REX HOPPER: DESIGN AND CONTROL OF A MONOPOD HOPPER WITH REACTION WHEELS

Benjamin Bokser
Graduate Research Assistant
Robotic Exploration Laboratory
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: bbokser@andrew.cmu.edu

Zachary Manchester
Assistant Professor
Robotic Exploration Laboratory
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: zmanches@andrew.cmu.edu

ABSTRACT

In this paper, we present a novel monopodal hopping robot through a systematic overview of the robot's hardware, software, and control framework. The REx Hopper is a reaction wheel-controlled monopodal hopping robot designed to perform highly dynamic leaping maneuvers with inertial reorientation. The design takes advantage of reaction wheels to minimize complexity and inertial behavior while maintaining controllability, and carries a powerful onboard computer to run expensive control algorithms. We present a hybrid floating-body model predictive control framework which allows the robot to track sinusoidal hopping reference trajectories online using either point mass dynamics or centroidal dynamics. This approach is compared to a more conventional Raibert hopping method. Through our systems approach to hardware, software, and control design, we achieve successful dynamic behavior on several hopping reference trajectories in simulation.

NOMENCLATURE

α	Angular acceleration
Δt	Timestep size (seconds)
$\mathcal{A}$	Amplitude (meters)
$\mathcal{J}$	Polar moment of inertia
μ	Coefficient of friction
ω	Motor speed in (radians/s)
ϕ	Gait phase
ϕ_{shift}	Phase shift
ϕ_{switch}	Gait switching phase
ψ	Yaw angle
τ	Motor torque (Nm)
A	State matrix
B	Input matrix
C	Contact map
f	Force vector
G	Gravity vector
h	Starting height (meters)
I	Inertia matrix
i	Current (A)
J	Forward kinematic Jacobian matrix
k	Timestep
K_T	Motor torque constant (Nm/A)
m	Body mass (kg)
P	Power (W)
R	Rotation matrix
r	Armature resistance (Ohms)
T	Gait period (seconds)
t	Time (seconds)
U	Control vector
V	Voltage (V)
X	State vector
AHRS	Attitude and Heading Reference System
CFRP	Carbon Fiber Reinforced Polymer
CoM	Center of Mass
IMU	Inertial Measurement Unit
MPC	Model Predictive Control/Controller
PID	Proportional-Integral-Derivative

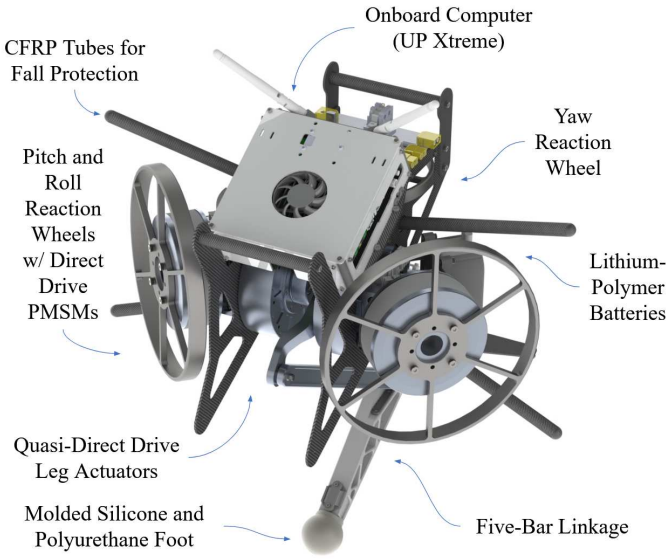

FIGURE 1: Annotated rendering of the REx Hopper

1 INTRODUCTION

Thanks to a combination of improved actuation and control methods, great advances are being made in the field of legged robots with quadrupedal and bipedal morphologies. As the complexity of these robots increases and the goalposts for dynamic behavior shift, the choice of dynamics model becomes increasingly important. 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 are limited in their ability to track complex center of mass (CoM) reference trajectories. 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, regardless of whether a full body or simplified dynamics model is used, complex inertial behavior is an obstacle in the path of an accurate and robust control system.

In the face of such a challenge, it is useful to consider the problem not just from a controls perspective but also a design perspective. One way to drastically reduce complex inertial behavior is to use a momentum control system composed of reaction wheels, as reaction wheels can provide precise internal torque control without altering the total inertia dyadic of the system [5]. And with the use of reaction wheels to handle internal torque control, it becomes possible to reduce the inertial complexity of the system further by using only one leg to traverse a 3-dimensional environment.

Many monopod jumping robots have been developed over the years, but few are designed to confront the state of the art in dynamical controls. Of the hoppers capable of continuous athletic hopping, a large portion of them, such as [6], are constrained to a 2D plane and require external support or a tether. Salto-1P [7], the 3D One-Leg Hopper [8], and the Disney single-legged hopper [9] were capable of 3 dimensional traversal. However, the 3D One-Leg Hopper was both powered and controlled externally, Salto-1P is too small to carry significant computational power, and the Disney hopper is somewhat limited in its dynamic capabilities due to its design. Furthermore, none of the three have yet been implemented with optimal control.

