2019 International Conference on Unmanned Aircraft Systems (ICUAS)
Atlanta, GA, USA, June 11-14, 2019

Error-State LQR Control of a Multirotor UAV

Michael Farrell[12], James Jackson[2], Jerel Nielsen[3], Craig Bidstrup[3], and Tim McLain[2]

Abstract— **We propose an implementation of an LQR controller for the full-state tracking of a time-dependent trajectory with a multirotor UAV. The proposed LQR formulation is based in Lie theory and linearized at each time step according to the multirotor's current state. We show experiments in both simulation and hardware that demonstrate the proposed control scheme's ability to accurately reach and track a given trajectory. The implementation is shown to run onboard at the full rate of a UAV's estimated state.**

I. Introduction

Over the past two decades, multirotor unmanned air vehicles (UAVs) have become a popular platform for robotics research and the base for a variety of consumer and commercial products. UAVs are currently used all over the world for everything from military surveillance to package delivery. Larger multirotor vehicles have even been recently introduced to transport humans. Whatever the application, multirotors must but able to safely navigate in their environment, requiring a combination of complex perception, motion planning, and control algorithms. This paper describes a novel control algorithm that allows a multirotor UAV to accurately track a desired trajectory in time and space.

The Linear Quadratic Regulator (LQR) is a well-known feedback controller that computes the optimal feedback gains for a linear time-invariant (LTI) system given a quadratic cost function. LQR has been used to control multirotor UAVs with a variety of approaches. Almost all of these approaches linearize the system at a given stable state and use a fixed LQR gain [1]. Some have used a gain scheduling approach with a library of LQR gains for different magnitudes of deviation from the desired state [2]. Recently, an approach was proposed that relinearizes the system at a fixed rate, slower than the control loop, and then recomputes the LQR gains at that rate [3]. Our proposed solution takes a similar approach while relinearizing and recomputing the LQR gains at every control step.

Recently there has been a movement in the robotics community to appropriately deal with the evolution of a robot's state along a manifold using Lie theory [4]. Though these methods have widely been used in the field of state estimation [5] [6], a few methods have emerged that also apply Lie theory to control [7] [8]. We propose a formulation of the LQR problem that properly deals with the manifold nature of the state, specifically the attitude component. Most previous LQR solutions for a multirotor UAV use an Euler angle representation of attitude and treat the tuple of ZYX Euler angles as if it were a vector space [1], even though it is not [9]. While some methods use unit quaternions or rotation matrices to properly represent atttiude, these are also not inherently a vector space and extra steps are required to orthonormalize or otherwise force the attitude to stay on the manifold [2] [3]. The proposed solution is derived from Lie theory and care is taken to ensure that all vector manipulations are done with appropriate vector quantities so that the state remains on the manifold.

Sec. II explains the multirotor UAV model used in the proposed controller formulation. Sec. III presents traditional LQR theory and shows how the proposed LQR formulation is a natural extension when care is taken to apply Lie theory to the multirotor problem. Sec. IV describes the experiments used to demonstrate the proposed control scheme both in simulation and in hardware. Sec. V discusses the results of the experiments and Sec. VI provides concluding remarks.

The research was supported by the National Science Foundation STTR Phase II: Autonomous Landing of Small Unmanned Aircraft Systems onto Moving Platforms, Award No. IIP-1758678 in partnership with Planck Aerosystems.

[1]The corresponding author can be contacted at michaeldavidfarrell at gmail.com.

[2]Author is with the Department of Mechanical Engineering, Brigham Young University, Provo, UT, 84602, USA.

[3]Author is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT, 84602, USA.

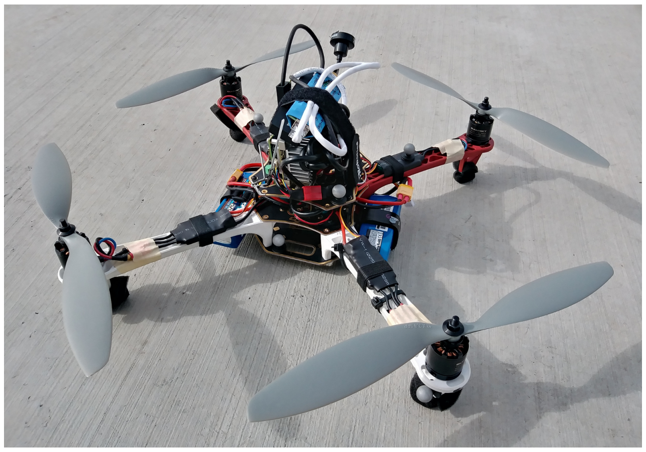

Fig. 1. Hardware platform used in experiments.

