Model Predictive Control of a Monopod Hopper with Reaction Wheels

Benjamin Bokser[1]

***Abstract*— In this paper, we present a floating-body model predictive control framework for the REx Hopper, an RWA-controlled monopodal hopping robot designed to perform highly dynamic leaping maneuvers with inertial reorientation. To demonstrate its effectiveness, this controller is tested successfully on several hopping reference trajectories.**

I. Introduction

Achieving highly acrobatic planned motion with a robot that has many degrees of freedom often requires a choice between the use of a full body dynamics model and a simplified, lumped-mass dynamics model. While a full body dynamics model will usually provide more accurate prediction, its nonlinearity often makes it impractical for real-time applications. Furthermore, the increase in state dimension greatly increases computation time. On the other hand, many successful approaches to dynamic robot motion in recent years, such as the Mini Cheetah [1], Hume [2], and Cassie [3], and have made extensive use of simplified, lumped mass dynamics models with model predictive control. However, these solutions were only able to demonstrate the tracking of simple center of mass (CoM) reference trajectories such as straight lines. For more dynamic behavior, such as performing backflips, Katz et al. used offline nonlinear optimization with a more complex dynamics model [4]. This is because the use of simplified, lumped mass dynamics can introduce inaccuracies in a robot with complex inertial behavior.

Therefore, it is advantageous in the design of highly dynamic robots to minimize complexity and inertial behavior while maintaining controllability, whether a full body dynamics model or a simplified dynamics model is used. This makes a significant case for the use of reaction wheel actuators (RWAs) in acrobatic robots, as a momentum control system composed of RWAs can provide precise internal torque control of a legged robot without altering the total inertia dyadic of the system [5].

As such, we present the REx Hopper (Fig. 1), a monopodal hopping robot with reaction wheel actuators designed to perform highly dynamic leaping maneuvers with inertial reorientation. The REx Hopper will serve as a testbed for state-of-the-art control algorithms, including contact-implicit model predictive control [6].

In this paper, we aim to demonstrate the use of MPC with a lumped-mass dynamics model to allow the REx Hopper to track a sinusoidal hopping trajectory. Section II provides a brief description of the robot system. Section III provides the control framework. Section IV explores the model predictive controller. And Section V contains a discussion of the results.

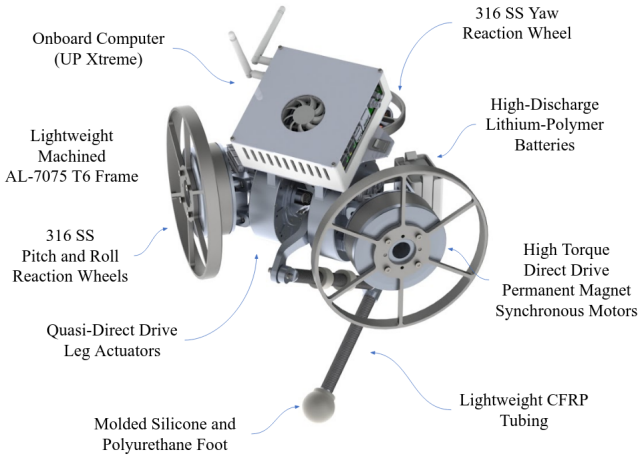

Fig. 1: Rendering of the REx Hopper.

II. System Overview

The REx Hopper has been designed with the specific intent of performing highly dynamic leaping maneuvers. As such, the design prioritizes maximum flight time per unit mass, mid-air maneuverability, impact mitigation, and force proprioception.

In its default state, the REx Hopper's Center of Mass (CoM) rests approximately 0.3 m off the ground and weighs approximately 7.5 kg in total. The vast majority of this mass is concentrated in the body for several reasons. Firstly, it is important to minimize distal leg mass in order to better approximate a lumped mass dynamics model. Secondly, minimizing distal leg mass provides optimal actuation bandwidth and impact mitigation [7]. Parallel actuation of the leg is used to increase maximum output force, and the use of a five-bar linkage allows the leg to utilize a parallel spring mechanism which improves efficiency, impact mitigation, and force output. The dynamics of the spring are not investigated in this paper.