As such, we present the REx Hopper (Fig. 1) as a testbed for state-of-the-art control algorithms, including contact-implicit model predictive control [10]. The REx Hopper is a monopodal hopping robot with reaction wheels designed to perform highly dynamic leaping maneuvers with inertial reorientation.

In this work, we provide an overview of the REx Hopper's hardware design, software design, and control framework. We then demonstrate the use of MPC with a lumped-mass dynamics model to allow the REx Hopper to track a sinusoidal hopping trajectory. Finally, we discuss results and future work.

2 HARDWARE DESIGN

The REx Hopper is meant to serve as a platform for testing control algorithms involving dynamic leaping maneuvers. As such, the design prioritizes maximum flight time per unit mass, mid-air maneuverability, impact mitigation, and force proprioception. The robot has three reaction wheels: two for the pitch and roll axes, and one for the yaw axis. Two actuators power the Hopper's single leg.

An iterative model-simulate-test cycle played a substantial role in the overall design of the REx Hopper. An important driving factor in the design of the robot's overall shape was the location of the center of mass (CoM). During early simulations, it was found that excessive reaction torques were being generated during the push-off phase of the robot's gait due to a relatively small CoM offset. Because of this, the robot's components are arranged to keep its CoM as close to the hip axis as possible.

2.1 Scale

The REx Hopper weighs approximately 8 kg. As a general rule, the less massive a robot is, the better its performance in terms of relative jump height and impact mitigation. However, design requirements for this project bounded the REx Hopper's minimum size. For example, whereas most hopping robots lack on-board computing power, one the REx Hopper requires

a powerful on-board computer to run MPC online. In addition, available actuators, controllers, and other components limited the practical size of the robot given the time frame involved.

2.2 Actuator Design

Quasi-direct drive (QDD) actuators, typically permanent magnet synchronous outrunner motors with large air gap radii connected to single stage planetary gearing, have taken the forefront in dynamic robotic actuation over the past decade due to their transparency, backdrivability, and force proprioception capabilities [11]. The REx Hopper takes full advantage of this design paradigm for leg actuation. Both leg actuators are composed of modified RMD-X10 actuators, which have a 1:7 gear reduction. The outer shell of each actuator is replaced with a custom version machined from 7075-T6 aluminum, which has additional mounting points for various sections of the robot frame as well as the motor controllers (ODrive Pro). In addition, the custom shell is designed to serve as a heat sink for the motor controller with the help of thermal pads.

2.3 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 [12], 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.

In addition, a low-pass filter is applied to the output to represent the specified torque bandwidth of the motor controllers.

This model is applied to the control inputs at every simulation step based on each actuator's specifications.

2.4 Leg Design

The vast majority of the REx Hopper's mass is concentrated in the body. This is because it is important to minimize distal leg mass in order to better approximate a lumped mass dynamics model. In addition, minimizing distal leg mass provides optimal actuation bandwidth and impact mitigation [13]. The need for distal actuation narrowed the mechanism selection for the leg to three main types:

1. Two-actuator serial actuation as seen in [6] and [14]
2. Two-actuator parallel actuation as seen in [15]
3. A single-actuator belt-driven straight-line mechanism

While there are other methods for achieving a single-actuator straight line mechanism, they were ruled out due to concerns involving interrelated issues such as design complexity, fragility, backlash, and cost.

A PyBullet simulation based on a simple mockup of the robot was performed early in the design process to select the optimal mechanism type. All three mechanism types were compared on a basis of vertical jump height for a given body mass, and actuator specifications using the actuator model described earlier were taken into account. Types 1 and 2 were tested with both a belt drive and a linkage. Type 2 was found to provide the best ratio of jump height to actuator torque, an important factor due to the torque limitations of quasi-direct drive electric actuators.

A five-bar linkage was chosen over a belt drive due to its potential synergy with parallel spring mechanisms, which would improve efficiency and flight time through energy storage. However, due to the control complexities it would introduce, the parallel spring is a future goal and is not currently implemented.

The leg design is depicted in Fig. 2 below. The approximate forces acting on the joints were found through simulation, and were used to perform finite element analysis on all of the leg links. Links 0 through 2 are machined from 7075-T6 aluminum, whereas Link 3 is composed of 3D-printed PLA. The hard stops are damped with 3D-printed TPU. There are also hard stops built into the base to prevent the Hopper from "kicking" itself.

The foot is composed of a machined aluminum 6061 core with a polyurethane inner layer and a silicone outer layer, which screws into Link 3 for easy interchangeability. A two-shot mold was designed and used for foot fabrication.

