Mesh-based Methods for Quantifying and Improving Robustness of a Planar Biped Model to Random Push Disturbances

Nihar Talele and Katie Byl

***Abstract*— In this paper, we apply meshing tools to improve and analyze the performance of a 5-link planar biped model to random push perturbations. Creating a mesh for a 14-dimensional state space would typically be infeasible. However, as we show in this paper, low level controllers can restrict the reachable space of the system to a much lower dimensional manifold, which makes it possible to apply our tools to improve the performance. To validate the effectiveness of our tools in analyzing, quantifying and improving the performance of a system, we conduct simulations on two different sets of trajectories: one consisting of trajectories having only a single support phase, and a second set consisting of trajectories having both a double support as well as a single support phase.**

I. INTRODUCTION

Humanoid locomotion control is challenging due to the presence of underactuated dynamics, with constraints at the ground-foot contact imposing dynamic limitations on feasible motions. At the same time, deliberate underactuation in bipeds can potentially provide more energy efficient locomotion, making trade-offs between efficiency and stability a particularly interesting problem in biped control.

Use of optimization to generate energy-optimal motions for underactuated systems such as bipeds has been popular for some time now. In 1999, Hardt et al. used DIRCOL [1] to implement direct collocation in formulating the problem of minimizing energy use for a planar 5-link biped walker. In recent years, much effort has been focused on increasing the robustness of such energy-optimal trajectories. In [2] Kuindersma et. al. use quadratic programming to incorporate constraints in the feedback policy. In [3], Majumdar and Tedrake use the concept of funnel libraries to increase the robustness of optimal trajectories. In [4], Nguyen, Grizzle, Sreenath et. al. use 2-step gait optimization and gait interpolation to navigate across rough terrain. And while all these results are very impressive, a motivating goal in this work is to more effectively quantify and compare the robustness gains achieved by such otherwise disparate techniques. Toward this aim, we build upon a framework based on metastability.

In [5], Byl and Tedrake use the concept of metastability to quantify the robustness of simple rough-terrain walking models, proposing mean time to falling, or mean first-passage time (MFPT), as an important metric of merit. In [6], Saglam and Byl build on the previous work by analyzing the trade-offs between energy and robustness for a 5-link biped model (Fig. 1), using once-per-step switching between low-level sliding mode controllers for walking on rough terrain. As the approach models a discretized approximation of the resulting dynamics as a Markov decision process (MDP), feasibility of the methods in [6] depends upon the "meshability" of the resulting system; i.e., the stochastic system dynamics must visit only a relatively low-dimensional manifold within the full (i.e., 10-D set of positions and velocities, in [6]) state space, to avoid the curse of dimensionality.

In this paper, we further extend the applicability of these tools to analyze the effects not only of "noise" in the terrain height, which determines when an otherwise-deterministic continuous-time trajectory ends (i.e., with swing leg impact at an unknown ground height), but also of more general perturbation disturbances typical of real-world scenarios, such as push disturbances that can happen at any time during the gait cycle. Additionally, we demonstrate applicability of these tools for other low-level control schemes, aside from sliding mode control (SMC), to explore whether the finite-time convergence of SMC to lower-dimensional manifolds is a critical key to meshability; we conclude that it is not.

The rest of the paper is organized as follows. In Section II, we present the 5-link model we use for our simulation as well as the control structure that we use to control the energy optimal trajectories that we have generated. In Section III, we provide more details on how we generate the energy optimal trajectories using our trajectory optimization framework. Section IV gives a brief overview on our meshing techniques and some important results that we use for analyses in the rest of the paper. In Section V, we present our results, and we discuss and present our conclusions in Section VI.

II. MODEL AND CONTROL

A. Model

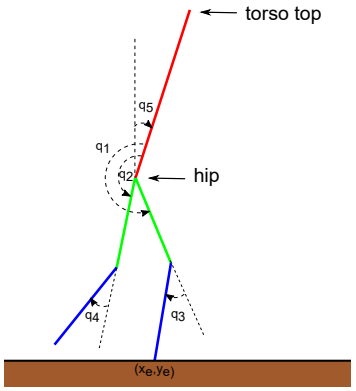

Fig. 1. Model of the 5-link biped system used in simulations.

*This work was funded in part by NSF NRI award 1526424.

