function that maps a state to an action $a = \pi(s)$. As such, a policy lives in the combined state and action spaces, which we term Q-space.[2]

B. Transition Map

We compute high-resolution 800×800 grids of state-action pairs, as is commonly done for these types of problems [9], [24], [41], [56], [63]. We, thus, obtain a lookup table of the transition map $P(s_k, a_k)$, visualized in the state-action space Q in Fig. 3 for the SLIP model and in Fig. 4 for the NSLIP model.

To highlight the limit cycles, we use a color-map centered around $s_k - s_{k+1} = 0$. The warm and cool colored regions correspond to state-action pairs that result in a higher or lower state, respectively. The gray regions are state-action pairs, which result in a failure state $P(s_k, a_k) \in S_F$. The black region is composed of infeasible points in which the foot would start underground, and as such is not part of the Q-space.

We call the gray region the set of failing state-action pairs Q_F. Its complement, the colored region, is the set of non-failing state-action pairs. More formally

$$Q_N := \{(s_k, a_k) : P(s_k, a_k) \notin S_F\}. \tag{1}$$

[2]This term is chosen in reference to Q-learning in reinforcement learning.

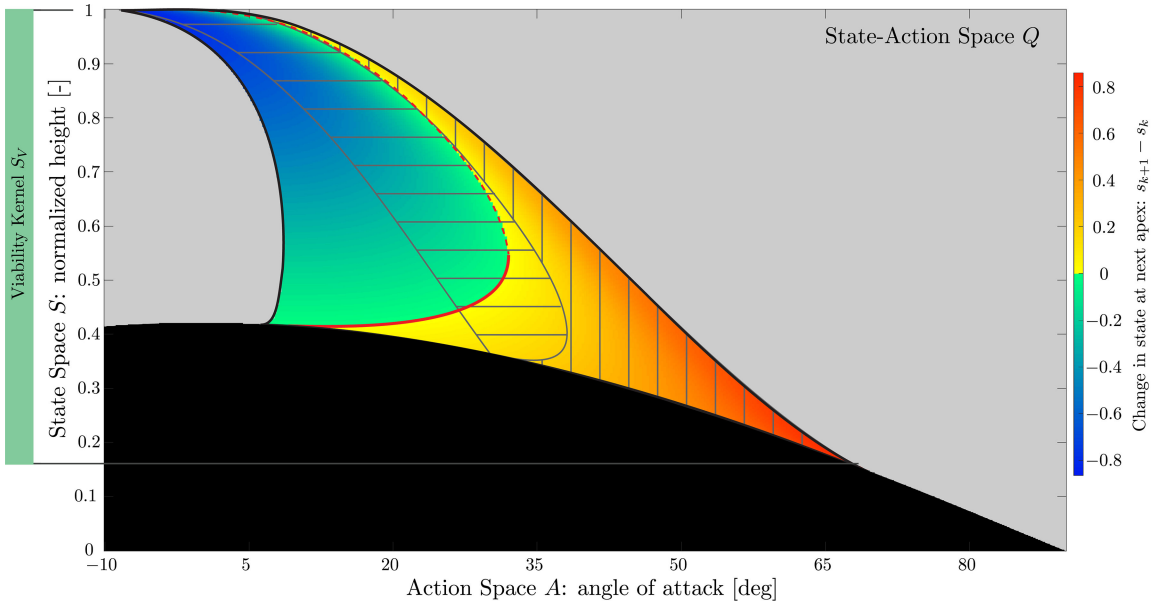

Fig. 4. Although the viability kernel S_V remains the same for both models, the size of Q_V of the NSLIP is 36% larger. This allows for more flexibility and robustness in designing a control policy for the NSLIP model. For reference, the Q_V of the SLIP model is overlaid in gray lines in horizontal and vertical for the cold and warm colored regions, respectively.

We denote the projection of Q_N onto the state space S as the set $S_N = \text{proj}_S(Q_N)$. Throughout the paper, we always use orthogonal projections, that is

$$\text{proj}_S\,(s, a) = s \tag{2}$$

S_N is the set of controllable states, from which actions that avoid immediate failure can be selected. More formally

$$S_N := \{s_k : \exists\, a_k \text{ such that } P\,(s_k, a_k) \notin S_F\}\,.$$

The upper bound between Q_N and Q_F are state-action pairs that convert all kinetic energy into potential energy in one step, resulting in a state of $s = 1$. In other words, these are the equivalent of 1-step capture points [60]. The lower bound is a boundary to falling, meaning that the point-mass hits the ground without reaching a second flight apex.

C. Viable Sets

A viability kernel is the set of all states for which there is at least one time evolution of the system which remains in the set for all time [51]. Since all state-action pairs $(s, a) \in Q_N$ result in at least a second step, all $s \in S_N$ have at least a one failure-preventing action available. However, it is possible for a nonfailing state-action pair to reach a state from which all solutions eventually reach a failed state, as was examined in [64]. In other words, there can be states from which immediate failure can be avoided, but from which the system will fail within some finite time. Thus, the viability kernel, which we will call S_V, is a subset of S_N and the set of viable state-action pairs Q_V is a subset of Q_N.

We can compute the discretized set of viable state-action pairs Q_V and its projection S_V iteratively, as in Algorithm 1. In this process, we begin with an estimated $Q_V = Q_N$ and $S_V = \text{proj}_S(Q_V)$. Then, we alternate trimming both estimates

Algorithm 1: Compute Viable Sets.

procedure VIABLE SETS(P, Q_N)
  $Q_V \leftarrow Q_N$
  $S_V \leftarrow \{\,\}$
  while $S_V \neq \text{proj}_S(Q_V)$ **do**
    $S_V \leftarrow \text{proj}_S(Q_V)$
    for all $s_{k+1} = P(s_k, a_k), (s_k, a_k) \in Q_V$ **do**
      if $s_{k+1} \notin S_V$ **then**
        Remove (s_k, a_k) from Q_V
  return $Q_V,\ S_V$

of Q_V and S_V: first, we check if any state-action pairs (s, a) in the estimated Q_V maps to a state outside of S_V and exclude these from Q_V. Then, we update the estimate of S_V as the projection of the new Q_V estimate and repeat. If the projection does not change, each state in S_V has an action available that maps back into the set and the algorithm terminates.

For the models we examine, Q_V is equal or almost equal to Q_N except in unusual corner cases.

We can now compare the resulting Q_V and S_V for the SLIP and the NSLIP models (see Fig. 5). Although the set of viable states S_V is the same in both models, the set of viable state-action pairs Q_V is much larger for the NSLIP model. This suggests unexplored benefits of nonlinear leg compliance.

D. Family of Viable Control Policies

A control policy $\pi(s)$ must sample from Q_N with nonzero probability; otherwise, it will always fail in a single step. All meaningful policies must sample from Q_V with nonzero probability, or it will always fail in finite time. In order to avoid failure from every viable state for all time, a policy must sample *exclusively* from Q_V, which we call the viable policy design space.

