Proceedings of the ASME 2020 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2020
August 16-August 19, 2020, St. Louis, USA

IDETC2020-18604

DRAFT: APPROXIMATION OF THE STEP-TO-STEP DYNAMICS ENABLES COMPUTATIONALLY EFFICIENT AND FAST OPTIMAL CONTROL OF LEGGED ROBOTS

Pranav A. Bhounsule[*]
Myunhgee Kim
Department of Mechanical and Industrial Engineering
University of Illinois at Chicago
842 W. Taylor St., Chicago, IL, 60607
pranav@uic.edu, myheekim@uic.edu

Adel Alaeddini
Department of MechanicalEngineering
The University of Texas at San Antonio
1 UTSA Circle, San Antonio, TX 78249
Email: adel.alaeddini@utsa.edu

ABSTRACT

Legged robots with a point or small feet are impossible to control instantaneously but are controllable over the time scale of one or more steps. For example, a runner when pushed, cannot instantaneously stabilize in the ballistic or stance phase but can take a step in the direction of the push to neutralize the effect of the destabilization over a step. We call this form of stabilization as step-to-step control. Previous approaches to achieve step-to-step control either perform (1) online optimization with a nonlinear model to achieve a large stability region but is expensive, or (2) create a linear approximation of the nonlinear step-to-step dynamics followed by optimization which is cheaper but offers a limited stability region. Our method combines the advantages of both. We use a simulator that takes as its input, the robot states at an instant of the motion and control commands during the step and produces the output, the robot states at the same instant at the next step. This input, output data is curve fitted using a suitable regression model to get a closed-form model of the step-to-step map. We then use this model for optimal control. Although we use many offline computations for curve fitting, the main advantage of the approach stems from the use of a compact, accurate, step-to-step model that enables efficient and fast optimization. In this paper, using the spring-load inverted pendulum model of running, we show that both parametric (e.g., polynomial, neural network) and non-parametric (Gaussian process regression) approximations can capture the step-to-step dynamics. We show that using this closed-form approximation, we can stabilize a wide range of initial conditions. We suggest that closed-form approximation of the step-to-step dynamics provides a simple model for fast optimal control of legged robots.

1 Introduction

Legged robots with a point or small feet are under-actuated systems (i.e., the number of degrees of freedom exceeds the number of actuators on the degrees of freedom). Because of under-actuation, such systems cannot are difficult to control instantaneously (e.g., balancing over the upright position when perturbed). However, it is possible to control these systems over a finite time horizon, typically of the order of one step. For example, one can correct a push given to a standing robot with point feet by taking a step in the direction of the push, thus achieving balance control over one step [1]. We call this control as step-to-step control in this paper.

There are two widely used methods for step-to-step control. One, linearize the step-to-step dynamics around a nominal solution, known as the limit cycle [2, 3], and use this linear approximation (i.e., floquet theory) for control using linear quadratic regulator [4–6]. Two use trajectory optimization by integrating the equations of motion from one step to the next to find the optimal control input [7]. The first method is computationally simple

[*]Address all correspondence to this author.

but works well only for small perturbations around the limit cycle [8]. The second method works well for large perturbations but is computationally expensive as it requires repeated integration of the equations over a step. In this work, we blend both these methods, thus combining their advantages and offsetting limitations of either.

Our approach is to generate a closed-form approximation of the step-to-step dynamics, which we use for optimization (see Fig. 1). Since the approximation is valid over a wide region of state space, it is valid for large perturbation, but since it is also a closed-form approximate model, it is a computationally simpler optimization. To obtain the closed-form approximation we proceed as follows. First, we integrate the equations of motion over a complete step for a range of input robot states and input control actions to obtain the robot state at the next step, the outputs. Second, we fit the input data to the output data by assuming regression model (e.g., polynomials, neural nets, Gaussian process regression). In this way, we may use any black-box model of the system for closed-form model generation. We may also use the closed-form approximate model for online optimization as needed.

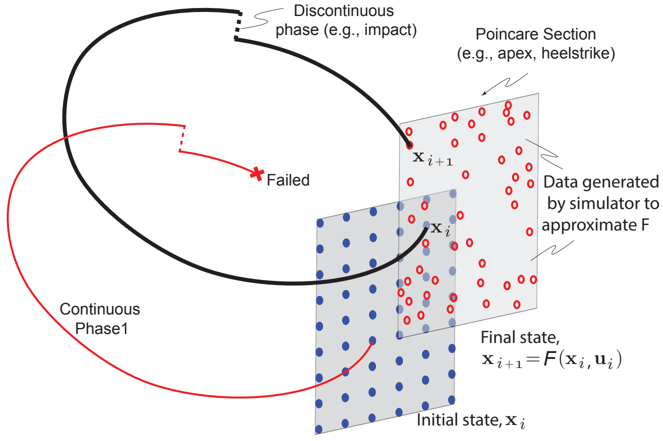
Discontinuous phase (e.g., impact)
Poincare Section (e.g., apex, heelstrike)
$\mathbf{x}_{i+1}$
Failed
$\mathbf{x}_i$
Data generated by simulator to approximate F
Continuous Phase1
Final state, $\mathbf{x}_{i+1}=F(\mathbf{x}_i,\mathbf{u}_i)$
Initial state, $\mathbf{x}_i$

FIGURE 1. **An overview of the approximation to the step-to-step dynamics:** Initial state $\mathbf{x}_i$ (blue filled circle) are chosen at a fixed instant in the locomotion cycle (e.g., apex) at step i. Then a forward simulation is performed while applying appropriate control action $\mathbf{u}_i$ (e.g., amplitudes, gain) during the step. The forward simulation system leads to the state, $\mathbf{x}_{i+1}$ (red unfilled circle) at the same instant at the next step. Applying the same procedure, a range of input states $\mathbf{x}_i$ (blue filled circles) and input controls $\mathbf{u}_i$ and the corresponding output states $\mathbf{x}_{i+1}$ (red unfilled circles) are recorded. Only successful states (e.g., (i)) are saved and failed ones (e.g., (ii)) are discarded. Once the input/output pairs are obtained, a curve fitting is done to get a closed-form expression for the step-to-step dynamics represented by $\mathbf{x}_{i+1} = \overline{\mathbf{F}}(\mathbf{x}_i, \mathbf{u}_i)$.

2 Background and related Work

Raibert and colleagues built the very first dynamically balancing single-legged hopping robot [9]. Initially, they used offline computations to create controllers and implemented them using look-up tables [10]. Later on; they found simple heuristics to achieve step-to-step balance control by using three decoupled controllers in tandem: foot placement to regulate horizontal speed, axial thrust for height control, and hip torque during ground contact to stabilize the torso. Raibert et al. then demonstrated the same ideas can control biped and quadruped robots [9].