Nihar Talele and Katie Byl are with the Electrical Engineering Department at the University of California, Santa Barbara CA 93106 nihar@ucsb.edu, katiebyl@ucsb.edu

We use a point-footed, 5-link model as shown in Fig. 1 for our simulations. This model is studied extensively in [7]. We set the mass and length parameters such that the model has a total mass of 70 Kg and height of 1.6m. To calculate inertia values, we consider the links to be thin rods with uniform mass distribution. Although [7] studies only single-support gaits, we use an extended model that allows us to design and simulate double-support trajectories, as well. Our model has 7 degrees of freedom: $q := [q_1\ q_2\ q_3\ q_4\ q_5\ xe\ ye]^T$, where q_1, q_2, q_3, q_4, q_5 are the five angles as shown, and *xe* and *ye* are the x and y coordinates of the end of the stance foot respectively. The simulation model can be compactly expressed in the following mathematical model:

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

Here, $D(q) \in \mathbb{R}^{7\times7}$ is the mass inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{7\times7}$ is the Coriolis matrix, $G \in \mathbb{R}^{7\times1}$ is the vector of gravitational forces, $B \in \mathbb{R}^{7\times4}$ is the input mapping and $E \in \mathbb{R}^{7\times4}$ is the mapping of forces acting on the stance leg and swing legs, respectively. During single support, the system is underactuated, as the input mapping B has more rows than columns. During double support, F_{ext} ensures no penetration of the leg into the ground and provides Coulumbic friction limits laterally. Sliding and/or bouncing of the point feet can occur in this model. Note that this differs from many formulations, which (artificially) treat such eventualities as failure modes in the dynamics, for simplicity. Swing leg impacts at the ground are modeled as instantaneous, inelastic collisions.

B. Control

We use a partial feedback linearization (PFL) controller to simulate the optimal trajectories obtained from the optimization framework. For our current work, we use a semi-implicit Euler integration scheme, which is also used by many physics-based simulators in use today, such as MuJoCo [8]. Instead of using the current velocity to calculate the next position, we first calculate the velocity for the next state and then use that velocity to calculate the position for the next time step. For our meshing algorithm, the time step for our integration scheme is 0.0005s. The choice of a fixed-step integration scheme instead of higher-order ODE functions in MATLAB was made to reduce the time required for meshing, with a focus on methodology in this work. Our controller has the following structure:

$$U_{fb} = (SD^{-1}B)^{-1}(v + SD^{-1}(C\dot{q} + G)), \tag{2}$$

where S is a matrix determining which degree or combination of degrees of freedom (DOFs) are controlled for the system. With five DOFs for the 5-link walker, and a degree of underactuation of one (at the passive ankle contact at the ground), S selects 4 acceleration terms to be set. We use,

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

so that we set $v = [\ddot{q}_{2des}, \ddot{q}_{1des} + \ddot{q}_{3des}, \ddot{q}_{4des}, \ddot{q}_{5des}]^\mathsf{T}$ where $\ddot{q}_{des} = -K_p(q - q_{des}) - K_d(\dot{q} - \dot{q}_{des})$. Here, $K_p = \omega_n^2$ and $K_d = 2\zeta\omega_n$ and q_{des}, $\dot{q}_{des}$ are the optimal positions and velocities obtained from our trajectory optimization as will be described in Section III.

For all our simulations in this paper we set $\zeta = 1$ and $\omega_n = 100$. The total input to the system is then given by $U = U_{ff} + U_{fb}$, where U_{ff} is the feedforward torque, which is nominal torque, u, output by the optimization framework, as described in the next Section.

III. TRAJECTORY OPTIMIZATION

Our optimization framework here is very similar to our previous work [9], where we have

$$\min_{q,\dot{q},u,F_{ext}} \quad COT \tag{4}$$

$$\text{such that} \quad D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + EF_{ext} \tag{5}$$

$$\Phi(q,\dot{q},u,F_{ext}) \leq 0 \tag{6}$$

Here, $\Phi(q,\dot{q},u,F_{ext})$ is the vector of constraints and COT is given by Eq. 10. In addition, we also impose constraints on forces on the stance leg and swing legs. We generate trajectories assuming the coefficient of friction is 0.2, but we simulate the trajectories with a coefficient of friction of 0.5. A tighter bound on the friction is used because even though the controlled trajectory deviates from the optimal trajectory by a very small margin, it may cause slipping in certain cases. We use a Direct Collocation framework for optimization. For the purpose of trajectory optimization, we discretize the dynamics with a resolution of 0.01s and use the trapezoid rule for integration. For the inputs, we use a zero-order hold instead of first-order as this provided smoother results. We use IPOPT [10] for optimization, along with CasADi [11] to calculate the gradients for optimization using automatic differentiation. All the trajectories are designed for a fixed time length of 0.6s and are optimized for minimal energy use.