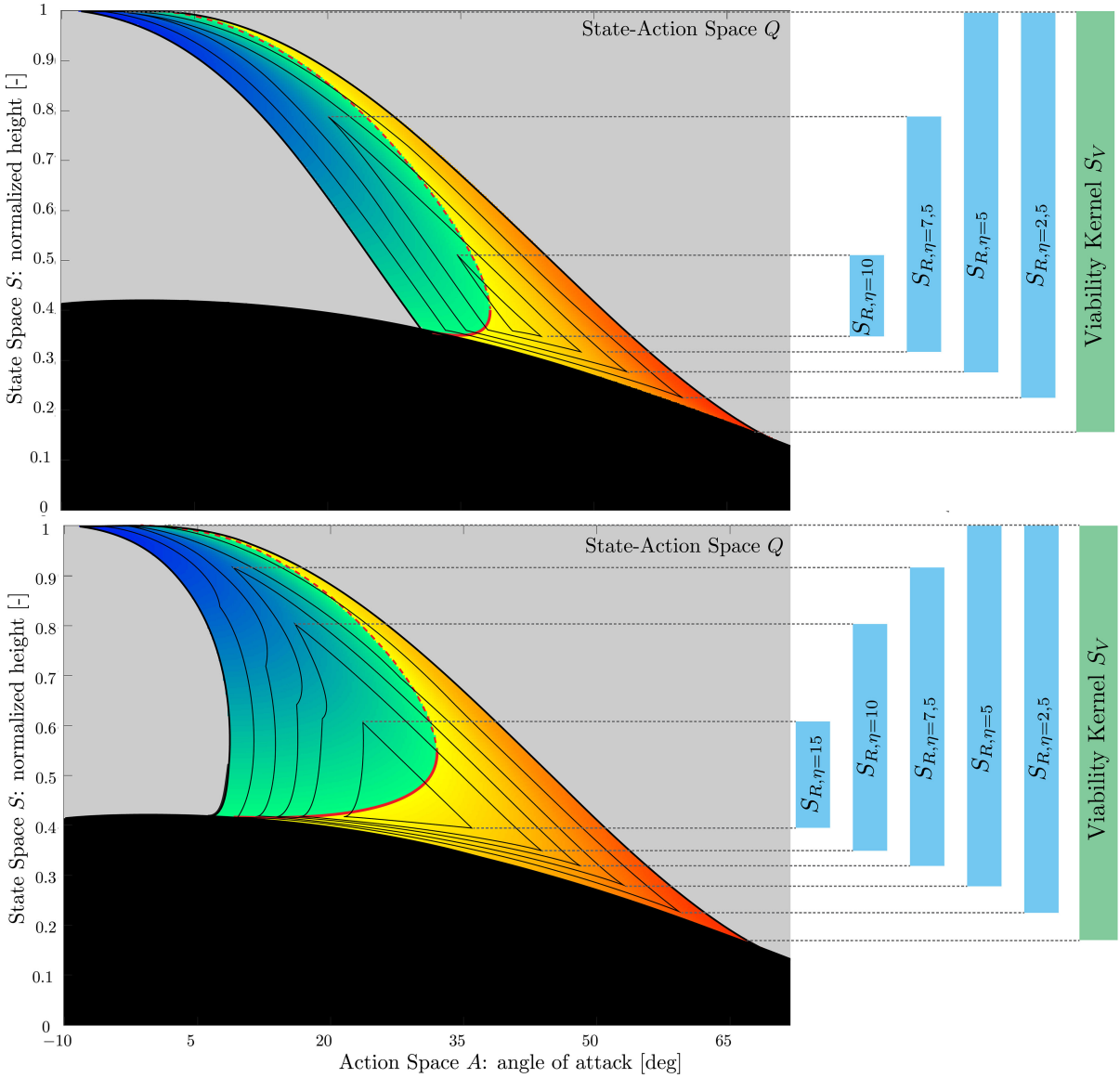

Fig. 5. Robust sets for different amounts of noise are computed for the SLIP (top) and NSLIP (bottom). The NSLIP benefits from much larger robust sets Q_R for any amount of noise, which makes it easier to design or learn a robust control policy. Also, the set of robust states S_R are not only larger for the NSLIP, but remain relatively large even for rather imprecise control.

We call the set of all such policies the *family of viable control policies*. More formally, if the set Q_V is nonempty, we also have a nonempty set of viable policies Π_V, where

$$\forall s_k \in S_V \; \exists\, \pi(s_k) \in \Pi_V,\; a_k = \pi(s_k) : \\ (s_k, a_k) \in Q_V, \text{ and } P(s_k, a_k) \in S_V \; \forall k.$$

The shape of Q_V in the dimensions of S and A poses different constraints on the control policies $\pi(s) \in \Pi_V$ that we can design. The projection of Q_V onto the dimensions of state space S is the viability kernel S_V itself.

The volume of Q_V in the dimensions of action space A, on the other hand, allows more flexibility in designing a viable control policy since more viable actions are available to choose from.

Imagine, for example, a set Q_V defined by a single line[3] covering all of S, a surjective function $f(s)$. While the viability kernel $S_V = S$ is maximal, there is exactly one deterministic control policy $\pi(s) = f(s)$, which remains viable. This can make the control policy not only difficult to design or learn, but also very sensitive to uncertainty, as we will discuss in the following section.

IV. Robust Natural Dynamics

We define robustness as the ability of a system to avoid failure in the face of uncertainty. A key objective of this paper is to evaluate the robustness inherent to the natural dynamics: we

[3] A hypersurface for arbitrary dimensional state-action space.

care about the robustness resulting from the system design, before specifying the policy parameterization or even the control objective (such as converging to a specific limit cycle).

To this end, we focus on uncertainty in action space, in other words, the effect of noise on the control policy output. We will use this as a basis to also examine robustness to perturbations in state space for the family of all robust controllers. We briefly discuss the link of action noise to state-estimation noise. We do not consider model uncertainty, and leave this to future work.

A. Computing Robust Sets

Noise in the action space causes the system to sample a state-action pair with a different action than chosen by the policy

$$a = \pi(s_k) + \eta_a \tag{3}$$

$$s_{k+1} = P(s_k,\, \pi(s_k) + \eta_a) \tag{4}$$

where η_a is some form of noise. A robust control policy needs to ensure that the chosen output never causes the system to fail despite this noise, for all time. More formally

$$\text{If } \pi(s_k) \in \Pi_R \text{ and } \eta_a \in H_a$$

$$\text{Then } s_{k+1} = P(s_k,\, \pi(s_k) + \eta_a) \notin S_F \;\forall\, k \tag{5}$$

where Π_R is the family of all robust control policies. For simplicity, we will consider noise sampled from a symmetrical bounded set $\eta_a \in H_a = [-\eta, \eta]$, where η is some finite scalar.

When considering unbounded noise (such as Gaussian noise), similar arguments hold in a probabilistic sense: instead of being able to guarantee that state-action pairs allow the system to never fail, we can only guarantee that it will not fail within a finite-time horizon with a certain probability.