II. Model

A. Notation

We define some common notation used throughout the paper, first noting that vectors are represented with a bold letter (e.g., $\mathbf{v}$) and matrices with a capital letter (e.g., A).

R_a^b	Rotation matrix from reference frame a to b
$\mathbf{v}_{a/b}^c$	Vector state $\mathbf{v}$ of frame a w.r.t. frame b, expressed in frame c
$\check{a}$	Desired value of a
$\dot{a}$	Time derivative of a
$\tilde{a}$	Error of variable a, i.e., $\tilde{a} \triangleq a - \check{a}$

We also define the following coordinate frames:

I	The inertial coordinate frame in north-east-down
ℓ	The aircraft's vehicle-1 (body-level) coordinate frame
b	The aircraft's body-fixed coordinate frame

We make frequent use of the skew-symmetric matrix operator defined by

$$[\mathbf{v}]_\times \triangleq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{1}$$

which is related to the cross-product between two vectors as

$$\mathbf{v} \times \mathbf{w} = [\mathbf{v}]_\times \mathbf{w}. \tag{2}$$

We also use the standard basis vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N$, where $\mathbf{e}_1 = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}^\top$ and so forth.

B. Quaternion Representation

A quaternion $\mathbf{q}$ is a hyper-complex number of rank four. It is well known that a unit quaternion $\in S^3$ can be used to efficiently represent attitude, as S^3 is a double cover of $SO(3)$. Quaternions have the advantage over $SO(3)$ of being more efficient to implement on modern hardware [10], therefore in the software implementation of the described algorithm, we use quaternions, rather than rotation matrices.

We use Hamiltonian notation for unit quaternions $\in S^3$

$$\mathbf{q} = \begin{pmatrix} q_0 & q_x i & q_y j & q_z k \end{pmatrix} \tag{3}$$

and define the complex numbers i, j, and k, such that

$$\begin{aligned} ij &= -ji &&= k, \\ jk &= -kj &&= i, \\ ki &= -ik &&= j, \\ i^2 = j^2 = k^2 &= ijk &&= -1. \end{aligned} \tag{4}$$

For convenience, we sometimes refer to the complex portion of the quaternion as

$$\bar{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top \tag{5}$$

and write quaternions as the tuple of the real and complex portions

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{\mathbf{q}} \end{pmatrix}. \tag{6}$$

Given our use of the Hamiltonian notation, the quaternion group operator $\otimes$ can be written as the following matrix-like product

$$\mathbf{q}^a \otimes \mathbf{q}^b = \begin{pmatrix} -q_0^a & (-\bar{\mathbf{q}}^a)^\top \\ \bar{\mathbf{q}}^a & q_0^a I + [\bar{\mathbf{q}}^a]_\times \end{pmatrix} \begin{pmatrix} q^b \\ \bar{\mathbf{q}}^b \end{pmatrix}. \tag{7}$$

It is often convenient to convert a quaternion $\mathbf{q}$ to its associated passive rotation matrix. This is done with

$$R(\mathbf{q}) = \left(2q_0^2 - 1\right) I - 2q_0 [\bar{\mathbf{q}}]_\times + 2\bar{\mathbf{q}}\bar{\mathbf{q}}^\top \in SO(3). \tag{8}$$

We also need to frequently convert between the Lie group, S^3, and the Lie algebra, $\mathbb{R}^3$, which enables us to operate in a vector space. This is done with the exponential and logarithmic mappings. The exponential mapping for a unit quaternion is defined as

$$\begin{aligned} \exp_{\mathbf{q}} &: \mathbb{R}^3 \to S^3 \\ \exp_{\mathbf{q}}(\boldsymbol{\delta}) &\triangleq \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{\delta}\|}{2}\right) \\ \sin\left(\frac{\|\boldsymbol{\delta}\|}{2}\right) \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|} \end{bmatrix}, \end{aligned} \tag{9}$$

with the corresponding logarithmic map defined as

$$\begin{aligned} \log_{\mathbf{q}} &: S^3 \to \mathbb{R}^3 \\ \log_{\mathbf{q}}(\mathbf{q}) &\triangleq 2 \operatorname{atan2}(\|\bar{\mathbf{q}}\|, q_0) \frac{\bar{\mathbf{q}}}{\|\bar{\mathbf{q}}\|}. \end{aligned} \tag{10}$$

To avoid numerical issues when $\|\boldsymbol{\delta}\| \approx 0$, we also employ the small-angle approximations of the quaternion exponential and logarithm

$$\exp_{\mathbf{q}}(\boldsymbol{\delta}) \approx \begin{bmatrix} 1 \\ \frac{\boldsymbol{\delta}}{2} \end{bmatrix} \tag{11}$$

$$\log_{\mathbf{q}}(\mathbf{q}) \approx 2 \operatorname{sign}(q_0) \bar{\mathbf{q}}. \tag{12}$$

