Meshing Hybrid Zero Dynamics for Rough Terrain Walking

Cenk Oguz Saglam and Katie Byl

***Abstract*— For an underactuated biped on a constant-slope terrain, the Hybrid Zero Dynamics (HZD) controller framework provides exponentially stable walking motions. In this paper, we quantify the stability of such a control system on rough terrain by estimating the average number of steps before failure. In addition, we show how to switch between multiple HZD controllers (optionally using terrain look-ahead) to increase the stability dramatically, e.g., 10 thousand steps compared to 10. To do this robustly, we make use of the new meshing method proposed in this paper.**

I. INTRODUCTION

Humans can benefit from humanoids in many ways. A major challenge in developing human-like robots is obtaining bipedal locomotion what is both energy efficient and stable. Toward this goal, passive walkers [1] have motivated *dynamic walking*, where underactuation is not avoided as in humans. In this paper we study a 5-Link planar biped with point feet that are modeled as unactuated, free pivots. This robot has hybrid dynamics, i.e., it experiences both continuous phases and discontinuous jumps [2]. For this walker on flat terrain, [3] provides a systematic design of a Hybrid Zero Dynamics (HZD) controller, which achieve stable walking motions. However, our simulations show that such a strategy will fail with several steps if even mild variability is added to the terrain profile. For such cases, e.g., on rough terrain, one needs to quantify stability. An intuitive approach is estimating the average number of steps before falling, aka the Mean First Passage Time (MFPT) before failure [4].

In this paper we first show how to design a HZD controller for constant-inclined terrain by slightly modifying [3]. After that, we have two main goals in this paper as follows.

1) Estimating MFPT: In [5] we showed how to estimate MFPT for any controller. Our method involves starting from the limit cycle walking motion to explore and mesh the reachable part of the state space. In this paper, in addition to applying the same method, we propose an alternative algorithm for HZD controllers. The latter will not be as accurate, however computational cost will be significantly lower due to the elegance of the HZD formulation.

2) Increasing MFPT: Again, in [5] we showed how to switch between multiple controllers designed a priori to increase the MFPT significantly. Our motivation is the fact that humans modify their motion while walking on rough terrain. Our method, based on the mesh mentioned in the previous paragraph, can also make use of the upcoming terrain information. In [5], we addressed the problem of noisy slope estimation. We also introduced the mesh-robustness problem by evaluating the performance of a policy obtained with a mesh approximation of the true dynamics. In the mentioned work, both meshes were obtained using the same method. Ensuring mesh-robustness was not obvious and our previous solution was ad-hoc. In the current paper, we will show how to achieve the same goal easily using the new meshing method proposed in this paper.

C. O. Saglam and K. Byl are with the Electrical and Computer Engineering Department, University of California, Santa Barbara, CA 93106 USA `saglam@ece.ucsb.edu, katiebyl@ece.ucsb.edu`

II. MODEL

A. The Biped

Figure 1 shows the 5-Link model of RABBIT [6]. We assume point feet. Ankles are not actuated, and this model has 1 DOF underactuation. The angles are given by $q := [q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5]^T$.

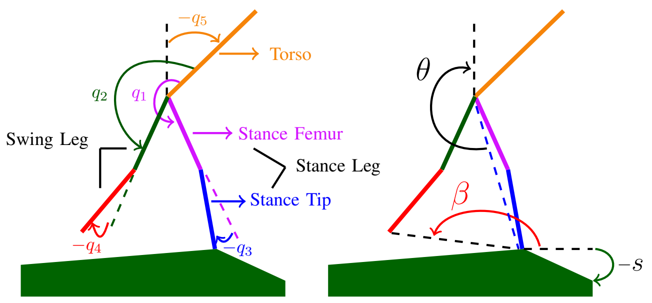


Fig. 1. Illustration of the five-link robot with identical legs. As we will explain later, s is the slope ahead of the robot, θ is called the phase variable, and β is angle from stance foot to swing foot.

Walking consists of steps. A step has two phases. In the single support (swing) phase, the robot will have only one leg (stance leg) in contact with the ground. The dynamics are described by

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{1}$$

where u is the input. This equation can be obtained by a Lagrangian approach. By defining the ten dimensional state as $x := [q^T \ \dot{q}^T]^T$, same dynamics can be equivalently expressed as

$$\dot{x} = f(x) + g(x)u. \tag{2}$$

A swing phase starts and ends with a double support phase, which can be modeled as an instantaneous impact event by

$$x^+ = \Delta(x^-) := \begin{bmatrix} \Delta_q \ q^- \\ \Delta_{\dot{q}}(q^-) \ \dot{q}^- \end{bmatrix}, \tag{3}$$

where $x^- = [(q^-)^T \ (\dot{q}^-)^T]^T$ and x^+ are the states just before and after the impact respectively. Although angular positions are assumed to stay constant at the impact, we relabel the stance and swing legs. So, Δ_q is such that we have $\Delta_q[q_1\ \ q_2\ \ q_3\ \ q_4\ \ q_5]^T = [q_2\ \ q_1\ \ q_4\ \ q_3\ \ q_5]^T$. The function $\Delta_{\dot{q}}(q)$ is obtained using conservation of energy and the principle of virtual work [6], [2].

B. The Terrain

In this paper we assume the terrain ahead of the robot is a constant slope until an impact occurs. So each step will experience a slope and the terrain will be angular. As shown in Figure 1, we will denote the slope by s. This terrain assumption captures the fact that to calculate the pre-impact state, the terrain for each step can simply be interpreted as a ramp with the appropriate slope. Then, the next state of the robot, $x[n+1]$, is a function of the current state $x[n]$, the slope ahead $s[n]$, and the controller used $\zeta[n]$, i.e.,

$$x[n+1] = h^t(x[n], s[n], \zeta[n]), \tag{4}$$

where superscript t denotes the terrain assumption we made.