The effect of action noise reduces the space available for controller design in two ways. First, the output of the control policy $\pi(s_k)$ must be sufficiently distant from failing state-action pairs, such that the added noise never causes an immediate failure. The second requirement is similar to that for viability: the system must always land in a state from which it can continue to sample robustly, for all time. More formally, we want that

$$s_k \in S_R,\; \pi(s_k) \in \Pi_R,\; \eta_a \in H_a :$$

$$P(s_k,\, \pi(s_k) + \eta_a) \in S_R \;\forall\, k. \tag{6}$$

We call Q_R the *robust control policy design set*. Similar to the relation between Π_V and Q_V, policies in the set Π_R must sample exclusively from Q_R in order to avoid failure for any state $s_k \in S_R$ where $S_R = \text{proj}_S(Q_R)$. Such sets are shown in Fig. 5 for various amounts of noise η. Each of these sets is computed with the iterative process in Algorithm 2. This is essentially the same as the algorithm for computing the viable set, while also considering additional possible transitions caused by noise. Note that, if the system dynamics have certain properties, only the worst-case noise needs to be considered [59]. Even without these properties, a worst-case only assumption is often sufficiently accurate in practice. Importantly, the computation of Q_R depends only on the set Q_V and, thus, the set of failure states S_F, the transition map P and the noise set H. It does not depend on the exact choice of policy $\pi(s_k)$, but is valid for the family of all robust control policies Π_R. In other words, we can evaluate the robustness inherent to the natural dynamics, before we design the control policy or define a control objective other than "avoid failure".

Algorithm 2: Compute Robust Sets.

procedure ROBUST SETS(P, Q_V, H)
  $Q_R \leftarrow Q_V$
  $S_R \leftarrow \{\,\}$
  while $S_R \neq \text{proj}_S(Q_R)$ **do**
    $S_R \leftarrow \text{proj}_S(Q_R)$
    for all $(s_k, a_k) \in Q_R$ **do**
      for all $\eta_a \in H_a$ **do**
        if $(s_k, a_k + \eta_a) \notin Q_R$ **then**
          Remove (s_k, a_k) from Q_R
          Break
        if $s_{k+1} = P(s_k, a_k + \eta_a) \notin S_R$ **then**
          Remove (s_k, a_k) from Q_R
          Break
  return $Q_R,\; S_R$

B. Evaluating Robustness of Different Legs

We compare the robustness of the SLIP and NSLIP models for varying amounts of noise, as shown in Fig. 5.

With the SLIP model, Q_R and S_R become empty sets for noise greater than $\pm 10.75^\circ$, whereas in the NSLIP model the upper threshold is almost twice as large, at $\pm 20.00^\circ$.

For any given amount of noise, the size of the set Q_R is also much greater for the NSLIP than for the regular SLIP model. The larger size of Q_R means there is more flexibility to fulfill robustness requirements while also designing a control policy around other criteria.

Furthermore, action noise is one of the most common methods of introducing exploration in learning, for example with Gaussian policies [65]. The amount of noise needs to be carefully balanced: more noise allows for more aggressive exploration, but it can also keep the agent from converging to the true optimum, as well as lead to unstable behaviors ending in failed states. This can be particularly troublesome for learning in hardware, requiring more samples as well as potentially damaging the robot. Robustness to action uncertainty allows for more aggressive and effective exploration during learning.

C. Robustness to State Perturbations

The projection of the robust policy design set onto state space, $S_R = \text{proj}_S(Q_R)$, is the set of robust states, from which any robust policy $\pi \in \Pi_R$ can always recover. Interestingly, with small amounts of noise up to $\eta < 5^\circ$, S_R remains the same for both the SLIP and NSLIP models (see Fig. 6). For greater amounts of noise, it shrinks much more rapidly for the SLIP model.

The set S_R is particularly useful for choosing the specific control objective. For example, if we expect perturbations in state space to have a symmetrical distribution, we would want to stabilize a limit cycle near the center of S_R. On the other

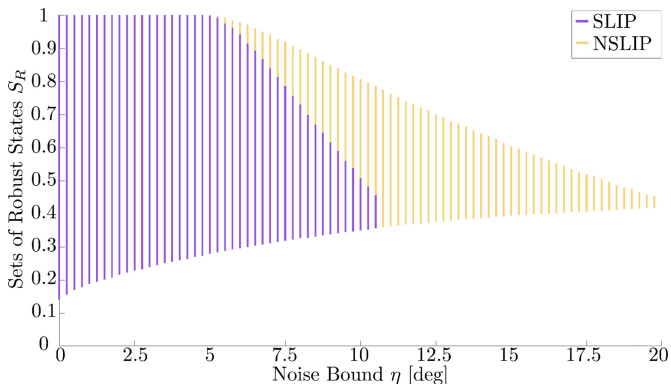

Fig. 6. The size of the sets of robust states S_R remains equal for the SLIP and NSLIP models for noise bounded to less than 5°. For greater amounts of noise, the sets shrink much more rapidly for the SLIP model.

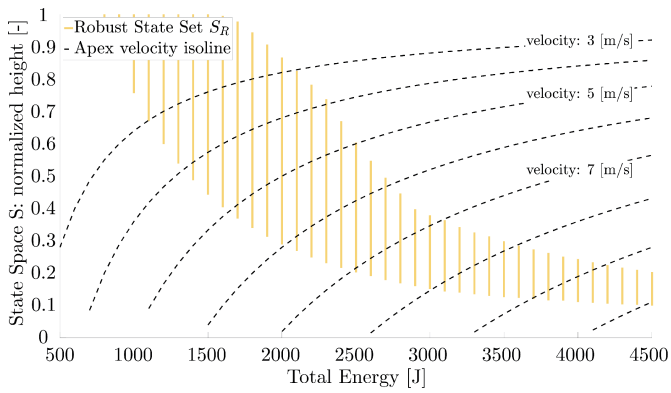

Fig. 7. S_R for different amounts of total energy for the NSLIP model, with noise fixed at $\eta = 7.5°$. For a change in ground height, the system state travels along the forward velocity isolines (dashed black). For reference, the author runs recreationally at roughly 3.2 [m/s], Eliud Kipchoge ran the Breaking2 marathon event at roughly 5.8 [m/s] and Usain Bolt holds the 100 meter dash world record at roughly 10.8 [m/s]. The simulations shown in other graphs are all for the fixed energy level of 1860 J.

hand, if we expect a specific type of perturbation to occur more frequently, we can choose a limit cycle with a larger margin in that specific direction.

As a specific example, a well-studied state perturbation is a change of ground height between steps [1], [13], [24]. This type of perturbation involves a change in total energy: the forward velocity at apex remains the same, though the effective height (and, thus, potential energy) changes. We can compute S_R at different energy levels to then pick out operating points that remain robustly controllable across different energy levels, as shown in Fig. 7. Assuming symmetric distribution of perturbations, the control objectives should be chosen to maximize the distance from the edge of the viability kernel in each direction. For a given desired forward velocity, we can choose a total energy that centers the normalized height to perturbation along the vertical axis (constant energy perturbation) and along the forward velocity isolines (ground height change).

D. Robustness to State Estimation Uncertainty

Sensory noise causes the control policy to sample an action based on a noisy estimate of the state

$$a = \pi(s + \eta_s) \tag{7}$$

where η_s is the noise in state space. There is an equivalence between η_s and η_a: the action used deviates from what a control policy would determine under perfect conditions, whether this is due to noise in action space or state estimation. This equivalence can be directly calculated using 4 and 7

$$\pi(s) + \eta_a = \pi(s + \eta_s)$$
$$\eta_a = \pi(s + \eta_s) - \pi(s).$$

If the control policy π is affine, the equivalence is trivially $\eta_a = \pi(\eta_s)$ and for bounded estimation noise η_s the equivalent action noise η_a is also bounded. Otherwise, we cannot guarantee bounds are available. Since this equivalence is dependent on the specific control policy, we do not investigate it further here. Suffice it to say, increasing robustness to action uncertainty can only improve robustness to state-estimation uncertainty as well.

