Quantifying and Optimizing Robustness of Bipedal Walking Gaits on Rough Terrain

Cenk Oguz Saglam and Katie Byl

Electrical and Computer Engineering Department,
University of California, Santa Barbara, CA 93106 USA
{saglam,katiebyl}@ece.ucsb.edu
http://robotics.ece.ucsb.edu/

Abstract. Legged robots need "good" disturbance rejection to operate reliably in real-world environments, and achieving this goal arguably requires quantifying robustness. In this work, we consider a point-foot biped on variable-height terrain and measure robustness by the expected number of steps before failure. Unlike our previous work, in which we always assumed a fixed set of low-level gait controllers exist and focused on high-level control design, in this work we finally use quantification of robustness to benchmark and optimize a given (low-level) controller itself. Specifically, we study two particular control 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 with a sliding mode control. This work provides a methodology for optimization of a broad variety of parameterizable gait control strategies and illustrates dramatic increases in robustness due to both gait optimization and choice of control strategy.

Keywords: bipedal locomotion, rough terrain, stability quantification and optimization, metastability, mean first-passage time

1 Introduction

Quantifying robustness of legged locomotion is essential toward developing more capable robots with legs. In this work, we study underactuated biped walking models. For such systems, various sources of disturbance can be introduced for robustness analysis. While keeping the methods generic, this paper will focus on two-legged locomotion and study *stability on rough terrain*, or equivalently, *robustness to terrain disturbance*.

An intuitive and capacious approach is to use two levels for controlling bipedal locomotion. Fixed low-level controllers are blind to environmental information, such as the terrain estimation. Given environment and state information, the high-level control problem defines a policy to choose the right low-level controller at each step. Our previous work has always assumed a fixed set of low-level gait controllers exist and focused on the high-level control design [1]; in this work, we finally address the more fundamental issue of tuning a particular gait (low-level controller) itself.

For optimization of the low-level controller for stability, quantification is a critical step. As in many zero moment point approaches, stability is often conservatively defined as a binary metric to avoid the rotation of the stance foot and ensure not-falling [2]. However, robust, dynamic, fast, agile, and energy efficient human walking exploits under-actuation by foot rolling. For robots that are designed with this inspiration, such as point-feet walkers, walking motion is often analyzed in terms of limit cycles. Then, the local stability of the limit cycle can be studied by investigating deviations from the trajectories (gait sensitivity norm [3], H_∞ cost [4], and L2 gain [5]), or the speed of convergence back after such deviations (using Floquet theory [6, 7]). The L2 gain calculation in [5] was successfully extended and implemented on a real robot in [8]. Alternatively, largest terrain disturbance was maximized in [9] and trajectories were optimized to replicate human-walking data in [10].

Another approach to robustness quantification begins by stochastic modeling of the disturbances and (conservatively) defining what a failure is, e.g., slippage, scuffing, stance foot rotation, or a combination of such events. After discretizing the disturbance and state sets by meshing, step-to-step dynamics are studied to treat the system as a Markov Chain. Then, the likelihood of failure can be easily quantified by calculating the expected number of steps before falling, or mean first-passage time (MFPT) [11]. Optimizing the low-level controller for MFPT used to be impractical due to high computation time of MFPT for a given controller. However, our work over the years now allows us to estimate this number very quickly, and in turn, various low-level controllers can be optimized and benchmarked.

The rest of this paper is organized as follows. The walker model we study and the terrain model we employ are presented in Section 2. We then present two low-level control schemes in Section 3: (1) A hybrid zero dynamics strategy, with trajectories based on Bézier polynomials and joint-tracking via PD control as suggested in [12], and (2) sliding mode control with time-invariant piece-wise constant joint references adopted in [1]. Section 4 shows the discretization of the dynamics. Tools for generating and working on a Markov chain are presented in Section 5. Section 6 gives results, including both performance benchmarks, 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 7 gives conclusions and discusses future work.

2 Model

2.1 The Biped

The planar 5-link biped with point feet and rigid links illustrated in Figure 1 is adopted as the walker model in this paper. The ankles have no torque, so the model is underactuated. The ten dimensional state of the robot is given by $x := [q\ ;\ \dot{q}]$, where $q := [q_1\ \ q_2\ \ q_3\ \ q_4\ \ q_5]^T$ is the vector of angles shown in the figure.