The cost function for our optimization is given by

$$Cost = \int_0^t \sum_{n=1}^{7} |P_n(t)|dt, \tag{7}$$

where $P_n(t) = \tau_n\omega_n$. Here τ_n is the torque and ω_n is the angular velocity for the corresponding degree of freedom. We sum the absolute value of both the positive as well as the negative mechanical power. Since absolute value functions can be problematic for the convergence of optimization, we use the following approximation, where we have also taken discretization into account:

$$Cost = \sum_{k=0}^{N-1} \sum_{n=1}^{7} \widetilde{P}_n(k)\Delta t. \tag{8}$$

Here

$$\widetilde{P}_n = \sqrt{P_n^2 + \varepsilon}, \tag{9}$$

where ε is a smoothing factor. For our experiments, we used $\varepsilon = 0.5$. Experimentally, smaller values of ε will work but also increase the time to find an optimal solution with not much difference in the final trajectories obtained. In addition to the cost, we also add some regularization terms to smooth the velocities and external forces. We found that smoothing

the external forces was essential in order to control the double support phase trajectories on the model. Also of note is the fact that double support phase trajectories have a higher cost of transport as compared to single support phase trajectories. For the same stride length and ground clearance, the cost of transport for double support phase trajectories was 2-3 times that of single support phase trajectories. Detailed values are given in Table I; all trajectories in this Table are for a constant time length of 0.6s. We calculate the Cost of Transport as

$$COT = \frac{Cost}{m_t g d}, \tag{10}$$

where *Cost* is as described in Eq. 7, m_t is the total mass of the system, g is the acceleration due to gravity and d is the stride length.

Stride Length(m)	$COT(SS)$	$COT(DS)$
0.4	0.0501	0.1385
0.5	0.0809	0.1995
0.6	0.1332	0.3133

TABLE I
COT FOR SINGLE SUPPORT(SS) AND DOUBLE SUPPORT(DS) TRAJECTORIES, FOR DIFFERENT STRIDE LENGTHS.

IV. MESHING

We record the state at the end of each step and use these discrete states (on a Poincaré section) to build a mesh $\mathbb{M}$, toward modeling stochastic, step-to-step dynamics. For our current work, we build and analyze a mesh that includes random disturbances, in the form of a push that can occur some time in the middle of the trajectory. Regardless of when (if at all) that a push occurs, we record the subsequent post-impact state with the ground to create our mesh. We denote this post-impact state as $s = x^+$. The mesh also includes an absorbing failure state. All failure events transition to (and subsequently remain in) this state, in our discrete model of the dynamics. We then build the mesh, starting with each of the optimal limit cycles, along with the failure state. Non-failure states are simulated, as described in Section II-B, for each possible optimal trajectory found as described in Section III, and for each of a finite set of possible push disturbances γ in our uncertainty set. That is, the control action is the choice of which reference trajectory, ξ, to use. Our meshing algorithm performs an exhaustive search, sequentially and deterministically exploring all possible states, s, that can be visited, to generate a mesh approximating the reachable state space of the modeled system, for the given control and disturbance sets. New states visited are added to the mesh if and only if they exceed a defined distance threshold from all previously created states in the mesh. More specifically, we use a simple threshold criterion, based on a Euclidean distance metric given by

$$d(s_1, s_2) = \min_{s_2 \in \mathbb{M}} \sqrt{\sum_{i=1}^{n} (s_1(i) - s_2(i))^2}. \tag{11}$$

States are added to the mesh if $d(s_1, s_2) > d_{thr}$. When any part of the biped other than one or both of the point feet touches the ground, this is modeled as a transition to the failure state. Because we allow for sliding and bouncing in our simulation, we pick a threshold time limit of 0.4s after which the first impact of the swing foot with the ground terminates the current step.

A deterministic state transition matrix $T_{det}(s, \xi, \gamma)$ which contains the information of all state transitions is updated every iteration. When calculating the distance, we ignore the x position of the ground contact of the stance leg as it has no effect on the next step being taken.

For our uncertainty set, we create push scenarios, γ, with different magnitudes in forward as well as backward directions that occur at various times throughout the trajectory. For our current work, we restrict the maximum number of disturbances per simulation to 1. Some disturbances are applied at the hip and some occur at the top of the torso. Our goal is to "push" the trajectories during walking away from their nominal paths in state space, to explore the mixing effects over time of cumulative disturbances that vary in direction, magnitude, and timing during the gait. Our underlying hypothesis is that while the system does not converge to any particular fixed point under such conditions, it is also limited to a relatively low-dimensional subset of the full state space, making long-term failure probabilities calculable.

