Through-contact Monopod Motion Generation via MPC: Convex Formulation and Locomotion Analysis

Jason White and Christian Hubicki

***Abstract*— This paper presents a convex motion planner for single-legged robots that generates behaviors in real-time without using predefined contact sequences or mode durations. Similar to existing model-predictive controllers (MPC), this method iteratively solves quadratic programs (QPs) to generate trajectories for linearized plant models over a receding planning horizon. However, the presented method also iteratively updates the predicted contact sequence with each control cycle. This allows for through-contact motion planning without smoothing contact conditions or including any nonlinear constraints. Despite the contact predictions being locally and iteratively updated, planar simulations of the controller exhibit a wide range of monopod behaviors. By only setting a desired mean speed and mean height in the optimization cost, the monopod hops with a stable limit cycle, transitions between hopping and standing within one gait cycle, and demonstrates deadbeat disturbance rejection. Analysis of the exhibited gait cycles indicates that key markers such as gait period are highly dependent on the dynamics and task, not controller hyperparameters such as the time horizon. This suggests that the design choices which render the through-contact MPC fast-solving are not impeding behaviors that are well suited to the robot's dynamics and task.**

I. Introduction

Hopping and jumping are critical locomotion modes for highly agile animals, from parkour-performing humans to arboreal monkeys to leaping goats, and have served as inspiration for legged robots for decades [1], [2]. However, leaping between contact and no-contact modes induces hybrid dynamics and underactuation which have been historical challenges for control. In the late-2010's, real-time motion planning (e.g. MPC) emerged as a powerful tool for controlling legged robots in agile locomotion regimes (e.g. trotting quadrupeds [3], jumping humanoids [4]). While these methods generate more robust locomotion to disturbances, they assume specific contact sequences and timings *a priori* to speed up their computation – this limits the range of extemporaneous behaviors they can exhibit. This paper presents a real-time MPC approach for monopod hoppers that plans through-contact, assuming no contact sequence or timings *a priori*.

Controllers and heuristics for monopods date back to the Raibert controllers [1]. These controllers specifically regulated gait quantities such as energy [5] to achieve bounding at a desired height and speed but required manual gain tuning. Later methods specifically developed control policies for canonical models, such as the spring-loaded inverted pendulum (SLIP) [6] (and variations thereof) for stable [7], [8], robust [9], [10], and deadbeat control [11]. However, designing these policies requires simulations that span the state space in a brute force manner, making policy adjustments impractical during operation. Other approaches use feedback linearization to allow for on-the-fly planning computations with specific dynamic hopping models [12].

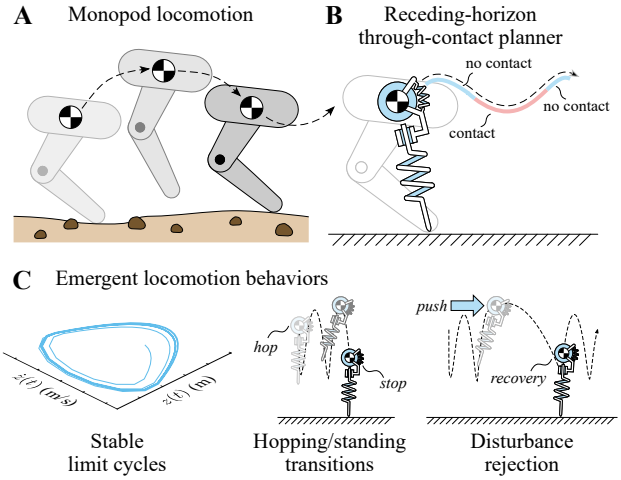


Fig. 1: **A.** An illustration of monopod locomotion. **B.** This work presents a through-contact planner for control of monopods. **C.** The controller yields canonical legged behaviors such as (1) stable limit cycles, (2) hopping/standing transitions, and (3) disturbance rejection.

