Nonlinear Model Predictive Control of a 3D Hopping Robot: Leveraging Lie Group Integrators for Dynamically Stable Behaviors

Noel Csomay-Shanklin, Victor D. Dorobantu, and Aaron D. Ames

Abstract—Achieving stable hopping has been a hallmark challenge in the field of dynamic legged locomotion. Controlled hopping is notably difficult due to extended periods of underactuation combined with very short ground phases wherein ground interactions must be modulated to regulate global state. In this work, we explore the use of hybrid nonlinear model predictive control paired with a low-level feedback controller in a multi-rate hierarchy to achieve dynamically stable motions on a novel 3D hopping robot. In order to demonstrate richer behaviors on the manifold of rotations, both the planning and feedback layers must be designed in a geometrically consistent fashion; therefore, we develop the necessary tools to employ Lie group integrators and appropriate feedback controllers. We experimentally demonstrate stable 3D hopping on a novel robot, as well as trajectory tracking and flipping in simulation.

I. INTRODUCTION

Hopping has been a benchmark challenge in the field of robotic locomotion dating back to the seminal work of Marc Raibert in the 1980’s [1]. The lessons learned during hopping have inspired generations of researchers, and have enabled complex behaviors such as walking and running on bipedal [2]–[4] and quadrupedal robots [5], [6]. Since the original work of Raibert, the landscape of control has changed dramatically, and has recently been fueled by advances in computation power allowing for previously prohibitively costly methodologies to be employed in real-time on robotic platforms. The goal of this work is to investigate the extent to which this new modern computation can be leveraged in the context of the canonical example of hopping.

The control of hopping robots is particularly challenging due to intermittent continuous and discrete dynamics, periods of extreme underactuation, and exceptionally short ground phases during which the robot can apply forces to regulate its global position. These pose unique difficulties for conventional control algorithms, and necessitate the ability to decide control actions based on predictions of where the robot will be in the future. Raibert addressed these challenges through the use of reduced order models, e.g., the spring loaded inverted pendulum (SLIP) model [7], which has since proven effective in walking and running generation on bipedal robots [8]. Besides the work of Raibert, many methods have been developed to stabilize hopping robots [9], including reinforcement learning based approaches [10], [11], nonholonomic motion planning [12], a mix of offline and online hierarchical motion planning strategies [13], [14], and model predictive control of simplified models [15], [16]. Compared to the abundance of work that exists for planar hopping robots, the literature for the control of 3D hopping robots is comparatively sparse.

Recently, model predictive control (MPC) [17] has been used effectively for the control of dynamic robotic systems, including hybrid systems (systems with continuous dynamics and discrete impacts [18]) like legged robots [19]–[21]. MPC has been brought into the realm of real-time dynamic robotic control due to modern computing power and increased algorithmic efficiency [22]; however, the implementation of MPC on hardware platforms, as well as theoretical justifications of its performance for nonlinear systems remains an active area of research. As nonlinear MPC is predicated on taking local approximations of system dynamics, its success heavily relies on correctly constructing these approximations and remaining within the regions in which they are valid.

When the system states are manifold-valued, these local approximations must be carefully constructed. We will specifically be concerned with Lie groups, groups with smooth manifold structure whose operator and inverse are also smooth (see [23], [24]), as they are often used in the field of robotics to model the space of orientations. Lie groups have additionally been studied from the perspectives of discrete mechanics [25] and numerical analysis [26], [27]. The estimation of robotic systems on Lie groups was outlined in [28], and the control of legged robotic systems on Lie groups has recently been investigated in [29]. Finally, the application of optimal control techniques over differentiable manifolds to the control of hybrid systems was explored in [30] with experimental results achieved in [31], [32].

The goal of this work is to experimentally realize dynamic hopping behaviors on a novel 3D hopping robot through...
the use of nonlinear MPC. To this end, we: (1) develop a framework for hybrid nonlinear MPC in a geometrically consistent fashion via Lie group integrators, and (2) experimentally demonstrate on a novel 3D robot that MPC is an effective tool for accomplishing highly dynamic behaviors such as hopping.