A. Stochastic State Transition matrix

Once the mesh is complete and we have a deterministic state transition matrix as described above, we proceed with calculating the optimal policy for the mesh, which depends on the assumed probability distribution of the disturbances. The optimal policy given by $\pi(s)$ is a function of the state. Once we have this policy, the controller for the next step is selected by $\xi = \pi(s)$. We study a range of values for the probability distribution, $P(\gamma)$, for our uncertainty set. For each, we use value iteration to converge to a stable policy, where the value of being in state i is given by

$$V(i) := \max_{\xi} (\sum_{\gamma} P(\gamma)(R_j + \alpha V(j))), \tag{12}$$

and the optimal policy is given by

$$\pi(i) := \operatorname*{argmax}_{\xi} (\sum_{\gamma} P(\gamma)(R_j + \alpha V(j))), \tag{13}$$

where α is the learning rate, which we set to 0.99 for our experiments, and R_j is the reward function which is given by

$$R_j = \begin{cases} 0, & \text{if } j \text{ is 1} \\ 1, & \text{else.} \end{cases} \tag{14}$$

This function rewards the control action so long as it does not lead to a failure state. Using this optimal policy, we calculate the stochastic state transition matrix:

$$T(i, j) := \sum_{\gamma} P(\gamma) f_j \tag{15}$$

where

$$f_j = \begin{cases} 1, & \text{if } T_{det}(i,\pi(i),\gamma) = j \\ 0, & \text{else.} \end{cases} \tag{16}$$

That is, the action depends deterministically on the present state, each particular disturbance then maps deterministically to one particular future state under this control, and the probability mass function for disturbances sets the relative weights in each possible arrival state. Note, each row in the stochastic transition matrix correspondingly sums to one.

B. Mean First Passage Time

The concept of mean first passage time (MFPT) for metastable walking systems is explained in detail in [12] and [5]. We highlight important results here that we will be using in the rest of the paper; they focus on an eigen analysis of the *transpose* of the stochastic transition matrix, T. First, we define a metastable distribution, ϕ, as the stationary distribution in state space, conditioned on not having entered the failure state:

$$\phi_i = \lim_{n\to\infty} Pr(X(n) = x_i | X(n) \neq x_1). \tag{17}$$

The mean first passage vector m where m_i is the mean time for the state i to go into failure is given by

$$m_i = \begin{cases} 0, & \text{if } i = 1 \\ 1 + \sum_{j>1} T_{ij} m_j, & \text{else.} \end{cases} \tag{18}$$

Vector m can be calculated by

$$m = \begin{bmatrix} 0 \\ (I - \hat{T})^{-1}, \end{bmatrix} \tag{19}$$

where $\hat{T}$ is T with the first column and row (corresponding to the failure state, for which $m(1) = 0$) removed. This in turn lets us calculate the system wide MFPT:

$$M = \sum_i \phi_i m_i. \tag{20}$$

For a more efficient calculation of M, we use the eigenvalues of the stochastic transition matrix. Because of the structure of T, with $T(1,1) = 1$ corresponding to the absorbing failure state, the largest eigenvalue has the value 1: failure is a persistent state. Let λ_2 be the second largest eigenvalue of T. λ_2 denotes the probability of taking a successful step, assuming initial conditions have been forgotten and the walker is in a non-failure state; i.e., the probability of failure is $1 - \lambda_2$. The probability of taking only n steps is then

$$Pr(X(n) = x_1, X(n-1) \neq x_1) = \lambda_2^{n-1}(1 - \lambda_2). \tag{21}$$

By calculating the expectation of this probability distribution, we obtain the approximate MFPT for the system given by

$$M = \frac{1}{(1 - \lambda_2)}. \tag{22}$$

Finally, note that normalizing all the non-failure-state elements in the eigenvector associated with the 2^{nd}-largest eigenvalue of T^T (the *transpose* of T) correspondingly yields the metastable distribution, ϕ, in Eq. 17.