McGeer [11] formalized the step-to-step control within the context of dynamical systems theory by. First, McGeer demonstrated that a design that resembles a human's lower body when suitably tuned for mass distribution and geometry can walk down a shallow slope with no control. Second, he interpreted the resulting periodic motion as a limit cycle (a repeating trajectory) and used the eigenvalues of the linearization of the step-to-step map, also known as the Poincaré map (i.e., the function that maps the initial conditions from one step at a particular instant in the locomotion cycle to the same instant at the next step) to analyze the stability of the system [12]. Finally, he showed that by using a linear controller based on the step-to-step map (a mapping from initial conditions from one step to the next), it was possible to increase the robustness of the passive limit cycle [13]

Here, we consider the spring-loaded inverted pendulum (SLIP) model of running. The model comprises a point mass body and a springy leg. Geyer et al. [14] showed that the model describes the mechanics of walking and running in human. In a pure SLIP model, energy is conserved and the only control variable is the foot placement angle during the transition from flight to stance. Seyfarth et al. [15] showed that by controlling the foot placement angle at every step it is possible to achieve a wide range of stable solutions. A slight backward motion of the swing leg (also known as swing leg retraction) just before touchdown improves robustness to changes in terrain height [16]. To further increase the range of stable initial conditions in the SLIP model, there needs to be a control mechanism to add or remove energy from the system. These may include changing the leg length during stance phase [17], changing the spring stiffness during stance phase [18], and adding an impulse along during the stance phase [19]. However, all these studies rely on the numerical integration of the step-to-step map for control.

Unlike a few systems such as the point mass inverted pendulum with impulsive push-off and foot placement [20] and the rimless wheel [21] that have an exact closed-form solution for the step-to-step map, an exact solution to the step-to-step dynamics of the SLIP model does not exist. However, past studies that have tried to approximate the step-to-step map by assuming a small angular sweep and stiff springs to demonstrate the equations of motion may into a closed-form for both, the conservative and non-conservative SLIP model [22, 23]. While such approximation may capture 90% of the dynamics [24], it is unclear if it is possible to generalize to other complex systems (e.g., non-conservative SLIP model with a torso [25]).

In this paper, we use a data-driven approach to approximating the step-to-step map as illustrated in Fig. 1. We assume that a simulator is already available that can integrate the equations of motion from a chosen instant in the locomotion cycle (e.g., apex) at step i (say) to the same instant (i.e., apex) of the next step $i+1$. For example, given initial condition $\mathbf{x}_i$ at the instant (blue filled circle), the simulator integrates the motion through all continuous phases and discontinuous phase/events and applying the control action $\mathbf{u}_i$ (e.g., setpoint, gain, amplitude, foot placement angle) at pre-determined times in the motion. Note that these control actions are control parameters that held constant during the step but updated only at a single chosen instant in the locomotion cycle (e.g., apex, foot-strike). Depending on the initial state and control, the step might be successful (see (i)) to give the state at the same instant at the next step $\mathbf{x}_{i+1}$ (red unfilled circle) or unsuccessful (see (ii)). We repeat this procedure for multiple initial states $\mathbf{x}_i$ and input control actions $\mathbf{u}_i$. We only record the initial states and controls that lead to a successful step or a valid $\mathbf{x}_{i+1}$ and discard the data for the failed step. Then we use an appropriate parametric or non-parametric regression model to curve fit the outputs $\mathbf{x}_{i+1}$ to the inputs $\mathbf{x}_i$ and $\mathbf{u}_i$, thus generating the approximation to the step-to-step map $\mathbf{x}_{i+1} = \overline{\mathbf{F}}(\mathbf{x}_i, \mathbf{u}_i)$ where $\overline{\mathbf{F}}$ is the approximation to the true function $\mathbf{F}$. We use this closed-form approximation $\overline{\mathbf{F}}$ for the optimization. The novelty of the paper is the idea that closed-form approximations of the step-to-step map, although of low fidelity (e.g., usually with an accuracy of 85% or more), can still stabilize the system for a relatively wide range of state perturbations. In a broader sense, we can use the method to generate closed-form approximation for complex models (e.g., quadruped, humanoid) and even in the presence of nonlinearities (e.g., Coriolis acceleration) and dissipative effects (e.g., viscous or Coulomb friction). We may use these simple closed-form approximate step-to-step models for fast online optimization.

3 Methods

3.1 Poincaré section and Poincaré map

We define some preliminaries that are used to analyze and consequently develop controllers for legged systems. Figure 2 (a) (red solid line) shows the trajectory of the robot in state space. We define the Poincaré section as an instant or event in the gait (e.g., a foot-strike event occurs at the instant swinging leg hits the ground, apex event occurs when the vertical velocity is zero during flight phase for a running robot). We choose an initial condition at the Poincaré section at step i, $\mathbf{x}_i$, and trace its movement on the application of control $\mathbf{u}_i$ for a single step. At the Poincaré section at step $i+1$, the state of the robot is $\mathbf{x}_{i+1}$. There is a function $\mathbf{F}$, known as the step-to-step map or Poincaré map that relates robot state between steps given by

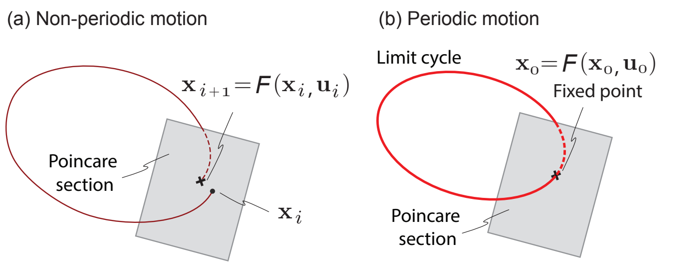

FIGURE 2. Dynamical systems tools use for analysis: (a) Non-periodic motion: An initial condition $\mathbf{x}_i$ chosen at the Poincaré section maps to a new state $\mathbf{x}_{i+1}$ after one step leading to non-periodic motion. (b) Periodic motion: An initial condition $\mathbf{x}_0$ maps onto itself after one step leading to periodic motion, i.e., the trajectory is closed at the Poincaré section.

$$\mathbf{x}_{i+1} = \mathbf{F}(\mathbf{x}_i, \mathbf{u}_i). \tag{1}$$

The trajectory between steps is not closed or $x_{i+1} \neq x_i$ as shown in Fig. 2 (a). Such a motion is aperiodic or non-steady state. However, we can find a state $\mathbf{x}_0$ and a related control $\mathbf{u}_0$ such that

$$\mathbf{x}_0 = \mathbf{F}(\mathbf{x}_0, \mathbf{u}_0). \tag{2}$$

This motion leads to a closed trajectory as shown in Fig. 2 (b). The resulting motion is periodic or steady-state locomotion or also known as a limit cycle motion. The initial condition $\mathbf{x}_0$ is the fixed point of the limit cycle.