E. Model Comparison

Previous studies of spring-mass models by Rummel and Seyfarth and others [23], [66], [67] have suggested that nonlinear effective leg compliance can improve stability. These studies focus on finding basins of attraction with a fixed parameter set. As such, they focus specifically on limit-cycle motion and only provide insight to robustness to state perturbations.

With their numerical studies, Rummel *et al.* show that, compared to a linear leg compliance, a nonlinear leg compliance has a broader range of parameters which exhibit passively stable limit cycles. These limit cycles also tend to have larger basins of attraction. However, at higher velocities, the model with nonlinear spring stiffness no longer exhibits passively stable limit cycles, whereas with a linear spring this property is retained. These results suggested that nonlinear compliance is only beneficial at lower running speeds [23].

Using our formulation, we can evaluate robustness to state perturbations not only for an open-loop system but for any robust control policy. Our results confirm that, even with a maximally robust control policy, the set of robust states S_R shrinks at higher speeds (see Fig. 7), though not as drastically as the basins of attraction studied by Rummel *et al.*

V. Optimizing Natural Dynamics for Robustness

As an example application, we use our quantification to optimize the robustness of a simulated planar monoped with a 2-segment leg with a hierarchical control structure, shown in Fig. 8. The kinematic tree of the simulated system matches a robot testbed we currently use in our lab, though we have adjusted the parameters to be consistent with the models in the previous section. The system consists of three links: a floating-base free to move in the plane, but without rotation, and a two-link leg. Both hip and knee joints are actuated, resulting in an

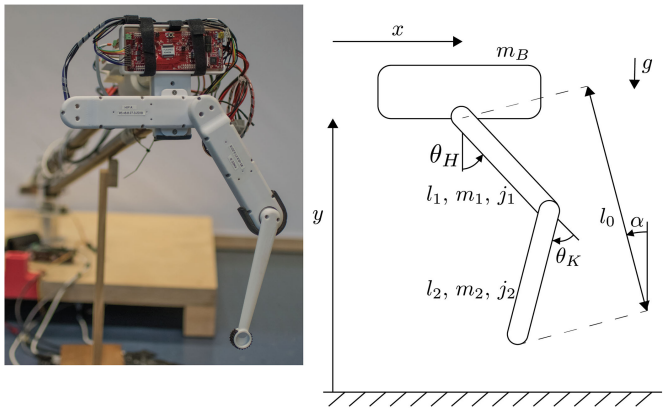

Fig. 8. Simulated system is based on a hardware testbed, which is rigidly attached to a boom. Thus, the floating base is limited to two degrees of freedom. Two additional degrees of freedom, the hip and knee joints, are both actuated. Thus, the system has four position coordinates $q = [x, y, \theta_H, \theta_K]^\intercal$, 8-D state space $[q, \dot{q}]^\intercal$ and 2-D action space $[\tau_H, \tau_K]^\intercal$, where τ_H and τ_K are the hip and knee torques, respectively. The robot shown is designed by our colleague Felix Grimminger.

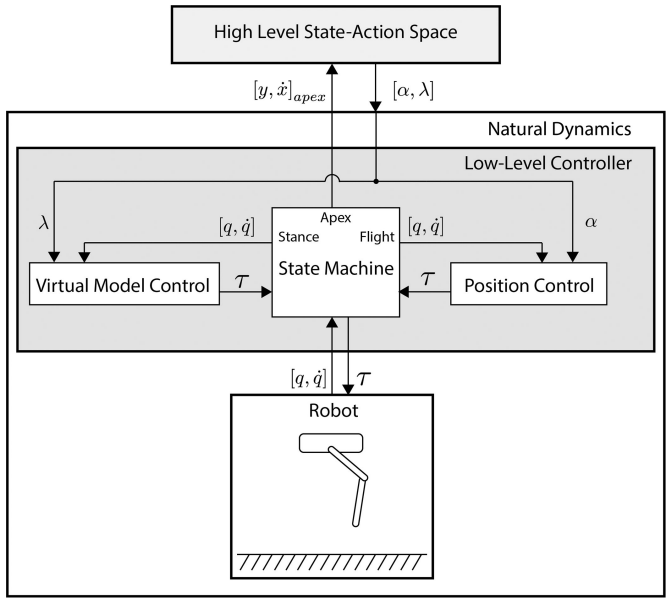

Fig. 9. High-level state-action space is composed of the height and forward velocity of the floating base at apex $[y, \dot{x}]^\intercal_{\text{apex}}$, the desired landing angle of attack α and the thrust factor λ. The natural dynamics considered are those relative to the high level. These include both the rigid-body dynamics of the simulated robot as well as the embedded low-level controller.

eight-dimensional (8-D) state space and a two-dimensional (2-D) action space. Rigid impacts and ground-reaction forces are solved as described in [43] and [68].

We use the volume of the robust set Q_V as the fitness function for a particle swarm optimization (PSO), a standard gradient-free optimization scheme. Thus, instead of requiring the low-level controller to enforce a specific template model, we improve its robustness in a general sense. The resulting natural dynamics allow for a high-level control policy to be implemented more reliably.

A. High-Level State-Action Space

The choice of the high-level state-action space is based on the spring-mass models and classic Raibert control [69], which share many similarities. The structure is shown in Fig. 9.

The state is defined on the Poincaré section at flight apex, as introduced in Section II. Since the system is not energy conservative, both the height and forward velocity of the floating base at apex must be considered, resulting in the state vector $[y, \dot{x}]^\intercal_{\text{apex}}$.

The action space is defined as a desired landing angle of attack α, constrained within 0 and 45°, and a thrust factor λ applied during stance, constrained within 1 and 2. This results in a 4-D state-action space in the high level, which is amenable to direct computation of a sufficiently dense grid.

Although our choice of the state-action space is largely motivated by Raibert control, we make no restrictions on the high-level control policy and do not decouple the states and actions.

B. Low-Level Controller

The low-level controller is a state-machine that switches between flight and stance.

During flight, a standard PD position controller tracks the desired landing angle of attack α dictated by the high-level control policy. The resting length of the virtual leg l_0, is set as a constant parameter less than the maximum leg length to avoid reaching singularities. Thus, α uniquely determines the desired foot position during flight. Since there are two possible joint configurations for each desired foot position, this orientation is also set as a constant parameter in the computation of the inverse kinematics. Thus, α also uniquely determines the desired joint angles. During the first flight phase, from apex till touchdown, α is freely chosen as the action. For the second flight phase, from liftoff till the next apex, α is reset to the default position 0. Thus, the initial leg configuration at each apex is expected to be the same.