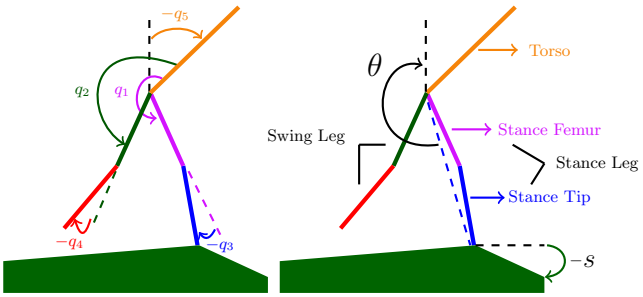

Fig. 1. Illustration of the five-link robot with identical legs. As will be explained, θ is called the phase variable.

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 be derived as

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

where u is the input. Equation (1) can be equivalently expressed as

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -D^{-1}(C\dot{q} + G) \end{bmatrix} + \begin{bmatrix} 0 \\ D^{-1}B \end{bmatrix} u =: f(x) + g(x)u. \tag{2}$$

On the other hand, if both legs are contacting the ground, then the robot is in its double support phase, which can be 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 [13], [14].

A step consists of a single support phase and an impact event. Since walking consists of steps in sequence, it has hybrid dynamics. For a step to be successful, certain "validity conditions" must be satisfied, which are listed 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. 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 no-slip constraint given by

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

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.

2.2 The Terrain

In this paper we assume the terrain ahead of the robot is a constant slope until an impact. 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.

An alternative and more common choice is modeling the rough terrain with varying heights like stairs. Both models of the rough terrain are equally complex, valid, and important for this paper's purpose and combining the two is a topic of future work.

3 Control Scheme

This section summarizes two low-level controller strategies that will be used to demonstrate the applicability of our method.

1. Hybrid Zero Dynamics using Proportional-Derivative Control and Bézier Polynomials

The hybrid zero dynamics (HZD) controller framework provides stable walking motions on flat ground. We summarize some key points here and refer interested reader to [12] for details.

While forming trajectories, instead of time, the HZD framework uses a phase variable denoted by θ. Since it will be an internal-clock, phase needs to be monotonic through the step. As the phase variable, we use θ drawn in Figure 1, which corresponds to $\theta = cq$ with $c = [-1\ 0\ -1/2\ 0\ -1]$. Second, since there are only four actuators, four angles to be controlled need to be chosen, which are denoted by h_0. Controlling the relative (internal) angles means $h_0 := [q_1\ q_2\ q_3\ q_4]^T$. 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

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

Taking the first derivative with respect to time reveals

$$\dot{h} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x} f(x) =: \mathcal{L}_f h, \tag{6}$$

where we used the fact that $\frac{\partial h}{\partial x} g(x) = 0$. Then, the second derivative of tracking error with respect to time is given by

$$\ddot{h} = \mathcal{L}_f^2 h + \mathcal{L}_g \mathcal{L}_f h\ u. \tag{7}$$

Substituting the controller structure

$$u(x) = (\mathcal{L}_g \mathcal{L}_f h)^{-1}(-\mathcal{L}_f^2 h + v) \tag{8}$$

to (7) yields

$$\ddot{h} = v. \tag{9}$$

To force h (and $\dot{h}$) to zero, a simple PD controller given by

$$v = -K_P y - K_D \dot{y} \tag{10}$$

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

As suggested in [12], we use Bézier polynomials to form the reference (h_d). Let θ^+ and θ^- be the phase at the beginning and end of limit cycle walking on flat terrain respectively. An internal clock which ticks from 0 to 1 during this limit cycle can be defined by

$$\tau(q) := \frac{\theta(q) - \theta^+}{\theta^- - \theta^+}. \tag{11}$$

Then, the Bézier curves are in the form of

$$b_i(\tau) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{12}$$

where M is the degree and α_k^i are the coefficients. Then, the reference trajectory is determined as

$$h_d(\theta) := \begin{bmatrix} b_1(\tau) \\ b_2(\tau) \\ b_3(\tau) \\ b_4(\tau) \end{bmatrix}. \tag{13}$$

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 [12]. In total, 20+2=22 parameters will be optimized including the PD controller gains.

2. Sliding Mode Control with Time-Invariant Piece-Wise Constant References

The second controller strategy of this paper is adopting sliding mode control (SMC) to track piece-wise constant references [15, 1].