III. HZD FOR INCLINED TERRAIN

In this section we summarize the Hybrid Zero Dynamics (HZD) control for the 5-Link biped explained in [6]. In addition, we will extend it for inclined (constant-sloped) terrain as in [7]. This generalization can be made for all Hybrid Zero Dynamics (HZD) control frameworks. However, we will follow the specific choices made for 5-Link biped for clarity and space considerations.

A. The Structure

First, we choose a phase-variable, which will work as an internal-clock. It should be monotonic through the step, e.g., hip location moves forward with an almost constant velocity in humans [8]. We choose θ shown in Figure 1 as our phase variable. Since femur and tip lengths are equal in RABBIT [6], this will correspond to having $\theta = cq$, where $c = [-1\ 0\ -1/2\ 0\ -1]$.

Secondly we decide on four independent variables to control, namely h_0. It is intuitive to control the relative (internal) angles, i.e., $h_0 := [q_1\ q_2\ q_2\ q_3\ q_4]^T$. Then h_0 is in the form of $h_0 = H_0 q$, where $H_0 = [I_4\ 0]$.

Thirdly, let $h_d(\theta)$ be the references for h_0. Then the tracking error is given by

$$h(q) := h_0(q) - h_d(\theta) = H_0 q - h_d(cq) \tag{5}$$

Taking the first derivative with respect to time we get

$$\dot{h} = \frac{\partial h}{\partial q}\dot{q}, \tag{6}$$

which can be equivalently written as

$$\dot{h} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x}f(x) =: \mathcal{L}_f h = \langle \nabla h, f \rangle, \tag{7}$$

where we used the fact that $\frac{\partial h}{\partial x}g(x) = 0$. For the clarity of later equations, we will use the Lie derivative ($\mathcal{L}$) notation. Then, we have

$$\ddot{h} = \mathcal{L}_f^2 h + \mathcal{L}_g \mathcal{L}_f h\ u. \tag{8}$$

Substituting the controller structure

$$u = (\mathcal{L}_g \mathcal{L}_f h)^{-1}(-\mathcal{L}_f^2 h + v) \tag{9}$$

to (8) yields

$$\ddot{h} = v. \tag{10}$$

There are various methods to design v to force h (and $\dot{h}$) to zero [9]. While even a PD controller would do the job, a Sliding Mode Control (SMC) would be preferable for finite time convergence [10], which is summarized in the Appendix.

On the "zero dynamics manifold", noted by $\mathcal{Z}$, we have $h = 0$ and $\dot{h} = 0$. The rest of this section considers the dynamics on this manifold. The goal is to design h_d such that h and $\dot{h}$ stay as zero even at impacts (hybrid invariance) and a stable walking motion is correspondingly generated.

B. Swing Phase Zero Dynamics

Let V be the potential energy of the robot, $\gamma_0(q)$ be the last row of D and define $\gamma := \gamma_0 \dot{q}$. Then, $\mathcal{L}_g \gamma = 0$ and $\mathcal{L}_f \gamma = -\frac{\partial V}{\partial q_5}\Big|_{\mathcal{Z}}$. For $x \in \mathcal{Z}$ we transform coordinates by

$$\xi_1 = \theta, \quad \xi_2 = \gamma, \tag{11}$$

to equivalently express the system dynamics during the swing phase as

$$\dot{\xi}_1 = c\dot{q}, \quad \dot{\xi}_2 = \mathcal{L}_f \gamma, \tag{12}$$

where the right-hand sides are evaluated at

$$\begin{gathered} q = H^{-1}\begin{bmatrix} h_d \\ \xi_1 \end{bmatrix} \\ \text{(because } \xi_1 = \theta = cq \text{ and } H_0 q = h_0 = h_d \text{ for } x \in \mathcal{Z}\text{)} \\ \dot{q} = \begin{bmatrix} \frac{\partial h}{\partial q} \\ \gamma_0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \xi_2 \end{bmatrix} \\ \text{(because } \dot{h} = 0 \text{ for } x \in \mathcal{Z} \text{ and } \xi_2 = \gamma = \gamma_0 q\text{)} \end{gathered} \tag{13}$$

To evaluate $\dot{q}$, we can alternatively differentiate q from above to get

$$\dot{q} = H^{-1}\begin{bmatrix} \frac{\partial h_d}{\partial \theta} \\ 1 \end{bmatrix}\dot{\theta} \tag{14}$$

Using (13) we can write (12) in the following form.

$$\dot{\xi}_1 = \kappa_1(\xi_1)\xi_2, \quad \dot{\xi}_2 = \kappa_2(\xi_1), \tag{15}$$

C. Hybrid Zero Dynamics Impact

As we will see, h_d will be designed such that $x^- \in \mathcal{Z}$ will imply $x^+ \in \mathcal{Z}$. Now assume the swing foot passed the stance foot and had an impact. Such an impact will happen when $\beta = s$, cf. Figure 1. Let q_0^- denote the angles just before the impact. Then, q_0^- will be the solution to

$$\begin{bmatrix} h(q) \\ \beta \end{bmatrix} = \begin{bmatrix} 0 \\ s \end{bmatrix}. \tag{16}$$

At the impact, the angles will be relabeled as in (3). So, after the impact we will have $q_0^+ = \Delta_q q_0^-$. The phases at the beginning and end of the step are simply $\theta^+ := cq_0^+$ and $\theta^- := cq_0^-$ respectively.