Theoretically, we consider both the manifold structure of the configuration space and the hybrid nature of the dynamics (Sec. II), wherein an MPC problem is synthesized through the use of sequential linearizations that leverage Lie group and Lie algebra structures (Sec. III). The MPC problem is translated to hardware via a multi-rate control paradigm. To experimentally verify this framework (Sec. IV), we leverage a novel hardware platform: ARCHER [33] (which builds upon earlier generations of hopping robots [34]-[36]). This robot has three reaction wheels for attitude control, and one motor connected via a rope to control foot spring compression. We first show the capabilities of this robot in a high-fidelity simulation wherein a variety of dynamic motions are demonstrated: from path following to flipping. Finally, we experimentally realize sustained 3D hopping on this new hardware platform, marking the first demonstration of 3D hopping using online motion planning strategies.

II. PRELIMINARIES

A. Hybrid System Dynamics

The configuration of the hopping robot is given by \( q = (p, q, \theta, \ell) \in \mathbb{Q} \), where \( p \in \mathbb{R}^3 \) is the Cartesian position, \( q \in S^3 \) is the unit quaternion representing the orientation, \( \theta \in \mathbb{R}^3 \) represents the flywheel angles, and \( \ell \in \mathbb{R} \) is the foot deflection. Next, let \( v = (\dot{p}, \omega, \dot{\theta}, \ell) \in \mathbb{V} \triangleq \mathbb{R}^3 \times s^3 \times \mathbb{R}^3 \times \mathbb{R} \), where \( \omega \in s^3 \) is a purely imaginary quaternion representing the angular rate of the body. The complete robot state can then be written as \( x = (q, v) \in \mathcal{X} = \mathbb{Q} \times \mathbb{V} \).

Hopping consists of alternating sequences of continuous and discrete dynamics; therefore, it is naturally modeled as a hybrid system. Two distinct continuous phases of dynamics exist, the flight phase \( f \) when the robot is in the air, and the ground phase \( g \) when the foot is contacting the floor. We can construct a directed graph with vertices \( v \in \mathcal{V} \triangleq \{ f, g \} \) and edges \( e \in \mathcal{E} \triangleq \{ f \rightarrow g, g \rightarrow f \} \) to characterize how the robot traverses the hybrid modes, as shown in Figure 2.

For each vertex \( v \in \mathcal{V} \), let \( \mathcal{D}_v \subset \mathcal{X} \) represent the admissible domain in which the system state evolves, and \( n_v \) denote the number of holonomic constraints restricting the motion of the robot. Note that \( n_f = 0 \), and \( n_g = 3 \), which pin the foot to the ground. Omitting the details for manifold value variables (which can be found in [37]), let \( J_v : \mathbb{Q} \rightarrow \mathbb{R}^{n_a \times n} \) denote the Jacobian of the holonomic constraints where \( n = |\mathbb{Q}| \). We can define the dynamics as:

\[
D(q)\dot{q} + H(q, \dot{q}) = Bu + J_v(q)\lambda_v,
\]

\[
J_v(q, \dot{q}) + J_v(q)\lambda_v = 0.
\]

where \( D : \mathbb{Q} \rightarrow \mathbb{R}^{n \times n} \) is the mass-inertia matrix, \( H : \mathcal{X} \rightarrow \mathbb{R}^n \) contains the Coriolis and gravity terms, \( B \in \mathbb{R}^{n \times m} \) is the actuation matrix, \( u \in \mathbb{R}^m \) is the control input, and \( \lambda_v \in \mathbb{R}^{n_v} \) are the Lagrange variables describing the constraint forces. These equations can be rearranged and forces \( \lambda_v \) solved for in order to produce the constraint implicit dynamics:

\[
\dot{x} = f_v(x, u),
\]

where \( f_v : \mathcal{X} \times \mathbb{R}^m \rightarrow \mathbb{R}^{2n} \) is of control-affine form.