Optimization-based legged control methods offer more flexibility with respect to plant models. Solving trajectory optimizations offline generates fast and stable hopping [13] when periodic gaits are enforced via constraints [14] or given sufficiently long multi-step time horizons [15]. However, these behaviors are limited to the trajectories generated prior to operation. In contrast, model-predictive control (MPC) solves these trajectory optimizations in real-time with a short and receding time horizon. MPC is responsible for highly dynamic maneuvers for many legged robot morphologies [3], [16]. However, to achieve the fast computation speeds required for real-time opeartion, contact sequences and mode duration (*i.e.* footfall timing) are assumed *a priori*.

Through-contact and contact-implicit optimizations offer the freedom to change contact sequence and timing on-the-fly. A notable example is the use of linear complementarity problems (or LCPs) to express contact as a constraint relating distance-to-contact to contact force [17]. Recent work has accelerated the solve time of complementarity-constrained trajectory optimizations [18], [19] to 15-30Hz, but methods

The authors are with the Department of Mechanical Engineering at the Florida A&M University-Florida State University College of Engineering, Tallahassee, FL

that rely on convex quadratic optimizations are considered faster and the solvers more robust [20]. Differential dynamic programming methods can achieve similar solve times to LCP methods by using invertible contact models [21]. Smoothing the contact model can accelerate through-contact motion planning [22] at some cost to dynamical accuracy while globally optimal contact plans can be found using convex (but-slower) mixed-integer approaches [23]. While quickly and reliably solving the general through-contact motion planning problem is an ongoing challenge, we posit that the dynamics of monopod locomotion are a special case which can be effectively planned through-contact with a simpler and faster algorithm.

The contributions of this work are as follows:

1) A real-time through-contact motion planner that uses the model prediction to iteratively update a mode schedule prediction
2) Planar simulations showing of the receding horizon planner's ability to achieve stable limit cycles despite lacking any periodicity constraints
3) Demonstration of standing/hopping transitions and deadbeat (single gait cycle) disturbance rejection
4) Numerical experiments showing the relative insensitivity of the gait dynamics to the choice of MPC time horizon.

II. Methods

A. Conventions and Assumptions

1) Conventions: The following notation is used throughout the paper: $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$ represent the space of real numbers, vectors with length n, and matrices with n rows and m columns. Scalar values are lowercase and italicized (e.g. $x \in \mathbb{R}$); vectors are lowercase and bold (e.g. $\mathbf{q} \in \mathbb{R}^n$); and matrices are uppercase and bold (e.g. $\mathbf{A} \in \mathbb{R}^{n\times m}$). Variables ($\mathbf{p}$, $\mathbf{v}$, $\mathbf{u}$) are positions, velocities, and control inputs respectively. Finally we define the robot state vector as $\mathbf{q} = [\mathbf{p},\ \mathbf{v}]^T$ and its derivative $\dot{\mathbf{q}} = [\dot{\mathbf{p}},\ \dot{\mathbf{v}}]^T$. For these variables and others in the following sections, notations represent a single node in the optimization unless explicitly stated with a k. Similarly, all variables presented are time varying except for the model's physical properties mass m, gravity g, spring stiffness $[k_1, k_2]$, nominal lengths $[l_o, \theta_o]$, and maximum length l_{max}.

2) Assumptions: For the models used here, we make some general simplifying assumptions. First, we assume the actuated spring model leg is massless. During stance, we also assume the actuated spring model's ground contact point is fixed. Finally the model predictions are integrated using a trapezoidal scheme but the controller is simulated using MATLAB's ode45 variable-time-step differential equation solver.

B. Through-contact Planning

Here we discuss present our approach which allows us to formulate a through-contact motion planning optimization as a convex quadratic program (QP). It is well established that linear dynamic models of legged systems, when given a constant contact sequence and mode duration, can have their motion plans generated using convex quadratic programs [24]. Our method leverages this fact by predicting a mode schedule, assuming the constant mode schedule for a single optimization, and then iteratively updating the predicted schedule between MPC optimizations. The steps are as follows and illustrated in Fig. 2.

