Quantifying and Optimizing Stability of Bipedal Walking Gaits

Cenk Oguz Saglam and Katie Byl

Abstract— **In this paper, we examine the problem of quantifying and optimizing stability for two different low-level gait control schemes for point-foot bipedal walking. This work relies on recent methods for estimating the average number of steps, or Mean First Passage Time (MFPT), of a biped before falling down during rough-terrain walking. Our previous work in quantifying and optimizing the MFPT has always assumed a fixed set of low-level gait controllers exist; in this work, we finally address the more fundamental issue of tuning a particular gait controller itself. Our methods are applicable for a wide variety of low-level control schemes, with the goal of allowing researchers to optimize and compare various strategies, and we present results on each of two particular strategies, as case demonstrations. One scheme is the now-familiar Hybrid Zero Dynamics approach and the other is a method using piece-wise reference trajectories and sliding-mode control.**

I. INTRODUCTION

The fact that even passive bipeds introduced in [1] can achieve human-like locomotion has motivated work on "dynamic walking", and the study of point-foot walkers is in turn useful toward developing control addressing the underactuated nature of dynamic walking. The Hybrid Zero Dynamics (HZD) approach [2] can be used to guarantee stability on flat terrain and can simultaneously be designed to minimize energy consumption. However, just like passive walkers, this approach results in designs that are sensitive to perturbations, thus performing poorly on rough terrain. To increase robustness, we initially quantify it.

A basic motivation of dynamic walking is to achieve efficient, human-like walking. However, simultaneously achieving robustness to perturbations is challenging. Hybrid Zero Dynamics fundamentally assume that terrain is deterministic. One can design HZD trajectories to minimize energy use, and the gaits that result look human-like. However, on rough enough terrain robots are destined to eventually fall. The important question is "how frequently".

Our previous work has investigated tools to generate high-level (switching) control policies to maximize the number of steps, our Mean First Passage Time (MFPT) to failure, given a fixed set of low-level controllers. In this paper, we are finally able to do something we have had in mind for a long time: optimizing the parameters of a low-level controller itself to maximize MFPT. The reason we were not able to do this was because calculation of MFPT used to take an impractical time to compute. However, our work over the years now allows us to estimate this number very quickly, in turn allowing us to optimize and bench-mark the resulting performance of various low-level controllers.

There are a wide range of strategies various researchers have developed to achieve stable walking gaits on flat terrain. Published data on competing methods emphasize particular strengths or weaknesses of a given approach but do not easily facilitate intuitive bench-marking of performance, particularly when comparisons of reliability and robustness on variable terrain are desired. In previous work, we have examined the high-level problem of developing optimal switching control for a set of fixes low-level controllers. Here, we instead address the issue of tuning a particular low-level control itself. We demonstrate that optimization of MFPT improves the reliability of each method dramatically and that optimization of energetics (i.e., cost of transport) results in worse reliability. Additionally, we illustrate that our in-house control strategy outperforms the HZD approach dramatically on rough terrain, emphasizing the importance of apples-to-apples performance analyses. Our primary motivation in this work is to provide practical computation tools to achieve head-to-head bench-marking of any of a wide range of particular low-level control strategies researchers may wish to investigate.

The rest of this paper is organized as follows. The 5-link planar walker model we study is presented in Section II. We then present each of two low-level control schemes in Section III: (1) sliding mode control with time-invariant piece-wise constant joint references and (2) a hybrid zero dynamics strategy, with trajectories based on Bézier polynomials and joint-tracking via PD control. Tools for meshing the dynamics and generating a Markov chain are presented in Section IV and V, respectively. Section VI gives results, including both performance bench-marks, using the MFPT metric, and also tables documenting the optimal control parameters found using our algorithm. The latter will be of particular use to anyone wishing to repeat and build upon our methods. Finally, Section VII gives conclusions and discusses future work.

II. MODEL

Figure 1 shows our walker model, namely a planar 5-link biped with point feet and rigid links. It is underactuated by 1 DOF because there is no ankle torque. The ten dimensional state of the robot is given by $x := [q^T \ \dot{q}^T]^T$, where $q := [q_1 \ \ q_2 \ \ q_3 \ \ q_4 \ \ q_5]^T$ is the vector of angles shown in the figure.