We also note that rotations may be written equivalently as $\mathbf{q}_a^b = R\left(\mathbf{q}_a^b\right) = R_a^b$, where the choice of these is dictated by convenience. We use passive rotation matrices, meaning that the rotation matrix R_a^b acts on a vector $\mathbf{r}^a$, expressed in frame a, and results in the same vector, now expressed in frame b as

$$\mathbf{r}^b = R_a^b \mathbf{r}^a. \tag{13}$$

C. Quadrotor Dynamics

If we define the state of a quadrotor as the tuple of position, velocity, and attitude

$$\mathbf{x} = \left(\mathbf{p}_{b/I}^I, \mathbf{v}_{b/I}^b, \mathbf{q}_I^b\right) \in \mathbb{R}^3 \times \mathbb{R}^3 \times S^3$$

and the input to our system as the tuple of the throttle signal, s, and angular velocity, $\boldsymbol{\omega}_{b/I}^b$,

$$\mathbf{u} = \left(s, \boldsymbol{\omega}_{b/i}^b\right) \in \mathbb{R}^1 \times \mathbb{R}^3,$$

then the rigid body dynamics of a multirotor UAV are as follows [11]:

$$\begin{aligned} \dot{\mathbf{p}}_{b/I}^I &= \left(R_I^b\right)^\top \mathbf{v}_{b/I}^b \\ \dot{\mathbf{v}}_{b/I}^b &= g R_I^b \mathbf{e}_3 - g \frac{s}{s_e} \mathbf{e}_3 - c_d \left(I - \mathbf{e}_3 \mathbf{e}_3^\top\right) \mathbf{v}_{b/I}^b - \\ &\quad \left[\boldsymbol{\omega}_{b/I}^b\right]_\times \mathbf{v}_{b/I}^b \\ \dot{\mathbf{q}}_I^b &= \mathbf{q}_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\omega_{b/I}^b \end{pmatrix}, \end{aligned} \tag{14}$$

where c_d is a linear drag constant, s_e is the throttle command required to hover, and g is the magnitude of gravity. This model assumes a linear relationship between throttle signal and thrust, which is not always the case. Although we use this simple model, more sophisticated approaches, such as [12] estimate this relationship online and compensate for it in real time.

D. Error-State Dynamics

It is useful to consider what is known as the *error-state* of the quadrotor. This concept has a long history in state estimation and is used in the error-state Kalman filter [5], [13]. The error-state is used in state estimation as a principled way to represent the covariance about attitude in terms of a vector space, as opposed to some local approximation. This relationship is also useful in control for the same reason. Performing control in the vector space of error-state provides a principled way to leverage well-understood and efficient linear algebra machinery to solve control problems over non-vector quantities, such as attitude.

We define the error-state of some quantity $\mathbf{y}$ as

$$\tilde{\mathbf{y}} = \mathbf{y} \boxminus \check{\mathbf{y}}, \tag{15}$$

where $\boxminus$ is an appropriate difference operator, as described by [14]. For instance, if $\mathbf{y}$, $\check{\mathbf{y}} \in \mathbb{R}^n$, the $\boxminus$ operator may be defined as the vector subtraction operator. However, due to the attitude component of our state, the vector subtraction operator is not defined between $\mathbf{x}$ and $\check{\mathbf{x}}$. We instead define the error-state piecewise for each component of the state and combine these into an error-state vector

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{p}}^I_{b/I} & \tilde{\mathbf{v}}^b_{b/I} & \tilde{\mathbf{r}}^b_I \end{bmatrix}^\top \in \mathbb{R}^{9\times 1}, \tag{16}$$

where $\tilde{\mathbf{p}}^I_{b/I}$ is the error-state associated with position, $\tilde{\mathbf{v}}^b_{b/I}$ is the error-state associated with velocity, and $\tilde{\mathbf{r}}^b_I$ is the error-state associated with attitude.

In our case, the error-states associated with position and velocity are simply defined using vector subtraction

$$\tilde{\mathbf{p}}^I_{b/I} = \mathbf{p}^I_{b/I} - \check{\mathbf{p}}^I_{b/I} \tag{17}$$

$$\tilde{\mathbf{v}}^b_{b/I} = \mathbf{v}^b_{b/I} - \check{\mathbf{v}}^b_{b/I}, \tag{18}$$

however, the error-state associated with attitude is more complicated.