As a general rule, smaller robots perform better than larger robots in terms of relative jump height and impact mitigation due to significantly reduced mass. However, design requirements for this project bounded the REx Hopper's minimum size to future-proof the robot for larger-scale applications. In addition, available electronics, actuators, and other components, as well as design complexity, limit the practical size of the robot.

[1]B. Bokser is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213 USA bbokser at andrew.cmu.edu

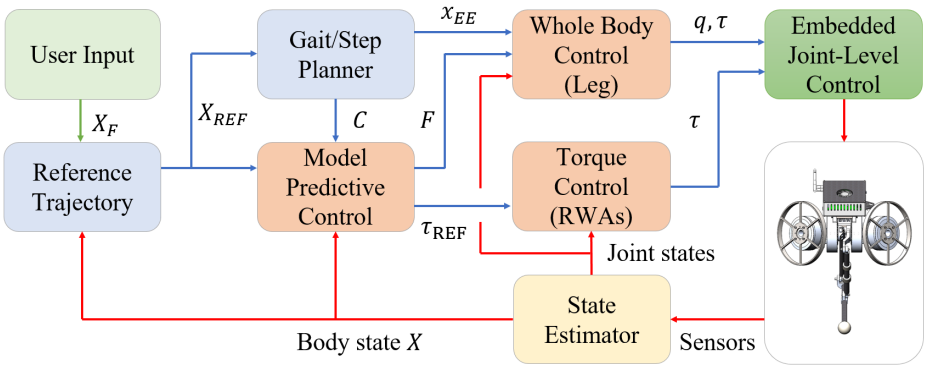

Fig. 2: Control Framework.

A. Actuator Design

High torque density electric motors with quasi-direct drive (QDD) actuation have taken the forefront in dynamic robotic actuation over the past decade due to their transparency, backdrivability, and force proprioception capabilities [8]. The REx Hopper takes full advantage of this design paradigm for leg actuation. Both leg actuators are composed of permanent magnet synchronous outrunner motors with large air gap radii connected to single stage planetary gearing with a 1:7 reduction.

B. Actuator Model

For the purposes of simulation fidelity, it is important to accurately represent the physical behavior of the motors and gearing through the use of an actuator model. An electric motor's torque saturates as a function of its speed. As described in [9], equations (1) through (3) below are used to saturate the torque output of each actuator given a current input and the motor's angular velocity at timestep k:

$$\tau_k = k_t * i_k \tag{1}$$

$$\tau_{k,max} = -\omega_k(k_t^2)/R + V_{max} * k_t/R \tag{2}$$

$$-\tau_{k,max} \leq \tau_k \leq \tau_{k,max} \tag{3}$$

where i is the current input, τ is motor torque, k_t is the torque constant, R is armature resistance, $\tau_{k,max}$ is the maximum torque at the current timestep, ω is motor speed in radians, and V_{max} is maximum voltage.

III. Control Framework

The hybrid control framework is illustrated in (Fig. 2). The model predictive controller computes linear and angular impulses based on its lumped mass dynamics formulation to match the reference trajectory. During the stance phase, the operational space controller converts linear impulses to leg actuator torques, and during the flight phase it tracks the world frame position of the next footstep setpoint. The reaction wheel controller converts angular impulses to reaction wheel torques in both the stance and flight phases.

A. Gait Scheduling

The REx Hopper gait is characterized by two phases: stance and flight. A "contact map", C, encodes the sequence of planned phases across each timestep in the planned trajectory. C is generated during the initialization routine using the following formula as presented in [10]:

$$\phi_k = \text{mod}(\frac{t - t_0}{T}, 1) \tag{4}$$

$$C_k = \begin{cases} 0 & \phi_k > \phi_{\text{switch}} \quad \text{(Flight)} \\ 1 & \phi_k \leq \phi_{\text{switch}} \quad \text{(Stance)} \end{cases} \tag{5}$$

Where the planned gait period T and the switching phase ϕ_{switch} are constants which must be manually tuned. Based on experimental results with Raibert hopping, a gait period of 0.8 seconds and a switching phase of 0.5 were found to work well.

During initialization, t_0 is chosen depending on where in the gait cycle the reference trajectory should start. Usually, to start from a standing position requires starting the gait cycle halfway through the stance phase, so $t_0 = \frac{1}{2}T\phi_{\text{switch}}$.