When only one of the legs is in contact with the ground, the robot is in the single support phase, which has continuous dynamics. Using the Lagrangian approach, the dynamics can

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

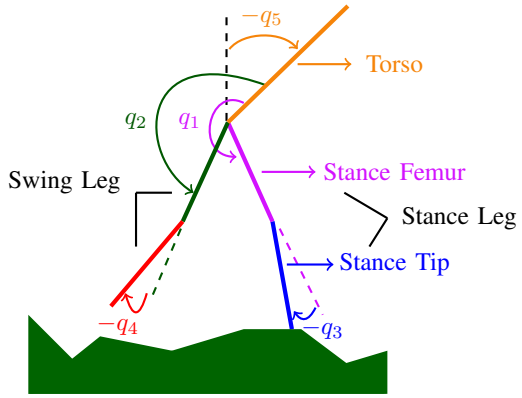

Fig. 1. Illustration of the five-link robot with identical legs

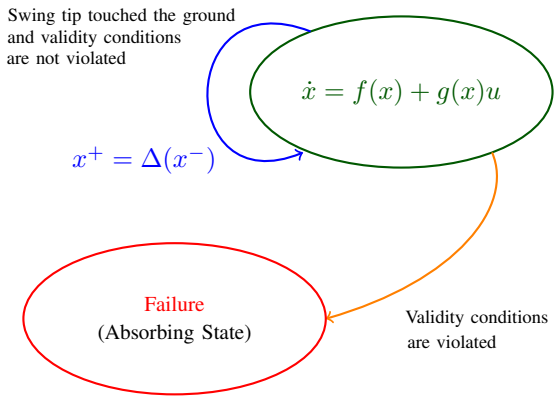

Fig. 2. Hybrid model of a step and the failure state

be derived as

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

where u is the input. We can equivalently express these dynamics as:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ D^{-1}(-C\dot{q} - G + Bu) \end{bmatrix} =: f(x) + g(x)u. \tag{2}$$

On the other hand, if both legs are contacting the ground, then the robot is in its double support phase, which can be well approximated as an impact map given by

$$x^+ = \Delta(x^-), \tag{3}$$

where x^- and x^+ are the states just before and after the impact respectively. Conservation of energy and the principle of virtual work give the mapping Δ [3], [4].

A step consists of a single support phase and an impact event. Since walking consists of steps in sequence, it has hybrid dynamics as illustrated in Figure 2. For a step to be successful, certain "validity conditions" must be satisfied, which we list next. After impact, the former stance leg must lift from ground with no further interaction with the ground until the next impact. Also, the swing foot must have progressed past the stance foot before the impact of the next step occurs. Only the feet should contact the ground. Furthermore, the force on stance tip during the swing phase and the force on the swing tip at impact should satisfy the following no-slip constraint:

$$F_{friction} = F_{normal}\ \mu_s > |F_{transversal}|. \tag{4}$$

If validity conditions are not met, the step is unsuccessful and the system is modeled as transitioning to an absorbing failure state. This is a conservative model because in reality violating these conditions does not necessarily mean failure.

III. CONTROL SCHEME

Our methods in the next section for quantifying and optimizing controllers are applicable for different low-level controller schemes. In this paper, we study two schemes.

1. Sliding Mode Control (SMC) with Time-Invariant Piece-Wise Constant References

For faster optimization time, it is preferable to have fewer parameters to optimize. This section explains the controller scheme used in [5]. It involves piece-wise constant references and adopts a Sliding Mode Control (SMC) scheme for finite time convergence, as explained in [6] and applied in [7] .

Although the walker has five angles, since there are only four actuators, we need to choose four variables to control. As a result of our experience [8], we proceed with

$$q_c := [\theta_2 \quad q_3 \quad q_4 \quad q_5]^T, \tag{5}$$

where $\theta_2 := q_2 + q_5$ is an absolute angle and q_c stands for angles to be controlled. We select the input to the system in the following form:

$$u = (ED^{-1}B)^{-1}(v + ED^{-1}(C\dot{q} + G)),$$
$$where\ E = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