2.5 Reaction Wheel Design

Reaction wheels consume power to produce torque as a function of rotor speed, in the following relationship: $P = \tau\omega$. This results in a very large power and energy cost per unit torque generated, especially as rotor speed increases. As such, it is highly advantageous to minimize rotor speed. There are further reasons to reduce maximum rotor speed–for example, vibration due to an offset center of mass is proportional to the square of the

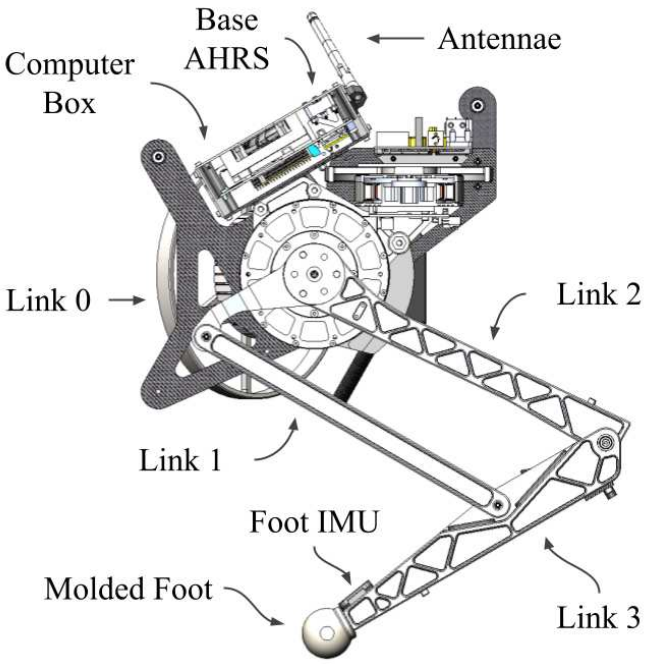

FIGURE 2: Annotated cutaway view showing the leg design

rotor velocity [5]. In addition, a lower maximum speed requirement for a given torque requirement allows for the use of axially shorter, lighter motors. On the other hand, friction increases with lower rotor speed, but this is of lesser concern.

To reduce rotor speed, acceleration should be minimized. Angular acceleration is related to torque as $\tau = \mathcal{J}\alpha$, so the reaction wheel's polar moment of inertia $\mathcal{J}$ should be maximized. On the other hand, mass should always be minimized, so the ideal reaction wheel is as large in diameter and as thin as possible. This rule is only limited by practical concerns–for example, if the reaction wheels are too large, they would come in contact with the floor every time the Hopper makes a landing. To keep the reaction wheels as thin as possible, the material of choice should be rigid, durable, and dense, which is why they are machined from stainless steel.

A rudimentary mockup of the robot design was tested in the PyBullet with a number of control policies, including Raibert hopping (See Appendix B for more details). The maximum required torque and speed of each reaction wheel was recorded after every design-and-test iteration and used to inform the next version of the design. This informed the choice of motors. At 11 Nm for the pitch-and-roll wheels and 4 Nm for the yaw reaction wheel, torque requirements for the REx Hopper's reaction wheel actuators are significantly higher than those on satellites of similar size. On the other hand, rotor speed requirements are lower, with a maximum motor speed of 3800 RPM. The reaction wheel actuators are direct drive to completely avoid the issues of

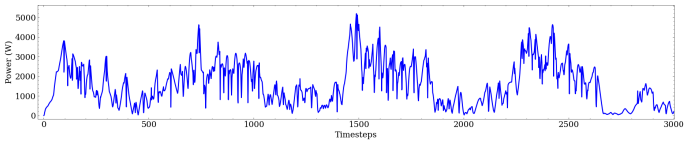

FIGURE 3: Simulated power draw over a 3-second period

reflected inertia and backlash.

2.6 Batteries

The robot is powered by two high discharge lithium polymer (LiPo) batteries through a custom designed power distribution PCB. The batteries are rated at 22.2V each and connected in series for a nominal voltage of 44.4V. To minimize the weight of the robot, the batteries have a capacity of just 1200 mAh. Using the actuator model shown in equations (1) through (3), a method for estimating the power draw of the robot was developed for the PyBullet simulation. Fig. 3 shows the robot's considerable power draw during a highly athletic hopping sequence. In extreme situations, power draw can peak at 5 kW, and the robot's batteries can be depleted in as quickly as two minutes (Although this makes the robot impractical, it's acceptable during research and testing). This is mainly due to the inefficiency of reaction wheels, as mentioned earlier. The discharge requirements observed here have been taken into account during battery selection–combined in parallel, the batteries have a nominal power draw rating of 4.3 kW.

2.7 Power Distribution

The power distribution board connects the two batteries in serial and outputs to the motor drivers. It also distributes power to the main computer through a 60W isolated DC-DC converter. Power is activated by using a hex key to turn a screw that closes the circuit. When the hex key is inserted, it first activates a limit switch connecting the batteries to the pre-charge circuit, which charges the motor controller capacitors over the course of about three seconds. This protects the capacitors from high inrush current when the main circuit is closed. In addition, an arc suppression circuit protects the main power switch from sparking.