As in the HZD control, let h_0 denote the four variables to control. As a result of our experience in previous work [16], we proceed with

$$h_0 := [\theta_2 \quad q_3 \quad q_4 \quad q_5]^T, \tag{14}$$

where $\theta_2 := q_2 + q_5$ is an absolute angle. Equivalently we can write $h_0 = H_0 q$, where

$$H_0 = \begin{bmatrix} 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix}. \tag{15}$$

Substituting the control input

$$u = (H_0 D^{-1} B)^{-1}(v + H_0 D^{-1}(C\dot{q} + G)), \tag{16}$$

into (1) yields

$$\ddot{h}_0 = v. \tag{17}$$

We will then design v such that h_0 acts as desired (h_d). The tracking error is again given by $h = h_0 - h_d$ and the generalized error is defined as

$$\sigma_i = \dot{h_i} + h_i/\tau_i, \quad i = \{1, 2, 3, 4\}, \tag{18}$$

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

$$0 = \dot{h_i} + h_i/\tau_i. \tag{19}$$

The solution to this equation is given by

$$h_i(t) = h_i(t_0)\ exp(-(t - t_0)/\tau_i), \tag{20}$$

which drives h_i to 0 exponentially fast. Next, v in (17) is chosen to be

$$v_i = -k_i |\sigma_i|^{2\alpha_i - 1} sign(\sigma_i), \quad i = \{1, 2, 3, 4\}, \tag{21}$$

where $k_i > 0$ and $0.5 < \alpha_i < 1$ are called the convergence coefficient and convergence exponent respectively. Note that if we had $\alpha_i = 1$, this would simply be a standard PD controller. Then, τ_i and k_i are analogous to the proportional gain and derivative time of a PD controller. However, $0.5 < \alpha_i < 1$ ensures finite time convergence. For further reading on SMC please refer to [17]. Note that SMC has $4 \times 3 = 12$ parameters to be optimized.

For faster optimization, it is preferable to have fewer parameters to optimize. Motivated by simplicity, we use references in the form of

$$h_d = \begin{cases} [\theta_2^{ref1} \quad q_3^{ref} \quad q_4^{ref1} \quad q_5^{ref}]^T, & \theta_1 := q_1 + q_5 > \pi, \\ [\theta_2^{ref2} \quad q_3^{ref} \quad q_4^{ref2} \quad q_5^{ref}]^T, & \text{otherwise.} \end{cases} \tag{22}$$

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, $12 + 6 = 18$ parameters will be optimized.

4 Discretization

4.1 Discretization of the dynamics

The impacts when a foot comes into contact with the ground provide a natural discretization of the robot motion. For the terrain profile described in Section 2.2, using (2) and (3) the step-to-step dynamics can be written as

$$x[n+1] = \rho^t(x[n], s[n], \zeta), \tag{23}$$

where $x[n]$ is the state of the robot at step n, $s[n]$ is the slope ahead at step n, ζ is the controller, and superscript t denotes the terrain assumption we made.

4.2 Discretization of the slope set

Our method requires a finite slope set S, which we typically set as

$$S = \left\{ k^\circ \ : \ \frac{k}{d_s} \in \mathbb{Z}, \ -20 \leq k \leq 20 \right\}, \tag{24}$$

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

4.3 Meshing Reachable State Space

There are two key goals in meshing. First, while the actual state x might be any value in the 10 dimensional state space, the reachable state space for system will be a lower dimensional manifold once we implement particular low-level control options and allow only terrain height variation as a perturbation source. The meshed set of states, X, needs to well cover the (reachable) part of state space the robot can visit. This set should be dense enough for accuracy while not having "too many" elements for computational efficiency. Second, we want to learn what the step-to-step transition mapping, $\rho^t(x, s, \zeta)$, is for all $x \in X$ and $s \in S$. Next, an initial mesh, X_i, should be chosen. In this study, we use an initial mesh consisting of only two points. One of these points (x_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 (x_2) should be in the controllable subspace. In other words, it should be in the basin of attraction for controller ζ.

Then, our algorithm explores the reachable state space deterministically. We initially start by a queue pf "unexplored" states, $\overline{X} = x_2$, which corresponds to all the states that are not simulated yet for all possible terrains. Then we start the following iteration: As long as there is a state $x \in \overline{X}$, simulate to find all possible $\rho^t(x, s, \zeta)$ and remove x from $\overline{X}$. For the points newly found, check their distance to other states in X. If the minimum such distance exceeds some threshold, a new point is then added to X and $\overline{X}$.