B. Reference Trajectory

During initialization, a reference trajectory X_{REF} of the state at each timestep is generated. In this paper, we explore reference trajectories generated using a starting state X_0 and a final state X_F. Changes in position are interpolated between X_0 and X_F.

The quality of the reference trajectory is critical to the performance of the model predictive controller. It should be as close to dynamically feasible as possible. Because the robot is a hopper, its CoM moves up and down in a sinusoidal pattern. As such, it is necessary to generate a sinusoidal trajectory for the z-axis position that matches the hopper's natural behavior.

$$z_k = h + A\sin(\frac{2\pi}{T}k\Delta t + \phi_{\text{shift}}) \tag{6}$$

Where h is the starting height, A is the amplitude, and ϕ_{shift} is the phase shift. Again, these constants must be experimentally or heuristically chosen. A good heuristic for the amplitude was found to be $A = \frac{T}{4}$. This works well for

a range of gait periods. The phase shift should be equal to $\frac{3\pi}{2}$ given that the hopper starts at the bottom of the trough. Note that it is important for the reference trajectory to match the contact map. This should occur naturally given the use of the same T as well as a t_0 and ϕ_{shift} that match each other.

As a final step in the formation of the reference trajectory, velocities are calculated based on the changes in position. Reference attitudes are left constant in the scope of this paper, although more complex reference trajectories with attitude changes will be tested in the future.

C. Footstep Planning

Planned footstep locations are based directly on the reference trajectory: one footstep location is placed underneath each trough of the sine wave. It was experimentally found that placing footsteps slightly behind the local minima results in the best performance. This works well for trajectories that travel straight forward, but a more complex footstep planning scheme will be necessary in the future.

An array of footstep positions is generated such that each timestep k corresponds to the appropriate footstep position in the reference trajectory. New footstep locations correspond to the start of the flight phase in each gait cycle. This allows the flight phase leg position controller to begin re-orienting the leg toward the next footstep position mid-flight.

D. Gait and Footstep Adaptation

When contact is made either early or late, it is advantageous to update the contact map to reflect the new gait timing. This prevents error from compounding further by ensuring that the appropriate amount of time is allotted for the next stance phase.

In addition, the footstep position must be updated when contact is made, changing from the planned footstep location to the actual footstep location. This is critical because the centroidal MPC receives footstep position as an input.

It is notable that while the contact map and footstep position are updated every time the robot makes contact with the ground, the reference trajectory is not. It is likely that better performance can be achieved by updating the reference trajectory as well, and this will be investigated in the future.

E. Task Space Leg Control

During the stance phase, desired ground reaction forces are translated into joint torques using

$$\tau = J^\top R^\top f \tag{7}$$

where J is the forward kinematic jacobian, R is the rotation matrix from the world frame to the body frame, and f is a vector of world frame forces generated by the MPC. During the flight phase, the end effector (foot) is commanded to point in the direction of the next footstep location using the swing leg control described in [11].

IV. Model Predictive Control

Model predictive control was explored with two different lumped mass dynamics formulations: a point mass model and a centroidal model.

A. Point Mass Dynamics

By offloading the attitude control task to a low-level controller, it is theoretically possible to remove attitude from the MPC dynamics through the use of a point-mass dynamics model–and as a result, vastly improve computational speed. This made possible by the three-axis reaction wheel actuator array, which can theoretically provide full control over the robot's attitude.

The continuous-time point mass dynamics are shown below in standard linear time-invariant form. In this formulation, the robot is approximated as a single point mass body subject to gravity and capable of exerting linear impulses.

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\top \tag{8}$$

$$U = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^\top \tag{9}$$

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -g \end{bmatrix}^\top \tag{10}$$

$$A = \begin{bmatrix} 0_3 & 1_3 \\ 0_3 & 0_3 \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 0_3 \\ 1_3 \frac{1}{m} \end{bmatrix} \tag{12}$$

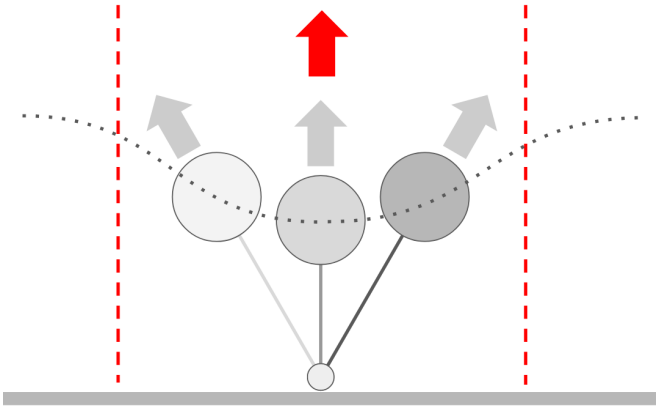

Fig. 3: Comparison of the net impulse (red) calculated by the point mass MPC and the force vectors (gray) outputted by the leg as the CoM passes over the footstep position.