3.2 Stabilizing periodic motion

For an initial condition $\mathbf{x}_i \neq \mathbf{x}_0$, we need to find a control input $\mathbf{u}_i$ such that the system tries to reach back to the fixed point $\mathbf{x}_0$. We list three approaches.

3.2.1 Discrete Linear Quadratic Regulator using linearized Poincaré map:
In the discrete linear quadratic regulator, we assume a quadratic cost function (Eqn 3) and linearization of the Poincaré map closed to the fixed point (Eqn. 4)

$$\underset{\mathbf{u}^i}{\text{minimize}} \left\{ (\Delta\mathbf{x}_{i+1})^T \mathbf{Q} (\Delta\mathbf{x}_{i+1}) + (\Delta\mathbf{u}_i)^T \mathbf{R} (\Delta\mathbf{u}_i) \right\} \tag{3}$$
$$\Delta\mathbf{x}_{i+1} = \mathbf{A}\Delta\mathbf{x}_i + \mathbf{B}\Delta\mathbf{u}_i \tag{4}$$

where $(\Delta\mathbf{x}_i) = \mathbf{x}_i - \mathbf{x}_0$, $(\Delta\mathbf{u}_i) = \mathbf{u}_i - \mathbf{u}_0$, the model comes from the following linearizations $\mathbf{A} = \frac{\partial F}{\partial x}|_{(\mathbf{x}_0,\mathbf{u}_0)}$, $\mathbf{B} = \frac{\partial F}{\partial u}|_{(\mathbf{x}_0,\mathbf{u}_0)}$ and

$\mathbf{Q}$, $\mathbf{R}$ are appropriately sized constant matrices that are designer choices. Note that this is a finite time horizon discrete linear regulator unlike the infinite horizon discrete linear regulator that has as its solution, a constant gain matrix $\mathbf{K}$ such that $\Delta\mathbf{u}_i = -\mathbf{K}\Delta\mathbf{x}_i$ [26].

3.2.2 Discrete Nonlinear Quadratic Regulator using an approximated Poincaré map: In the discrete nonlinear quadratic regulator problem, we assume one-step cost function (Eqn. 5) and an approximation to the Poincaré map (Eqn 6)

$$\underset{\mathbf{u}^i}{\text{minimize}} \left\{ (\Delta\mathbf{x}_{i+1})^T \mathbf{Q} (\Delta\mathbf{x}_{i+1}) + (\Delta\mathbf{u}_i)^T \mathbf{R} (\Delta\mathbf{u}_i) \right\} \tag{5}$$

$$\mathbf{x}_{i+1} = \overline{\mathbf{F}}(\mathbf{x}_i, \mathbf{u}_i). \tag{6}$$

We use MATLAB constrained optimization function *fmincon* to solve the above optimization problem.

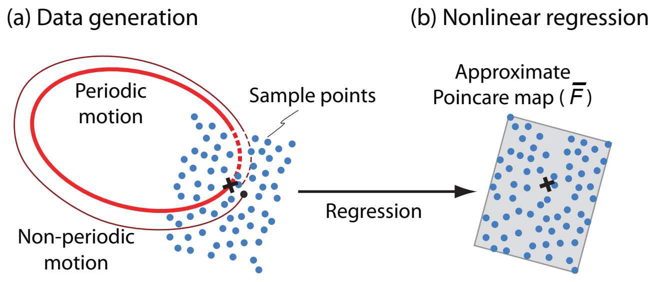

FIGURE 3. Regression to estimate the Poincaré map function $\overline{\mathbf{F}}$: (a) For a range of values of $(\mathbf{x}_i, \mathbf{u}_i)$ at the Poincaré section, we use the forward simulation to generate data $\mathbf{x}_{i+1}$, shown by blue dots on the left plot. (b) Then using an assumed regression model, we fit an approximate function for the Poincaré map $\overline{\mathbf{F}}$, i.e., $\mathbf{x}_{i+1} = \overline{\mathbf{F}}(\mathbf{x}_i, \mathbf{u}_i)$, shown by the gray plane.

We compute the approximation to the Poincaré map ($\overline{\mathbf{F}}$) as follows. We choose a range of initial states on the Poincaré section ($\mathbf{x}_i$) and a range of control actions in the step ($\mathbf{u}_i$). Next, we use the forward simulation of the system given by $\mathbf{x}_{i+1} = \mathbf{F}(\mathbf{x}_i, \mathbf{u}_i)$, to obtain the numerical value for the system state at the next step $\mathbf{x}_{i+1}$. We show these points as blue dots in Fig. 3 (a). Some of these inputs will lead to a failure, i.e., the state at the next step $\mathbf{x}_{i+1}$ is not defined. We ignore these initial conditions in our curve fitting. Finally, using the generated data, we fit a regression model for $\overline{\mathbf{F}}$ as shown by the gray plane in Fig. 3 (b). In this paper, we try three regression models to obtain $\overline{\mathbf{F}}$: (a) a quadratic model; (b) a gaussian process regression model; and (c) a neural network model.

3.2.3 Benchmark controller – Discrete Nonlinear Quadratic Regulator using true Poincaré map: To benchmark to compare these controllers, we solve the optimization problem defined above using the MATLAB function *fmincon* but using the exact Poincaré map $\mathbf{F}$. The optimization relies on the repeated integration of the equations of motion to find $\mathbf{x}_{i+1}$.

4 Model and computer simulation

4.1 Model

The model comprises a point mass m a springy leg of length ℓ (see Figure 4 (a) middle figure). The linear spring constant is k and the free length of the leg is ℓ_0. The force applied by the springy leg on the body when in contact with the ground is $F_s = k(\ell_0 - \ell)$. There is a hip actuator (not shown) that allows places the swing leg at an angle θ to the vertical before touchdown. Gravity vector is g and points downwards.

4.2 Equations of motion

The states of the model are $\{x, \dot{x}, y, \dot{y}\}$ where x, y are the x- and y- position of the center of mass and $\dot{x}, \dot{y}$ are the respective velocities. Figure 4 (a) shows the model of the hopper including a few events (e.g., apex, touchdown) that make up a single step. We give the complete set of events and phases in Fig. 4 (b).