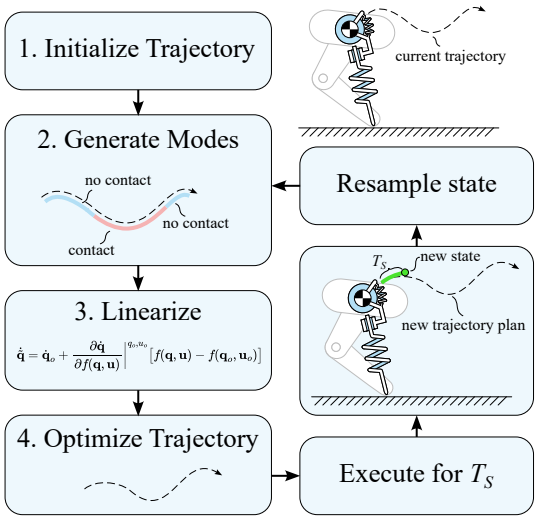


Fig. 2: Overview of the through-contact model-predictive control procedure.

1) **Initialize:** Generate a guess of the planned trajectory for a fixed time horizon.
2) **Generate Modes:** Check contact guards along planned trajectory to generate a mode schedule guess
3) **Linearize:** Linearize the plant dynamics along the trajectory given the current mode schedule
4) **Optimize:** Formulate and solve the motion planning problem over the time horizon, T, as a convex quadratic program
5) **Execute and Resample:** Enact the motion plan for sampling time, T_s, and sample the current state as the new initial condition

Utilizing an estimated planned trajectory allows us to both estimate a contact sequence and develop an accurate linearization of the dynamics. The first iteration assumes the system is stationary. Each successive iteration uses the dynamic set associated with the projected mode (stance or flight) to generate a motion plan. Changes in the contact guards are then identified after the motion plan is generated, and the mode schedule is updated accordingly. The contact guards are determined by identifying breaches in the kinematic limits assigned to each node from the dynamics. This approach will update mode changes along the motion plan, and find up to one additional mode change per iteration. The first few iterations are run without simulating the system's motion to essentially warm-start the optimization and generate a reasonable initial trajectory.

For the nonlinear monopod presented here, we identify when the toe would be above or below ground using the leg length and angle. This allows us to determine which

set of dynamics to use for each node in the optimization. These nonlinear dynamics are used to simulate the system, but are not suited for use with a convex optimization. After this determination, the states in the previously generated trajectory are used with a first order Taylor Series to linearize the dynamics.

$$\dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_o + \frac{\partial \dot{\mathbf{q}}}{\partial f(\mathbf{q},\mathbf{u})}\bigg|^{q_o,u_o} \left[f(\mathbf{q},\mathbf{u}) - f(\mathbf{q}_o,\mathbf{u}_o)\right] \tag{1}$$

where the robot state vector, its derivative, control inputs, and design vector $f(\mathbf{q},\mathbf{u})$ are evaluated at operating points $\mathbf{q}_o$, $\dot{\mathbf{q}}_o$, $\mathbf{u}_o$ to find a linear form of the dynamics $\dot{\tilde{\mathbf{q}}}$. The approach presented mimics the methods presented in [25], where the operating point values come from the previous optimal trajectory generated by the MPC.

Typically these dynamics are then used with some integration scheme to constrain the nodes to each other (as seen in Sec. II-D). When we encounter mode changes we remove the continuity constraint for the discontinuous states (i.e. the toe position and velocity), and act as though the first state of the phase has it's own set of initial conditions. Finally we optimize the problem, and simulate the nonlinear dynamics using the inputs developed from optimization. The results of the simulation then becomes our initial state, and the process starts over.