C. Average Cost of Transport

We can calculate the average COT for the system, based on the metastable distribution ϕ. To do this, we first calculate the average COT for each state in the mesh using

$$\overline{COT}_i = \sum_\gamma COT(i,\pi(i),\gamma) P(\gamma) \tag{23}$$

where we assume $COT(i,\pi(i),\gamma)P(\gamma) = 0$ if $T_{det}(i,\pi(i),\gamma) = 1$, i.e., if state i transitions to the failure state. The average COT for the whole system is then

$$\overline{COT} = \sum_i \overline{COT}_i \phi_i. \tag{24}$$

D. Fractional Dimensionality

As the dimensionality of a system increases, discretized (e.g., mesh-based) machine learning methods become impractical, due to Bellman's so-called "curse of dimensionality". For a D dimensional system, to make the state-state transition mappings more accurate we can reduce the distance between mesh points. The dimensionality of the system can be used to predict the resulting change in the mesh size. More specifically, if we wish to change the distance between mesh points from s to s/k, then we would expect the number of mesh points to scale as $N \propto k^D$. Figure 2(a) shows some simple examples of this. For example, for a 3D cuboid ($D = 3$) if we reduce the side length (s) of each element in a uniform grid to $1/2$ of its original side, there will be $N = 2^3 = 8$ grid elements.

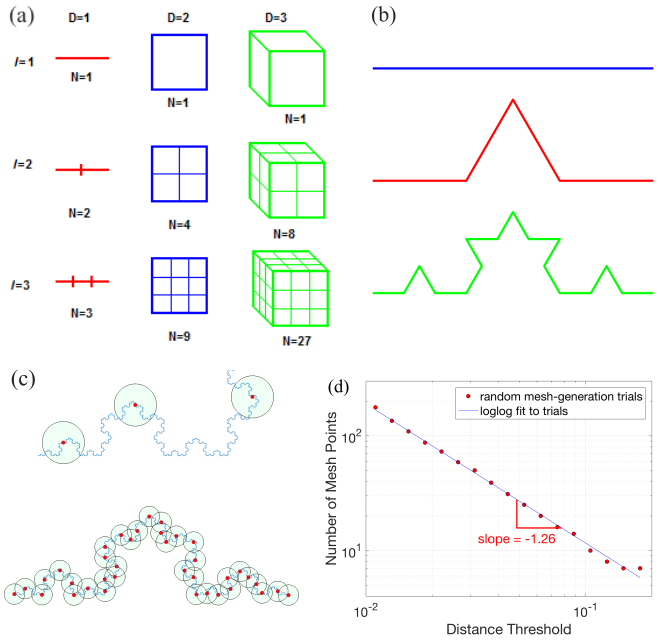

Fig. 2. Fractional Dimensionality. (a) Uniform meshing examples [13], (b) Koch snowflake segment, (c) Non-uniform meshing example (see text for details), (d) slope of a loglog plot of "Distance Threshold" versus "Number of Mesh Points" is -1.26, which is correspondingly an estimate for the negative of the dimensionality of the Koch snowflake.

For some systems, this scaling is not an integer, due to some inherent structure of a system within state space.

Fractals are a common example of this. The Koch curve, also known as the Koch snowflake, is illustrated in Fig. 2(b).

The Koch snowflake has a dimensionality of about 1.26. To have a better understanding, imagine we wish to select a non-uniform set of points to approximate the full set. Paralleling the meshing approach used in this paper, we first pick a particular "distance threshold", d_{thr}, to compare candidate new mesh points with existing one, as in Equation 11. New candidate points are found and added to the mesh if and only if they are outside a "bubble" of radius d_{thr} of all the existing mesh points, resulting (after an exhaustive search) in a non-uniform mesh. Figure 2(c) shows a resulting example. Since $N \propto k^D$, and d_{thr} is in turn proportional to $1/k$, the negative slope of a log-log plot, such as shown in Fig. 2(d), gives the dimensionality, D. We use this method in Sec. V in estimating mesh dimensionality, which describes scaling of the number of discrete mesh points as d_{thr} varies.

V. RESULTS

We analyze the performance for two different scenarios. For the first case, our trajectory set is comprised only of single support phase trajectories. For the second case the trajectory set consists of trajectories which have both single support, as well as double support phase. All other parameters, except the trajectory and the structure of PFL, remain same for both the experiments. The following subsections detail each scenario.

A. Trajectories with only single support phase

We use 3 trajectories $\{\xi_1, \xi_2, \xi_3\}$ with stride lengths of 0.4m, 0.5m and 0.6m respectively in our trajectory set $\mathbb{Z}$ to generate the mesh. All these trajectories have a time length of 0.6s and have been optimized for energy. We then use PFL as stated above to control these trajectories. For the trajectories in this set, matrix S as given in Eq. 3 sets the four acceleration terms that are directly controlled. Fig. 3 shows one example gait trajectory. Because of the sensitive nature of the inelastic