2.8 Computing and Communication

The main computer of the Hopper is an UP Xtreme i7-8565U single board computer. A miniPCIe CAN bus card is connected to it and used for CAN communication with the five single-axis ODrive motor controllers. The CAN nodes are daisy chained where proximity allows it. A LORD Microstrain 3DM-CX5-25 AHRS is used to get the attitude estimate for the robot base, whereas a WT901 IMU provides useful acceleration data for the foot. Finally, the robot wirelessly receives motion capture

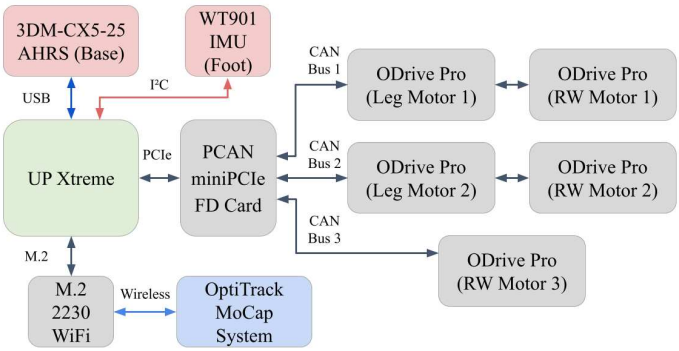

FIGURE 4: The communication architecture

data from an Optitrack system by subscribing to its ROS node. A diagram of the communication architecture is shown in Fig. 4.

2.9 Fall Protection

Any dynamic robot should be expected to fall over or crash during testing, and therefore must be durable enough to take a hit. There is reason for concern over the forces and torques acting on the REx Hopper's frame during a violent fall, especially because the reaction wheels extend so far from the CoM and would likely be spinning at high speed at the moment of impact.

As a last resort, a system of sacrificial guards protect the reaction wheels from a direct hit in the event of a fall. Four carbon fiber tubes are mounted to the center of the robot's frame with 3D-printed clamps. These clamps are expected to structurally fail on impact, and can be easily replaced. A pair of CFRP plates protect the reaction wheels from the front of the robot. These are also easily replaceable.

However, the REx Hopper is mainly protected during testing by being harnessed. There are mounting points on the front and back of the robot for this purpose.

Hardware Testing

So far all actuators and sensors are fully operational. The next step for hardware testing is to tune a simple PID controller to balance the robot in a standing position. Testing this is somewhat difficult because the robot cannot start from a standing position on its own, and having a human hold the robot in place is both dangerous and inconsistent. A testing jig is currently in the works to allow the robot to start from a standing position. However, to make future testing more practical, the robot should be capable of standing up from a sitting position. A sit-to-stand control sequence has already been successfully tested in simulation.

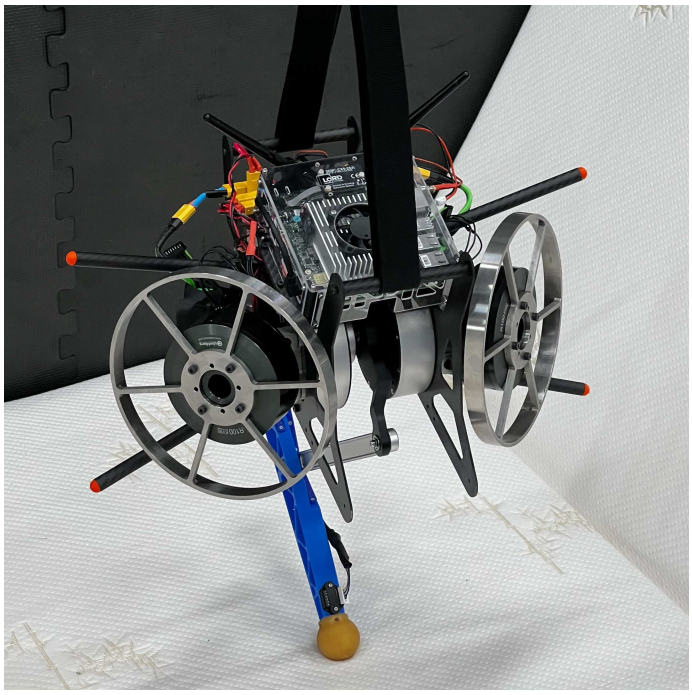

FIGURE 5: The REx Hopper harnessed to a gantry

3 SOFTWARE DESIGN

The majority of the simulated control experiments discussed in this work were written in Python, using PyBullet for simulation. However, the Python control stack (See Appendix for more details) is far too slow to run on the physical robot, so the control stack has been rewritten in C++17. The new C++ control stack uses MuJoCo for simulation.

The control stack is written to be highly polymorphic; both the simulation and hardware can be called from the same executable without re-compilation.

3.1 Parallelism and Timing

The main control loop can currently run at 1 kHz in real time; however, MPC has not yet been implemented on the C++ stack and this may change the situation. On the other hand, there are plans to switch the UP Xtreme's Linux kernel to a pseudo-realtime version to improve the timing accuracy of the system.