Substituting this u into (1) and noting $q_c = E\,q$, we obtain a simple equation for the second derivatives of the angles to be controlled:

$$\ddot{q}_c = v \tag{7}$$

We will then design v such that q_c acts as desired (q_c^{ref}). The error is given by

$$e = q_c - q_c^{ref}, \tag{8}$$

and the generalized error is defined as

$$\sigma_i = \dot{e}_i + e_i/\tau_i, \quad i = \{1,2,3,4\}, \tag{9}$$

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

$$0 = \dot{e}_i + e_i/\tau_i. \tag{10}$$

The solution to this equation is given by

$$e_i(t) = e_i(t_0)\ exp(-(t - t_0)/\tau_i), \tag{11}$$

which drives q_c to q_c^{ref} exponentially fast and justifies the name time constant for τ_i, which can be also seen as the ratio of proportional and derivative gains of a PD controller. Then, v in (7) is chosen to be

$$v_i = -k_i|\sigma_i|^{2\alpha_i - 1} sign(\sigma_i), \quad i = \{1, 2, 3, 4\}, \tag{12}$$

where $k_i > 0$ and $0.5 < \alpha_i < 1$ are called the convergence coefficient and convergence exponent, respectively. k_i is analogous to a gain common to both e_i and $\dot{e}_i$. $0.5 < \alpha_i < 1$ ensures finite time convergence. Note that if we had $\alpha_i = 1$, this would simply be a standard PD controller. SMC has $4 \times 3 = 12$ controller parameters to optimize.

For simplicity, we use references in the following form

$$q_c^{ref} = \begin{cases} [\theta_2^{ref1} \;\; q_3^{ref} \;\; q_4^{ref1} \;\; q_5^{ref}]^T, & \text{condition}, \\ [\theta_2^{ref2} \;\; q_3^{ref} \;\; q_4^{ref2} \;\; q_5^{ref}]^T, & \text{otherwise}, \end{cases} \tag{13}$$

where we selected "condition", above, to be $\theta_1 := q_1 + q_5 < \pi$. Note that the references are piecewise constant and time-invariant. What makes this reference structure appealing is the fact that there are only 6 parameters to optimize. So, in total, we will be optimizing $12 + 6 = 18$ parameters.

2. Hybrid Zero Dynamics (HZD) using Proportional-Derivative Control and Bézier Polynomials

As a second approach, we will be using the well-known Hybrid Zero Dynamics (HZD) control [2].

To begin with, we need to choose a phase-variable, θ, which will increase monotonically during a step, and four independent variables to control, h_0. As in [3], we use $\theta := cq$, where $c := [-1 \;\; 0 \;\; -1/2 \;\; 0 \;\; -1]$, and control the relative (internal) angles, i.e., $h_0 := [q_1 \;\; q_2 \;\; q_3 \;\; q_4]^T$. Then h_0 is in the form of $h_0 = H_0 q$, where $H_0 = [I_4 \;\; 0]$.

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

$$h(q) := h_0(q) - h_d(\theta) = H_0 q - h_d(cq) \tag{14}$$

Taking the first derivative with respect to time we get

$$\dot{h} = \frac{\partial h}{\partial q}\dot{q}, \tag{15}$$

which can be equivalently written as

$$\dot{h} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x} f(x) =: \mathcal{L}_f h, \tag{16}$$

where we used the fact that $\frac{\partial h}{\partial x} g(x) = 0$. Then, we have

$$\ddot{h} = \mathcal{L}_f^2 h + \mathcal{L}_g \mathcal{L}_f h \; u. \tag{17}$$

Substituting the controller structure

$$u(x) = (\mathcal{L}_g \mathcal{L}_f h)^{-1}(-\mathcal{L}_f^2 h + v) \tag{18}$$

to (17) yields

$$\ddot{h} = v. \tag{19}$$

There are various methods to design v to force h (and $\dot{h}$) to zero [9]. We will use PD control for simplicity. Then, we choose

$$v = -K_P y - K_D \dot{y}, \tag{20}$$

where K_P, and K_D are the proportional and derivative gains respectively.

As suggested in the original paper [2], we use Bézier polynomials to form the reference (h_d). First, we scale and shift θ to have an internal clock which ticks from 0 to 1 during limit cycle walking on flat terrain.

$$\tau(q) := \frac{\theta(q) - \theta^+}{\theta^- - \theta^+} \tag{21}$$

The Bézier curves are given by

$$b_i(\tau) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{22}$$

which form the reference trajectory:

$$h_d(\theta) := \begin{bmatrix} b_1(\tau) \\ b_2(\tau) \\ b_3(\tau) \\ b_4(\tau) \end{bmatrix}. \tag{23}$$

Choosing $M = 6$ yields $(6 + 1) \times 4 = 28$ α_k^i parameters to optimize. However, for hybrid invariance, $h = \dot{h} = 0$ just before an impact on flat terrain should imply $h = \dot{h} = 0$ after the impact. This constraint will eliminate $2 \times 4 = 8$ of the parameters as explained in [2]. In total, we will need to optimize 20+2=22 parameters, including the PD controller gains.