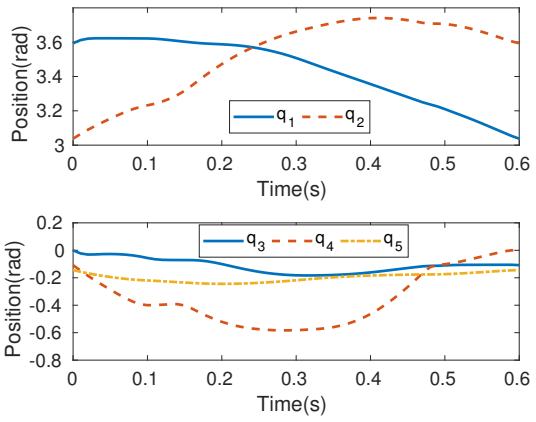

Fig. 3. Plot of position trajectories for a 0.5m stride length trajectory having only single support phase

collision at the end of each footstep, COT of controlled gaits (using integration as described in Sec. II-B) deviates from the value predicted through more coarse integration during trajectory optimization in Sec. III. Table II shows this deviation. Table III shows the number of states for different

Stride Length(m)	Predicted COT	Controlled COT
0.4	0.0501	0.1507
0.5	0.0809	0.1021
0.6	0.1332	0.1417

TABLE II
COT FOR SINGLE SUPPORT PHASE TRAJECTORIES.

meshes that we generate. As expected from Sec. IV-D, the number of states increase by an exponential factor as d_{thr} decreases. If we plot the data in this table on a logarithmic

d_{thr}	0.3	0.4	0.45	0.5	0.6
# of points	6089	2333	1856	946	779

TABLE III
NUMBER OF POINTS FOR DIFFERENT THRESHOLD VALUES FOR MESH GENERATED WITH SINGLE SUPPORT PHASE TRAJECTORIES

scale, the slope gives the dimensionality by which the mesh grows. Fig. 4 shows the said plot. For the case of trajectories

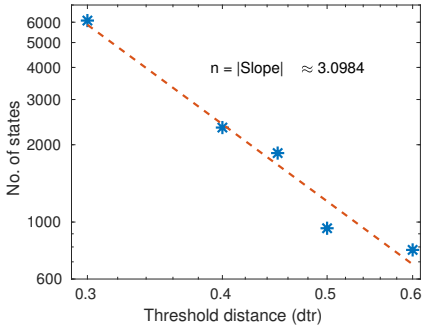

Fig. 4. Dimensionality of mesh growth for Single Support trajectories.

with only single support phase, the dimensionality comes out to be approximately $n \approx 3.1$ which is much less than the total 14 dimensional state space. It is this low dimensionality that allows us to use our meshing techniques even for real world noise scenarios like push disturbances. Once we have the mesh, we can calculate the MFPT and other important metrics. The following results are obtained for a mesh with $d_{thr} = 0.6$, along with the following probability distribution for the uncertainty set.

$$P(\gamma) = \begin{cases} 0.8, & \text{if no disturbance} \\ 0.2/28, & \text{else.} \end{cases} \tag{25}$$

Our uncertainty set $\mathbb{U}$ consists of 29 scenarios which are shown in Fig. 5. Of the 29 cases, 1 case is a no-disturbance scenario.

Using Eq. 22, the MFPT for our mesh comes out to 103.25, suggesting that with our optimal policy and the

disturbance probability given, the robot would take approximately 103 steps before falling. From Eq. 24, we get the average cost of transport for the metastable distribution of the system to be 0.2751. Using our tools, we can perform various interesting analyses, such as calculating the sensitivity to disturbances. To do so, we use the following probability distribution. Here, γ_i is the disturbance for which we are calculating the sensitivity.

$$P(\gamma) = \begin{cases} 0.7, & \text{if no disturbance} \\ 0.2, & \text{if } \gamma = \gamma_i \\ 0.1/27, & \text{else.} \end{cases} \tag{26}$$

We then proceed to calculate the MFPT for the given probability distribution. We do this for all the disturbances in $\mathbb{U}$ and the results are as shown in Fig. 5.