A crucial question is how to set the (threshold) distance metric so that the resulting X has a small number of states while accurately covering the reachable state space? The standardized (normalized) Euclidean distance turns out to be extremely useful, because it dynamically adjusts the weights for each dimension. The distance of a vector $\bar{x}$ from X is calculated as

$$d(\bar{x}, X) := \min_{x \in X} \left\{ \sum_i \left(\frac{\bar{x}_i - x_i}{r_i} \right)^2 \right\}, \tag{25}$$

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

$$c(\bar{x}, X) := \operatorname*{argmin}_{x \in X} \left\{ \sum_i \left(\frac{\bar{x}_i - x_i}{r_i} \right)^2 \right\}. \tag{26}$$

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 (preceding) 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 x. We can simulate $\rho^t(x, -20°, \zeta)$ just once and then extract $\rho^t(x, s, \zeta)$ 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 possible (less steep) impact points in the slope set.

Algorithm 1 Meshing algorithm

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

```
1: X̄ ← X_i (except x_1)
2: X ← X_i
3: while X̄ is non-empty do
4:     X̄_2 ← X̄
5:     empty X̄
6:     for each state x̄ ∈ X̄_2 do
7:         for each slope s ∈ S do
8:             Simulate a single step to get the final state x,
               when initial state is x̄, slope ahead is s,
               and controller ζ is used. Store this information
               in the state-transition map
9:             if robot did not fall and d(x, X) > d_thr then
10:                add x to X̄
11:                add x to X
12:            end if
13:        end for
14:    end for
15: end while
```

While meshing the whole 10D state space is infeasible, this algorithm is able to avoid the curse of dimensionality because the reachable state space is actually a quasi-2D manifold [18]. As a result, the meshing can be done with a relatively small number of iteration steps.

5 Metastable Markov Chains

5.1 Obtaining a Markov Chain

Meshing provides $\rho^t(x, s, \zeta)$ information for all $x \in X$, and $s \in S$. Using (26), we then write the approximate step-to-step dynamics as

$$x[n+1] = \rho^a(x[n], s[n], \zeta) := c(\rho^t(x[n], s[n], \zeta), X), \tag{27}$$

where the superscript a stands for approximation. The idea is approximating $x[n+1]$ as the closes point in X to $\rho^t(x[n], s[n], \zeta)$. After that, the deterministic state transition matrix can be written as

$$T^d_{ij}(s,\zeta) = \begin{cases} 1, & \text{if } x_j = \rho^a(x_i, s, \zeta) \\ 0, & otherwise. \end{cases} \tag{28}$$

Note that (28) 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

$$T^s_{ij} := Pr(x[n+1] = x_j \mid x[n] = x_i). \tag{29}$$

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

$$P_S(s) = Pr(s[n] = s). \tag{30}$$

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

$$s[n] \sim \mathcal{N}(\mu_s, \sigma_s^2) \tag{31}$$

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

$$T^s(\zeta) = \sum_{s \in S} P_S(s)\ T^d(s,\zeta). \tag{32}$$

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.

5.2 Expected Number of Steps Before Failure

This section serves as a summary on how we estimate the expected number of steps before failure, or mean first-passage time (MFPT). For details, we invite interested reader to [19].

The eigenvalues of T^s cannot have magnitude larger than one. However, the largest eigenvalue is equal to 1 because we model failure as absorbing. Also, the second largest eigenvalue, denoted by λ_2, is non-negative and real.

No matter what the initial condition is, if the robot does not fall within several steps, then the probability density function for the system converges to its "metastable distribution". Starting with this distribution, with $1 - \lambda_2$ probability the walker is going to fall on the next step, otherwise nothing the probability distribution will not change. Then, the probability of taking n steps only, equivalently falling at the nth step is simply

$$Pr(x[n] = x_1,\ x[n-1] \neq x_1) = \lambda_2^{n-1}(1 - \lambda_2). \tag{33}$$

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

$$\begin{aligned} MFPT &= E[FPT] \\ &= \sum_{n=1}^{\infty} n \; Pr(x[n] = x_1, \; x[n-1] \neq x_1) \\ &= \sum_{n=1}^{\infty} n\lambda_2^{n-1}(1-\lambda_2) = \frac{1}{1-\lambda_2}, \end{aligned} \tag{34}$$

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

$$M = \begin{cases} \infty & \lambda_2 = 1 \\ \frac{1}{1-\lambda_2} & \lambda_2 < 1. \end{cases} \tag{35}$$

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.