Each hybrid transition \( e \in \mathcal{E} \) occurs when the system state intersects the guard, defined as:

\[
\mathcal{S}_f = \{ x \in \mathcal{D}_f : p_z(q) = 0, \dot{p}_z(x) < 0 \} \quad e = f \rightarrow g,
\]

\[
\mathcal{S}_g = \{ x \in \mathcal{D}_g : \ell = 0, \dot{\ell} < 0 \} \quad e = g \rightarrow f,
\]

where \( p_z : \mathbb{Q} \rightarrow \mathbb{R} \) returns the vertical position of the foot. For an edge \( e = v_1 \rightarrow v_2 \), upon striking the guard \( \mathcal{S}_{v_1} \), the system undergoes a discrete jump in states as described by:

\[
x^+ = \Delta_e(x^-),
\]

where \( \Delta_e : \mathcal{S}_{v_1} \rightarrow \mathcal{D}_{v_2} \) is the reset map describing the momentum transfer though impact, and \( x^- \in \mathcal{D}_{v_1} \) and \( x^+ \in \mathcal{D}_{v_2} \) are the pre and post-impact states, respectively. Collecting the various objects \( D = \{ \mathcal{D}_v \}_{v \in \mathcal{V}}, S = \{ \mathcal{S}_v \}_{v \in \mathcal{V}}, \Delta = \{ \Delta_e \}_{e \in \mathcal{E}} \) and \( F = \{ f_v \}_{v \in \mathcal{V}} \), we can describe the hybrid control system of the hopping robot via the tuple:

\[
\mathcal{HC} = (V, E, D, S, \Delta, F).
\]

B. Lie Group Integrators

In this section, we focus our attention to the orientation coordinates of the robot and discuss how to perform one form of Lie group integration. We represent the orientation of a rigid body via a unit quaternion \( q \in S^3 = \{ q \in \mathbb{H} : |q| = 1 \} \); quaternionic representations of orientation have been extensively explored in attitude control of spacecrafts [38], [39]. As opposed to Euler angles, quaternions do not suffer from issues of singularities, and provide a straightforward interpolation method – this will be helpful when constructing continuous-time signals from the discrete points that MPC produces. Importantly, quaternions and their associated quaternion multiplication define a Lie group structure on \( S^3 \).

The angular rate of the body is given by an element of the associated Lie algebra \( \omega \in s^3 \), the tangent space of \( S^3 \) at the identity element \( q_E \), the elements of which are purely imaginary quaternions.
The time rate of change of a unit quaternion at a point \( q \) is given by an element of the tangent space at \( q \), i.e. \( \dot{q} \in T_qS^3 \). Given the angular rate of the body \( \omega \), we can calculate \( \dot{q} \) as:

\[
\dot{q} = q \omega, \tag{2}
\]

using standard quaternion multiplication. The formulation (2) is possible since the tangent map of left multiplication by a quaternion is also given by left multiplication, mapping from the Lie algebra to the tangent space at \( q \). Integrating equation (2) results in:

\[
q(t) = q(0) \exp(\omega(t)) \tag{3}
\]

where \( \exp : s^3 \to S^3 \) is termed the exponential map and maps elements of the Lie algebra \( s^3 \) back to the Lie group \( S^3 \). This map is injective for imaginary quaternions with magnitude less than \( \pi \); over this neighborhood, the inverse map is denoted as \( \log : S^3 \to s^3 \). If instead of directly integrating we take a Lie-Euler step, (3) becomes:

\[
q_{k+1} = q_k \exp(\omega_k h) \tag{4}
\]

for a (small) time step \( h \in \mathbb{R} \). This is a simple example of a Lie group integrator.

III. GEOMETRIC MPC AND MULTI-RATE CONTROL

A. Linearized Dynamics

In order to avoid the nonlinearity present in (4), we propose the following change of coordinates:

\[
\xi_k = \log(q_0^{-1} q_k), \tag{5}
\]

which first pulls the variables back to the vicinity of \( q_0 \), and then to the Lie algebra as shown in Figure 3. Substituting in the Lie-Euler step from (4) into the above expression for the first few values of \( k \), we have:

\[
\begin{align*}
\xi_0 &= 0 \\
\xi_1 &= \log(q_0^{-1} q_1) = \log(q_0^{-1} q_0 \exp(\omega_0 h)) \\
&= \omega_0 h \\
\xi_2 &= \log(q_0^{-1} q_2) = \log(q_0^{-1} q_1 \exp(\omega_1 h)) \\
&= \log(q_0^{-1} q_0 \exp(\omega_0 h) \exp(\omega_1 h)) \\
&= \omega_0 h + \omega_1 h + \frac{1}{2} [\omega_0 h, \omega_1 h] + O(h^3)
\end{align*}
\]