The model starts at the apex for step i as shown in Fig. 4 (a) (i) where the state vector is, $\{x_i, \dot{x}_i, y_i, 0\}$. The model then falls under the effect of gravity,

$$\ddot{x} = 0, \qquad \ddot{y} = -g. \tag{7}$$

The model then sticks out its leg at the foot placement angle θ which is the control parameter. Ground contact occurs when $y - \ell_0 \cos(\theta) = 0$. Thereafter, the ground contact interaction occurs at touchdown as shown in Fig. 4 (a) (iii), and given by

$$m\ddot{x} = F_s \frac{x - x_c}{\ell}, \qquad m\ddot{y} = F_s \frac{y}{\ell} - mg \tag{8}$$

where x_c is the x-coordinate of the foot contact point that is set at every step when the leg makes ground contact, $F_s = k(\ell_0 - \ell)$, where $\ell = \sqrt{(x - x_c)^2 + y^2}$ is the instantaneous leg length. The take-off phase occurs when the ground reaction force is zero, $k(\ell - \ell_0) = 0$. Thereafter, the model has a flight phase and ends up in the next apex state, $\{x_{i+1}, \dot{x}_{i+1}, y_{i+1}, 0\}$ as shown in Fig. 4 (vii).

4.3 Poincaré map (reduced state)

As mentioned earlier, the hopper motion in continuous time and given by 4 state variables given by $\{x, \dot{x}, y, \dot{y}\}$. The use of

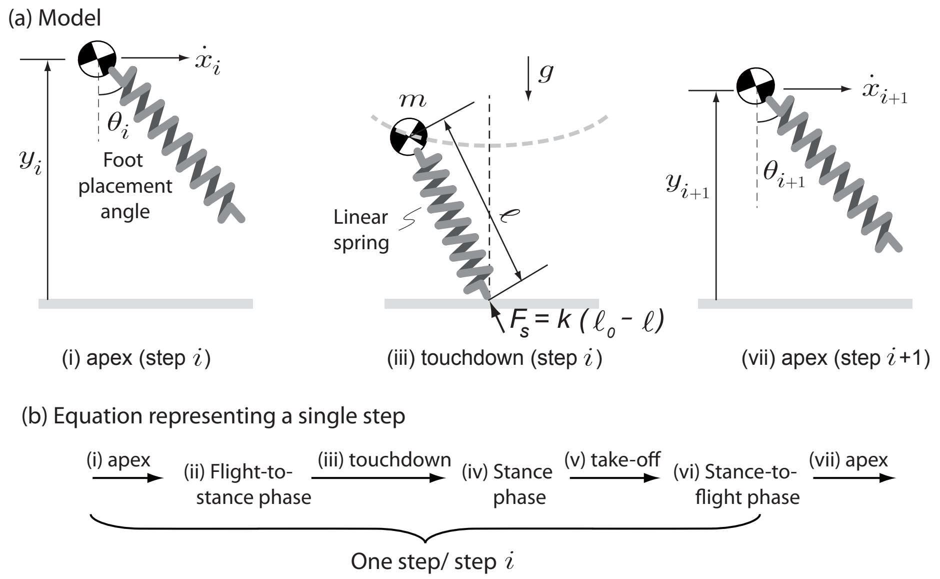

FIGURE 4. **(a) Model going through a single step:** The model starts in the flight phase at the apex position (vertical velocity is zero), followed by the stance phase, and finally ending in the flight phase at the apex position of the next step. There is a linear spring along the leg that applies the spring force F and a hip actuator (not shown) that can control the leg to the position θ with respect to the vertical before touchdown. **(b) Equation representing a single step:** Phases (e.g., stance phase) are shown between arrows and events are shown above the arrows. The equation represents a single step, step i for the model of running.

Poincaré map at the apex when $\dot{y}_i = 0$ eliminates the vertical velocity variable from the map. The state variable denoting the horizontal position at the apex x_i also drops as it does not capture the periodicity of the motion. Thus, using the Poincaré map for analysis, we reduce the system state to 2 variables $\{\dot{x}_i, y_i\}$. Thus the Poincaré map is $\mathbf{x}_{i+1} = \mathbf{F}(\mathbf{x}_i, \mathbf{u}_i)$ has $\mathbf{x}_i = \{\dot{x}_i, y_i\}$ and $\mathbf{u}_i = \theta_i$ or foot placement angle at touchdown.

4.4 Computer simulations

The model parameters for the simulation are mass $m = 100$ kg, nominal leg length $\ell_0 = 1$ m, spring constant $k = 10{,}000$ N/m, and gravity $g = 9.81$ m/s^2 based on human idealized to the SLIP model [14]. We build the forward simulator using MATLAB 2016a. It involves simulating a single step using phases/event stated in Fig. 4 (b) and Eqns. 7 and 8. We integrate these equations using *ode113* with an integration tolerance of 10^{-13} with in-built function *events* to detect a change in phase. Since we treat the simulator as a black-box, we also put checks to detect simulation failure. We consider the robot has failed if it meets any of the following conditions: (1) the horizontal speed ($\dot{x}$) is negative indicating falling backward; (2) the height of the point mass (y) is below the ground; (3) during take-off from the ground, the vertical velocity is negative ($\dot{y}_{\text{take-off}} < 0$); and (4) at the apex of the flight phase it meets the following condition, $y_{\text{apex-of-flight-phase}} < 0.95\ell_0$. The last condition is a arbitrarily set to 0.95 and is used to ensure that there is sufficient ground clearance for the swing leg.

5 Results

Fixed point: The Poincaré map presented in Sec. 4.3 has 2 equations but three variables, two states $\dot{x}_i$, y_i, and one control θ_i. We fix one of these three variables and solve for the remaining two using the two equations. Here, we chose a foot placement angle as 20° (0.349 rad) and using the computer simulation, solved for the fixed point $\mathbf{x}_0$ that satisfies Eqn. 2. We used the MATLAB function *fsolve* to get $\mathbf{x}_0 = \{\dot{x}_0, y_0\} = \{2.04, 1.038\}$. For all simulations we chose the design matrices in the cost function to be $\mathbf{Q} = diag(1, 100)$, $\mathbf{R} = 0.1$. In $\mathbf{Q}$ we have different values in the diagonal matrix based to normalize the states $\dot{x}$ and y which have different unit and scales.