During stance, we do not enforce the dynamics of a spring mass template model. Instead, compliant behavior is achieved via virtual model control [47], [49]. Torques are computed to mimic a relatively arbitrary leg compliance

$$[\tau_H, \tau_K]^\intercal = \begin{cases} BJ_c^\intercal \, k_v \Delta l \, + \, K_j \Delta\theta & \text{if } \dot{y} < 0 \\ \lambda \, (BJ_c^\intercal \, k_v \, \Delta l \, + \, K_j \Delta\theta) & \text{otherwise} \end{cases} \tag{8}$$

where $[\tau_H, \tau_K]^\intercal$ are the hip and knee torques, B is the actuator selection matrix, J_c is the contact Jacobian, k_v is the stiffness coefficient of a virtual linear spring between hip and foot, Δl is the deflection of the virtual leg from rest, K is a symmetric linear matrix, and $\Delta\theta$ is the joint deflection of the leg from rest. The diagonal coefficients of K can be interpreted as virtual springs on the corresponding joints, while the off-diagonal coefficient serves as a mixing term. As long as K is positive definite, K results in a nonlinear compliance with respect to the virtual leg deflection Δl. In similar fashion to classic Raibert control [69], additional thrust is triggered once the body reverses direction by amplifying joint torques by the thrust factor λ, as dictated by the high-level control policy.

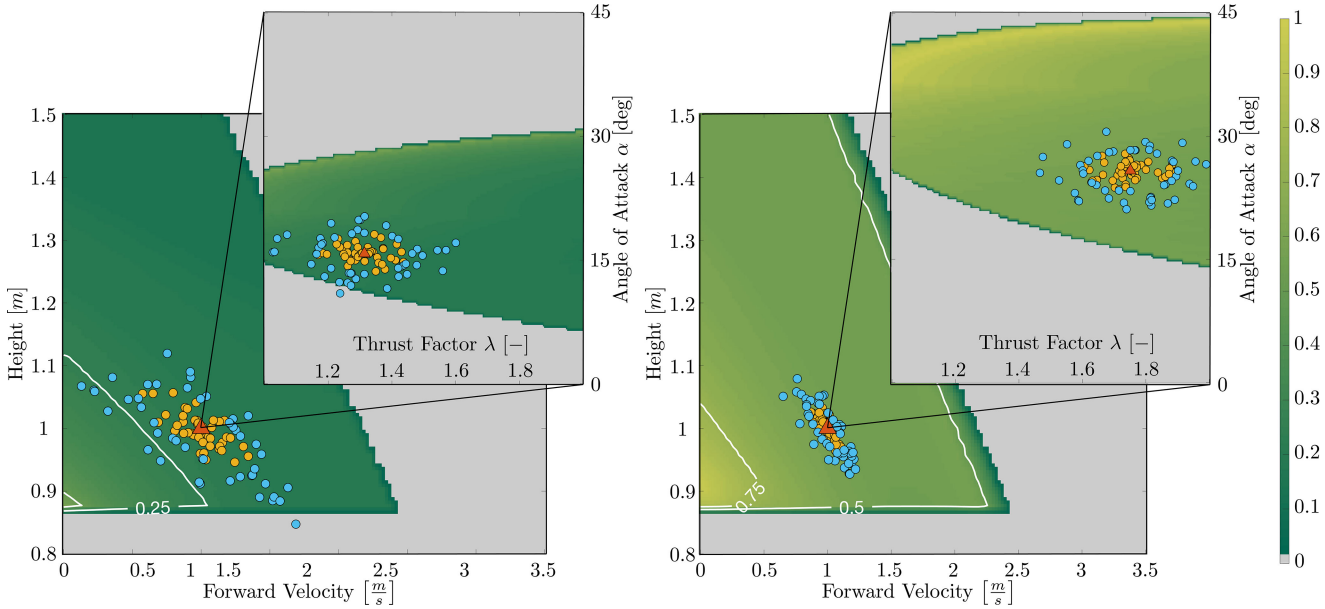

Fig. 10. Viability kernels in the high-level space for the initial monoped (left) and after optimizing virtual compliance (right), along with the action space for the operating point $[y,\ \dot{x}]^\intercal = [1, 1]^\intercal$, shown in the image insets. A red triangle marks the state-action pair which leads to limit-cycle motion on the operating point, in both the state and action spaces. In the action space, the orange and blue circles mark actions randomly sampled around the operating point and with bounded noise $\eta = [5°, 0.1]^\intercal$, and between η and 2η, respectively. The states reached by these state-action pairs are marked with their respective colors in the state space, which shows the much lower sensitivity experienced by the optimized monoped. The intensity of the color-map indicates for each point in state space, the portion of the action space which is viable. In the action space (image insets), the color-map indicates the intensity of the state that would be reached if that state-action pair were sampled.

We assume accurate tracking of α during flight phase, which is achieved through proper tuning of the PD gains. This is important to ensure well-behaved high-level dynamics for two reasons. First, to ensure that each high-level state-action pair results in a unique state at touchdown. Second, to ensure that the robot leg returns to the same resting configuration at each apex. In this manner, the leg masses can be lumped with the floating base to determine potential and kinetic energy, meaning that the high-level state $[y, \dot{x}]^\intercal_{\text{apex}}$ fully describes the system energy. Thus, the transition map P provides a unique map for each high-level state-action pair, and the viable sets S_V and Q_V can be directly computed in the high-level state-action space.

C. Optimization Setup

We use a standard PSO implementation based on [70]. The parameters optimized are the stiffness coefficients of the virtual leg in the low-level stance controller, $[k_v, k_{11}, k_{22}, k_{ij}]$, where k_{11} and k_{22} form the diagonal of the symmetric matrix K, and k_{ij} is the off-diagonal term.

As fitness function, we choose to maximize the hypervolume enclosed by the viable set Q_V in the high-level state-action space. For our systems, we have found that maximizing the hypervolume of Q_V and Q_R generally leads to the same results for reasonable amounts of noise. Each dimension of the state is normalized by heuristically determined bounds on maximum height and forward velocity, and the dimensions of the action space are bounded by their corresponding constraints. The hypervolume is calculated by summing and then normalizing the points inside the set. Thus, a fitness of 1 means that for any state, all actions are viable. A fitness of 0 means that for any state, all actions are outside the viable set.

For the results shown, 25 particles were initialized at random. Convergence tolerance on the fitness variance was set to 10^{-5}, which was reached after 12 iterations, taking roughly 3.5 h on a 28-core desktop. During the optimization, we used a low-resolution grid with 160 000 points to speed up computation. Note that a lower resolution will result in a more conservative estimate of the sets, but not in mislabeled points in the set. The simulation parameters used are:

Mechanical Parameters			
gravitational constant	g :	9.81	$[m/s^2]$
body mass	m_B :	65	$[kg]$
upper leg length	l_1:	0.5	$[m]$
upper leg mass	m_1:	10	$[kg]$
upper leg inertia	j_1:	2	$[Kgm^2]$
lower leg length	l_2:	0.5	$[m]$
lower leg mass	m_2:	5	$[kg]$
lower leg inertia	j_2:	2	$[Kgm^2]$
Low Level Control Parameters			
leg resting length	l_0 :	0.85	$[m]$
saturation torque	τ_{max}:	2000	$[Nm]$
Hip joint PD gains	$[k_p, k_d]$:	$[500, 50]$	$[-]$
Knee joint PD gains	$[k_p, k_d]$:	$[500, 25]$	$[-]$

D. Optimization Results

We compare the robustness with a virtual leg compliance roughly matching that of the SLIP model, with stiffness coefficient $[k_v, k_{11}, k_{22}, k_{ij}] = [8, 0, 0, 0]\,10^3$, versus one with the stiffness coefficients resulting from the optimization, $[k_v, k_{11}, k_{22}, k_{ij}] = [8.1, 5.0, 0.9, -0.5]\,10^3$. The viability kernels S_V are visualized in Fig. 10. The intensity of the color-map indicates the portion of the action space which is viable for each point in state space. The red triangle marks an arbitrary operating point, $[y,\ \dot{x}]^\intercal = [1, 1]^\intercal$, and the action space for this state is shown in the image inset. In the action space, the action pair leading to limit-cycle motion is also marked by a red triangle. To illustrate improved robustness, 50 actions are uniformly sampled around the operating point assuming bounded noise $\eta = [5^\circ, 0.1]^\intercal$ (orange circles) and an additional 50 with bounded noise between η and 2η (blue circles).