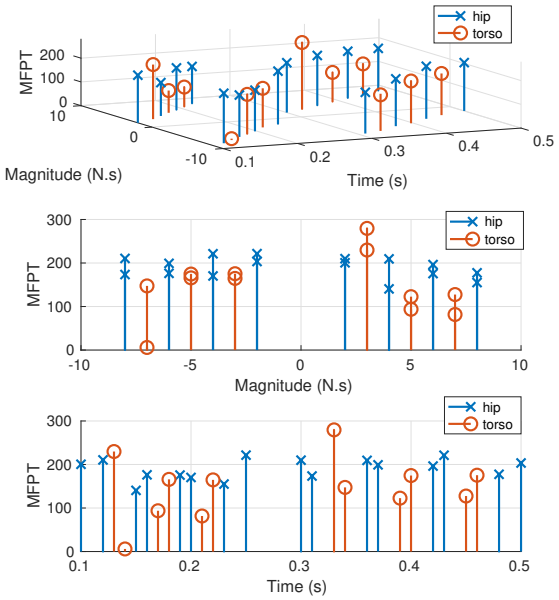

Fig. 5. Plot showing the MFPT for various disturbance probabilities. The top figure shows the MFPT as a function of both the magnitude as well as the time of disturbance. The middle and bottom plot show the MFPT vs magnitude and time respectively.

B. *Trajectories with single support and double support phase*

As in the previous section, we use 3 trajectories with a stride length of 0.4m, 0.5m and 0.6m to generate our mesh. These trajectories are also optimized for energy and have a time length of 0.6s. These trajectories differ from the trajectories of the previous section in that they have a double support phase for 20% of the gait duration. The gait starts in the double support phase and ends at the end of the single support phase with impact to the ground. These post-impact states are added to the mesh. Fig. 6 shows one such trajectory. To control these trajectories, we use the same PFL structure but with a switching scheme. We divide the trajectory into 3 different time intervals 0-0.15s, 0.15-0.53s and 0.53-0.6s. For time intervals 1 and 3 we set

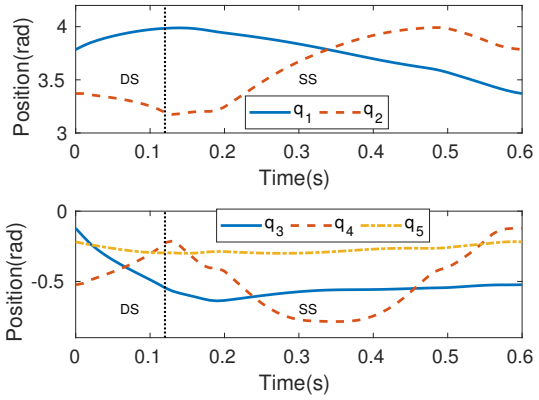

Fig. 6. Plot of position trajectories for a 0.5m stride length trajectory having both double support (DS) and single support (SS) phase. The transition from double support to single support happens at 0.12s mark.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{27}$$

and for time interval 2 we set S as given in Eq. 3. We introduced the switching scheme because the PFL scheme used in the previous section was not able to stabilize the trajectories having both a double support and a single support phase. The choice of the time intervals was based on the transition from double support to single support phase and some experimental simulations.

Tables IV and V show the COT for the trajectories and the number of points for mesh generated with different d_{thr} values respectively.

Stride Length(m)	COT	Stable COT
0.4	0.1385	0.2216
0.5	0.1995	0.2733
0.6	0.3133	0.3618

TABLE IV

COT FOR TRAJECTORIES HAVING BOTH DOUBLE SUPPORT AND SINGLE SUPPORT PHASE BEFORE AND AFTER PFL IS APPLIED.

d_{thr}	0.4	0.45	0.5	0.6
# of points	7260	5645	3925	2244

TABLE V

NUMBER OF POINTS FOR DIFFERENT THRESHOLD VALUES FOR, TRAJECTORIES HAVING DOUBLE AS WELL AS SINGLE SUPPORT PHASE

Fig. 7 shows the plot to calculate the dimensionality factor for this. The MFPT for the case with $d_{thr} = 0.6$ for the probability distribution given in Eq. 25 for these set of trajectories comes out to be 496.88 with an average cost of transport $\overline{COT}$ of 0.4993. This is significantly more robust than that for trajectories having only single support phase but

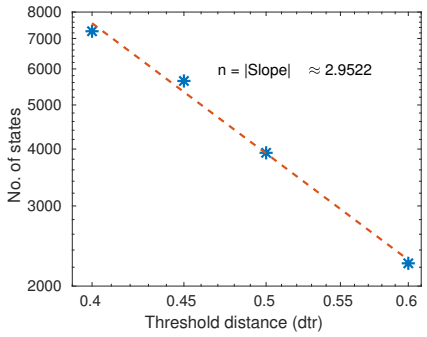

Fig. 7. Dimensionality of mesh growth for trajectories having both double support and single support phase.

at the same time there is a trade off in terms of the amount of energy expended. This may also be in part due to the different PFL schemes used in both cases, however, the fact that the dimensionality in both cases comes out nearly the same indicates that the significant performance improvement results not from the contracting effects of the low level controllers but mostly due to the fact that double support phase introduces an over actuated phase in the simulation where we can control all the degrees of freedom of the system.