Similarly let $\dot{q}_0^-$ and $\dot{q}_0^+$ denote the velocities just before and after the impact respectively. ξ_2^- and ξ_2^+ are the corresponding ξ_2 values. Then, using (13) we get

$$\dot{q}_0^- = \begin{bmatrix} \frac{\partial h}{\partial q}(q_0^-) \\ \gamma_0(q_0^-) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \xi_2^- =: \overline{\lambda}_{\dot{q}}\ \xi_2^- \tag{17}$$

From (3), we get $\dot{q}_0^+ = \Delta_{\dot{q}}(q_0^-)\ \dot{q}_0^-$and $\xi_2^+ = \gamma_0(q_0^+)\ \dot{q}_0^+$. Defining $\delta_{\text{zero}} := \gamma_0(q_0^+)\ \Delta_{\dot{q}}(q_0^-)\ \overline{\lambda}_{\dot{q}}$, the impact map on zero dynamics manifold is given by

$$\xi_1^+ = \theta^+, \quad \xi_2^+ = \delta_{\text{zero}}\ \xi_2^-. \tag{18}$$

D. Poincaré Analysis

We then define a Poincaré section just before the impact. We already know the angles are q_0^-. To find the fixed point on this Poincré map (if it exists), we also need a fixed ξ_2^-. Define $\xi_{2b} := \xi_2^2/2$. ξ_{2b}^+ and ξ_{2b}^- will denote its value at the beginning and end of the step respectively. Then, we have

$$\frac{d\xi_{2b}}{d\xi_1} = \frac{\kappa_2(\xi_1)}{\kappa_1(\xi_1)}. \tag{19}$$

Integration over a step gives

$$\xi_{2b}^- - \xi_{2b}^+ = \int_{\theta^+}^{\theta^-} \frac{\kappa_2(\xi_1)}{\kappa_1(\xi_1)} d\xi_1. \tag{20}$$

Since impact maps ξ_{2b} to $\delta_{\text{zero}}^2 \xi_{2b}$, the fixed point ξ_{2b}^* is obtained by solving

$$\xi_{2b}^* - \delta_{\text{zero}}^2 \xi_{2b}^* = \int_{\theta^+}^{\theta^-} \frac{\kappa_2(\xi_1)}{\kappa_1(\xi_1)} d\xi_1. \tag{21}$$

Then, the fixed points for the zero dynamics are

$$\xi_1^* = \theta^-, \quad \xi_2^* = \sqrt{\frac{-2}{1-\delta_{\text{zero}}^2} \int_{\theta^+}^{\theta^-} \frac{\kappa_2(\xi_1)}{\kappa_1(\xi_1)} d\xi_1}. \tag{22}$$

E. Constraints

Now using the fixed point as an initial condition, we can simulate the 2D dynamics described in (12) until $\theta = \theta^-$ and calculate the steady-state Cost of Transport (COT).

Note that this is a brief summary of [3]. There are some assumptions we made and many constraints we should check that are not repeated here. For the force constraints, we should keep in mind to consider the slope. If F^N and F^T are the forces calculated after simulation of the 2D dynamics as explained in [3], then they should be updated as

$$\begin{aligned} F^N_{\text{updated}} &= F^N cos(s) - F^T sin(s), \\ F^T_{\text{updated}} &= F^N sin(s) + F^T cos(s). \end{aligned} \tag{23}$$

F. Reference Design

As [3] suggests, we use Bézier polynomials to form h_d and next optimize for COT. While minimizing energy consumption, h_d should satisfy the constraints mentioned above. In particular, for hybrid invariance, $x^- \in \mathcal{Z}$ should imply $x^+ \in \mathcal{Z}$, i.e., once the walker is in the manifold it should stay there.

First, we scale and shift θ to have an internal clock which ticks from 0 to 1 in a step.

$$\tau(q) := \frac{\theta(q) - \theta^+}{\theta^- - \theta^+} \tag{24}$$

Then, Bézier curves are given by

$$b_i(\tau) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{25}$$

which form the reference as

$$h_d(\theta) := \begin{bmatrix} b_1(\tau) \\ b_2(\tau) \\ b_3(\tau) \\ b_4(\tau) \end{bmatrix}. \tag{26}$$

Choosing $M = 6$ yields $(6+1) \times 4 = 28$ α_k^i parameters to optimize. However, hybrid invariance constraint will eliminate $2 \times 4 = 8$ of them. First, assume $h = 0$ at the impact. To achieve $h = 0$ after the impact also, we require

$$\begin{aligned} q_0^+ &= \Delta_q q_0^- \\ H^{-1} \begin{bmatrix} \alpha_0 \\ \theta^+ \end{bmatrix} &= \Delta_q H^{-1} \begin{bmatrix} \alpha_M \\ \theta^- \end{bmatrix} \\ \begin{bmatrix} \alpha_0 \\ \theta^+ \end{bmatrix} &= H \Delta_q H^{-1} \begin{bmatrix} \alpha_M \\ \theta^- \end{bmatrix}, \end{aligned} \tag{27}$$

which gives Bézier coefficient α_0 using α_M.

Secondly, it is easy to verify

$$\begin{aligned} \left.\frac{\partial h_d}{\partial \theta}\right|_{\theta^+} &= \frac{M}{\theta^- - \theta^+}(\alpha_1 - \alpha_0), \\ \left.\frac{\partial h_d}{\partial \theta}\right|_{\theta^-} &= \frac{M}{\theta^- - \theta^+}(\alpha_M - \alpha_{M-1}). \end{aligned} \tag{28}$$

Assume $\dot{h} = 0$ at the impact. Remember (14). Achieving $\dot{h} = 0$ after the impact means

$$\dot{q}_0^+ = \Delta_{\dot{q}}(q_0^-)\ \dot{q}_0^-, \tag{29}$$

or equivalently

$$H^{-1} \begin{bmatrix} \left.\frac{\partial h_d}{\partial \theta}\right|_{\theta^+} \\ 1 \end{bmatrix} \dot{\theta}^+ = \Delta_{\dot{q}}(q_0^-)\ H^{-1} \begin{bmatrix} \left.\frac{\partial h_d}{\partial \theta}\right|_{\theta^-} \\ 1 \end{bmatrix} \dot{\theta}^-, \tag{30}$$

from which we calculate α_1.

This section is provided to make paper self contained. The equations are taken from [3]. The key change we made was in (16), which allows optimizing for different slopes. Code is available at MATLAB File Exchange [11] to implement this control. Optimizing for flat terrain we obtained the parameters provided in Table I and a COT of 0.18836.