Due to the linear time-invariant nature of its dynamics, the timestep size is only limited by the hybrid nature of the point mass MPC. As such, each timestep corresponds to either a stance or flight phase, vastly improving computational efficiency. As shown in Fig. 3, this means that the point-mass MPC only calculates net impulses over an entire stance phase, ignoring footstep position. This is an important consideration to note, because the force output vector of a hopper generally rotates as its center of mass (CoM) passes over the footstep location. As such, the task of converting these net impulses to footstep force vectors is left completely to a low-level controller.

The MPC is formulated as a convex QP using the point mass dynamics as a constraint, as shown in (9) through (17). The dynamics are discretized using the matrix exponential as shown in [12]. In addition, the horizontal output forces are limited by linearized friction cone constraints during the stance phase, and vertical force is limited by a user-specified maximum output force. During the flight phase, all output forces are constrained to zero.

$$\min_{X,U} \quad \sum_{k=0}^{N} \|X_k - X_{REF,k}\|_Q + \|U_k - U_{REF,k}\|_R \tag{13}$$

$$\text{s.t.} \quad X_{k+1} = A_d X_k + B_d U_k + G_d \qquad \forall k \tag{14}$$

$$0 \leq f_{z,k} \leq f_{z,max} \qquad \forall k \in \mathcal{C} \tag{15}$$

$$-\mu f_{z,k} \leq f_{x,k} \leq \mu f_{z,k} \qquad \forall k \in \mathcal{C} \tag{16}$$

$$-\mu f_{z,k} \leq f_{y,k} \leq \mu f_{z,k} \qquad \forall k \in \mathcal{C} \tag{17}$$

$$f_{x,k} = 0 \qquad \forall k \notin \mathcal{C} \tag{18}$$

$$f_{y,k} = 0 \qquad \forall k \notin \mathcal{C} \tag{19}$$

$$f_{z,k} = 0 \qquad \forall k \notin \mathcal{C} \tag{20}$$

B. Centroidal Dynamics

While the point mass formulation has the advantage of computational speed, planning with attitude can be highly advantageous for a dynamic robot. This motivates the use of the centroidal dynamics formulation shown in equations (22) through (27), which accounts for attitude and rotational dynamics.

Unlike the point mass formulation, here the control forces are calculated in the body frame. This is required because the REx Hopper leg is planar, with motion in the body frame's x- and z-axes only. This allows ${}^B f_y$ to be constrained or removed accordingly. The control torques are also calculated in the body frame in order to limit them based on the torque limits of the RWAs.

The rotation matrix $R_z(\psi)$ used here is a simplification of the rotation matrix translating angles in the world frame to the body frame as detailed in [11].

$$X = \begin{bmatrix} {}^W P_{3\times 1} \\ {}^W \Theta_{3\times 1} \\ {}^W \dot{P}_{3\times 1} \\ {}^W \omega_{3\times 1} \end{bmatrix} \tag{21}$$

$$U = \begin{bmatrix} {}^B f_{3\times 1} \\ {}^B \tau_{3\times 1} \end{bmatrix} \tag{22}$$

$$G = \begin{bmatrix} 0_{3\times 1} \\ 0_{3\times 1} \\ {}^W g_{3\times 1} \\ 0_{3\times 1} \end{bmatrix} \tag{23}$$

$$A(\psi) = \begin{bmatrix} 0_3 & 0_3 & 1_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & R_z(\psi) \\ 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix} \tag{24}$$

$$B(\psi, r) = \begin{bmatrix} 0_3 & 0_3 \\ 0_3 & 0_3 \\ \frac{1}{m} R_z(\psi)^\top & 0_3 \\ {}^W I^{-1} R_z(\psi)^\top [{}^B r]_\times & {}^W I^{-1} R_z(\psi)^\top \end{bmatrix} \tag{25}$$

$${}^W I^{-1} = R_z(\psi) {}^B I^{-1} R_z(\psi)^\top \tag{26}$$

Where ${}^B I$ and ${}^W I$ are the inertia matrix in the body frame and the inertia matrix in the world frame, respectively.