We also build a mesh by combining the trajectories from both sections above. As expected, this mesh provides a superior performance compared to either of the mesh generated in the previous section. For a $d_{thr} = 0.6$ this mesh has 10,603 states with a MFPT of 2,441.58 for a probability distribution given in Eq. 25. Next we analyze how sensitive the policy is to the changes in probability distribution. For this, we pick a probability value for each disturbance from a uniform distribution between 0 and 1. We then normalize these probabilities such that the probabilities of all disturbances sum to x. The probability for no push scenario is then $1-x$. For each x we collect 10 data points. Fig. 8 shows how much the performance is affected due to change in the probability of the disturbance. The % policy change gives the percentage of states that have a different policy when using the optimal policy versus the standard policy. The standard policy in this case is the optimal policy generated for the probability distribution given by Eq. 25.

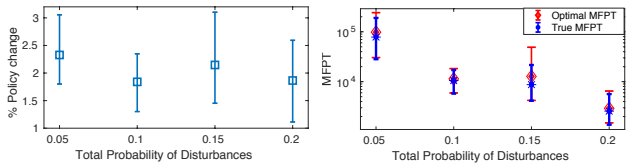

Fig. 8. Sensitivity of optimal policy (left) and MFPT variability (right) to changes in probability of disturbances.

VI. CONCLUSIONS

Our experiments have shown that our tools previously developed for improving and analyzing the performance for waking on rough terrain are also applicable to more random real world noise scenarios like push disturbances. In addition, our simulations also show that trajectories that have a double support phase are much more robust to pushes as compared to trajectories with only single support phase. This result is intuitive, but our tools allow us to quantify such differences in long-term performance. Our tools also allow us to do sensitivity analyses for different kinds of disturbances. The plots clearly indicate that the combined effects of magnitude as well as time of disturbance are significant to the performance of the system as the same magnitude disturbance accrues a different penalty if applied at different time. In addition we have also shown that the policy is quite robust to changes in probability distribution of the disturbances. In other words, even if the statistics of future disturbances are not well-known, the policy derived from the wrong statistical assumptions will perform nearly as well as the true optimal policy.

In showing that our tools are applicable to random noise scenarios, we have cleared an important criterion in applying these tools to real world systems. So far we have only focused on applying our tools to environmental disturbances. Future work will focus on adapting and improving these tools so that they can cope with changing or uncertain information about system parameters such as mass distribution and/or sensing capabilities as well as understanding the contracting nature of the closed loop dynamics that allow us to apply our meshing based tools to analyze and improve the performance of high dimensional systems.

REFERENCES

[1] M. Hardt, K. Kreutz-Delgado, and J. W. Helton, "Optimal biped walking with a complete dynamical model," in *Proc. IEEE Conference on Decision and Control (CDC)*, vol. 3, 1999, pp. 2999–3004.

[2] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Proc. IEEE Int. Conf. on Robotics and Auto. (ICRA)*. IEEE, 2014, pp. 2589–2594.

[3] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[4] Q. Nguyen, A. Agrawal, X. Da, W. C. Martin, H. Geyer, J. W. Grizzle, and K. Sreenath, "Dynamic walking on randomly-varying discrete terrain with one-step preview," in *Robotics: Science and Systems*, 2017.

[5] K. Byl and R. Tedrake, "Metastable walking machines," *I. J. Robotics Res.*, vol. 28, pp. 1040–1064, 2009.

[6] C. O. Saglam and K. Byl, "Quantifying the trade-offs between stability versus energy use for underactuated biped walking," in *Proc. IEEE/RSJ Int. Conf. on Intell. Robots and Systems (IROS)*, 2014, pp. 2550–2557.

[7] E. R. Westervelt, C. Chevallereau, J. H. Choi, B. Morris, and J. W. Grizzle, "Feedback control of dynamic bipedal robot locomotion." CRC press, 2007.

[8] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[9] G. Bellegarda, N. Talele, and K. Byl, "Exploring nonintuitive optima for dynamic locomotion," 2017.

[10] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[11] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Math. Programming Computation*, In Press, 2018.

[12] C. O. Saglam, "Tractable Quantication of Metastability for Robust Bipedal Locomotion," PhD thesis, UCSB, 2015.

[13] Wikimedia Commons, "File:fractaldimensionexample.png," 2017, [Online] https://commons.wikimedia.org/w/index.php?title=File:Fractaldimensionexample.PNG&oldid=247697185.