IV. MESHING

Our goal in meshing is to approximate the hybrid dynamics of walking as a Markov Chain. To do so, first we need to determine a controller set Z, a slope set S and a state set Y. The controller set is rather easy: it consists of all the available controllers we choose to design. Z should have at least one controller. Let ζ_s denote the controller optimized assuming

TABLE I

OPTIMIZED VALUES IN RADIANS FOR MINIMAL COT ON FLAT TERRAIN

i	1	2	3	4
α_0^i	3.6151	3.0349	-0.4162	-0.3200
α_1^i	3.6413	2.9006	-0.6657	-0.2484
α_2^i	3.3894	2.9544	-0.3732	-0.3690
α_3^i	3.2884	3.5470	-0.3728	-1.1041
α_4^i	3.1135	3.5186	-0.2359	-0.3973
α_5^i	3.1708	3.6851	-0.3780	-0.4260
α_6^i	3.0349	3.6151	-0.3200	-0.4162

slope is s and constrained to have a speed of 0.8m/s. In this paper we optimized 7 different controllers to obtain

$$Z = \{\zeta_{-3},\ \zeta_{-2},\ \zeta_{-1},\ \zeta_0\ \zeta_1,\ \zeta_2,\ \zeta_3\}. \tag{31}$$

Determining slope set is straightforward also. For the controllers we designed, by simulating we can easily have an idea about what slope range the robot may possibly walk on. The slope set should at least contain that range. It is perfectly fine to cover more. Then, we typically uniformly pick certain number of slopes. For example, in this paper we will use

$$S = \{k^\circ/3 \ \mid\ k \in \mathbb{Z},\ -30 \le k \le 30\}. \tag{32}$$

The state set should intuitively cover the reachable part of the state space. It should be dense enough for accuracy while not having "too many" elements. Determining the state set will be one of the goals in meshing, as we will see shortly. The second goal will be learning what $h^t(y[n], s[n], \zeta[n])$ is for all $y[n] \in Y$, $s[n] \in S$, and $\zeta[n] \in Z$. Without loss of generality we will reserve the first state (y_1) to represent an absorbing failure state.

A. Meshing Reachable State Space

Determining state set is difficult because we are studying high dimensional systems, e.g., the 5-Link walker has a 10 dimensional state (i.e., positions and velocities). So, it is *not* feasible to uniformly and densely cover a hypercube that includes the reachable state space. However, meshing the reachable state space can be achieved by starting from a very small number of states (one non-failure state is enough) and deterministically expanding by iteratively simulating [5]. Our algorithm works as follows. We initially start by setting $\overline{Y} = \{y \in Y,\ y \neq y_1\}$, which will correspond to all the states that are not simulated yet. Then we start the following iteration: As long as there is a state $y \in \overline{Y}$, simulate to find all possible $h^t(y[n], s[n], \zeta[n])$ and remove y from $\overline{Y}$. For the points newly found, check their distance to the other states in Y. If the distance is larger than some threshold d_{thr}, i.e., the point is far enough from all existing mesh points, then add that point to Y and $\overline{Y}$.

Using the right distance metric is key in ensuring that Y has a small number of states while accurately covering the reachable state space. Standardized (normalized) Euclidean distance turned out to be extremely useful as it dynamically adjusts the weights for each dimension according to its standard deviation at any mesh iteration [12]. The distance of a vector x from Y is calculated as

$$d(x, Y) := \min_{y \in Y} \left\{ \sqrt{\sum_i \left(\frac{x_i - y_i}{r_i}\right)^2} \right\}, \tag{33}$$

where r_i is the standard deviation of i^{th} dimension of all existing points in set Y. In addition, the closest point in Y to x is given by

$$c(x, Y) := \operatorname*{argmin}_{y \in Y} \left\{ \sum_i \left(\frac{x_i - y_i}{r_i}\right)^2 \right\}. \tag{34}$$

Our algorithm allows us to increase the accuracy of the final mesh at the expense of producing a higher number of states (larger Y) by decreasing threshold distance d_{thr}. For space considerations, we will refer interested reader to [5]. For this paper we obtained a mesh using $d_{thr} = 0.5$, which resulted with 115,990 points. We will refer to this as the *full-mesh.*

The key point for meshing the 10D space is the fact that the reachable state space is actually a quasi-2D manifold [13]. For the HZD controller, this corresponds to $\mathcal{Z}$, i.e., the zero dynamics manifold. However, we will see that this is an approximation on mildly rough terrain.

B. Meshing HZD Surface

In this paper, we propose a new strategy for meshing. Instead of expanding Y as we simulate, we pre-determine it initially. Remember that for each controller, we have a zero dynamics manifold as illustrated in Figure 2, where the x-axis is ξ_2 and left y-axis is ξ_1. We simulate from $(\xi_1, \xi_2) = (\theta^+, \delta_{\text{zero}}\ \xi_2^*)$ once, until ξ_1 stops increasing to obtain the blue trajectory in the figure. The non-linear right y-axis shows the corresponding β values in degrees. On flat terrain, the impact occurs when $\beta = 0$, where $\xi_1 = \theta^-$ and $\xi_2 = \xi_2^*$. As shown with dashed line in the figure, this impact maps back to $(\xi_1, \xi_2) = (\theta^+, \delta_{\text{zero}}\ \xi_2^*)$, where $\beta = 180°$. Thus, we have a limit cycle. However if the terrain is not flat, the blue curve would impact (intersect the Poincaré section) at another point on the curve. In addition, if the initial ξ_2 was lower (higher) than $\delta_{\text{zero}}\ \xi_2^*$, then the trajectory would shift to left (right) and deform. So, by starting with different ξ_2 values and experiencing different s values, the model will visit a variety of trajectories on this 2D manifold.