C. Dynamical Models

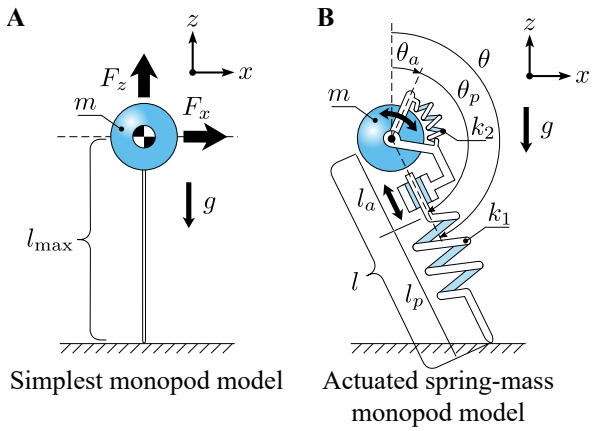
We demonstrate our through-contact MPC method using two dynamic monopod models in the sagittal plane. We first introduce a point-mass model we dub the **simplest monopod** (SM) model to clearly illustrate the method. Secondly, we introduce an **actuated spring-loaded monopod** (ASLM) model, similar to the spring-loaded inverted pendulum [6] but with rotational and prismatic actuation [10], to demonstrate extensibility to more-complex leg morphologies.

1) Simplest Monopod Model: The point mass model's state vector has two positions $\mathbf{p} = [x,z]^T$ and velocities $\mathbf{v} = [\dot{x},\dot{z}]$. The dynamics use simple Cartesian forces and gravity per

$$\dot{\mathbf{v}} = \left[\frac{F_x}{m}, \frac{F_z}{m} - g\right]^T \tag{2}$$

and assumes the control inputs $\mathbf{u} = [F_x, F_z]^T$ are zero once the point mass reaches a user defined vertical height.

2) Actuated Spring-loaded Monopod Model: The ASLM model state vector includes body positions $\mathbf{p}_b = [x_b, z_b]$, toe positions $\mathbf{p}_t = [x_t, z_t]$, and the leg actuation length & angle $\mathbf{p}_a = [l_a, \theta_a]$ for combined position vector of $\mathbf{p} = [\mathbf{p}_b, \mathbf{p}_t, \mathbf{p}_l]^T$. This model differs from the the traditional "minimalist coordinate" approach, but allows us to estimate and simulate when we expect ground contact will occur. We formulate the nonlinear dynamics for each phase (stance and flight) using a Newton Euler approach. The model has two springs, one prismatic along the leg, and a second rotational between the leg and rotational actuator. Both the leg length (l) and angle (θ) are determined using the actuator positions ($\mathbf{p}_a$) and spring deflections (l_p, θ_p)



Simplest monopod model | Actuated spring-mass monopod model

Fig. 3: Diagram of the two kinematic models used for the optimizations. **A.** The simplest monopod, showing the z based l_{max} limit which dictates what dynamic model to use. **B.** Actuated spring mass system with both a prismatic and rotational spring. The calculated l determines which set of dynamics to use.

$$l = l_a + l_p, \quad \theta = \theta_a + \theta_p \tag{3}$$

In flight, the spring's passive and nominal length are equal ($l_p = l_o$ and $\theta_p = \theta_o$), so toe positions are simply

$$\mathbf{p}_t = \mathbf{p}_b + [l\sin(\theta),\ -l\cos(\theta)]^T. \tag{4}$$

During this phase the body follows a simple ballistic trajectory, however, the toe acceleration can vary depending on the prismatic ($\ddot{l}_a$) and rotational ($\ddot{\theta}_a$) actuator accelerations.

$$\dot{\mathbf{v}}_b = [0,\ -g], \quad \dot{\mathbf{v}}_t = \frac{d^2}{dt^2}\mathbf{p}_t, \quad \dot{\mathbf{v}}_a = [\ddot{l}_a,\ \ddot{\theta}_a] \tag{5}$$

In stance, the springs deflect so the previous assumptions no longer hold true. Instead the distance between the body and toe is found via $\mathbf{r} = \mathbf{p}_t - \mathbf{p}_b$, which is then used to find the forces from spring deflections.