IV. MESHING

We will assume the terrain profile is angular. Each step will experience a constant slope, noted by s. The slope ahead will only change at impacts. 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, we define a Poincaré section at "just before the impact". For the rest of the paper, we will refer to states on this section simply as x. In this setting, 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 ζ, i.e.

$$x[n+1] = h^t(x[n], s[n], \zeta). \tag{24}$$

Next, we will mesh to obtain a Markov Decision Process model of the system. First, we define the slope set S as

$$S = \left\{ k^\circ \;\; : \;\; \frac{k}{d_s} \in \mathbb{Z}, \; -20 \leq k \leq 20 \right\}, \tag{25}$$

where d_s is a parameter determining the slope set density. The range of k is selected to be conservatively wide.

There are two key goals in meshing. First, we want to have a set of states, Y, which well covers the (reachable) part of state space the robot can visit. Secondly, we want to learn what $h^t(y, s, \zeta)$ is for all $y \in Y$ and $s \in S$. Next, an initial mesh, Y_i, should be chosen. In this study, we use an initial mesh consisting of only two points. One of these points (y_1) represents all (conservatively defined) failure states, no matter how the robot failed, e.g. a foot slipped, or the torso touched the ground. The other point (y_2) should be in the

controllable subspace. In other words, it should be in the basin of attraction of controller ζ.

Then, our algorithm explores the reachable space deterministically. We initially start by setting $\overline{Y} = \{y \in Y \mid y \neq y_1\}$, 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 $h^t(y[n], s[n], \zeta)$ and remove y from $\overline{Y}$. For the points newly found, check their distance to other states in Y. If the minimum such distance exceeds some threshold, a new point is then added to Y and $\overline{Y}$.

A crucial question is how should the (threshold) distance metric be so that the resulting Y has small number of states while accurately covering the reachable state space? We found using standardized (normalized) Euclidean distance to be extremely useful. When a is a vector, and B is a set of vectors (growing in size, during meshing) each with the same dimension as a, the distance of a from B is calculated as

$$d(a, B) := \min_{b \in B} \left\{ \sqrt{\sum_i \left(\frac{a_i - b_i}{r_i} \right)^2} \right\}, \tag{26}$$

where r_i is the standard deviation of b_i elements. In addition, the closest point in B to a is given by

$$c(a, B) := \operatorname*{argmin}_{b \in B} \left\{ \sum_i \left(\frac{a_i - b_i}{r_i} \right)^2 \right\}. \tag{27}$$

We are now ready to present the pseudocode in Algorithm 1. Two important tricks to make the algorithm run faster are as follows. First, the slope set allows a natural cluster analysis. We can classify states using the inter-leg angle they possess. So, the distance comparison for a new point can be made only with the points that are associated with the same slope. This might result in more points in the final mesh, but it speeds up the meshing and later calculations significantly. Secondly, consider a state y. We can simulate $h^t(y, -20^\circ, \zeta)$ just once and then extract $h^t(y, s, \zeta)$ for all $s \in S$. The reason is, for example, in order robot to experience an impact at -20 degree, it has to pass through all the possible impact points for less extreme slopes in the set.

We estimate the intrinsic dimension of the reachable space to be around 2 [10]. As a result, the curse of dimensionality is avoided and meshing can be done with a relatively small number of iteration steps.