Discrete Linear Quadratic Regulator (DLQR): For the discrete linear quadratic controller we use finite difference (specifically, central difference with a perturbation size of 10^{-4}) and the exact Poincaré map $\mathbf{F}$ to obtain the numerical values for $\mathbf{A}$ and $\mathbf{B}$

$$\mathbf{A} = \begin{bmatrix} 2.503, & -2.568 \\ -0.312, & 1.534 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -11.663 \\ 2.426 \end{bmatrix}$$

The stability of open-loop system is given by the eigenvalues of $\mathbf{A}$ and are 1.0 and 3.03. The system is stable if all the eigenvalues are less than 1. For the spring loaded inverted pendulum, one eigenvalue is always 1 because the system is conservative, i.e., it conserves the energy while the other eigenvalue of 3.03 showing that the limit cycle is unstable.

Closed-form approximation of the Poincaré map: To find an approximation of the Poincaré map we proceed as follows. Our data range for the input are: apex horizontal velocity in the range 0.5 m/s $\leq \dot{x}_i \leq$ 5 m/s with 0.5 m/s increments; apex height in the range 0.95 m $\leq y_i \leq$ 1.15 m in with 0.025 m increments; and foot placement in the range 5° (0.087 rad) $\leq \theta_i \leq 60^\circ$ (1.042 rad) with 5° (0.087 rad) increments. This generates 1080 input data points $\{\dot{x}_i, y_i, \theta_i\}$. For each input data point, we run a forward simulation (see Sec. 4.4) and save the output data, $\{\dot{x}_{i+1}, y_{i+1}\}$. Out of these 1080 data points, the robot failed 916 times and successfully took a step 164 times. This means that we only have 164 data points to fit a closed-form expression. Figure 6 (a) shows the input data $\mathbf{x}_i$ that resulted in a successful step and the corresponding Poincaré map $\mathbf{x}_{i+1}$. We use 75% or 119 successful steps for training and the rest 25% or 45 for testing. Each of the two outputs $\dot{x}_{i+1}$ and y_{i+1} was curve fitted to the inputs, $\mathbf{x}_i$ and u_i separately using each of the three regression models.

1. **Polynomial regression model:** A second order polynomial (P2) for each of the two outputs. Each polynomial has 10 constants. We use MATLAB function *lsqnonlin* to fit each polynomial. In our testing, we found that 87% and 100% of the fit for $\dot{x}_i$ and y_i respectively was within 90% accuracy.
2. **Gaussian process regression model:** A Gaussian process (GP) with squared exponential kernel as the covariance function and a constant basis function for each of the two outputs. The number of parameters in this model is only 2. We use MATLAB function *fitrgp* to fit the GP. In our testing, we found that 91% and 100% of the fit for $\dot{x}_i$ and y_i respectively was within 90% accuracy.
3. **Neural network regression model:** A neural network (NN) with 14 hidden layers for each of the two outputs. The model contains 71 constants. We use the MATLAB function *fitnet* to fit the NN. In our testing, we found that 91% and 100% of the fit for $\dot{x}_i$ and y_i respectively was within 90% accuracy.

To summarize, our metric for 90% accuracy was met by 90% of the $\dot{x}_i$ data and 100% of the y_i data.

Evaluating the controllers We have discussed 5 controllers, (1) Discrete linear quadratic regulator (see Sec. 3.2.1), (2) (3) (4) Discrete nonlinear quadratic regulator (see Sec. 3.2.2) with closed-form approximations for the Poincaré map P2, GP, and NN respectively and (5) Discrete nonlinear quadratic regulator (see Sec. 3.2.3) with exact Poincaré map. We tested each of these controllers on the same set of 90 initial conditions generated as follows. We generated 10 apex horizontal velocities, 0.5 m/s $\leq \dot{x}_i \leq$ 5 m/s in increments of 0.5 m/s and 9 apex vertical heights 0.95 m $\leq y_i \leq$ 1.15 m in increments of 0.025 m. By combining apex horizontal velocities and apex vertical heights we obtained $10 \times 9 = 90$ initial conditions to test the 5 controllers. For the non-linear optimizations (see Secs. 3.2.2 and 3.2.3), we constrained the foot placement angle to be in the range 5° (0.087 rad) $\leq \theta_i \leq 60^\circ$ (1.042 rad). For each of the 90 initial conditions, we use each of the 5 controllers to compute the foot placement angle for 5 consecutive steps. Given an initial condition for the start of step 1, if the robot can run successfully for 5 consecutive steps without falling down, then we consider the controller to stabilize the robot for that initial condition. We simulated only 5 consecutive steps because we found that if the system failed, it usually did so on steps 1 to 3, thus 5 was long enough to find out stability without leading to extensive forward simulations.

Figure 6 shows the results for each of the 5 controllers. The shaded region indicates the stable initial conditions (i.e., the robot does not fall in 5 consecutive steps). The percentages of initial conditions successfully stabilized for each controller were: linear optimization with linearized model for F was 48.89%, non-linear optimization with F approximated with P2 was 91.11%, with GP was 82.22%, with NN was 93%, and non-linear optimization with exact F obtained through integration was 96.67%.

Figure 7 gives the results for the 5 controllers when starting from the same initial condition $\mathbf{x}_i = \{2.5, 1.075\}$, an arbitrary choice. In (a), we show the foot placement angle as a function of the step length. We can see that except DLQR, other four controllers give almost the same foot placement angle after about 2 steps indicating that the robot has reached a steady-state. We can infer the steady-state value from the plot for $\dot{x}_i$ in (b) and for y_i in (c). We show the one-step cost in (d), where the lowest cost indicates a more optimal controller. We can see that optimization with neural network model approximation and exact model have the lowest cost, while the optimization with polynomial and Gaussian process regression have the highest cost. The cost of DLQR is in between but keeps changing every step, it does not achieve a steady-state value. Figures 7 (e) shows the time needed for computation of the foot placement control and (f) shows the time from apex to foot placement. Note that both these plots have the same y-scale to enable easy comparison. For the optimization to find the controller in real-time, the computation time should be less than the time taken to go from apex to stance. We can see that only GP, P2, and DLQR can achieve fast computation time within the time constraints while NN and SIM-based computation takes too much time. We did all computations using MATLAB on a Macbook circa 2012. We do not see that the computation time is a real impediment in this case because it is possible to speed up the computations using either a faster computer or using a compiled language (e.g., C).