As in the comparison between the SLIP and NSLIP models in the previous section, the viability kernel S_V in state space remains nearly identical for both systems. The volume of the set of viable state-action pairs, however, increases from 0.08 to 0.23, over 2.8 times. The noisy sampling of actions around the operating point shows the decreased sensitivity to action noise with the optimized nonlinear compliance. In Fig. 10, we chose an arbitrary operating point for the sake of simplicity and fair comparison. In practice, an operating point can be chosen based on the robustness of that point in state space. Conversely, instead of optimizing the overall robustness of the system, the fitness function can be weighted to bias robustness near a predetermined operating point.

VI. Conclusion

In this paper, we have presented a formulation for computing viable and robust sets in state-action space which allows the inherent robustness of a system to be quantified, prior to specifying the control policy parameterization or objective. Different system designs can thus be compared quantitatively.

We have illustrated this formulation on the spring-mass model, a low-dimensional system commonly used to synthesize control strategies for running robots. Furthermore, we have shown an example application using our quantification to perform gradient-free optimization. The system optimized is a simulated planar monoped with a two-segment leg and a hierarchical control structure. The low-level controller parameters are optimized to improve robustness of the natural dynamics, as relative to the high-level state-action space.

An important advantage of this formulation is that the robustness of the natural dynamics can be optimized without enforcing the dynamics of a specific template model, which is often challenging and requires extensive tuning, developing accurate models as well as state estimation [44], [71], [72]. Instead, the inherent robustness will allow control policies designed on simple model abstractions to be leveraged despite inaccuracies.

To the best of our knowledge, prior work in viability theory focused on evaluating robustness of a specific control policy, or on synthesizing control policies directly, and computation was limited to viability kernels in state space.

The notable exception is the work of Zaytsev *et al.* [56], which also computes viable sets in state-action space. Aside from the minor difference in studying walking instead of running models, Zaytsev *et al.* focus on the connection between controllability and viability. This is used to qualify how robust a given control policy is, how appropriate different templates may be for a given control task and given robot, and to motivate the statement that planning two steps ahead is sufficient. While we use the same state-action space formulation, we take a different approach to quantification by evaluating bounded noise in action space, which is more suitable for our motivating question: how to design natural dynamics that are easy to exploit? Indeed, we show why this is the only type of uncertainty that can be considered for the family of all robust control policies, without setting any assumptions on the control policy structure or objective. As such, we find our methods to be highly complementary, and applicable at different stages of robot design.

One of the main challenges with viability-based approaches is tractability [53], [59]. While we show how, in principle, a hierarchical control scheme reduces dimensionality, this approach alone is rarely sufficient in dealing with the curse of dimensionality on real systems. There is much recent progress on different scalable approaches to computing viable and back-reachable sets (see Section I-C), and the specific choice will depend greatly on the properties of the system in question.

For running motion, characterized by nonlinear, nonsmooth hybrid dynamics, we believe that, in addition to dimensionality reduction through hierarchical control, the use of heuristics such as computing ahead only two steps [56] are among the most promising tools to scaling this to real hardware.

We are also interested in using sampling-based approaches to make probabilistic estimates. There has been keen interest recently in applying machine learning techniques to tune control parameters directly in hardware [10], [73]–[75]. In these situations, safe exploration of the state-action space is particularly important. Active sampling to add samples close to the edge of the viable set would significantly increase sample efficiency for estimating the sets, while at the same time allowing safe exploration, making this a logical next step.

There is also potential for improvement in the definition of failures, the starting point of any viability approach. In this paper, we have used a very general and intuitive definition for failure (falling and direction reversal), however other definitions may be equivalent while offering earlier detection when computing viability kernels. Conservative definitions which lead to inner approximations may also be useful if they substantially speed up computation. It may also be possible to decouple the system dynamics, a common approach to simplifying control [15], [69], [76], and identify different failure conditions for each decoupled subsystem. This divide and conquer approach would also allow substantially higher dimensional systems to be tackled.

Appendix
SLIP and NSLIP Models

The SLIP and NSLIP models are shown in Fig. 11. Integration between two apex events is split into three phases: a flight phase that terminates with a touchdown event, a stance phase which

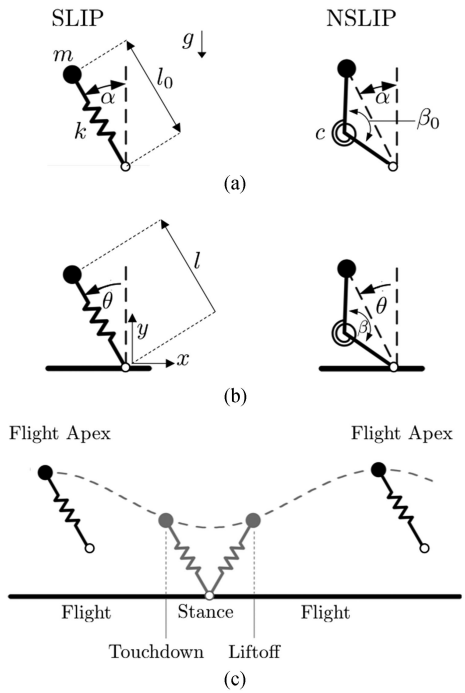

Fig. 11. (a) Parameters of the SLIP and NSLIP models. (b) States. The reference frame is reset to the foot position at each touchdown. (c) Qualitative trajectory over a full cycle, with the relevant phases and events.

terminates with a liftoff event, and another flight phase which terminates with an apex event.

The flight phase equations of motion are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

where x and y are the body position and g is the gravitational acceleration. The stance phase equations of motion are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{F_{\text{leg}}}{m} \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix}$$

$$\theta = \arctan 2 \left(\frac{y}{x} \right) - \frac{\pi}{2}$$

where θ is the incident angle between the body and the foot (the rotation by $\frac{\pi}{2}$ serves to keep it consistent with the landing angle of attack) and F_{leg} is the force acting on the body due to the spring. In the SLIP model

$$\text{SLIP: } F_{\text{leg}} = k(l_0 - l)$$

$$l = \sqrt{(x^2 + y^2)}$$

where k is the spring coefficient, l_0 is the spring resting length, and l is the leg length. In the NSLIP model

$$\text{NSLIP: } F_{\text{leg}} = \frac{4lc(\beta_0 - \beta)}{l_0^2 \sin(\beta)}$$

$$\beta = \arccos\left(1 - \frac{2l^2}{l_0^2}\right)$$

where c is the torsional spring coefficient, β_0 is the spring resting angle, and β is the knee angle. The three events are

$$\text{touchdown: } l = l_0$$

$$\text{liftoff: } \theta = \arctan 2 \left(\frac{y}{x} \right) - \frac{\pi}{2}$$

$$\text{apex: } \dot{y} = 0.$$

At each touchdown, the reference frame is reset to the foot position, which allows the equations of motion to be written more compactly. In the simulation, we also keep track of the foot position in an auxiliary variable.