\[\text{where} \ [\cdot, \cdot] \text{represents the Lie bracket, the last line follows from the Campbell-Baker-Hausdorff theorem [40], and the higher order terms consist of linear combinations of iterated Lie brackets, which due to linearity are multiplied by terms of } h^3 \text{ or higher}^2. \text{ This means that, neglecting terms of } O(h^2) \text{ or higher, we are able to write our dynamics update law as:}

\[
\xi_{k+1} = \sum_{i=0}^{k} \omega_i h = \xi_k + \omega_k h,
\]

which is linear and will therefore be straightforward to include in the MPC program. The next challenge concerns the construction of local approximations of the acceleration dynamics. As most of the coordinates lie in Euclidean space and therefore have straightforward Taylor approximations, we will limit our attention on the manifold valued variables. Specifically, in a continuous domain \( \nu \) consider a function \( f : S^3 \times s^3 \times \mathbb{R}^m \to s^3 \) satisfying:

\[
\dot{\omega} = f(q, \omega, u).
\]

Given a perturbation \( \eta \in s^3 \), a local representation of \( f \) in exponential coordinates \( \tilde{f} : S^3 \times s^3 \times \mathbb{R}^m \times s^3 \to \mathbb{R}^3 \) can be defined as:

\[
\tilde{f}(q, \omega, u, \eta) \triangleq f(q \exp(\eta), \omega, u).
\]

Given \( q \in S^3, \omega \in s^3, \text{ and } u \in \mathbb{R}^m \), as well as additional perturbations \( \Delta \omega \in \mathbb{R}^3 \) and \( \Delta u \in \mathbb{R}^m \), we can compute a Taylor expansion of \( \tilde{f} \) about the point \( (q, \omega, u, 0) \) at a perturbed point \( (q, \omega + \Delta \omega, u + \Delta u, \eta) \) via:

\[
\begin{align*}
f(q \exp(\eta), \omega + \Delta \omega, u + \Delta u) &\approx f(q, \omega, u) + \frac{\partial f}{\partial \eta} \cdot \eta + \frac{\partial f}{\partial \omega} \cdot \Delta \omega + \frac{\partial f}{\partial u} \cdot \Delta u \\
&= f(q, \omega, u) + \frac{\partial f}{\partial \eta} \cdot \eta + \frac{\partial f}{\partial \omega} \cdot \Delta \omega + \frac{\partial f}{\partial u} \cdot \Delta u.
\end{align*}
\]

Then, we can write the continuous-time linearized dynamics of \( \omega \) about the point \( (q, \omega, u) \) as:

\[
\frac{d}{dt} \Delta \omega = \frac{\partial \tilde{f}}{\partial \eta}(q, \omega, u, 0) \cdot \eta + \frac{\partial \tilde{f}}{\partial \omega} \cdot \Delta \omega + \frac{\partial \tilde{f}}{\partial u} \cdot \Delta u. \tag{6}
\]

Next, we consider the dynamics of the variables \( \xi_k \) around a reference trajectory \( q_k \in S^3, \omega_k \in s^3, \text{ and } u_k \in \mathbb{R}^m \). Define \( \xi_k = \log(q_0^{-1} q_k) \) with \( \xi_0 = 0 \) and suppose the reference trajectory satisfies the Lie-Euler step, i.e.:

\[
\xi_{k+1} = \xi_k + \omega_k h.
\]

\[\text{If quaternion multiplication commuted, then the iterated Lie brackets would be zero – but alas, no such pleasantries exist.}\]
For a trajectory $\xi_k \in s^3$ similarly satisfying the Lie-Euler step, and vectors $\omega_k \in s^3$, and $u_k \in \mathbb{R}^m$, we have:

$$ (\xi_{k+1} - \xi_{k+1}) = (\xi_k - \xi_k) + (\omega_k - \omega_k) h + \xi_k + \omega_k h - \xi_{k+1}, $$

yielding continuous-time linearized dynamics:

$$ \frac{d}{dt} \delta \xi = \delta \omega. \quad (7) $$

Combining expressions (6) and (7), we obtain:

$$ \frac{d}{dt} \begin{bmatrix} \delta \xi \\ \delta \omega \end{bmatrix} = \begin{bmatrix} 0 & I \\ \frac{\partial f}{\partial \eta}(q, \omega, u, 0) & \frac{\partial f}{\partial \omega} \end{bmatrix} \begin{bmatrix} \delta \xi \\ \delta \omega \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\partial f}{\partial u} \end{bmatrix} \delta u. \quad (8) $$