$$\begin{aligned} F_l &= k_1(l_o + l_a - l) \quad \textit{where} \quad l = ||\mathbf{r}||_2 \\ F_\theta &= k_2(\theta_a - \theta) \quad \textit{where} \quad \theta = \sin^{-1}\left(\frac{r_x}{l}\right) \end{aligned} \tag{6}$$

The equations of motion are determined by parsing forces into their respective cartesian directions $\left(\text{e.g. } F_{l,x} = F_l\frac{r_x}{l}\right)$.

$$\begin{aligned} \dot{\mathbf{v}}_b &= \left[\frac{1}{m}(F_{\theta,x} - F_{l,x}),\ -\frac{1}{m}(F_{\theta,z} + F_{l,z} + mg)\right], \\ \dot{\mathbf{v}}_t &= [0,\ 0], \qquad \dot{\mathbf{v}}_a = [\ddot{l}_a,\ \ddot{\theta}_a] \end{aligned} \tag{7}$$

D. Model Predictive Control

MPC utilizes online optimizations to track desired trajectories, but can also be used for online motion generation with loose (or non-existent) tracking requirements. We use it to generate trajectories (**p,v**) and associated control inputs (**u**). As is typical in MPC, we facilitate model prediction in our equality constraints using an Trapezoidal integration

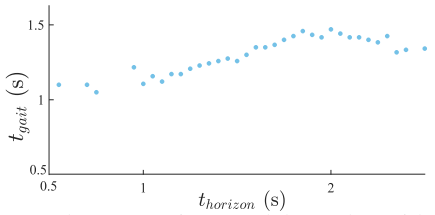


Fig. 4: Actuated mass-spring model results with a defined MPC time horizon and the corresponding gait cycle time for an unchanged objective

of our system dynamics. At each control loop, we constrain the measured states to equal the initial trajectory states ($\mathbf{q}_1$). The optimization uses a multi-part quadratic cost, and linear constraints per:

$$\begin{aligned} \min_{\mathbf{q}_k,\mathbf{u}_k} \quad & J(\mathbf{q},\mathbf{u}) \\ \text{s.t.} \quad & \dot{\mathbf{q}}_k = \mathbf{A}\mathbf{q}_k + \mathbf{B}\mathbf{u}_k && \text{(Dynamics)} \\ & \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta t && \text{(Model Prediction)} \\ & h(\mathbf{q}_k,\mathbf{u}_k) \leq 0 && \text{(Task Constraints)} \end{aligned}$$

and is detailed in later sections.

E. Cost Function

The cost function $J(\mathbf{q},\mathbf{u})$ has a corresponding weighting matrix, $\mathbf{W}$, and is specific to each task and model (J for brevity). The cost function consists of two components, the desired state error $J(\mathbf{q})$ and an actuator acceleration penalty $J(\mathbf{u})$.

$$J(\mathbf{q}) = ||\mathbf{q}_k - \mathbf{q}_{ref}||^2_{\mathbf{W}_{\text{target}}} \qquad J(\mathbf{u}) = ||\mathbf{u}_k||^2_{\mathbf{W}_{\text{actuator}}} \tag{8}$$

The spring deflection between the actuator and hopper components are what produces the forces acting on the body. The optimization problem is minimizing the actuator acceleration rather than the forces produced. The ratio of the target to actuator weight matricies are 5000:1 for the simplified model and 500:1 for the complex model respectively. These weights make the primary focus of the optimization to reach the target, while preventing unrealistic accelerations and actuator inputs.

F. Additional Constraints

The system has a leg actuation length limit, which is included as an inequality constraint to limit the leg extension. In many cases this constraint causes the system to enter a flight phase, which poses an interesting challenge for convex optimizations, and is discussed further in other sections.

III. Results

Simulations were run in MATLAB 2019b using the quadratic programming solver `quadprog` from the optimization toolbox. A MSI Trident 3 with an Intel i7-8700 processor, 16GB of ram, and a Nvidia GeForce GTX 1060 was used for all the results presented. The constraints and cost function are generated prior to the control loop using the `MatlabFunction` code generator and symbolic toolbox. After each elapsed sample period, the states are sampled, the planning optimization solved, and the planned control inputs are applied to the system dynamics–which are subsequently integrated with MATLAB's `ode45` medium-order differential equation solver.