Algorithm 1 Meshing algorithm

Input: Controller, Initial set of states Y_i, Slope set S and threshold distance d_{thr}
Output: Final set of states Y, and state-transition map

$\overline{Y} \leftarrow Y_i$ (except y_1)
$Y \leftarrow Y_i$
while $\overline{Y}$ is non-empty **do**
  $\overline{Y}_2 \leftarrow \overline{Y}$
  empty $\overline{Y}$
  for each state $\overline{y} \in \overline{Y}_2$ **do**
    for each slope $s \in S$ **do**
      Simulate a single step to get the final state x, when initial state is $\overline{y}$, slope ahead is s, and controller ζ is used. Store this information in the state-transition map
      if robot did not fall and $d(x, Y) > d_{thr}$ **then**
        add x to $\overline{Y}$
        add x to Y
      end if
    end for
  end for
end while
return Y, state-transition map

V. METASTABLE MARKOV CHAINS

A. Obtaining Markov Chain

Now, thanks to the meshing (state-transition map) approximation, we know $h^t(y, s, \zeta)$ for all $y \in Y$, and $s \in S$. We define

$$h^a(x[n], \gamma[n], \zeta) := c(h^t(x[n], \gamma[n], \zeta), Y), \tag{28}$$

where (27) is used and the superscript a stands for approximation. Then we write the approximate step-to-step dynamics:

$$y[n+1] = h^a(y[n], s[n], \zeta) \tag{29}$$

After that, the deterministic state transition matrix can be written as

$$T^d_{ij}(s, \zeta) = \begin{cases} 1, & \text{if } y_j = h^a(y_i, s, \zeta) \\ 0, & otherwise. \end{cases} \tag{30}$$

Note that (30) is the result of our basic, nearest-neighbor approximation, which appears to work well. More sophisticated approximations result with the matrix not just having one or zero elements, but also fractional values in between. This increases the memory and computational costs while, to our experience, not providing much accuracy increase.

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

$$T^s_{ij} := Pr(y[n+1] = y_j \mid y[n] = y_i). \tag{31}$$

To calculate this matrix, the first thing we need to do is assume a distribution over slope set, noted by P_S.

$$P_S(s) = Pr(s[n] = s) \tag{32}$$

In this paper, we will assume a normal distribution for P_S, with mean μ_s, and standard deviation σ_s.

$$s[n] \sim \mathcal{N}(\mu_s, \sigma_s^2) \tag{33}$$

After distributing s values, T^s can be calculated as

$$T^s(\zeta) = \sum_{s \in S} P_S(s)\ T^d(s, \zeta) \tag{34}$$

As we make d_{thr} and d_s smaller, we have more accurate representations of the full dynamics at the expense of higher numbers of states in the final mesh.

B. Metastability

We study bipedal walking as a metastable system [11]. In the presence of noise (e.g., rough terrain), the robot will eventually fall, but the number of steps before doing so might be very high (millions). This section serves as a summary on how we estimate this number. We invite interested reader to [12].

The eigenvalues of T^s cannot have magnitude larger than one. However, the largest eigenvalue will be 1 because we model failure as absorbing. Let λ_2 be the second largest eigenvalue. λ_2 will be non-negative and real. For metastable (rarely-falling) walking, it will be close to but smaller than 1.

If the robot does not fall within several steps, then it converges to so called metastable distribution, ϕ. The fascinating fact is what happens when the robot takes a single step after that. With $1-\lambda_2$ probability the robot is going to fall, otherwise nothing (the distribution) will not change. Then, the probability of taking n steps only, equivalently falling at the nth step is simply

$$Pr(y[n] = y_1,\ y[n-1] \neq y_1) = \lambda_2^{n-1}(1-\lambda_2). \tag{35}$$

For $\lambda_2 < 1$, realize that as $n \to \infty$, the right hand side goes to zero, i.e., the system will eventually fail. Note that we also count the step which ended up falling as a step. An intuitive check is to consider "falling down" at the first step (taking 1 step only). When $n = 1$ is substituted, we get $1-\lambda_2$ as expected. Then, the average number of steps can be then calculated as

$$\begin{aligned} MFPT &= E[FPT] \\ &= \sum_{n=1}^{\infty} n\ Pr(y[n] = y_1,\ y[n-1] \neq y_1) \\ &= \sum_{n=1}^{\infty} n\lambda_2^{n-1}(1-\lambda_2) = \frac{1}{1-\lambda_2}, \end{aligned} \tag{36}$$

where we used the fact that $\lambda_2 < 1$. As a result, MFPT can then be calculated using

$$M = \begin{cases} \infty & \lambda_2 = 1 \\ \frac{1}{1-\lambda_2} & \lambda_2 < 1. \end{cases} \tag{37}$$

Note that being stable corresponds to $\lambda_2 = 1$, but we will introduce enough roughness so that we always have $\lambda_2 < 1$. This will be achieved with a wide-enough slope set and high enough σ_s.