We then choose a rectangle to mesh. The y-axis points are taken from the slope set, e.g., (32). However, the HZD controller we obtained will not necessarily reach all values in this set. To illustrate, the controller ζ_0 will not reach $s < -8$ while ξ_1 is monotonically increasing. So we mesh points with $s \in S$ such that $s \ge -8$, corresponding to $\zeta_1 \ge -156.3°$. Limits for the x-axis can be selected conservatively. In this paper, we pick points from $-2\xi_2^*$ to 0. As a result, we mesh the shaded rectangle shown in Figure 2, where a toy mesh with $5 \times 5 = 25$ states is also shown. We will obtain the actual mesh by uniformly distributing $[-2\xi_2^*, 0]$ for 100 points and use the reachable part of the slope set, which is also uniform. In the end, our combined mesh will consist of multiple uniform 2D grids, i.e., one 2D grid for each of the controllers (7 in this paper) available. That is, a given mesh point includes three values: ξ_1, ξ_2, and the (previous) controller, ζ, used in driving the system to a particular HZD manifold during the step immediately preceding the (pre-impact) Poincaré mesh state. We will refer to this full mesh as a *uniform-mesh*, which turns out to have 29,701 states in this work.

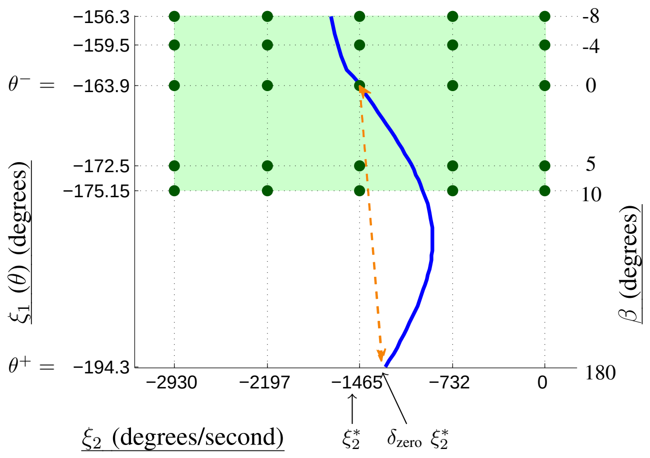


Fig. 2. Zero dynamics manifold for ζ_0. Axis names are shown underlined and y-axis on the right is not linear. Blue curve is the trajectory going through the limit cycle. Limit cycle on flat terrain includes the section of this curve until $\zeta_1 = \theta^-$. Then, an impact (shown as dashed line) maps it back to the beginning of the same curve. The green shaded area is where we would like to mesh. Toy manifold illustrates meshing with $5 \times 5 = 25$ states. Recall that ζ_1 and ζ_2 represent weighted sums of joint positions and velocities, respectively.

However, this meshing will not be able to capture the full dynamics as accurately as the full meshing of previous section. The reason is, the robot will leave this manifold on rough terrain, e.g., impact at $\beta \neq 0$ for blue trajectory on Figure 2 causes $x^+ \notin \mathcal{Z}$. Fortunately, as we hoped in using this approach and also observe, this mesh will still give very good information about the full dynamics. This is because, although impacts push away from $\mathcal{Z}$, our low-level sliding-mode control is designed such that trajectories will converge back to one of the HZD manifolds before the next impact. There are three main advantages of using this kind of meshing over the first one. (1) It ends up with fewer states, because it is uniform. Imagine we would like to mesh the line segment given by [0,1]. In this case dispersion would correspond to the longest segment in [0,1] without any points [14]. For a fixed dispersion, say 0.1, uniform-mesh would give the least number of states (9 points). (2) It is much faster to mesh and simulate using this method. This is partly because of the number of states in the mesh, but also expanding iteratively and checking distances repeatedly make full-meshing take much longer and make parallel computation less practical. (3) Ensuring robustness of the policies is much more trivial. This is a very important aspect for our work as we will explain at the end of Section VII.

V. OBTAINING MARKOV CHAIN

By meshing and simulating in the previous section we obtain $h^t(y, s, \zeta)$ for all $y \in Y$, $s \in S$, and $\zeta \in Z$. We define

$$h^a(x[n], s[n], \zeta[n]) := c(h^t(x[n], s[n], \zeta[n]), Y). \tag{35}$$

where (34) is used, and the superscript a stands for approximation. Then the approximate step-to-step dynamics are given by

$$y[n+1] = h^a(y[n], s[n], \zeta[n]). \tag{36}$$

Using this we can write the deterministic state transition as

$$T^d_{ij}(s, \zeta) = \begin{cases} 1, & \text{if } y_j = h^a(y_i, s, \zeta) \\ 0, & otherwise. \end{cases} \tag{37}$$

A Markov Chain can be represented by a stochastic state-transition matrix T^s is defined by

$$T^s_{ij} := Pr(y[n+1] = y_j \mid y[n] = y_i). \tag{38}$$

To calculate this matrix, the first thing we need to do is assume a distribution over slope set, noted by $P_S := Pr(s[n] = s)$. In this paper, we will assume a normal distribution for P_S, with mean μ_s, and standard deviation σ_s. After distributing s values, we end up with a Markov Decision Process model. The last step to end up with a Markov Chain is to decide on which controller to use. If only one of the controllers, say ζ_i is used (no switching), T^s can be calculated as

$$T^s = \sum_{s \in S} P_S(s)\ T^d(s, \zeta_i). \tag{39}$$

More generally, we consider policies in the form of

$$\zeta[n] = \pi(y[n], \tilde{s}[n]), \tag{40}$$

where π is the function determining which controller to use and $\tilde{s}$ is the noisy slope information given by

$$\tilde{s} = max(min(S), min(max(S), s + l)), \tag{41}$$

where l is the noise. Note that this says $\tilde{s} = s + l$ except at boundaries of the slope set. We define $P_L(l) := Pr(l[n] = l)$. l is normally distributed with zero mean and standard deviation σ_l. Then the approximate dynamics (36) will be

$$y[n+1] = h^a(y[n], s[n], \pi(y[n], \tilde{s}[n])). \tag{42}$$

As a result the stochastic state-transition matrix is given by

$$T^s_{ij} = \sum_{s \in S} \sum_{l \in S} P_S(s)\ P_L(l)\ T^d_{ij}(s, \pi(y_i, \tilde{s})). \tag{43}$$

VI. MEAN FIRST PASSAGE TIME (MFPT)

In the presence of enough noise, e.g., roughness of the terrain, the robot is destined to fall. However, if it takes many steps (thousands) before doing so, the system can be said to be metastable [4]. In this section, we will summarize how to estimate the average number of steps before failing (MFPT). We invite interested readers to see [15] for details.