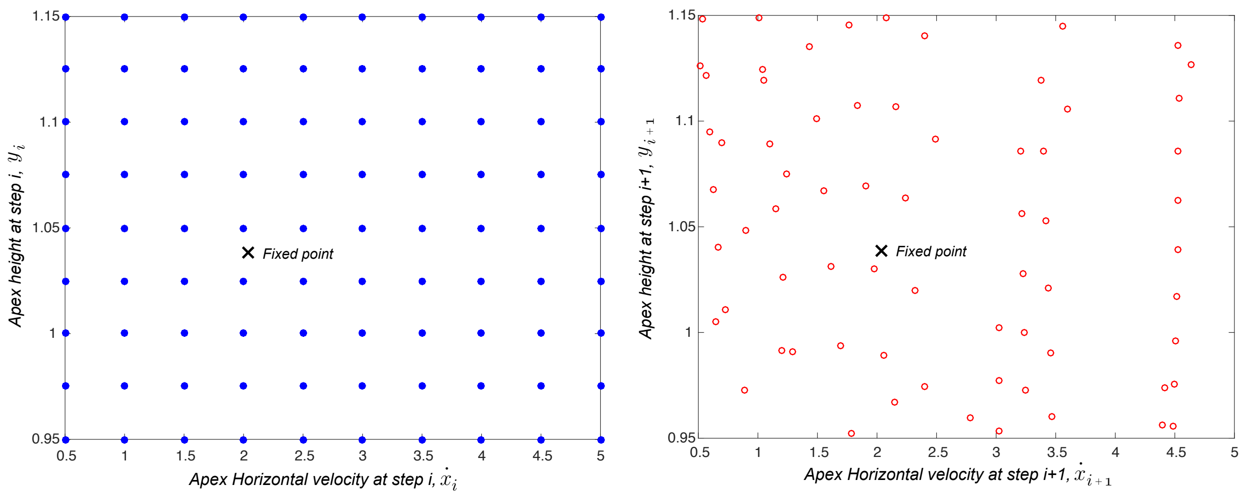

FIGURE 5. **Data points for fitting the Poincaré section:** (a) State values at step i, $\mathbf{x}_i = \{\dot{x}_i, y_i\}$ were chosen and (b) State values that resulted in a successful step $i+1$, $\mathbf{x}_{i+1} = \{\dot{x}_{i+1}, y_{i+1}\}$ were obtained using the forward simulation. The cross denotes the fixed point $\mathbf{x}_0$.

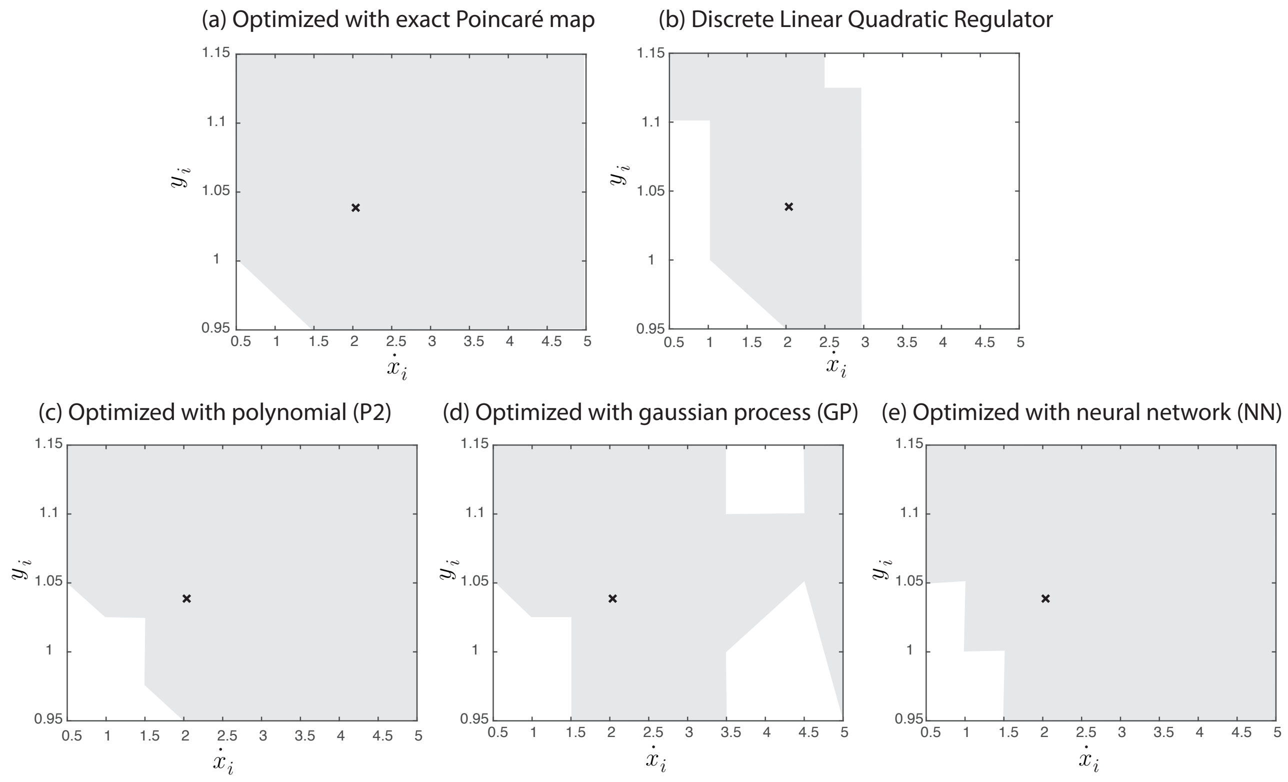

FIGURE 6. **Initial conditions that are stabilized by the different controllers:** The horizontal axis is the apex horizontal speed $\dot{x}_i$ and vertical axis is the apex height y_i for: (a) non-linear optimization with exact F, (b) discrete linear quadratic regulator based on linearized model F, (c) optimization using polynomial regression of order 2 for F (P2), (d) optimization using Gaussian process (GP) regression for F , and (e) optimization using neural network (NN) regression approximation to F . The cross denotes the fixed point $\mathbf{x}_0$, optimization points are shown by blue dots, and gray region is the set of initial conditions that are stabilized by the particular controller.