The CAN buses (See Fig. 4) run the CAN 2.0 protocol, communicating with the ODrives at a baud rate of 1M. Joint space position, velocity, and torque commands are sent from the main computer to the ODrives, which then execute low-level PID control at 8 kHz. The main control loop does not hang to receive updated joint positions and velocities from the ODrives. Instead,

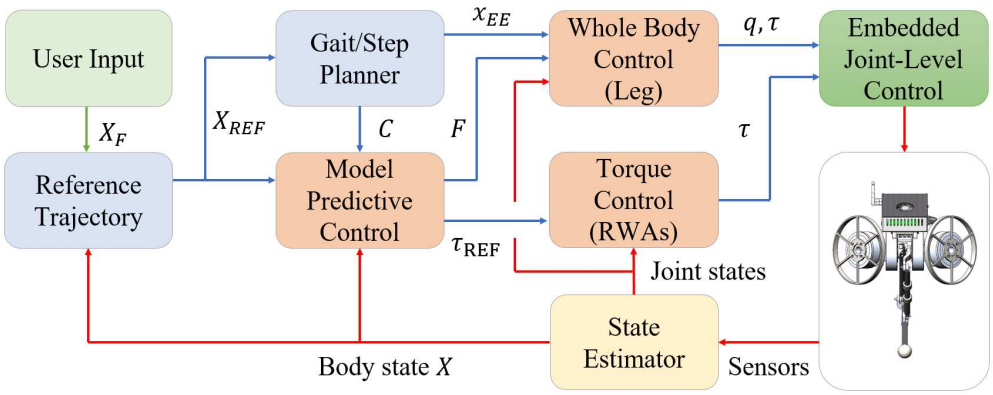

FIGURE 6: The control framework

a regular "heartbeat" message from the ODrives containing joint information is picked up asynchronously.

Both the Optitrack system and the LORD Microstrain AHRS communicate with the main computer through ROS Nodes. ROS provides an easy method for parallelization, preventing the main control loop from hanging while the external devices are queried. A ROS publisher and subscriber for the WT901 IMU is currently in the works as well.

It is notable that the Optitrack system and the WT901 IMU update at a sluggish rate of 200 Hz, and the LORD Microstrain AHRS updates at 500 Hz. This presents a state estimation challenge. One potential solution in the works is to use polynomial regression to feed extrapolated predictions of the current state into the Kalman filter.

4 CONTROL FRAMEWORK

The hybrid control framework is illustrated in (Fig. 6). 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 impulse commands to leg actuator torque commands, and during the flight phase it tracks the world frame position of the next footstep setpoint. The reaction wheel controller converts angular impulse commands to reaction wheel torque commands in both the stance and flight phases.

4.1 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 [16]:

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

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

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 $t_0 = \frac{1}{2} T \phi_{\text{switch}}$.

4.2 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.

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

Where h is the starting height, $\mathcal{A}$ 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 $\mathcal{A} = \frac{T}{4}$. 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.

4.3 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. At the start of the flight phase, a foot position trajectory is generated as a cubic spline from the foot's current location to the next footstep location, and a foot velocity trajectory is generated based on the position trajectory. The foot is then commanded to track these trajectories in the world frame during the flight phase. This prevents jerky behavior during flight and allows the foot to make contact with the ground at the correct location, on schedule.

4.4 Footstep Adaptation

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

4.5 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 follow the foot reference trajectory using the swing leg control described in [17].

4.6 MPC with Point Mass Dynamics

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

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.

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

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

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

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

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

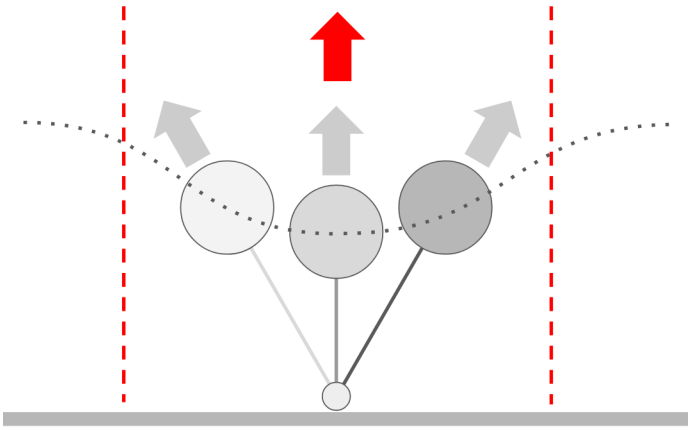

FIGURE 7: Comparison of the net impulse (red) calculated by the point mass MPC and the force vectors (gray) outputted by the leg as the CoM passes over the footstep position.

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. 7, 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 [18]. 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.

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

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

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

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

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

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

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

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

4.7 MPC with 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 (21) through (26), 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 reaction wheel actuators.