We can similarly construct local approximations of the impact maps. On an edge $e$, consider a function $\Delta : S \to s^3$ which satisfies:

$$ \omega^+ = \Delta(q^-, \omega^-). \quad (9) $$

Here, $S$ represents the guard as a submanifold of $S^3 \times s^3$ (though the guard is actually a submanifold of $Q$). As defined, this reset map is the restriction of the momentum transfer of the system at impact to the guard. Therefore, we can naturally extend the domain of the reset map by considering the same momentum transfer applied anywhere in the state space, yielding $\Delta_{\text{ext}} : S^3 \times s^3 \to s^3$. This is needed because in our Taylor expansion of the discrete dynamics, we consider perturbations of the full system state (not just perturbations tangent to the guard) as before, we can locally approximate the function $\Delta_{\text{ext}}$ via:

$$ \Delta_{\text{ext}}(q^-, \omega^- + \Delta \omega^-) \approx \Delta_{\text{ext}}(q^-, \omega^-) + \frac{\partial \Delta_{\text{ext}}}{\partial \eta}(q^-, \omega^-, 0) \cdot \eta + \frac{\partial \Delta_{\text{ext}}}{\partial \omega^-} \cdot \Delta \omega^-,$n

where again we can represent $\Delta_{\text{ext}}$ locally as:

$$ \tilde{\Delta}_{\text{ext}}(q^-, \omega^-, \eta) \approx \Delta_{\text{ext}}(q^- \exp(\eta), \omega^-). $$

Noting that the $q^+ = q^-$, we can represent the linearization of the discrete map as:

$$ \begin{bmatrix} \delta \xi^+ \\ \delta \omega^+ \end{bmatrix} = \begin{bmatrix} I & 0 \\ \frac{\partial \Delta_{\text{ext}}}{\partial \eta}(q^-, \omega^-, 0) & \frac{\partial \Delta_{\text{ext}}}{\partial \omega^-} \end{bmatrix} \begin{bmatrix} \delta \xi^- \\ \delta \omega^- \end{bmatrix}. \quad (10) $$

### B. Geometric Model Predictive Control

This section represents the mid-level of the control hierarchy, as shown in Figure 4. High level target base positions $p_{\text{ref}} \in \mathbb{R}^3$ are provided by the user, and MPC produces reference trajectories to pass to the low level. This architecture maintains the advantage of having a horizon and is paired with a low level feedback controller which adds robustness to model error and delays induced by computation time.

For a given MPC horizon $N \in \mathbb{N}$, we begin by constructing a vertex sequence $v_k \in V$ for $k = 0, \ldots, N - 1$ describing the continuous modes that the robot will be in.

3This extension allows us to avoid from having to only consider perturbations along the guard, which is an interesting area of future work.
C. Quaternionic Feedback

Once MPC produces a solution, a desired trajectory and feedforward input can be produced as:

\[ q_d(\tau) = \bar{q}_0 \exp(\xi^1), \quad \omega_d(\tau) = \omega^1, \quad u_{ff}(\tau) = u^0, \]

for \( \tau \in [0, dt) \subset \mathbb{R} \). An interesting area of future work
is using the MPC signals to produce dynamically admissible trajectories in the inter-MPC times, but this was not explored due to communication bandwidth limitations. The FTOCP is implemented in a receding horizon fashion, where the low level controller only ever receives the first control input and desired trajectory.