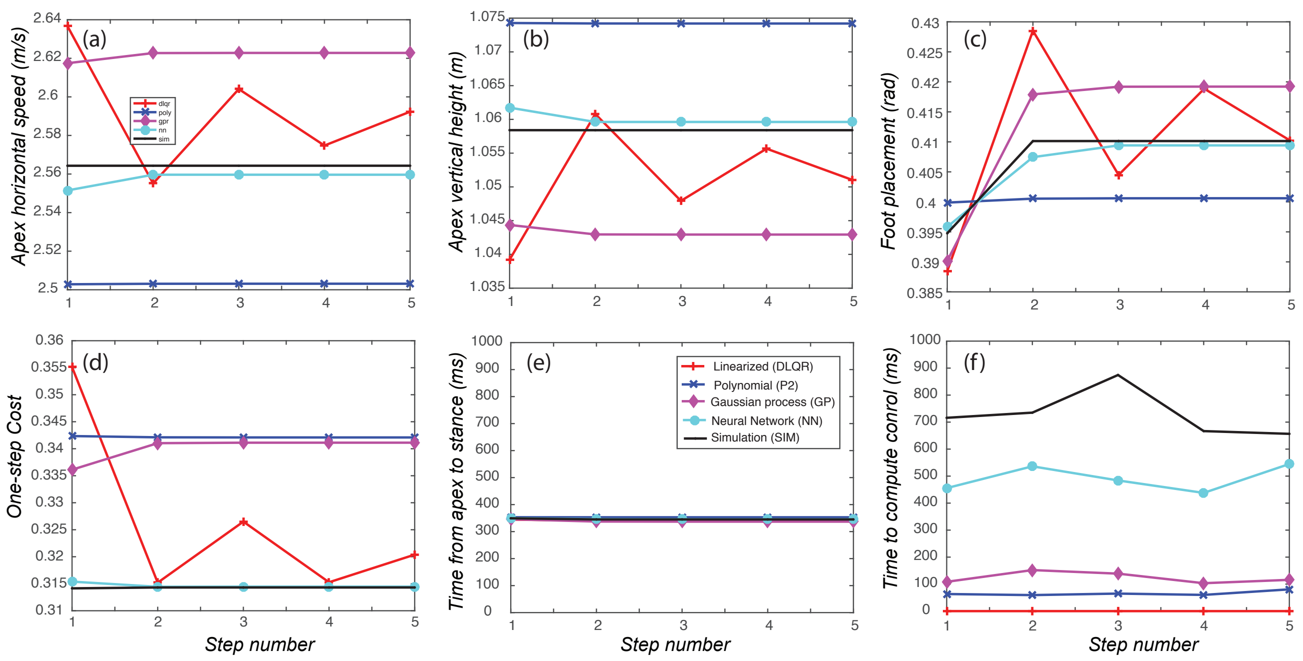

FIGURE 7. **Result obtained by starting with the same initial condition and evaluation on each of the** 5 **controllers:** $\mathbf{x}_i = \{2.5, 1.075\}$ for the 5 controllers: (a) apex horizontal velocity $\dot{x}$ m/s (b) apex height y m, (c) foot placement θ rad, (d) one-step cost, (e) time from apex to stance in milliseconds, (f) time to compute the controller.

6 Discussion

The work presented multiple regression methods (linear, quadratic, neural network, and Gaussian process regression) to create a closed-form approximation of the step-to-step map for the spring-loaded inverted pendulum model of running followed by optimization using these approximations. These results were compared against optimization using the accurate step-to-step map obtained through numerical integration. We found that the quadratic model approximation (P2) offers the best compromise between the range of initial conditions stabilized and computationally costs.

The main goal of this paper was to investigate if the closed-form approximation of the step-to-step dynamics can: (1) stabilize a wide range of initial conditions, and (2) allows fast online optimization. We found that polynomial (P2) and neural network (NN) stabilizes a similar range of initial conditions, followed very closely by the Gaussian process (GP). However, only P2 and GP are fast enough for real-time implementation as the optimization time is less than half the step-time (see Fig. 7 (e), (f)). The discrete linear quadratic regulator (DLQR), although the fastest, can stabilize a relatively small range of initial conditions.

The speed of online computation of the controller depends on the complexity of the regression model used. In the case of P2 there are only 10 parameters and GP there are only 2 parameters. On the other hand, NN has 71 parameters and this adds more time to the optimization. One could potentially speed up these computations by writing C code but requires additional work (e.g., MEX files) and/or using smaller parameter set, which could potentially decrease the accuracy of the fit and consequently the number of initial conditions that may be stabilized.

The region of attraction (ROA) or the range of initial conditions that are guaranteed to be stabilized is another metric for the stability [27]. Figure 6 demonstrates the region of attraction for the 5 controllers. The optimization with an exact step-to-step map has the largest region of attraction followed by the NN and P2 with similar ROA's, followed closely by GP, and finally DLQR. These results suggest that approximating the step-to-step or Poincaré map using low-order polynomial models is a viable approach for online optimization.

A large volume of work exists on approximating the dynamics near the neighborhood of the fixed point using a linear or quadratic model of the step-to-step map (e.g., [5, 28]). In this paper, although we used a quadratic model (P2), we used data from a wider range of initial conditions near the fixed point to curve fit the parameters of the quadratic model. Our P2 approximation was accurate to about 87% for horizontal speed, yet they stabilized almost 91% of initial conditions. This result suggests that using approximation over a wider range around the fixed point, not in the small neighborhood region, has the potential to stabilize a wider range of initial conditions.

How accurate should the model be for control? Schwind and Koditschek [24] suggest that 90% or more model accuracy is sufficient. Using the thumb rule of 90% model accuracy, we found that our regression models can achieve this accuracy in $87\% - 91\%$ of the data depending on the model used. Our control achieves stable running for $82\% - 93\%$ of initial conditions compared to 96% achieved by the exact step-to-step map. These results suggest that 90% model accuracy is perhaps a good metric to aim for to enable high fidelity control.

One of the prime advantages of using the Poincaré map for control is that the map is a continuous function of the state and controls [29], although the instantaneous dynamics are piecewise continuous. We exploit the fact to approximate the Poincaré map with a smooth function. The other advantages of our methods are that we can use any black-box simulator for generating the data needed for closed-form approximation and the use of the closed-form model allows for fast computation that may enable real-time optimal control.

Our method has several limitations too. Our limited exploration with the spring-loaded inverted pendulum suggests that the quality of the result depends on the choice of parameterization, the number of constants used in the fit, etc. These aspects are specific to the particular system and we need deal with them on a case-by-case basis. We can interpret the closed-form approximation errors as a model mismatch that would lead to substantial tracking errors although stability may not be an issue. Although we have shown that the method works for a simple system with 2 states and 1 control, we need to investigate more complex models to demonstrate that the method scales to high degree of freedom systems.