VI. RESULTS

Unless stated otherwise, we will be using "minimize" function from [13] in MATLAB to optimize. In this paper we will optimize for $\mu_s = 0$. However, σ_s choice for optimization depends on the controller. If it is too small, then the MFPT will turn out to be very large, which may not be calculated due to numeric software capabilities. Using MATLAB, we were able to calculate values up to around 10^{14} reliably. On the other hand, σ_s should not be too large, otherwise it may not be as easy to differentiate different controller's performance. Also, MFPT is more 'valid' when it is higher. Appropriate range for σ_s can be easily decided by calculating MFPT for different values with a single mesh. Once we decide on σ_s, we pick d_s. $d_s = \sigma_s/2$ is the rule of thumb we applied in this paper. Just like d_s, d_{thr} can be made smaller for higher accuracy in the expense of higher computation time. Iteration time varies greatly. If the MFPT turns out to be very small, it takes less than a second to compute. However, the computation time is not upper bounded, it may be longer and longer as we decrease d_s and d_{thr}. In this paper, we adjusted them such that it does not take more than several minutes in a single PC and used a computer cluster with 64 cores so that optimization is very fast. However, d_s and d_{thr} must be small enough for a decent amount of accuracy. This can be checked after the optimization by using smaller d_s and d_{thr} values and confirming MFPT estimation does not change much.

1. Sliding Mode Control (SMC) with Time-Invariant Piece-Wise Constant References

We start optimizing with the hand-tuned controller from [8], which we will refer to with ζ^1_{Base}. We first optimize for Cost of Transport (COT) of the limit cycle gait on flat terrain to obtain ζ^1_{COT}. We then optimize for MFPT with $\sigma_s = 2$, $d_s = 1$ and $d_{thr} = 1$. This results with controller ζ^1_{MFPT}. The parameters for each controller is given in Table I.

TABLE I
PARAMETERS FOR THE FIRST CONTROLLER SCHEME

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
θ_2^{ref1}	225°	190.5977°	224.9460°
θ_2^{ref2}	204°	200.92392°	203.7358°
q_3^{ref}	0°	-0.0008°	-0.0169°
q_4^{ref1}	-60°	-19.6094°	-60.0042°
q_4^{ref2}	-21°	-13.4718°	-24.0150°
θ_5^{ref}	0°	-0.0003°	0.0040°
k_1	50	49.1126	40.3791
k_2	100	84.2092	96.4343
k_3	75	83.1357	77.1343
k_4	10	7.5848	15.7245
α_1	0.7	0.7977	0.7003
α_2	0.7	0.6063	0.6954
α_3	0.7	0.6838	0.6991
α_4	0.7	0.4873	0.7001
τ_1	0.1	0.1354	0.0920
τ_2	0.1	0.0997	0.0905
τ_3	0.05	0.0679	0.0632
τ_4	0.2	0.1690	0.1918

Figure 3 compares the stability of each controller versus the roughness of the terrain. Noting the logarithmic y-axis, we immediately notice the huge improvement in stability by optimizing with the suggested method.

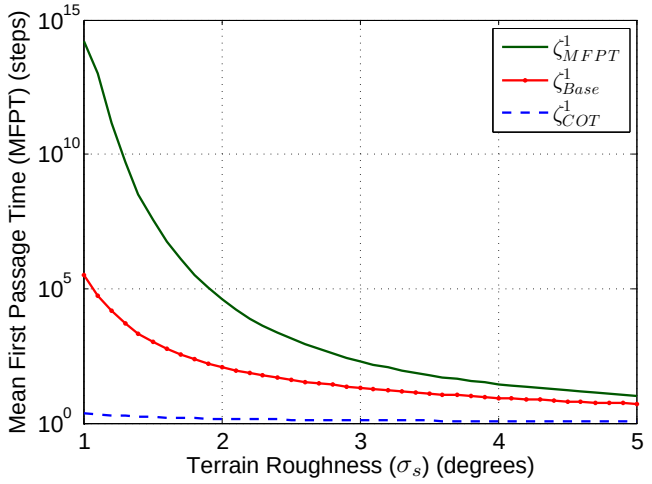

Fig. 3. Average number of steps before falling calculated using (36) versus σ_s for the first controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_s = 0$.