Given the measured quaternion \( q_a \), measured angular rate \( \omega_a \), desired quaternion \( q_d \), and desired angular rate \( \omega_d \) of the robot, we can construct our actuation as:

\[ u(x, t) = -K_p \Im(q_d(t)^{-1}q_a) - K_d(\omega_a - \omega_d(t)) + u_{ff}(t), \]

where \( K_p, K_d \in S^3 \) are positive definite gain matrices. The product \( q_a(t)^{-1}q_d \) represents a “difference” between elements of \( S^3 \); if \( q_a \) is in a small neighborhood of \( q_d \), then the product is in a small neighborhood of the identity element \( q_E \). The map \( \Im : S^3 \to s^3 \) takes the purely imaginary component of the error signal, and can be viewed as the Euclidean projection of the Lie group onto the Lie algebra, allowing us to base the control input over a vector valued error. Alternatively, the \log \ operation could be used instead of \Im, but the \Im \ operator was empirically found to work more reliably.

IV. IMPLEMENTATION AND RESULTS

A. Hardware

The ARCHER hardware platform consists of three KV115 T-Motors with 250 g flywheel masses attached for orientation control, and one U10-plus T-Motor attached to a 3-1 gear reduction to the foot via a cable and pulley system. The robot is powered by two 6 cell LiPo batteries connected in series, which can supply up to 50.8 V at over 100 A of current to the four ELMO Gold Solo Twitter motor controllers. The robot has two on-board Arduino Teensy microcontrollers for the low level feedback control, which run at 1kHz and communicate over WiFi via an ESP module to a desktop running the mid-level controller.

The MPC program runs at 100 Hz on an Ubuntu 20.04 machine with AMD Ryzen 5950x @ 3.4 GHz and 64 Gb RAM. The Pinocchio C++ library [22] is used, specifically the pinocchio3 preview branch, to produce fast evaluation of the system dynamics (constrained, unconstrained, continuous, and discrete), as well as their associated Jacobians, and the manif C++ library [28] is used to handle all Lie group operations (such as \log \ and \exp \). As seen in Figure 5 as well
as the supplemental video [41], the robot was successfully
able to hop stably in place, demonstrating the first instance
of 3D hopping using online motion planning.

B. Simulation

In order to thoroughly test the method, the torque limits
of the robot were increased in a Mujoco [42] simulation
environment from 1.5 Nm to 15 Nm. First, the tracking of
various global reference signals, including a square and a
Lissajous trajectory, were evaluated as seen in Figure 6. Note
the constant global velocity in the flight phase due to the lack
of control inputs when the robot is in the air. As such, the
robot must carefully plan its interactions with the ground in
order to track the desired reference signals. Specifically, it
is interesting to see how MPC is implicitly able to control
the actuated coordinates of the robot in order to stabilize
the underactuated ones. Also note the discontinuities in the
MPC planned trajectories around the impact events due to
the hybrid nature of the system dynamics. Next, disturbance
rejection and more dynamic maneuvers like flipping were
tested on the system, as seen in the accompanying video [41]
and Figure 7. Due to the geometrically consistent structure
of the planning algorithm, the robot is able to explore a
variety of states on its orientation manifold, and exhibits
exceptional robustness to disturbances. Note that the torque
limitations, but not limitations of the methodology, prohibit
such demonstrations on the hardware platform.

C. Implementation Details

The complete list of parameters used in the MPC pro-
gram are detailed in Table I. There is an inherent tradeoff
between tracking global position and maintaining a vertical
orientation – as such the associated gains need to be appro-
priately tuned. To avoid adding a nonlinear and mixed integer
constraint in to the optimization program, the impact time
was calculated as though the hopper was exactly vertical via
solving for the ballistic trajectory in the z-direction. As the
spring dynamics add significant stiffness to the optimization
problem, the foot torque was set to zero in the optimizer,
and instead the MPC program plans as though it is a passive
degree of freedom. Instead, the low level controller runs
its own feedback controller to regulate foot compression
between impact events. As the dynamics of the ground phase
are more challenging than the flight phase, the system was
discretized more finely in that domain. This means that the
lookahead time shrank whenever impact came into view
in the horizon, as the horizon length was kept constant.
We found that using the matrix exponential instead of the
Euclidean Euler step aided in performance, likely due to
the reduced one step prediction error. Finally, the quadratic
approximation of the cost to go was simply taken to be Q.
The complete code can be found at [43].