Because we model the failure state as absorbing, one (maximal) eigenvalue of T^s will be $\lambda_1 = 1$. We refer to the second largest real eigenvalue as λ_2. For metastable (rarely-falling) systems, λ_2 will be close to but smaller than 1.

After taking one or two non-falling steps, the robot converges very nearly to so called metastable distribution, ϕ. Taking a step afterwards, the robot will fall with $1-\lambda_2$ probability, otherwise its probabilistic distribution in state space will not change. As a result, the probability of taking n steps only, equivalently falling at the nth step is simply

$$Pr(y[n] = y_1,\ y[n-1] \neq y_1) = \lambda_2^{n-1}(1-\lambda_2). \tag{44}$$

For $\lambda_2 < 1$, as $n \to \infty$, the right hand side goes to zero, i.e., the system will eventually fail. When $n = 1$ is substituted, we get $1-\lambda_2$, which corresponds to "falling down" at the first step (taking 1 step only). Then, the average number of steps can be then calculated as

$$\begin{aligned} MFPT &= E[FPT] \\ &= \sum_{n=1}^{\infty} n\ Pr(y[n] = y_1,\ y[n-1] \neq y_1) \\ &= \sum_{n=1}^{\infty} n\lambda_2^{n-1}(1-\lambda_2) = \frac{1}{1-\lambda_2}, \end{aligned} \tag{45}$$

where we used the fact that $\lambda_2 < 1$.

For $\sigma_s = 1$, we present μ_s versus MFPT in Figure 3. The solid lines are obtained using the full-mesh, which captures the actual dynamics more accurately. Calculating T^s with a uniform-mesh gives the dashed lines. We see uniform-mesh is a good, but not perfect approximation of the full dynamics.

VII. SWITCHING CONTROL

In this section we will assume $\sigma_s = 1$ and $\sigma_l = 0.1$. To solve for optimal policies we will use value iteration [16]. We will iteratively calculate the value of y_i using

$$V(i) := \sum_{\tilde{s}\in S} \max_{\zeta} \left\{ \sum_{j} P_{ij}(\zeta,\tilde{s})\ (R(j) + \alpha\ V(j)) \right\}, \tag{46}$$

where $P_{ij}(\zeta,\tilde{s})$ is a probability we will explain soon, $R(j)$ is the reward for transitioning to y_j, and α is the discount factor, which is chosen to be 0.9.

Remember that the failure state is y_1. The value of the failure state will initially be zero, i.e.,

$$V(1) = 0, \tag{47}$$

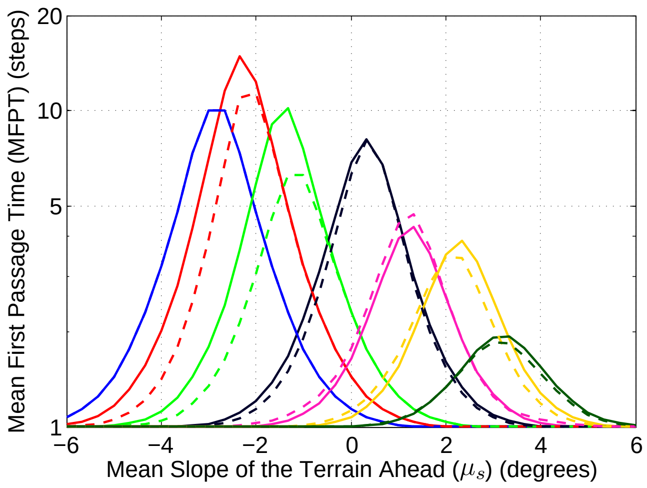


Fig. 3. Slopes ahead of the robot are assumed to be normally distributed with $\sigma_s = 1$. Figure shows average number of steps before falling calculated using (45) versus μ_s for two independently obtained meshes. Solid lines represent the full-mesh with 115,990 states. It is obtained using $d_{thr} = 0.1$ in Section IV-A. The dashed line is a result of using uniform-mesh with 29,701 states. It was obtained in Section IV-B. Each color (one solid and one dashed line) represents $\{\zeta_{-3},\ \zeta_{-2},\ \zeta_{-1},\ \zeta_0\ \zeta_1,\ \zeta_2,\ \zeta_3\}$ from left to right respectively.

and it will always stay as zero, because the reward for taking a successful step is one, while falling is zero.

$$R(j) = \begin{cases} 0, & j = 1 \\ 1, & \text{otherwise} \end{cases} \tag{48}$$

Note that the reward function we use does not depend on the controller, slope ahead, or current state. Use of more sophisticated reward functions (e.g., considering energy, speed, step width) is a topic of [17]. However, our focus is on stability in this paper. Using (47) and (48), we can rewrite (46) as

$$V(i) := \sum_{\tilde{s}\in S} \max_{\zeta} \left\{ \sum_{j\neq 1} P_{ij}(\zeta,\tilde{s})\ (1 + \alpha\ V(j)) \right\}. \tag{49}$$

The probability of 'thinking $\tilde{s}$ is the slope ahead' and 'transitioning from y_i to y_j when ζ is used' is then

$$P_{ij}(\zeta,\tilde{s}) = \sum_{s\in S} \sum_{l\in S} P_S(s)P_L(l)\ f_S(s,l,\tilde{s})\ T^d_{ij}(s,\zeta), \tag{50}$$

where

$$f_S(s,l,\tilde{s}) = \begin{cases} 1, & \tilde{s} = max(min(S), min(max(S), s+l)) \\ 0, & \text{otherwise.} \end{cases} \tag{51}$$

Using value iteration on uniform-mesh we obtain "optimal policy". To evaluate its performance, we will use the full-mesh, which we denote by Y_f. Then this policy corresponds to using

$$\zeta[n] = \pi(c(y_f[n], Y), \tilde{s}[n]), \tag{52}$$

where $y_f \in Y_f$ and c is the function defined in (34). The results are shown with dashed in Figure 4. We see that there is a great gain in switching.