We note that Monte Carlo simulations are not a computationally practical means of verifying MFPT when it is very high, which has motivated our methodology throughout. However, we present a Monte Carlo study in Table II for $\sigma_s = 5$, where MFPT is small. To obtain the second row in this table, we simulated 10 thousand times. To allow the robot to "forget" the initial condition, we omit the first step, i.e., we only consider cases where it took more than a single step and do not count that first step.

TABLE II

ESTIMATION OF MFPT FOR FIRST CONTROLLER SCHEME WITH $\mu_s = 0$ AND $\sigma_s = 5$

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
Estimation using (36)	5.1766	1.1470	10.6433
Monte Carlo Simulation	5.0738	1.5716	10.4813

TABLE III

PARAMETERS FOR THE SECOND CONTROLLER SCHEME IN RADIANS

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
K_P	100	100	169.2681
K_D	10	10	30.0166
α^1_0	3.6151	3.6151	3.6037
α^1_1	3.6413	3.6475	3.5957
α^1_2	3.3894	3.4675	3.3948
α^1_3	3.2884	3.2884	3.2914
α^1_4	3.1135	3.1135	3.1136
α^1_5	3.1708	3.1708	3.1701
α^1_6	3.0349	3.0349	3.0448
α^2_0	3.0349	3.0349	3.0448
α^2_1	2.9006	2.9081	2.9259
α^2_2	2.9544	3.4544	3.0162
α^2_3	3.5470	3.0939	3.5302
α^2_4	3.5186	3.5186	3.5255
α^2_5	3.6851	3.6929	3.7298
α^2_6	3.6151	3.6151	3.6037
α^3_0	-0.4162	-0.3693	-0.4113
α^3_1	-0.6657	-0.6079	-0.6018
α^3_2	-0.3732	0.0124	-0.3126
α^3_3	-0.3728	-0.6501	-0.3444
α^3_4	-0.2359	-0.1880	-0.2366
α^3_5	-0.3780	-0.3819	-0.3478
α^3_6	-0.3200	-0.3141	-0.3221
α^4_0	-0.3200	-0.3141	-0.3221
α^4_1	-0.2484	-0.2285	-0.2856
α^4_2	-0.3690	-0.7323	-0.3664
α^4_3	-1.1041	-0.1932	-1.1005
α^4_4	-0.3973	-0.3817	-0.3834
α^4_5	-0.4260	-0.5139	-0.5082
α^4_6	-0.4162	-0.3693	-0.4113

2. Hybrid Zero Dynamics (HZD) using Proportional-Derivative Control and Bézier Polynomials

For the HZD scheme, the base controller, ζ^2_{Base}, is obtained by fixing speed to be $0.8m/s$ and minimizing energy usage as in [2]. We then remove the speed constraint and optimize for Cost of Transport (COT) to obtain ζ^2_{COT}. Both of these controllers assume flat terrain, i.e., $\sigma_s = 0$. However, [2] shows how to obtain the trajectories only, but not the controller gains. So, we just picked $K_P = 100$ and $K_D = 10$, which works on flat terrain. To obtain ζ^2_{MFPT}, we used the "patternsearch" algorithm in MATLAB to optimize for MFPT with $\sigma_s = 1$, $d_s = 0.5$ and $d_{thr} = 0.3$. Table III lists the parameters for each controller.

We compare the stability of each controller versus the roughness of the terrain in Figure 4. Again noting the logarithmic y-axis, the suggested method provides a dramatic increase in the stability, just like in Figure 3. As an interesting side note, ζ^2_{MFPT} has a lower COT than ζ^2_{Base}. This is possible, because there is no speed constraint for ζ^2_{MFPT}.

Table IV presents the Monte Carlo study obtained assuming $\sigma_s = 2$. Just like in Table II, we omit the first step to allow robot "forget" the initial condition.

3. Comparison

We first note that all six controllers are stable on flat ground ($\sigma_s = 0$), because they all exhibit limit cycle. However, as Table V shows, there is a huge difference between ζ^1_{MFPT} and any of the HZD controllers. So, we conclude that the first controller scheme is much more capable in terms of stability.