The full optimization problem is as shown below.

$$\min_{X,U} \quad \sum_{k=0}^{N} \|X_k - X_{REF,k}\|_Q + \|U_k - U_{REF,k}\|_R \tag{27}$$

$$\text{s.t.} \quad X_{k+1} = A_k X_k + B_k U_k + G_d \qquad \forall k \tag{28}$$

$$0 \leq f_{z,k} \leq f_{z,max} \qquad \forall k \in \mathcal{C} \tag{29}$$

$$f_{z,k} = 0 \qquad \forall k \notin \mathcal{C} \tag{30}$$

$$f_{y,k} = 0 \qquad \forall k \tag{31}$$

$$-\mu f_{z,k} \leq f_{x,k} \leq \mu f_{z,k} \qquad \forall k \tag{32}$$

$$-\tau_{max} \leq \tau_k \leq \tau_{max} \qquad \forall k \tag{33}$$

It is critical to note that the dynamic system used here is by default nonlinear: $A(\psi)$ is a function of the yaw angle ψ, and $B(\psi, r)$ is a function of both ψ and the vector from the footstep location to the CoM, r. However, the system can be approximated as linear time-varying through sequential quadratic programming: at the beginning of each MPC run, the vector of states generated by the previous solve is shifted forward in time and concatenated with the new initial state X_0 to form a guess X_{guess} for the new optimization run. This X_{guess}, along with an array of planned footstep positions generated by the gait planner, provides the r_k and ψ_k predictions necessary to build a the predicted A_k and B_k matrices across the entire MPC horizon. On the first run, X_{guess} is generated using the reference trajectory. Here, A, B, and G are discretized using the forward Euler method for computational speed.

V. Results

The controllers were implemented in Python, and both MPC dynamics formulations were first tested in simplified simulation environments. The full control framework and MPC were then tested on a full simulation of the robot in PyBullet at 1 kHz (See Fig 5).

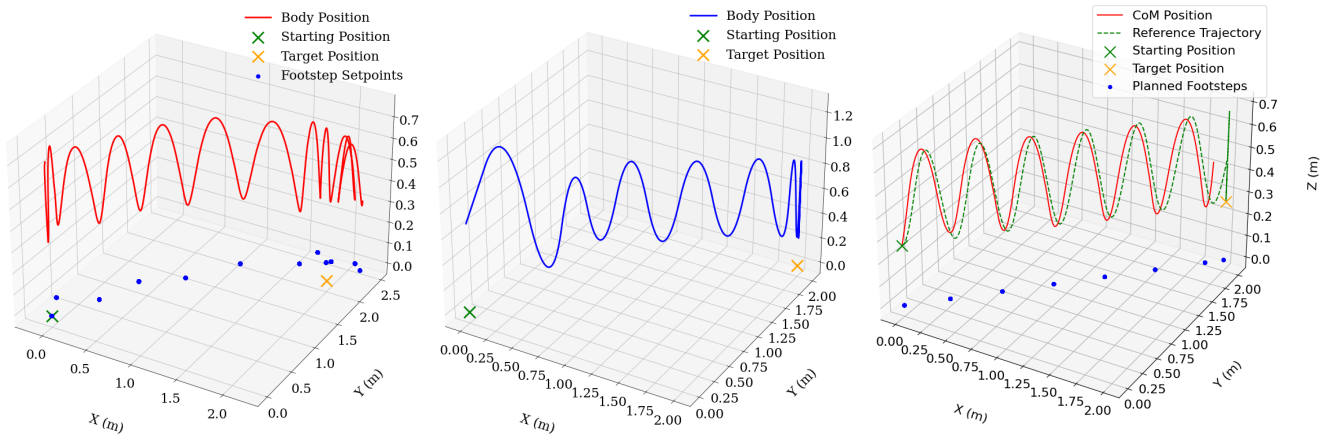

Fig. 4: (Left) The Raibert hopping trajectory simulated in PyBullet. (Middle) The point mass simulation of the hopper with point mass MPC. (Right) The SE(3) simulation of the hopper with centroidal MPC.