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 [17].

$$X = \begin{bmatrix} {}^{W}P_{3\times1} \\ {}^{W}\Theta_{3\times1} \\ {}^{W}\dot{P}_{3\times1} \\ {}^{W}\omega_{3\times1} \end{bmatrix} \tag{21}$$

$$U = \begin{bmatrix} {}^{B}f_{3\times1} \\ {}^{B}\tau_{3\times1} \end{bmatrix} \tag{22}$$

$$G = \begin{bmatrix} 0_{3\times1} \\ 0_{3\times1} \\ {}^{W}g_{3\times1} \\ 0_{3\times1} \end{bmatrix} \tag{23}$$

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

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

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

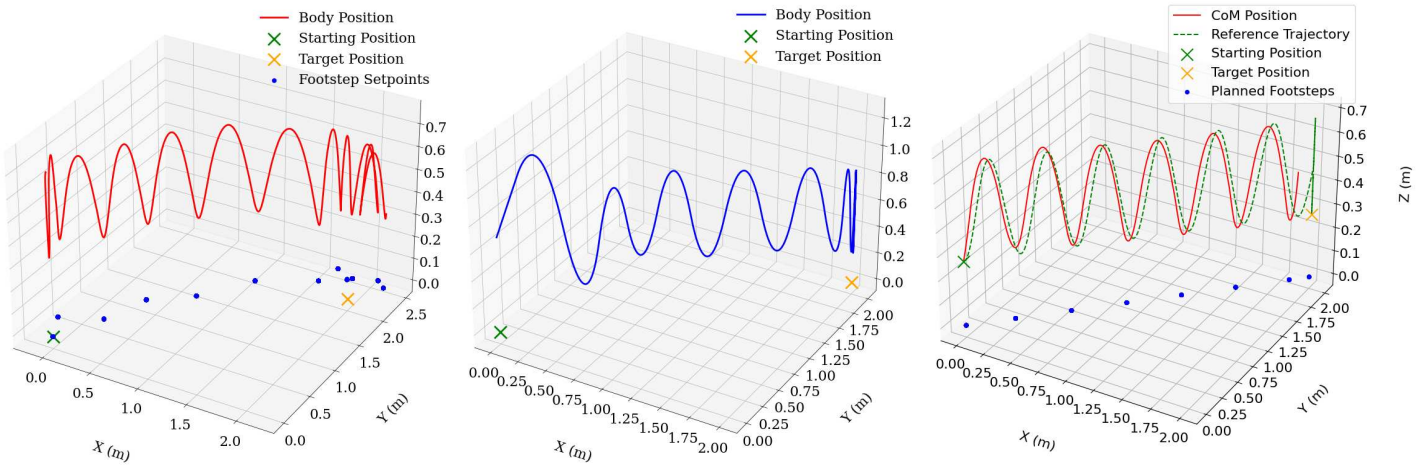

FIGURE 8: (Left) The Raibert hopping trajectory simulated in PyBullet. (Middle) The point mass simulation of the hopper with point mass MPC. (Right) The SE(3) simulation of the hopper with centroidal MPC.

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.

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

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

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

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

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

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

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

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.

5 RESULTS AND DISCUSSION

Control experiments were implemented in Python, and both 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.

5.1 Raibert Hopping Results

Raibert heuristic-based hopping control [8] was used as a basis of comparison against the MPC formulations. This "Raibert hopping" method is highly computationally efficient, but is somewhat inaccurate and requires extensive manual tuning for even small adjustments. 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. Another downside is that Raibert hopping cannot reliably manage both the robot's position and the reaction wheels' rotor speed, which leads to eventual failure once any one of the reaction wheels' rotor speeds saturate. Despite these disadvantages, highly dynamic continuous hopping has been achieved in simulation using the Raibert heuristic. As shown in Fig. 8, Raibert hopping was successfully used to command the robot to move from (x, y) position (0, 0) to position (2, 2).

For more information on the Raibert hopping method used here, see Appendix B.

5.2 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. 8.

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 reaction wheels. Attempting to track a world frame footstep position while applying force using this operational space controller results in reaction torques on the body which the attitude controller (See Appendix B) must reactively counter to maintain the correct body attitude. In practice, the attitude controller was unable to generate the correct torque profile.

5.3 Centroidal MPC Results

The centroidal MPC was first simulated with the SE(3) dynamics presented in [19] and classic Runge-Kutta integration. The MPC uses a horizon of 0.3 seconds divided into 30 steps of size 0.01 seconds. Results for hopping straight forward in the simplified simulation are as shown in Fig. 8. The centroidal MPC is also able to follow a curved reference trajectory, but only if the y-axis body rotation constraint is removed.

Fig. 10 shows results for full REx Hopper simulation in PyBullet with centroidal MPC attempting to track a sinusoidal reference trajectory over a timespan of 10 seconds. As you can see, the hopper diverges from the reference trajectory within the first 3.5 seconds. It is notable that despite the centroidal MPC's lack of robustness, it shows significantly greater accuracy than the Raibert heuristic-based method.

Fig. 9 shows the reference versus measured reaction forces between the foot and ground during that 3.5 second period, showing that they are accurate and on-time. This implies that the fault is in either the MPC or the reference trajectory, or both. It is not surprising that the reference trajectory is imperfect, as it was manually created and tuned. One way to improve the reference trajectory would be to generate it using offline nonlinear optimization with the full dynamics of the system. Another potential strategy would be to update the CoM and footstep trajectories online such that the robot can adapt to error. One challenge with this method would be generation of realistic roll trajectories for lateral movement.

5.4 Future Work

Testing of the C++ control stack on the hardware is ongoing, and we expect to perform a balancing test on the physical robot soon. We will also implement the centroidal MPC on the C++ control stack and test it on the hardware. On the controls side, we will continue to troubleshoot the centroidal MPC's robustness issues. We will experiment with adaptive reference trajectory planning. 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.