On the other hand, the first controller scheme has discontinuities in the references, which may or may not become a problem in a real experiment. Furthermore, the trajectory highly depends on the controller parameters, which we

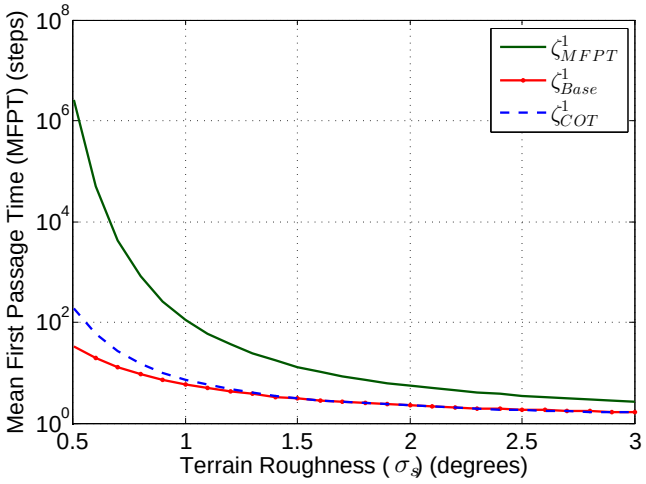

Fig. 4. Average number of steps before falling calculated using (36) versus σ_s for the second controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_s = 0$. Note that range of σ_s is different from Figure 3.

TABLE IV
ESTIMATION OF MFPT FOR SECOND CONTROLLER SCHEME WITH $\mu_s = 0$ AND $\sigma_s = 2$

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
Estimation using (36)	2.2085	2.2049	5.5206
Monte Carlo Simulation	2.1511	2.2487	6.1290

suspect makes the optimization even more non-convex. This might be the reason why many parameters of ζ^1_{Base} and ζ^1_{MFPT} in Table I are very close. We suspect that we only find local minimums. Indeed, starting with different initial conditions yields different final gaits.

A major problem in the second controller scheme, we believe, is the fact that reference is designed only for flat terrain. For example, the controller does not really know what to do after an impact does not occur at zero slope. This is because Bézier polynomials are designed for $0 \leq \tau(q) \leq 1$, and they quickly deviate outside this range. As a result, ζ^2_{Base} cannot take more than several steps on inclined terrain with a slope of -1 degrees. We discovered an easy fix to the problem by adopting the following policy: If $\tau(q) > 0.95$, then do not apply any torque. With this update, the controller can still walk on flat terrain. In addition, it seems to be stable on -9 degree sloped terrain! However, we did not present the result with this policy because it ends up with a low MFPT for $\mu_s = 0$. The reason is, it works very badly on uphill slopes.

TABLE V
COMPARISON OF CONTROLLER SCHEMES FOR $\mu_s = 0$ AND $\sigma_s = 1$

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
MFPT	3.2×10^5	2.2	1.6×10^{14}	5.9	7.3	113.1

This shows the need for a better reference parametrization.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we present a methodology for optimizing a low-level control scheme and of bench-marking resulting performance on rough terrain using the MFPT metric for reliability. We apply the approach to two particular control schemes as a motivating example; however, the approach is designed as a means of providing a systematic means of optimizing and bench-marking any of a wide variety of control strategies, not only for walking systems but also for other dynamic systems subject to stochastic environments, more generally.

As mentioned in the previous section, we ended up with a local minimum for the first controller scheme. We aim finding the global solution in a future study. However, our main intention is combining the two schemes by using a more capable and continuous reference parameterization, e.g., B-splines.

To build on this paper, we can also optimize under constraints, e.g., for desired speed, step width, or ground clearance. Furthermore, by designing multiple controllers for different mean slopes, we can increase the stability dramatically as illustrated in [5]. Finally, we may use costs that incorporate other performance metrics also, similar to [14]. For example, goal can be increasing stability while decreasing energy consumption.

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] 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.
[3] 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.
[4] 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.
[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] A. Sabanovic and K. Ohnishi, "Motion control systems," John Wiley & Sons, 2011.
[7] S. G. Tzafestas, T. E. Krikochoritis, and C. S. Tzafestas, "Robust sliding-mode control of nine-link biped robot walking," *Journal of Intelligent and Robotic Systems*, vol. 20, no. 2-4, pp. 375–402, 1997.
[8] 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.
[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] 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.

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

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

[13] R. Oldenhuis, "minimize - file exchange - MATLAB central." Retrieved October 1, 2014.

[14] 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.