For convenient comparison, we use the same parameters as in [23], which are similar to human averages:

gravitational acceleration	g :	9.81	$[m/s^2]$
body mass	m :	80	$[kg]$
prismatic spring resting length	l_0:	1	$[m]$
prismatic spring coefficient	k :	8200	$[N/m]$
torsional spring resting angle	β_0 :	170	$[^\circ]$
torsional spring coefficient	c :	704	$[Nm/rad]$

For the SLIP and NSLIP simulations shown, except in Fig. 7, the system energy simulated is 1860 J.

ACKNOWLEDGMENT

The authors would like to thank everyone who gave feedback during the writing of this manuscript. In particular, we appreciate the frequent and insightful discussions with M. Millard, B. Gillespie, and A. del Prete, as well as F. Solowjow's advice on mathematical notation, and also appreciate the editors and reviewers for their constructive suggestions and quick turnaround time.

REFERENCES

[1] M. A. Daley and A. A. Biewener, "Running over rough terrain reveals limb control for intrinsic stability," *Proc. Nat. Acad. Sci.*, vol. 103, no. 42, pp. 15681–15686, 2006.

[2] M. A. Daley and J. R. Usherwood, "Two explanations for the compliant running paradox: Reduced work of bouncing viscera and increased stability in uneven terrain," *Biol. Lett.*, vol. 6, no. 3, pp. 418–421, 2010.

[3] A. J. Ijspeert, "A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander," *Biol. Cybern.*, vol. 84, no. 5, pp. 331–348, 2001.

[4] J. Proctor and P. Holmes, "Reflexes and preflexes: On the role of sensory feedback on rhythmic patterns in insect locomotion," *Biol. Cybern.*, vol. 102, no. 6, pp. 513–531, 2010.

[5] D. Owaki, T. Kano, K. Nagasawa, A. Tero, and A. Ishiguro, "Simple robot suggests physical interlimb communication is essential for quadruped walking," *J. Roy. Soc. Interface*, vol. 10, no. 78, p. 20120669, 2013.

[6] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. New York, NY, USA: Taylor & Francis, 2007.

[7] T. Koolen *et al.*, "Design of a momentum-based control framework and application to the humanoid robot atlas," *Int. J. Humanoid Robot.*, vol. 13, no. 1, p. 1650007, 2016.

[8] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *Proc. IEEE 16th Int. Conf. Humanoid Robots*, 2016, pp. 842–849.

[9] Y. Zhao, B. R. Fernandez, and L. Sentis, "Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model," *Int. J. Robot. Res.*, vol. 36, no. 11, pp. 1211–1242, 2017.

[10] E. Heijmink, A. Radulescu, B. Ponton, V. Barasuol, D. G. Caldwell, and C. Semini, "Learning optimal gait parameters and impedance profiles for legged locomotion," in *Proc. IEEE 17th Int. Conf. Humanoid Robots*, 2017, pp. 339–346.

[11] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson, "Bayesian optimization using domain knowledge on the atrias biped," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1771–1778.

[12] R. Grandia, D. Pardo, and J. Buchli, "Contact invariant model learning for legged robot locomotion," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2291–2298, Jul. 2018.

[13] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 932–950, 2013.

[14] S. Rezazadeh *et al.*, "Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot," in *Proc. ASME Dyn. Syst. Control Conf.*, p. 23, 2015.

[15] S. W. Heim, M. Ajallooeian, P. Eckert, M. Vespignani, and A. J. Ijspeert, "On designing an active tail for legged robots: Simplifying control via decoupling of control objectives," *Ind. Robot, Int. J.*, vol. 43, no. 3, pp. 338–346, 2016.

[16] J. Ramos, B. Katz, M. Y. M. Chuah, and S. Kim, "Facilitating model-based control through software-hardware co-design," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 566–572.

[17] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation," *Sci. Robot.*, vol. 1, no. 1, 2016.

[18] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," in *Proc. 14th Yale Workshop Adaptive Learn. Syst.*, Yale University New Haven, 2005, vol. 95585, pp. 1939–1412.

[19] S. Heim, F. Ruppert, A. A. Sarvestani, and A. Spröwitz, "Shaping in practice: Training wheels to learn fast hopping directly in hardware," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–6.

[20] R. Ringrose, "Self-stabilizing running," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1997, vol. 1, pp. 487–493.

[21] A. Schwab and M. Wisse, "Basin of attraction of the simplest walking model," in *Proc. ASME Des. Eng. Tech. Conf.*, 2001, vol. 6, pp. 531–539.

[22] H. Geyer, R. Blickhan, and A. Seyfarth, "Natural dynamics of spring-like running: Emergence of selfstability," in *Proc. 5th Int. Conf. Climbing Walking Robots*, 2002, pp. 87–92.

[23] J. Rummel and A. Seyfarth, "Stable running with segmented legs," *Int. J. Robot. Res.*, vol. 27, no. 8, pp. 919–934, 2008.

[24] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1114–1124, Oct. 2013.

[25] T. McGeer, "Passive dynamic walking," *Int. J. Robot. Res.*, vol. 9, pp. 62–82, 1990.

[26] M. Wisse and J. Van Frankenhuyzen, "Design and construction of mike; a 2-D autonomous biped based on passive dynamic walking," in *Adaptive Motion of Animals and Machines*, Berlin, Germany: Springer, 2006, pp. 143–154.

[27] P. A. Bhounsule, J. Cortell, and A. Ruina, "Design and control of ranger: An energy-efficient, dynamic walking robot," in *Adaptive Mobile Robotics*. Singapore: World Scientific, 2012, pp. 441–448.

[28] F. Asano, "Stability principle underlying passive dynamic walking of rimless wheel," in *Proc. IEEE Int. Conf. Control Appl.*, 2012, pp. 1039–1044.

[29] A. D. Kuo, "Energetics of actively powered locomotion using the simplest walking model," *J. Biomechanical Eng.*, vol. 124, no. 1, pp. 113–120, 2002.

[30] R. Blickhan, "The spring-mass model for running and hopping," *J. Biomechanics*, vol. 22, pp. 1217–1227, 1989.

[31] R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land," *J. Exp. Biol.*, vol. 202, pp. 3325–3332, 1999.

[32] H.-M. Maus, S. Revzen, J. Guckenheimer, C. Ludwig, J. Reger, and A. Seyfarth, "Constructing predictive models of human running," *J. Roy. Soc. Interface*, vol. 12, 2015, Art. no. 20140899.