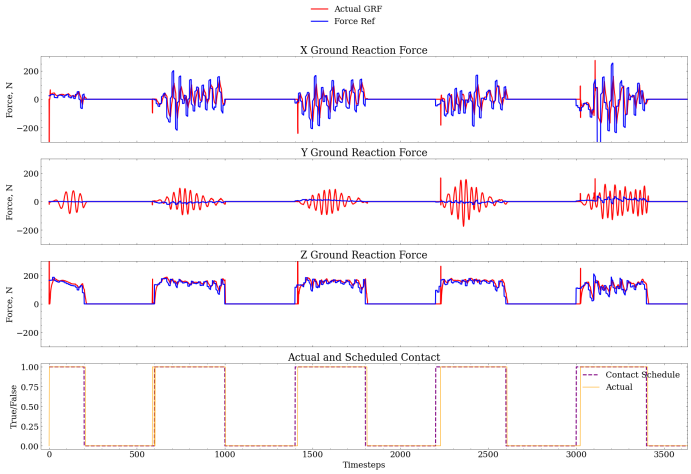

FIGURE 9: Ground reaction forces compared to reference forces in the world frame, as well as scheduled vs detected ground contact.

In terms of hardware design, the REx Hopper's main flaw is its excessive power draw. This is due in part to the Hopper's relatively high mass and inertia, which necessitates greater angular impulses from the reaction wheels, driving their acceleration and speeds higher, which wastes energy. Given the time frame for the project, we were more interested in developing a robot that "works" than a mass-optimized one. There is significant room for mass reduction in future iterations of the REx Hopper through topology optimization and other methods.

6 CONCLUSION

In this work, we introduced a system design and several corresponding control frameworks for a monopodal hopping robot with reaction wheels. The REx Hopper has power autonomy, is capable of expensive online computation, and is designed for agile movement in 3D. In full simulations of the REx Hopper, our MPC demonstrated the ability to follow a sinusoidal reference trajectory. The engineering work presented here lays the groundwork for the use of highly agile monopod robots to test novel control algorithms. Beyond the controllers discussed in this work, one such algorithm we plan to test with the REx Hopper is the contact-implicit MPC described in [10].

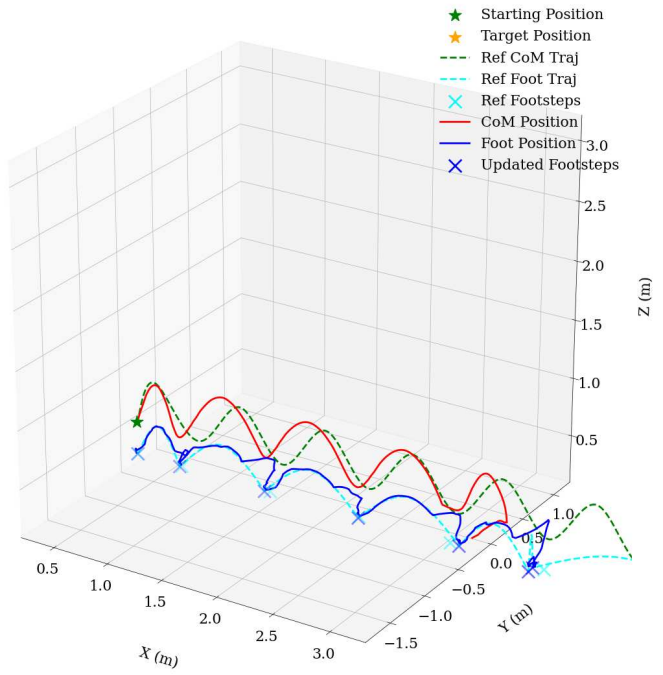

FIGURE 10: Trajectory of the REx Hopper in PyBullet simulation using MPC with centroidal dynamics.

ACKNOWLEDGMENT

Thanks go to Dr. Zachary Manchester (Carnegie Mellon University) for his generous guidance, encouragement, and support throughout my time at the REx Lab.

I would also like to thank Zhengyu Fu for his enthusiastic and industrious assistance on the software engineering side of this project, specifically setting up the C++ build environment and rewriting the CAN communication library.

REFERENCES

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

[2] Kim, D., Zhao, Y., Thomas, G., and Sentis, L., 2015. "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*.

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

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

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

[6] Ding, Y., and Park, H.-W., 2017. "Design and experimental implementation of a quasi-direct-drive leg for optimized jumping". In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 300–305.

[7] Haldane, D. W., Yim, J. K., and Fearing, R. S., 2017. "Repetitive extreme-acceleration (14-g) spatial jumping with salto-1p". In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 3345–3351.

[8] Raibert, M. H., Brown Jr, H. B., and Chepponis, M., 1984. "Experiments in balance with a 3d one-legged hopping machine". *The International Journal of Robotics Research,* ***3***(2), pp. 75–92.