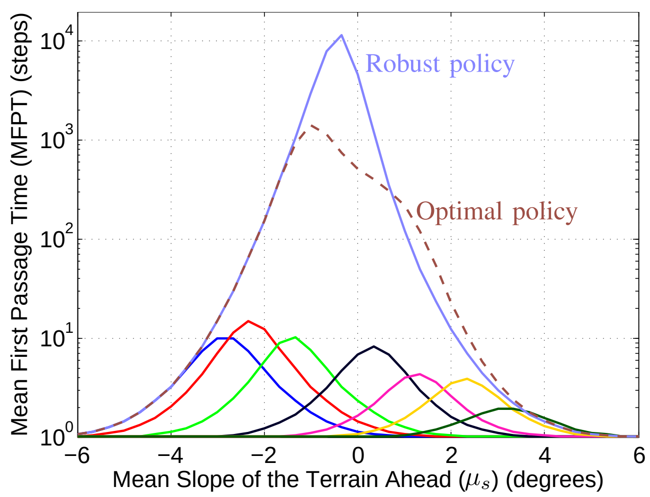


Fig. 4. Evaluation of policies obtained using uniform-mesh on full-mesh. Slopes ahead of the robot are assumed to be normally distributed with $\sigma_s = 1$. Figure shows average number of steps before falling calculated using (45) versus μ_s. Optimal policy is obtained using (49), whereas (53) will give the robust policy. Fixed controllers on the bottom are shown for reference.

Despite the success of the policy, we suspect we can do better based on the phenomenon introduced in [5]. We consider the following situation: While the actual state is y_i, the robot may think it is y_k. To make this clear, we rewrite the value iteration algorithm.

$$V(k) := \sum_{\tilde{s}\in S} \max_{\zeta} \left\{ \sum_{j\neq 1} P_{kj}(\zeta,\tilde{s})\ (1+\alpha\ V(j)) \right\} \quad (53)$$

Note that we only exchanged i with k, but this will make it easier to follow. The probability of 'thinking $\tilde{s}$ is the slope ahead', 'thinking the state is y_k', and 'transitioning to y_j when ζ is used' is

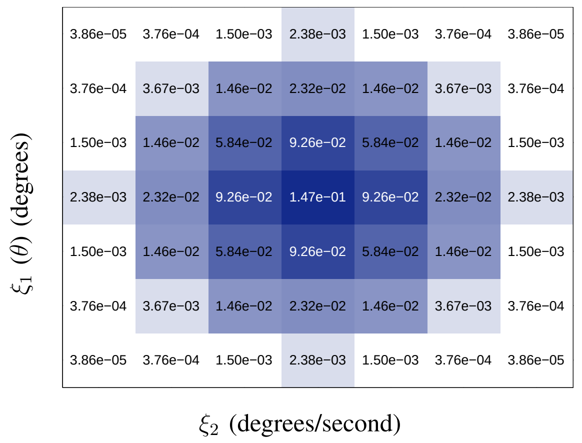
$$P_{kj}(\zeta,\tilde{s}) = \sum_{s\in S}\sum_{l\in S}\sum_{i} P_S(s)P_L(l)\ f_S(s,l,\tilde{s})\ P^P_{ik}\ T^d_{ij}(s,\zeta), \quad (54)$$

where P^P_{ik} is the probability of being at state y_i when robot thinks the state is y_k.

$$P^P_{ik} := Pr(y[n] = y_i \mid \tilde{y}[n] = y_k) \quad (55)$$

Determining P^P_{ik} for the full-mesh is not intuitive. We proposed an ad-hoc solution in [5]. However, for the uniform-mesh this will be very easy and visualizable as shown in Figure 5. This is in fact a table showing weights of the probability of being at neighbor states, when the robot thinks the states is the one in the center. So when the robot thinks the state is the one at the center, there is a 0.15 probability that it is actually there. To determine weights of the neighbor states, we simply used Multivariate normal distribution with mean $\mu = 0_2$ and covariance matrix $\Sigma = I_2$.

Using P^P_{ik} illustrated in Figure 5 for (53) gives the robust policy in Figure 4. Noting the logarithmic y-axis we see the increased robustness helps greatly. Overall, the robot takes 10 thousand steps on average where the fixed controllers can only take 10 steps!



3.86e−05	3.76e−04	1.50e−03	2.38e−03	1.50e−03	3.76e−04	3.86e−05
3.76e−04	3.67e−03	1.46e−02	2.32e−02	1.46e−02	3.67e−03	3.76e−04
1.50e−03	1.46e−02	5.84e−02	9.26e−02	5.84e−02	1.46e−02	1.50e−03
2.38e−03	2.32e−02	9.26e−02	1.47e−01	9.26e−02	2.32e−02	2.38e−03
1.50e−03	1.46e−02	5.84e−02	9.26e−02	5.84e−02	1.46e−02	1.50e−03
3.76e−04	3.67e−03	1.46e−02	2.32e−02	1.46e−02	3.67e−03	3.76e−04
3.86e−05	3.76e−04	1.50e−03	2.38e−03	1.50e−03	3.76e−04	3.86e−05

ξ_1 (θ) (degrees)
ξ_2 (degrees/second)

Fig. 5. Weight distribution for belief state. Figure shows how P^P_{ik} is determined. When the robot thinks the state is the one at the center, there is a 0.15 probability that it is actually there. With 0.09 probability, either ξ_1 and ξ_2 is neighbor to what it thinks.