[33] D. L. Jindrich and R. J. Full, "Dynamic stabilization of rapid hexapedal locomotion," *J. Exp. Biol.*, vol. 205, pp. 2803–2823, 2002.

[34] R. Altendorfer *et al.*, "RHex: A biologically inspired hexapod runner," *Auton. Robots*, vol. 11, no. 3, pp. 207–213, 2001.

[35] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *SIAM Rev.*, vol. 48, no. 2, pp. 207–304, 2006.

[36] B. Stephens and C. Atkeson, "Modeling and control of periodic humanoid balance using the linear biped model," in *Proc. IEEE 9th Int. Conf. Humanoid Robots*, Dec. 2009, pp. 379–384.

[37] E. W. Hawkes and M. R. Cutkosky, "Design of materials and mechanisms for responsive robots," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 359–384, 2018.

[38] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, "A simply stabilized running model," *SIAM Rev.*, vol. 47, no. 3, pp. 519–549, 2005.

[39] Ö. Arslan and U. Saranli, "Reactive planning and control of planar spring-mass running on rough terrain," *IEEE Trans. Robot.*, vol. 28, no. 3, pp. 567–579, Jun. 2012.

[40] G. Piovan and K. Byl, "Two-element control for the active slip model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5656–5662.

[41] T. Cnops, Z. Gan, and C. D. Remy, "The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2015, pp. 1586–1591.

[42] L. R. Palmer III and C. E. Eaton, "Periodic spring–mass running over uneven terrain through feedforward control of landing conditions," *Bioinspiration Biomimetics*, vol. 9, no. 3, 2014, Art. no. 036018.

[43] M. Hutter, C. D. Remy, M. A. Höpflinger, and R. Siegwart, "Slip running with an articulated robotic leg," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4934–4939.

[44] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, Dept. Electr. Eng., Stanford University, Stanford, CA, USA, 2007.

[45] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2013, pp. 5134–5140.

[46] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Trans. Autom. Control*, vol. 54, no. 8, pp. 1779–1793, Aug. 2009.

[47] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *Int. J. Robot. Res.*, vol. 20, no. 2, pp. 129–143, 2001.

[48] R. Altendorfer, D. E. Koditschek, and P. Holmes, "Stability analysis of a clock-driven rigid-body slip model for RHex," *Int. J. Robot. Res.*, vol. 23, no. 10/11, pp. 1001–1012, 2004.

[49] D. Renjewski, A. Spröwitz, A. Peekema, M. Jones, and J. Hurst, "Exciting engineered passive dynamics in a bipedal robot," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1244–1251, Oct. 2015.

[50] W. C. Martin, A. Wu, and H. Geyer, "Experimental evaluation of deadbeat running on the atrias biped," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1085–1092, Apr. 2017.

[51] J.-P. Aubin, *Viability Theory*. Berlin, Germany: Springer Science & Business Media, 2009.

[52] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. Oishi, and G. A. Dumont, "Lagrangian methods for approximating the viability kernel in high-dimensional systems," *Automatica*, vol. 49, no. 7, pp. 2017–2029, 2013.

[53] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 2, pp. 464–478, Mar. 2019.

[54] D. Panagou, K. Margellos, S. Summers, J. Lygeros, and K. J. Kyriakopoulos, "A viability approach for the stabilization of an underactuated underwater vehicle in the presence of current disturbances," in *Proc. 48th IEEE Conf. Decis. Control*, 2009, pp. 8612–8617.

[55] P.-B. Wieber, "Viability and predictive control for safe locomotion," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2008, pp. 1103–1108.

[56] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: Viability and controllability of simple models," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 336–352, Apr. 2018.

[57] D. Lakatos, W. Friedl, and A. Albu-Schäffer, "Eigenmodes of nonlinear dynamics: Definition, existence, and embodiment into legged robots with elastic elements," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 1062–1069, Apr. 2017.

[58] N. Smit-Anseeuw, C. D. Remy, and R. Vasudevan, "Walking with confidence: Safety regulation for full order biped models," 2019, *arXiv preprint arXiv:1903.08327*.

[59] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton–Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2242–2253.
[60] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *Int. J. Robot. Res.*, vol. 31, no. 9, pp. 1094–1113, 2012.
[61] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 387–394.
[62] A. Merker, D. Kaiser, A. Seyfarth, and M. Hermann, "Stable running with asymmetric legs: A bifurcation approach," *Int. J. Bifurcation Chaos*, vol. 25, no. 11, 2015, Art. no. 1550152.
[63] G. Piovan and K. Byl, "Reachability-based control for the active slip model," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 270–287, 2015.
[64] S. Heim and A. Spröwitz, "Learning from outside the viability kernel: Why we should build robots that can fall with grace," in *Proc. IEEE Int. Conf. Simul., Model., Program. Auton. Robots*, 2018, pp. 55–61.
[65] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.
[66] D. Owaki and A. Ishiguro, "Enhancing stability of a passive dynamic running biped by exploiting a nonlinear spring," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 4923–4928.
[67] J. D. Karssen and M. Wisse, "Running with improved disturbance rejection by using non-linear leg springs," *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 531–539, 2011.
[68] C. D. Remy, K. Buffinton, and R. Siegwart, "A matlab framework for efficient gait creation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2011, pp. 190–196.
[69] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.
[70] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[71] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Auton. Robots*, vol. 40, no. 3, pp. 473–491, 2016.
[72] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots," in *Proc. IEEE 17th Int. Conf. Humanoid Robots*, 2017, pp. 889–895.
[73] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion," in *Proc. IEEE 16th Int. Conf. Humanoid Robots*, 2016, pp. 22–28.
[74] V. C. Kumar, S. Ha, and K. Yamane, "Improving model-based balance controllers using reinforcement learning and adaptive sampling," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 7541–7547.
[75] A. von Rohr, S. Trimpe, A. Marco, P. Fischer, and S. Palagi, "Gait learning for soft microrobots controlled by light fields," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 6199–6206.
[76] E. C. Whitman and C. G. Atkeson, "Control of a walking biped using a combination of simple policies," in *Proc. IEEE 9th Int. Conf. Humanoid Robots*, 2009, pp. 520–527.

Steve Heim received the B.Sc. degree in mechanical engineering and the M.Sc. degree in robotics, systems and control from ETH Zürich, Switzerland, in 2012 and 2015, respectively. He is currently working toward a Ph.D. degree with the Dynamic Locomotion Group, at the Max Planck Institute for Intelligent Systems, Stuttgart, Germany.

He spent two years with the Ishiguro Lab, Tohoku University, Sendai, Japan. His research interests include nonlinear dynamics, control and learning, particularly in relation to legged locomotion.

Alexander Spröwitz received the Diploma in mechatronics from Ilmenau Technical University, Ilmenau, Germany, in 2005, and the Ph.D. degree in manufacturing systems and robotics from the Biorobotics Laboratory, the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2010.

Since 2016, he is the Max Planck Research Group Leader of the Dynamic Locomotion Group, and IMPRS-IS Faculty, at the Max Planck Institute for Intelligent Systems, Stuttgart, Germany. His current research focuses on the mechanisms underlying legged locomotion.

Dr. Sprowitz and his team design and experiment with legged robots and models to infer biomechanics and neurocontrol principles of motion in animals.