7 Conclusions

In this paper, we presented the closed-form approximation of the step-to-step map to enable fast optimization of the spring-loaded inverted model of running. We found that both parametric (e.g., polynomial, neural network) and non-parametric (e.g., Gaussian process regression) approximations can represent the step-to-step map with sufficient accuracy to enable balance control over a wide range of initial conditions while being computationally modest. We conclude that offline approximation of the step-to-step map and its use for fast online optimization is a viable method for on-the-fly control of legged robots.

ACKNOWLEDGMENT

This work was supported by NSF grants 1946282 and 2010736 to Pranav A. Bhounsule.

REFERENCES

[1] Pratt, J., Carff, J., Drakunov, S., and Goswami, A., 2006. "Capture point: A step toward humanoid push recovery". In 2006 6th IEEE-RAS international conference on humanoid robots, IEEE, pp. 200–207.

[2] Strogatz, S., 1994. *Nonlinear dynamics and chaos*. Addison-Wesley Reading.

[3] Kim, M., and Collins, S. H., 2013. "Stabilization of a three-dimensional limit cycle walking model through step-to-step ankle control". In 2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR), IEEE, pp. 1–6.

[4] Van Der Linde, R. Q., 1998. "Active leg compliance for passive walking". In Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Vol. 3, IEEE, pp. 2339–2344.

[5] Kuo, A. D., 1999. "Stabilization of lateral motion in passive dynamic walking". *The International journal of robotics research,* ***18***(9), pp. 917–930.

[6] Kim, M., and Collins, S. H., 2017. "Once-per-step control of ankle push-off work improves balance in a three-dimensional simulation of bipedal walking". *IEEE Transactions on Robotics,* ***33***(2), pp. 406–418.

[7] Dai, H., and Tedrake, R., 2012. "Optimizing robust limit cycles for legged locomotion on unknown terrain". In 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), IEEE, pp. 1207–1213.

[8] Koolen, T., Boer, T. D., Rebula, J., Goswami, A., and Pratt, J. E., 2012. "Capturability-based analysis and control of legged locomotion. part 1: Theory and application to three simple gait models". *International Journal of Robotics Research, to appear*.

[9] Raibert, M. H., Brown Jr, H. B., and Chepponis, M., 1984. "Experiments in balance with a 3d one-legged hopping machine". *The International Journal of Robotics Research,* ***3***(2), pp. 75–92.

[10] Raibert, M. H., and Wimberly, F. C., 1984. "Tabular control of balance in a dynamic legged system". *IEEE Transactions on systems, man, and Cybernetics*(2), pp. 334–339.

[11] McGeer, T., 1991. "Passive dynamic biped catalogue". In In Proc. of 2nd International Symposium on Experimental Robotics, pp. 465–490.

[12] McGeer, T., 1990. "Passive dynamic walking". *The International Journal of Robotics Research,* ***9***(2), pp. 62–82.

[13] McGeer, T., 1993. "Dynamics and control of bipedal locomotion". *Journal of Theoretical Biology,* ***163***(3), pp. 277–314.

[14] Geyer, H., Seyfarth, A., and Blickhan, R., 2006. "Compliant leg behaviour explains basic dynamics of walking and running". *Proceedings of the Royal Society of London B: Biological Sciences,* ***273***(1603), pp. 2861–2867.

[15] Seyfarth, A., Geyer, H., Günther, M., and Blickhan, R., 2002. "A movement criterion for running". *Journal of*

biomechanics, ***35***(5), pp. 649–655.

[16] Seyfarth, A., Geyer, H., and Herr, H., 2003. “Swing-leg retraction: a simple control model for stable running”. *Journal of Experimental Biology,* ***206***(15), pp. 2547–2555.

[17] Shemer, N., and Degani, A., 2017. “A flight-phase terrain following control strategy for stable and robust hopping of a one-legged robot under large terrain variations”. *Bioinspiration & Biomimetics*.

[18] Ernst, M., Geyer, H., and Blickhan, R., 2012. “Extension and customization of self-stability control in compliant legged systems”. *Bioinspiration & biomimetics,* ***7***(4), p. 046002.

[19] Andrews, B., Miller, B., Schmitt, J., and Clark, J. E., 2011. “Running over unknown rough terrain with a one-legged planar robot”. *Bioinspiration & biomimetics,* ***6***(2), p. 026009.

[20] Bhounsule, P. A., 2015. “Control of a compass gait walker based on energy regulation using ankle push-off and foot placement”. *Robotica,* ***33***(06), pp. 1314–1324.

[21] Coleman, M., 2009. “Numerical accuracy case studies of two systems with intermittent dynamics a 2d rimless spoked wheel and a 3d passive dynamic model of walking”. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Application and Algorithms,* ***16***, pp. 59–87.

[22] Geyer, H., Seyfarth, A., and Blickhan, R., 2005. “Spring-mass running: simple approximate solution and application to gait stability”. *Journal of theoretical biology,* ***232***(3), pp. 315–328.

[23] Saranlı, U., Arslan, Ö., Ankaralı, M. M., and Morgül, Ö., 2010. “Approximate analytic solutions to non-symmetric stance trajectories of the passive spring-loaded inverted pendulum with damping”. *Nonlinear Dynamics,* ***62***(4), pp. 729–742.

[24] Schwind, W. J., and Koditschek, D. E., 2000. “Approximating the stance map of a 2-dof monoped runner”. *Journal of Nonlinear Science,* ***10***(5), pp. 533–568.

[25] Poulakakis, I., and Grizzle, J. W., 2009. “Modeling and control of the monopedal robot thumper”. In 2009 IEEE International Conference on Robotics and Automation, IEEE, pp. 3327–3334.

[26] Bhounsule, P. A., Ruina, A., and Stiesberg, G., 2015. “Discrete-decision continuous-actuation control: balance of an inverted pendulum and pumping a pendulum swing”. *Journal of Dynamic Systems, Measurement, and Control,* ***137***(5), p. 051012.

[27] Schwab, A., and Wisse, M., 2001. “Basin of attraction of the simplest walking model”. In Proceedings of the ASME design engineering technical conference, Vol. 6, pp. 531–539.

[28] Hamed, K. A., Buss, B. G., and Grizzle, J. W., 2016. “Exponentially stabilizing continuous-time controllers for periodic orbits of hybrid systems: Application to bipedal locomotion with ground height variations”. *The International Journal of Robotics Research,* ***35***(8), pp. 977–999.

[29] Grizzle, J., Abba, G., and Plestan, F., 2001. “Asymptotically stable walking for biped robots: Analysis via systems with impulse effects”. *IEEE Transactions on Automatic Control,* ***46***(1), pp. 51–64.