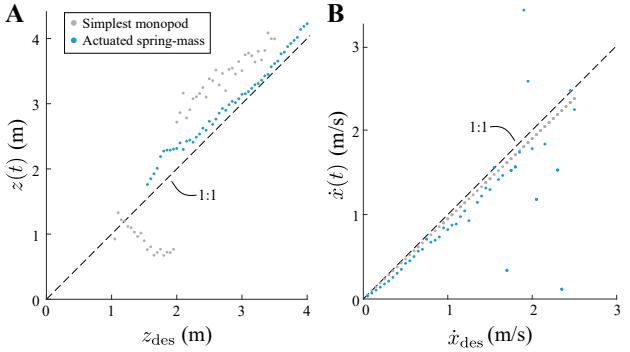


Fig. 5: Reliability testing of both models w.r.t. their objective over 10 seconds of operating time **A.** The average gait cycle peak height **B.** The average gait horizontal velocity

A. Reliability Results

The objective is for all nodes along the trajectory to reach a desired height and horizontal velocity. Three studies are run to test the accuracy of forward hopping, vertical hopping, and the effect of increasing the model prediction time horizon. The first two tests utilize a time horizon of 1.5sec with a node to node δt and simulated sample time of 50ms which is greater than the loop computation time of $\sim$27ms.

1) Variable Desired Velocity: Forward velocities from 0.1m/s to 2.5m/s are added to the objective function at a constant hop height of 2m to test the controllers ability to match a single desired hop velocity across all nodes. Excluding the outliers (detailed in the discussion) the velocities generally correspond to the intended speeds.

2) Variable Desired Height: Similar to the last analysis, we try hopping in place. Both models use a desired height of 1.6 to 4.0 meters with no horizontal velocity. In many cases we find that the optimal solution for the predictive horizon overshoots the desired state rather than reaching it exactly. Speculation of why this occurs is included in the discussion.

3) Variable Time Horizon: For the final analysis we use a desired height of 2m and forward velocity of 0.5m/s with a variable time horizon. For consistency, this study keeps the δt constant at 50ms and varies the number of nodes as the horizon increases. The goal of this study is to investigate the gait behavior using a constant task but a changing horizon changes. We notice some variance, but in general changing the time horizon does not have a large impact on gait cycle.

B. Transient Behaviors

Further investigating the gait behavior, we look at some of the phase plots developed with the aforementioned desired properties. We can see models start in flight with zero initial velocity, and quickly converge to a stable limit cycle near the desired speed and mean hop height.

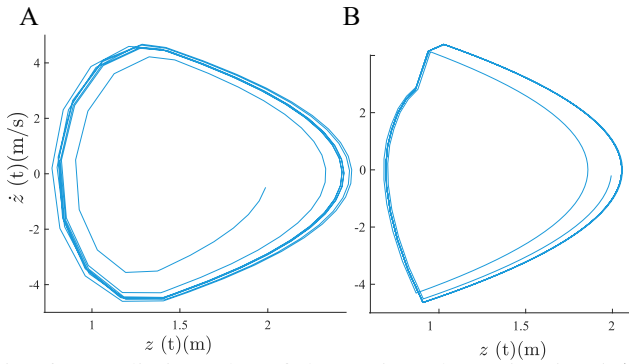


Fig. 6: Two limit cycles of data points shown in Fig. 5 **A.** Simulation of the actuated spring-mass model **B.** Simulation of the simplest monopod

1) Disturbance Rejection: To test the abilities of the controller we subject the model to a 2m/s disturbance while in flight. The leg almost instantaneous adjusts, and the systems recovers within a single stance phase to return back to its steady state.