Raibert heuristic-based hopping control [13] was used as a basis of comparison against the MPC formulations. As shown in Fig. 4, Raibert hopping was successfully used to command the robot to move from (x, y) position (0, 0) to position (2, 2). While implementation was relatively simple, Raibert hopping requires significant manual tuning and returns poor accuracy. In addition, Raibert hopping is limited to force output in the body frame z-axis and is unable to make use of the REx Hopper's x-axis force output. This results in greater attitude modulation to perform controlled hops with specific velocity vectors, which requires greater energy consumption from the RWAs. In addition, Raibert hopping does not generalize well for following arbitrary dynamic reference trajectories, as it requires manual re-tuning for even small modifications.

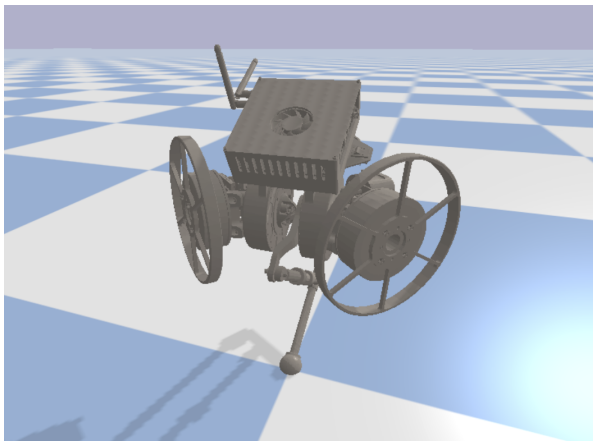

Fig. 5: Visualization of the PyBullet simulation.

A. *Point Mass MPC Results*

The point mass MPC was first simulated using Equations (8) through (12) with classic Runge-Kutta integration. The results are similar to those obtained with Raibert hopping, as shown in Fig. 4.

However, we encountered issues in the application of the point mass MPC to the full simulation and control framework in PyBullet. As previously mentioned, the point mass MPC only generates a net impulse over the stance phase and the low-level controller is left with the task of converting that net impulse into a series of force vectors for each low-level timestep. However, the operational space controller currently uses a fixed-base formulation, and does not consider the dynamical behavior of the robot base or the use of the RWAs. Attempting to track a world frame footstep position while applying force using this operational space controller results in body torques which the RWA controller must reactively counter to maintain the correct body attitude. In practice, the RWA controller was unable to generate the correct torque profile.

B. *Centroidal MPC Results*

The centroidal MPC was first simulated with the SE(3) dynamics presented in [14] and classic Runge-Kutta integration. The MPC uses a horizon of 0.8 seconds (equal to the gait period) divided into 40 steps of size 0.02 seconds. Results for hopping straight forward in the simplified simulation are as shown in Fig. 4. The centroidal MPC is also able to follow a curved reference trajectory, but only if the y-axis body rotation constraint is removed. It is notable that the current computational speed of the Python controller and simulation is significantly slower than real-time.

Fig. 6 shows results for full REx Hopper simulation in PyBullet with centroidal MPC attempting to track a sinusoidal reference trajectory from position (0, 0, 0.35) to position (1, 0, 0.35) over a timespan of 2 seconds. As you can see, significant drift in the CoM and footstep position over a short period of time causes the hopper to diverge from the reference trajectory.