6 Results

Unless stated otherwise, we use the "minimize" function from [20], which is based on fminsearch function, in MATLAB to optimize. In this paper we will optimize for $\mu_s = 0$, i.e., zero average slope. However, we optimize control for each of a range of particular values of σ_s. 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 can calculate MFPT 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 controllers' performance with all controllers performing "badly" as σ_s gets large enough. 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 apply in this paper. Just like d_s, d_{thr} can be made smaller for higher accuracy in the expense of higher computation time. Whether d_s and d_{thr} are small enough can be checked after the optimization by using smaller values and confirming MFPT estimation does not change much.

1. Hybrid Zero Dynamics using Proportional-Derivative Control and Bézier Polynomials

For the HZD scheme, the base controller, ζ^1_{Base}, is obtained by assuming flat terrain, fixing speed to be $0.8m/s$ and minimizing energy usage as in [12]. To

obtain ζ^1_{COT}, we remove the speed constraint and optimize for cost of transport (COT) given by

$$COT = \frac{W}{mgd}, \tag{36}$$

where m is the mass, g is the gravitational constant, and d is the distance traveled. In this paper we use a conservative definition of "energy spent" by regarding negative work is also done by the robot, i.e., $W = |W_{positive}| + |W_{negative}|$.

Both of these controllers assume flat terrain, i.e., $\sigma_s = 0$. However, the HZD framework 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^1_{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 1 lists the parameters for each controller.

Table 1. Parameters for the First Controller Scheme in Radians

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

We compare the stability of each controller versus the roughness of the terrain in Figure 2. Noting the logarithmic y-axis, we immediately notice the huge improvement in stability by optimizing with the suggested method. As an interesting side note, ζ^1_{MFPT} has a lower COT than ζ^1_{Base} (0.1411 compared to 0.1865). This is possible, because no speed constraint was set for ζ^1_{MFPT} to concentrate fully on stability. However, such speed constraints can be easily incorporated in a future work.

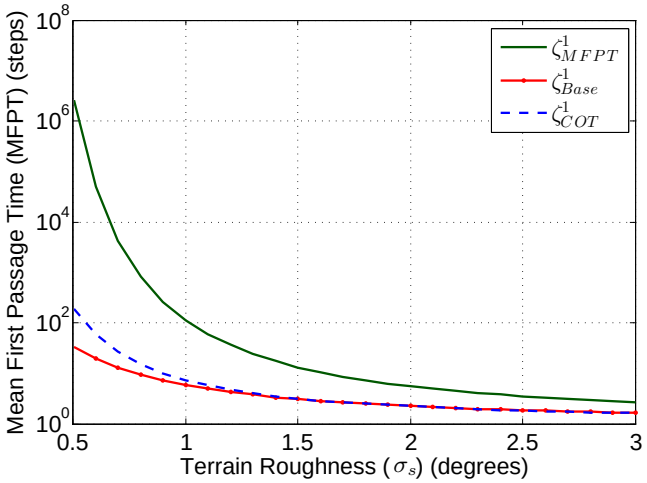

Fig. 2. Average number of steps before falling calculated using (34) versus σ_s for the first controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_s = 0$.

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 2 for $\sigma_s = 2$, 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 2. Estimation of MFPT for First Controller Scheme with $\mu_s = 0$ and $\sigma_s = 2$

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
Estimation using (34)	2.2085	2.2049	5.5206
Monte Carlo Simulation	2.1511	2.2487	6.1290

2. Sliding Mode Control with Time-Invariant Piece-Wise Constant References

We start optimizing second controller scheme with the hand-tuned parameters taken from [16], which we will refer to with ζ^2_{Base}. We first optimize for Cost of Transport (COT) of the limit cycle gait on flat terrain to obtain ζ^2_{COT}. We

then optimize for MFPT with $\sigma_s = 2$, $d_s = 1$ and $d_{thr} = 1$. This results with controller ζ^2_{MFPT}. The parameters for each controller are given in Table 3.

Table 3. Parameters for the Second Controller Scheme

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{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

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
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. Again noting the logarithmic y-axis, the suggested method provides a dramatic increase in the stability, just like in Figure 2.

Table 4 presents the Monte Carlo study obtained assuming $\sigma_s = 5$. Just like in Table 2, we omit the first step to allow robot "forget" the initial condition.

Table 4. Estimation of MFPT for Second Controller Scheme with $\mu_s = 0$ and $\sigma_s = 5$

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
Estimation using (34)	5.1766	1.1470	10.6433
Monte Carlo Simulation	5.0738	1.5716	10.4813

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 5 shows, there is a huge difference between ζ^2_{MFPT} and any of the HZD controllers. Comparing the results in Figures 2 and 3 also emphasizes this dramatic difference. So, we conclude that the second controller scheme is much more capable in terms of stability.