VIII. CONCLUSIONS

In this paper we propose a new method for meshing, called uniform-meshing, to approximate the dynamics of a walking robot as a Markov chain. Compared to our previous method in Section IV-A, uniform-meshing is computationally very efficient. In addition, although not quite as accurate as our full-mesh approach, it is still a decent approximation. We also showed that ensuring robustness of the policies is trivial for a uniform-mesh. We anticipate future work to improve generation of a Markov chain based on the uniform mesh can improve this approximation even further.

We conclude by emphasizing that our approach here relies on two basic ideas. First, the HZD framework creates holonomic constraints, so that only the position and velocity of single "clock" variable (which in this work is a linear combination, $\theta = cq$, of the joint angles) are needed to define the full (10D) state of all joint positions and velocities, if the dynamics are constrained to the zero dynamics manifold (i.e., corresponding to zero error in the desired constraint equations). Second, the purpose of sliding-mode control is to drive errors in trajectories to zero in finite time, exactly with the aim of ensuring the dynamics are driven to the HZD manifold for the controller (ζ) used for a given step before the swing foot impacts with terrain. There is one important catch, which is that the basin of attraction for the HZD manifold is limited, so there is no guaranteed that trajectories will actually converge, rather than spinning off and failing horribly. This caveat in fact captures a fundamental, motivating goal of our methods both here and in previous work: we can computationally quantify the long-term performance (usually converging, but very rarely falling down), even when global guarantees of stability do not exist.

APPENDIX

There are various methods to design v in (9) to force h (and $\dot{h}$) to zero [9]. While even a PD controller would work, a Sliding Mode Control (SMC) is preferable, due to its finite time convergence [10], which we summarize here. Remember that h corresponds to tracking error. The generalized error is defined as

$$\sigma_i = \dot{h}_i + h_i/\tau_i, \quad i = \{1, 2, 3, 4\}, \tag{56}$$

where τ_is are time constants for h_i. Note that when the generalized error is driven to zero, i.e. $\sigma_i = 0$, we have

$$0 = \dot{h}_i + h_i/\tau_i. \tag{57}$$

The solution to this equation is given by

$$h_i(t) = h_i(t_0)\ exp(-(t - t_0)/\tau_i), \tag{58}$$

which drives h_i to zero exponentially fast and justifies the name time constant. It can be also seen as the ratio of proportional and derivative gains of a PD controller.

Finally, v in (9) is given by

$$v_i = -k_i|\sigma_i|^{2\alpha_i - 1} sign(\sigma_i), \quad i = \{1, 2, 3, 4\}, \tag{59}$$

where $k_i > 0$ and $0.5 < \alpha_i < 1$ are called convergence coefficient and convergence exponent respectively. k_i can be seen as the common gain. $0.5 < \alpha_i < 1$ ensures finite time convergence. Note that if we had $\alpha_i = 1$, this would be a PD controller. We used $\alpha_i = 0.7$, $\tau_i = 0.1$, $k_i = 10$ for all controllers in this paper, but h_d was different for each controller.

ACKNOWLEDGMENT

This work is supported by the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

[1] T. McGeer, "Passive dynamic walking," *The International Journal of Robotics Research*, vol. 9, pp. 62–82, Apr. 1990.

[2] Y. Hurmuzlu and D. Marghitu, "Rigid body collisions of planar kinematic chains with multiple contact points," *The International Journal of Robotics Research*, vol. 13, pp. 82–92, Feb. 1994.

[3] E. Westervelt, J. W. Grizzle, and D. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, pp. 42–56, Jan. 2003.

[4] K. Byl and R. Tedrake, "Metastable walking machines," *The International Journal of Robotics Research*, vol. 28, pp. 1040–1064, Aug. 2009.

[5] C. O. Saglam and K. Byl, "Robust policies via meshing for metastable rough terrain walking," in *Proceedings of Robotics: Science and Systems*, (Berkeley, USA), 2014.

[6] E. Westervelt, C. Chevallereau, B. Morris, J. Grizzle, and J. Ho Choi, *Feedback Control of Dynamic Bipedal Robot Locomotion*, vol. 26 of *Automation and Control Engineering*. CRC Press, June 2007.

[7] S. Yadukumar, M. Pasupuleti, and A. Ames, "Human-inspired underactuated bipedal robotic walking with AMBER on flat-ground, upslope and uneven terrain," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2478–2483, Oct. 2012.

[8] A. D. Ames, "First steps toward automatically generating bipedal robotic walking from human data," in *Robot Motion and Control 2011* (K. Kozowski, ed.), no. 422 in Lecture Notes in Control and Information Sciences, pp. 89–116, Springer London, Jan. 2012.

[9] A. Ames, K. Galloway, and J. Grizzle, "Control lyapunov functions and hybrid zero dynamics," in *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, pp. 6837–6842, Dec. 2012.

[10] A. Sabanovic and K. Ohnishi, "Motion control systems," John Wiley & Sons, 2011.

[11] C. O. Saglam, "Hybrid zero dynamics - file exchange - MATLAB central." Retrieved October 1, 2014.

[12] C. O. Saglam and K. Byl, "Switching policies for metastable walking," in *Proc. of IEEE Conference on Decision and Control (CDC)*, pp. 977–983, Dec. 2013.

[13] C. O. Saglam and K. Byl, "Stability and gait transition of the five-link biped on stochastically rough terrain using a discrete set of sliding mode controllers," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5675–5682, May 2013.

[14] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*, vol. 63. SIAM, 1992.

[15] C. O. Saglam and K. Byl, "Metastable markov chains," in *IEEE Conference on Decision and Control (CDC)*, Dec. 2014. accepted for publication.

[16] R. Bellman, "A markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[17] C. O. Saglam and K. Byl, "Quantifying the trade-offs between stability versus energy use for underactuated biped walking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014. accepted for publication.