V. Conclusion and Future Work

In this work, we demonstrated that predictive control is
successfully able to regulate the underactuated coordinates
of a 3D hopping robot through intelligent control of the
actuated states. Through the use of geometrically consistent
model predictive control and feedback layers, we were
able to achieve stable hopping on hardware, and trajectory
tracking and flipping in simulation. Future work includes
providing theoretical justification of why predictive control
is able to regulate the underactuated coordinates of robotic
systems, as well as a proof of state and input constraint
satisfaction in continuous time for underactuated systems.
On the hardware side, planning with constrained footholds,
incorporating friction cone constraints, and the inclusion of
a high level decision making layer will be investigated.

VI. Acknowledgements

The authors would like to especially thank Igor Sadalski,
as well as Sergio Esteban and Adrian Boedtker Ghansah for
their help with simulation and hardware implementation, and
Will Compton for his experimental assistance.

| Horizon Length | 20 | SQP Iterations | 2 |
|----------------|----|----------------|---|
| \( p \) Weight  | 10 | \( v \) Weight | 1 |
| \( q \) Weight  | 10 | \( \omega \) Weight | 0.01 |
| \( u \) Weight  | 0.001 | \( u_{\text{max}} \) | 1.5 Nm |
| \( K_p \) Roll/Pitch Gain | 120 | \( K_g \) Roll/Pitch Gain | 4 |
| \( K_p \) Yaw Gain  | 15 | \( K_g \) Yaw Gain | 1 |
| \( dt_{\text{flight}} \) | 0.01 | \( dt_{\text{ground}} \) | 0.001 |
REFERENCES

[1] M. H. Raibert, H. B. Brown, 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, Jun. 1984, publisher: SAGE Publications Ltd. STM. [Online]. Available: https://doi.org/10.1177/027836498400300207

[2] M. H. Raibert, H. B. Brown, M. Chepponis, J. Koechling, and J. K. Hocgins, “Dynamically Stable Legged Locomotion,” p. 207.

[3] A. D. Ames, “Walking Toward Agile and Efficient Bipedal Robots,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 535–572, May 2021.

[4] M. Poulakakis, J. A. Smith, and M. Buehler, “Modeling and Experiments of Untethered Quadrupedal Running with a Bounding GaIT: The Scout II Robot,” *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 239–256, Apr. 2005. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364904059117

[5] H.-W. Park, P. M. Wensing, and S. Kim, “High-speed bounding with the MIT Cheetah 2: Control design and experiments,” *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, Feb. 2017. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364917694244

[6] H. Geyer, A. Seyfarth, and R. Blickhan, “Compliant leg behavior explains basic dynamics of walking and running,” *Proceedings. Bio. sciences / The Royal Society*, vol. 273, pp. 2661–7, Dec. 2006.

[7] J. Albro and J. Bobrow, “Optimal motion primitives for a 5 DOF hopping model using approximate step-to-step models for navigation,” *International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 5026–5033, iSSN: 2577-087X.

[8] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, “Models, feedback control, and open problems of 3D bipedal robots,” *Automatica*, vol. 50, no. 8, pp. 1955–1988, Aug. 2014.

[9] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, “Bipedal Locomotion with Nonlinear Model Predictive Control: Online Gait Generation using Whole-Body Dynamics,” *Mar. 2022, arXiv:2003.07429 [cs].

[10] M. Y. Galliker, N. Csomay-Shanklin, R. Grandia, A. J. Taylor, F. Farshidian, M. Hutter, and A. D. Ames, “Bipedal Locomotion with Nonlinear Model Predictive Control: Online Gait Generation using Whole-Body Dynamics,” *Mar. 2022, arXiv:2003.07429 [cs].

[11] F. Farshidian, E. Jelavic, A. Satpathy, M. Githalber, and J. Bucchi, “Real-time motion planning of legged robots: A model predictive control approach,” in *2017 IEEE-RAS 17th International Humanoids Conference*, Nov. 2017, pp. 577–584, ISSN: 2164-0580.

[12] J. Di Carlo, P. M. Wensing, B. Katz, G. Blest, 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), Oct. 2018, pp. 1–9, ISSN: 2153-0866.