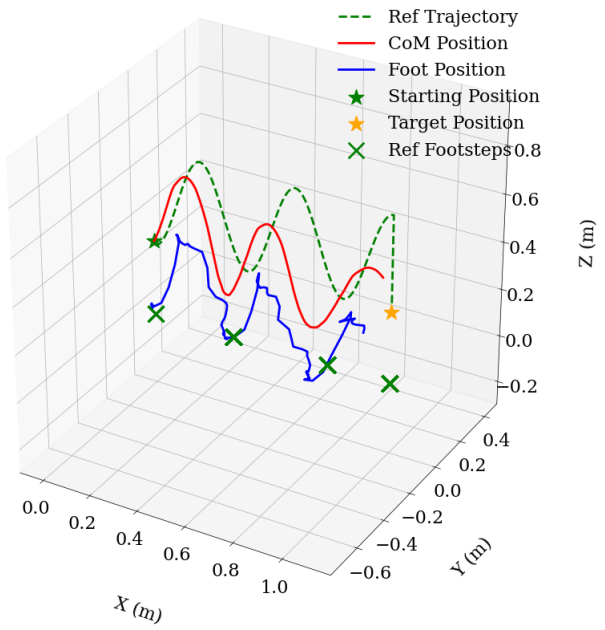

Fig. 6: Trajectory of the REx hopper in PyBullet simulation using MPC with centroidal dynamics.

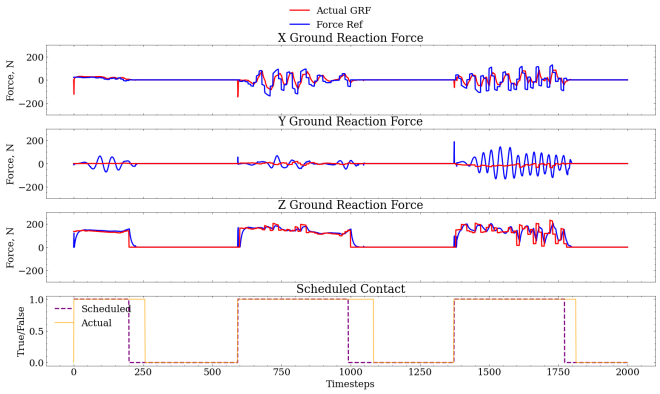

Fig. 7: Ground reaction forces compared to reference force over time in the world frame.

VI. Conclusions and Future Work

In this work, we introduced a promising model predictive control framework for a monopodal hopping robot using lumped mass dynamics. In full simulations of the REx Hopper, our MPC successfully demonstrated forward hopping control following a short sinusoidal reference trajectory.

Our immediate next steps will be to troubleshoot accuracy issues and to experiment with adaptive footstep location and adaptive reference trajectory planning. It is likely that attitude modulation will need to be included in the adaptive reference trajectory generator to allow for side-to-side hopping. Another helpful improvement would be the addition of an angular momentum state to the dynamics to minimize flywheel speed.

While the centroidal MPC showed better performance in the experiments conducted here, there still exist potential improvements to the point mass MPC formulation (such as the use of whole body control in the world frame) which may be explored if computational cost is enough of a concern.

In future work, we plan to implement the contact-implicit MPC with maximal coordinates and variational integration described in [6]. In addition, the code will be rewritten in C++ and optimized for computational efficiency in order to ensure that it is viable for real-time control on the hardware.

The source code for this project is available in the following repositories.

- https://github.com/bbokser/hopper-mpc-point
- https://github.com/bbokser/hopper-mpc-inertial
- https://github.com/RoboticExplorationLab/rex-hopper-python

Acknowledgment

I would like to thank Dr. Zachary Manchester (Carnegie Mellon University) for the generous advice he provided on this project.

References

[1] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[2] D. Kim, Y. Zhao, G. Thomas, and L. Sentis, "Assessing whole-body operational space control in a point-foot series elastic biped: Balance on split terrain and undirected walking," *arXiv preprint arXiv:1501.02855*, 2015.

[3] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot cassie.," in *Robotics: Science and Systems*, vol. 101, p. 14, 2018.

[4] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6295–6301, IEEE, 2019.

[5] F. A. Leve, B. J. Hamilton, and M. A. Peck, *Spacecraft momentum control systems*, vol. 1010. Springer, 2015.

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

[7] A. Abate, J. W. Hurst, and R. L. Hatton, "Mechanical antagonism in legged robots.," in *Robotics: Science and Systems*, vol. 6, Ann Arbor, MI, 2016.

[8] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *Ieee transactions on robotics*, vol. 33, no. 3, pp. 509–522, 2017.

[9] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[10] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4399–4406, IEEE, 2018.

[11] 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)*, pp. 1–9, IEEE, 2018.

[12] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM review*, vol. 20, no. 4, pp. 801–836, 1978.

[13] M. H. Raibert, H. B. Brown Jr, and M. Chepponis, "Experiments in balance with a 3d one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[14] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5658–5664, 2021.