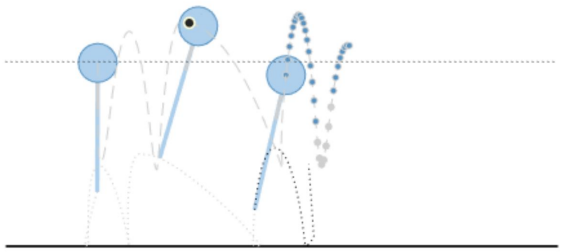


Fig. 7: Simulated disturbance and reaction of the actuated spring mass model to an instant velocity increase of 2m/s.

2) Change Directions: We further test the controller by changing the objective in the middle of a gait. While hopping forward, we reverse the desired velocity and introduce a new desired hop height. Immediately the generated gait changes, and the leg adjusts to match the new desired states. Once again the controller reacts within the first step to a stable cycle near the desired properties.

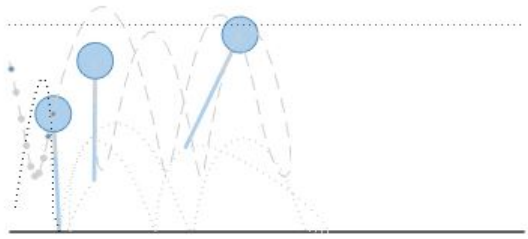


Fig. 8: Systems reaction to a changed objective mid-gait to hop in the opposite direction at a faster pace and different height

.

3) Stop in place: Finally we modify the desired height mid cycle to a length less than l_{max} with a horizontal velocity of zero. Almost immediately the ASLM model stops hopping and compensates to reduced the oscillation from the spring allowing it to remain at a stationary point.

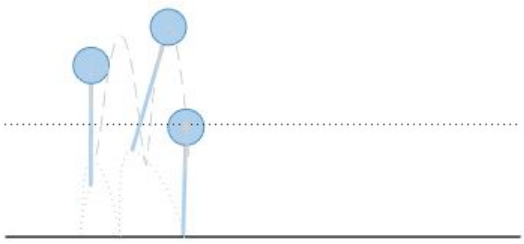


Fig. 9: A hop-to-stand transition as a consequence of changing the desired height below l_{max}.

IV. Discussion

The nonlinear ASLM model formulation utilizes actuator deflections thereby generating spring forces which are used for control. Although optimizations typically minimize control input, this model's cost function is minimized by reducing the actuator acceleration. This leads the to trudging style gaits observed throughout the paper. By keeping the actuator nearly stationary, the system is able to get both a vertical and horizontal forces while in stance with minimal cost.

Although we use 31 nodes throughout the reliability and reactivity tests, stable hopping has been maintained with as few as 5 nodes. As the nodes are reduced, the estimated trajectory becomes increasingly inaccurate leading to an obvious deviation from both the desired vs actual height (overshooting), and velocity (undershooting). The overshooting associated with vertical hopping is expected, as the optimization attempts to minimize the deviation between all nodes and the desired height, not the trajectory peak height. The velocity based undershooting observed is likely a consequence of the multipart objective the system attempts to achieve (a forward velocity, vertical height, and minimal actuator motion).

A. Limitations

As is the case with many research topics, complications were present within specific instances in the simulations. For one, if the desired height used in the optimization is less than the max leg length, the inputs developed from the MPC will never force a flight phase. The system will instead "stretch" to the furthest point possible, and sway back and forth. This happens because the model never applies enough force to cause a flight phase and thus the model never utilizes the flight dynamics to achieve a more optimal solution. This behavior propagates to the multi legged extension we formulated, where only one leg is utilized. For the single leg model, all forces are generated from one source, because of this the optimizer realizes that this sole source will be accessible when the foot hits the ground. When we introduce a second leg, one leg is picked up, and never used again (i.e. it hops around on one leg). Additionally in some instances, the simulation becomes unstable. This happens a few times above a desired speed of 1.5m/s, and always at speeds $>$ 2.5m/s (as seen in Fig.5B. Finally, instances of "procrastination" of the MPC are observed in some of the

simplest model results. This is where a trajectory is planned utilizing both stance and flight, however the control input extracted from the optimization never enforces this behavior. In term what happens is the model instead either "slides" along the horizon or barely hops off the ground, as seen in Fig. 5A around the 1.5m desired height.