[9] Batts, Z., Kim, J., and Yamane, K., 2017. "Untethered one-legged hopping in 3d using linear elastic actuator in parallel (leap)". *Springer Proceedings in Advanced Robotics*, p. 103–112.

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

[11] Wensing, P. M., Wang, A., Seok, S., Otten, D., Lang, J., and Kim, S., 2017. "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots". *Ieee transactions on robotics,* ***33***(3), pp. 509–522.

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

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

[14] Katz, B. G., 2018. "A low cost modular actuator for dynamic robots". PhD thesis, Massachusetts Institute of Technology.

[15] Hubicki, C., Grimes, J., Jones, M., Renjewski, D., Spröwitz, A., Abate, A., and Hurst, J., 2016. "Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot". *The International Journal of Robotics Research,* ***35***(12), pp. 1497–1521.

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

[17] Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., and Kim, S., 2018. "Dynamic locomotion in the mit chee-

tah 3 through convex model-predictive control". In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 1–9.
[18] Moler, C., and Van Loan, C., 1978. "Nineteen dubious ways to compute the exponential of a matrix". *SIAM review,* ***20***(4), pp. 801–836.
[19] Jackson, B. E., Tracy, K., and Manchester, Z., 2021. "Planning with attitude". *IEEE Robotics and Automation Letters,* ***6***(3), pp. 5658–5664.

Appendix A: Design Files and Code

The source code and design files for this project are available in the following repositories.

- Point-Mass Simulation with MPC
  https://github.com/bbokser/hopper-mpc-point
- Inertial Simulation with MPC
  https://github.com/bbokser/hopper-mpc-inertial
- Python Control and Simulation Stack
  https://github.com/RoboticExplorationLab/rex-hopper-python
- Symbolic Math for C++ Code Generation
  https://github.com/bbokser/hopper-symbolics
- C++ Control and Simulation Stack
  https://github.com/RoboticExplorationLab/RExHopper
- Power Distribution Board Design
  https://github.com/bbokser/hopper-power-dist

Appendix B: PID-Driven Attitude Control

A simple PID-driven attitude controller for the roll and pitch axes is presented here which is used for a standing controller and Raibert hopping. The attitude controller takes the attitude quaternion of the robot's base Q_{IN} as an input and attempts to rotate the robot into a reference attitude represented by Q_{REF}. To do this, the quaternions must be transformed into an angle error about each reaction wheel axis. Firstly, for the pitch and roll axes, the z (yaw) axis rotation should be removed. This can be done by transforming a reference forward vector by the quaternion and then calculating the rotation between the two vectors in the x-y plane:

$$\vec{v}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{34}$$

$$\vec{v}_2 = H^\top L(Q_{IN}) R(Q_{IN})^\top H \vec{v}_1 \tag{35}$$

$$\gamma = atan2(\vec{v}_{2x}, \vec{v}_{2y}) - atan2(\vec{v}_{1x}, \vec{v}_{1y}) \tag{36}$$

$$Q_\gamma = (\cos(\frac{\gamma}{2}), 0, 0, \sin(\frac{\gamma}{2}) \tag{37}$$

$$Q_{XY} = L(Q_\gamma^{-1})^\top Q_{IN} \tag{38}$$

Where $L(Q)$, $R(Q)$, and H are defined in [19]. Next, because the roll and pitch reaction wheels are mounted at 45 degrees from the sagittal plane, Q_{XY} must be rotated by $\zeta = \pm\frac{45\pi}{180}$ in the yaw axis before being used.

$$Q_\zeta = (\cos(\frac{\zeta}{2}), 0, 0, \sin(\frac{\zeta}{2}) \tag{39}$$

$$Q_{RW} = L(Q_\zeta)^\top Q_{XY} \tag{40}$$

A vector $\vec{v}_1$, representing a straight line pointing upward, is then rotated by Q_{RW} to get $\vec{v}_2$, a vector representing the robot's "up" vector in the reaction wheel frame due to its attitude.

$$\vec{v}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{41}$$

$$\vec{v}_4 = H^\top L(Q_{RW}) R(Q_{RW})^\top H \vec{v}_3 \tag{42}$$

Finally, the signed angle between $\vec{v}_3$ and $\vec{v}_4$ in the reaction wheel frame's y-z plane is found.

$$\theta_i = atan2(\vec{v}_{4z}, \vec{v}_{4y}) - atan2(\vec{v}_{3z}, \vec{v}_{3y}) \tag{43}$$

Where θ_i represents the angle of the robot base in the axis of the pitch and roll reaction wheels $i = 1, 2$. However, the calculation performed for the yaw reaction wheel is simpler: there is no yaw cancellation, no 45-degree rotation, and the signed angle is

found in the x-y plane. A similar set of operations is performed to obtain θ_{REF}, the desired angle in each reaction wheel frame. PID control is then performed on the difference between θ and θ_{REF}, with the output being reaction wheel torques. To prevent the reaction wheel rotor speeds from saturating, the PID loop is cascaded, with a velocity compensation term that attempts to drive the reaction wheel velocity to zero being fed into θ_{REF}.

The PID-driven attitude controller is combined with the methods described in [8] to test dynamic movement on the REx Hopper.