On the other hand, the second controller scheme has discontinuities in the references. Correspondingly, the actual joint trajectory highly depend on the

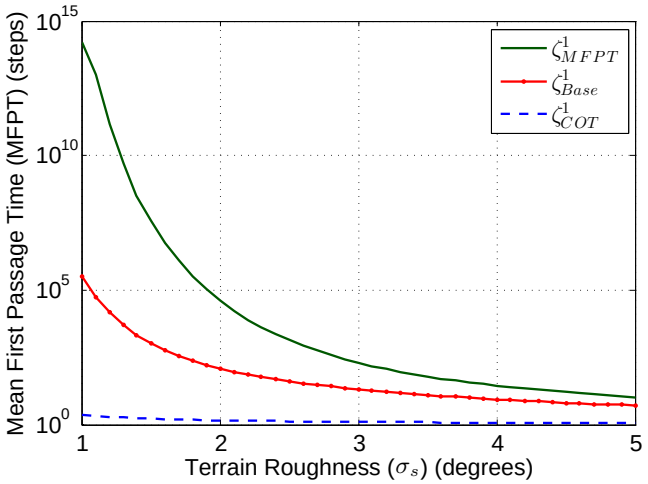

Fig. 3. Average number of steps before falling calculated using (34) versus σ_s for the second controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_s = 0$. Note that range of σ_s is different from Figure 3.

controller parameters, which may not be desired. Use of a smoother parameterization for the references might provide further improvements in performance. Also, note that many parameters of ζ^2_{Base} and ζ^2_{MFPT} in Table 3 are very close. We suspect that we only find local minimums. Indeed, starting with different initial conditions yields different final gaits.

Table 5. 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	5.9	7.3	113.1	3.2×10^{5}	2.2	1.6×10^{14}

A major problem in the first 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 when $\theta > \theta^+$ (or $\tau > 1$). 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^1_{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. The fact that turning the controller off greatly helps

when going downhill shows the need for a better reference parametrization to keep controller on at all times.

7 Conclusions and Future Work

In this work, we present a methodology for optimizing a low-level control scheme and of benchmarking final 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 to provide a systematic means of optimizing and benchmarking 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 end up with a local minimum for the second controller scheme. We aim to find 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 [1]. Finally, we may use costs that incorporate other performance metrics also, similar to [21]. For example, a practical goal is to increase stability while decreasing energy consumption, balancing the two aims as desired.

Acknowledgments. 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. 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.
2. M. Vukobratovic and B. Borovac, "Zero-moment point - thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.
3. D. Hobbelen and M. Wisse, "A Disturbance Rejection Measure for Limit Cycle Walkers: The Gait Sensitivity Norm," *IEEE Transactions on Robotics*, vol. 23, pp. 1213–1224, Dec. 2007.
4. J. Morimioto, G. Zeglin, and C. Atkeson, "Minimax differential dynamic programming: application to a biped walking robot," in *SICE 2003 Annual Conference*, vol. 3, pp. 2310–2315 Vol.3, Aug. 2003.
5. H. Dai and R. Tedrake, "L2-gain optimization for robust bipedal walking on unknown terrain," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 3116–3123, IEEE, 2013.

6. Y. Hurmuzlu and C. Basdogan, "On the Measurement of Dynamic Stability of Human Locomotion," *Journal of Biomechanical Engineering*, vol. 116, pp. 30–36, Feb. 1994.
7. T. McGeer, "Passive Dynamic Walking," *The International Journal of Robotics Research*, vol. 9, pp. 62–82, Apr. 1990.
8. B. Griffin and J. Grizzle, "Walking Gait Optimization for Accommodation of Unknown Terrain Height Variations," 2015. Submitted.
9. J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual Model Control: An Intuitive Approach for Bipedal Locomotion," *The International Journal of Robotics Research*, vol. 20, pp. 129–143, Feb. 2001.
10. 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.
11. K. Byl and R. Tedrake, "Metastable Walking Machines," *The International Journal of Robotics Research*, vol. 28, pp. 1040–1064, Aug. 2009.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. A. Sabanovic and K. Ohnishi, "Motion Control Systems," John Wiley & Sons, 2011.
18. 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.
19. C. O. Saglam and K. Byl, "Metastable Markov Chains," in *IEEE Conference on Decision and Control (CDC)*, Dec. 2014.
20. R. Oldenhuis, "minimize - File Exchange - MATLAB Central."
21. 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.