B. Future Work

Future work will extend and validate this method in 3D hopping. Further, we suspect that the method should be effective for multibody monopods with distal mass and body pitch. However, we anticipate that allow torso rotations may require incorporating additional terms in the cost function to promote a level torso. Additionally, future work will explore through-slip conditions and assess the ability for the presented through-contact MPC to generate stable gaits both with and without a sliding foot.

V. Conclusion

We presented a convex motion planner for single-legged robots that generates behaviors without using predefined contact sequences or mode durations in real-time. Overall the results here show the method can control monopod locomotion without a predefined motion plan while keeping a problem convex. In planar simulations, we show that by only changing the desired states, stable limit cycles are developed as an emergent property of the controller. Additionally the results indicate that limit cycles have limited dependency on the defined time horizon but are strongly shaped by the dynamics and task, suggesting that the MPC framework is flexible to the needs of dynamic monopods.

Acknowledgment

This work was supported by the Toyota Research Institute and the DEVCOM Army Research Laboratory under Cooperative Agreement Number W911NF-16-2-0008. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[2] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via series-elastic power modulation," *Science Robotics*, vol. 1, no. 1, p. eaag2048, 2016.

[3] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[4] K. Ishihara, T. D. Itoh, and J. Morimoto, "Full-body optimal control toward versatile and agile behaviors in a humanoid robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 119–126, 2019.

[5] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the arl-monopod ii," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 974–986, 2006.

[6] W. J. Schwind, *Spring loaded inverted pendulum running: A plant model*. University of Michigan, 1998.

[7] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1779–1793, 2009.

[8] B. Andrews, B. Miller, J. Schmitt, and J. E. Clark, "Running over unknown rough terrain with a one-legged planar robot," *Bioinspiration & biomimetics*, vol. 6, no. 2, p. 026009, 2011.

[9] D. Koepl and J. Hurst, "Impulse control for planar spring-mass running," *Journal of Intelligent & Robotic Systems*, vol. 74, pp. 589–603, 2014.

[10] J. Van Why, C. Hubicki, M. Jones, M. Daley, and J. Hurst, "Running into a trap: numerical design of task-optimal preflex behaviors for delayed disturbance responses," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2537–2542.

[11] A. Wu and H. Geyer, "The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[12] G. Piovan and K. Byl, "Approximation and control of the slip model dynamics via partial feedback linearization and two-element leg actuation strategy," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 399–412, 2016.

[13] W. Gao, C. Young, J. Nicholson, C. Hubicki, and J. Clark, "Fast, versatile, and open-loop stable running behaviors with proprioceptive-only sensing using model-based optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 483–489.

[14] W. Xi, Y. Yesilevskiy, and C. D. Remy, "Selecting gaits for economical locomotion of legged robots," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1140–1154, 2016.

[15] C. Hubicki, M. Jones, M. Daley, and J. Hurst, "Do limit cycles matter in the long run? stable orbits and sliding-mass dynamics emerge in task-optimal locomotion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5113–5120.

[16] J. Shen and D. Hong, "Convex model predictive control of single rigid body model on so (3) for versatile dynamic legged motions," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6586–6592.

[17] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[18] S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control," *arXiv preprint arXiv:2107.05616*, 2021.

[19] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3414–3421.

[20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[21] I. Chatzinikolaidis and Z. Li, "Trajectory optimization of contact-rich motions using implicit differential dynamic programming," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2626–2633, 2021.

[22] A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2447–2453.

[23] Y. Ding, C. Li, and H.-W. Park, "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3998–4005.

[24] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6724–6731.

[25] J. White, D. Jay, T. Wang, and C. Hubicki, "Avoiding dynamic obstacles with real-time motion planning using quadratic programming for varied locomotion modes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 626–13 633.