[13] J. Campion, G. Saurel, G. Buondonno, J. Mirbel, F. Lamiraux, O. Stasse, and N. Mansard, “The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives,” in *2019 IEEE/RSJ International Symposium on System Integration (SII)*, Jan. 2019, pp. 614–619, ISSN: 2474-2325.

[14] E. Celledoni, H. Marthinsen, and B. Owren, “An introduction to lie group integrators–basics, new developments and applications,” *Journal of Computational Physics*, vol. 257, pp. 1040–1061, 2014.

[15] M. Kobilarov, K. Crane, and M. Desbrun, “Lie group integrators for animation and control of vehicles,” *ACM transactions on Graphics (TOG)*, vol. 28, no. 2, pp. 1–14, 2009.

[16] O. Junge, J. E. Marsden, and S. Ober-Blöbaum, “Discrete mechanics and optimal control,” *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 538–543, 2005.

[17] M. Iwasaki, J. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, “Lie-group methods,” *Acta numerica*, vol. 9, pp. 215–365, 2000.

[18] E. Hairer, M. Hochbruck, A. Iserles, and C. Lubich, “Geometric numerical integration,” *Oberwolfach Reports*, vol. 3, pp. 805–882, 2006.

[19] J. Solá, J. Deray, and D. Atchutthan, “A micro Lie theory for state estimation in robotics,” Dec. 2021, arXiv:1812.01537 [cs].

[20] S. Teng, D. Chen, W. Clark, and M. Ghaffari, “An Error-State Model Predictive Control on Connected Matrix Lie Groups for Legged Robot Control,” Mar. 2022, arXiv:2003.08728 [cs, eess].

[21] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Campion, L. Righetti, S. Vijayakumar, and N. Mansard, “Crocosy: An Efficient and Versatile Framework for Multi-Contact Optimal Control,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[22] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Solà, N. Mansard, and S. Vijayakumar, “A feasibility-driven approach to control-limit DDP,” *Autonomous Robots*, vol. 46, no. 8, pp. 985–1005, Dec. 2022. [Online]. Available: https://doi.org/10.1007/s10514-022-10061-w

[23] C. Mastalli, S. P. Chhatot, T. Corberèes, S. Tonneau, and S. Vijayakumar, “Inverse-Dynamics MPC via Nullspace Resolution,” Jan. 2023, arXiv:2209.05375 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2209.05375

[24] E. R. Ambrose, “Creating ARCHER: A 3D Hopping Robot with Flywheels for Attitude Control,” Ph.D. thesis, Caltech, 2022.

[25] E. Ambrose, W.-L. Ma, and A. D. Ames, “Towards the Unification of System Design and Motion Synthesis for High-Performance Hopping Robots,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 7073–7078, ISSN: 2577-087X.

[26] E. Ambrose and A. D. Ames, “Improved Performance on Moving-Mass Hopping Robots with Parallel Elasticity,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2457–2463, ISSN: 2577-087X.

[27] E. Ambrose, N. Csomay-Shanklin, Y. Or, and A. Ames, “Design and Comparative Analysis of 1D Hopping Robots,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 5717–5724, ISSN: 2153-0866.

[28] R. Abraham and J. E. Marsden, *Foundations of Mechanics*. American Mathematical Soc., 2008, google-Books-ID: YAEKBAAQBAJ.

[29] U. V. Kalabič, R. Gupta, S. D. Cairano, A. M. Bloch, and I. V. Kolmanovsky, “MPC on manifolds with an application to the control of spacecraft attitude on SO(3),” *Automatica*, pp. 293–300, 2017.

[30] Y. Yang, “Spacecraft attitude determination and control: Quaternions-based method,” *Ann. Reviews in Control*, vol. 36, pp. 198–219, 2012.

[31] J. Stillwell, “The matrix logarithm,” in *Naive Lie Theory*, ser. Undergraduate Texts in Mathematics, J. Stillwell, Ed. New York, NY: Springer, 2008, pp. 139–159. [Online]. Available: https://doi.org/10.1007/978-0-387-78241-0_7

[32] “Video.” 2022. [Online]. Available: https://youtu.be/0XMbACG0s8

[33] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 5026–5033, ISSN: 2153-0866.

[34] “Complete Hopping Robot Code,” 2022. [Online]. Available: https://github.com/noelc-s/ARCHER_hopper