It is commonly understood that any representation of attitude has three underlying degrees of freedom. A unit quaternion has four parameters, but its error can be described in terms of three degrees of freedom that we wish to represent as a vector quantity. In a neighborhood sufficiently close to the identity, these behave similarly to the Euler angle representation of roll, pitch, and yaw. However, Euler angles are not a vector because the sequential rotation method used to define Euler angles nonlinearly couples the three degrees of freedom. Therefore, we define the vector

$$\mathbf{r}^b_I(t) = \mathbf{r}^b_I(t_0) + \int_{t_0}^{t} \boldsymbol{\omega}^b_{b/I}(\tau)\, d\tau, \tag{19}$$

such that $\mathbf{r}^b_I(t_0) = 0$ and $\dot{\mathbf{r}}^b_I = \boldsymbol{\omega}^b_{b/I}$. With this definition, we can use (9) and (10) to express

$$\mathbf{q}^b_I = \check{\mathbf{q}}^b_I \otimes \exp_{\mathbf{q}}(\tilde{\mathbf{r}}) \tag{20}$$

$$\tilde{\mathbf{r}} = \log_{\mathbf{q}}\left(\left(\check{\mathbf{q}}^b_I\right)^{-1} \otimes \mathbf{q}^b_I\right), \tag{21}$$

as described by [14].

Even though $\mathbf{r}^b_I$ is a vector, we cannot simply compute the error-state as $\tilde{\mathbf{r}}^b_I = \mathbf{r}^b_I - \check{\mathbf{r}}^b_I$ because $\mathbf{r}^b_I$ is a minimal representation of $\mathbf{q}^b_I$, which is a double cover of the Lie group $SO(3)$. Vector subtraction of members in this group is not valid. However, the derivative of $\mathbf{r}^b_I$ exists in the tangent space of $SO(3)$, so we can perform

$$\dot{\tilde{\mathbf{r}}}^b_I = \dot{\mathbf{r}}^b_I - R^b_I \left(\check{R}^b_I\right)^\top \dot{\check{\mathbf{r}}}^b_I, \tag{22}$$

where $R^b_I \left(\check{R}^b_I\right)^\top$ moves the desired vector derivative, $\dot{\check{\mathbf{r}}}^b_I$, from its own tangent space to the tangent space of $\dot{\mathbf{r}}^b_I$. With both vectors in the same tangent space, the vector subtraction in (22) is valid.

For use in control, we similarly define an error-state for the control input with the error-state being the difference between the current control input and some reference input. Using the same definition as in (15), we can see that

$$\tilde{s} = s - \check{s} \tag{23}$$

$$\tilde{\boldsymbol{\omega}}^b_{b/I} = \boldsymbol{\omega}^b_{b/I} - \check{\boldsymbol{\omega}}^b_{b/I} \tag{24}$$

where $\check{s}$ and $\check{\boldsymbol{\omega}}^b_{b/I}$ are respectively the reference throttle signal and reference angular velocity. Note that we do not model the dynamic response to these inputs. Instead, our model assumes that the multirotor instantaneously reaches any commanded throttle and angular velocity.

Using the error-state definitions above, we can derive the error-state dynamics of the quadrotor as

$$\begin{aligned}
\dot{\tilde{\mathbf{p}}}^I_{b/I} &= \left(R^b_I\right)^\top \tilde{\mathbf{v}}^b_{b/I} - \left(R^b_I\right)^\top \left[\mathbf{v}^b_{b/I}\right]_\times \tilde{\mathbf{r}}^b_I \\
\dot{\tilde{\mathbf{v}}}^b_{b/I} &= g\left[R^b_I \mathbf{e}_3\right]_\times \tilde{\mathbf{r}}^b_I - g\frac{\tilde{s}}{s_e}\mathbf{e}_3 - c_d\left(I - \mathbf{e}_3\mathbf{e}_3^\top\right)\tilde{\mathbf{v}}^b_{b/I} \\
&\quad - \left[\boldsymbol{\omega}^b_{b/I}\right]_\times \tilde{\mathbf{v}}^b_{b/I} + \left[\mathbf{v}^b_{b/I}\right]_\times \tilde{\boldsymbol{\omega}}^b_{b/I} \\
\dot{\tilde{\mathbf{r}}}^b_I &= \tilde{\boldsymbol{\omega}}^b_{b/I} - \left[\boldsymbol{\omega}^b_{b/I}\right]_\times \tilde{\mathbf{r}}^b_I,
\end{aligned} \tag{25}$$

or succinctly,

$$\dot{\tilde{\mathbf{x}}} = f\left(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}\right). \tag{26}$$

The derivation of these error-state dynamics can be found in the Appendix.

III. LQR Control

A. Traditional LQR

A linear-quadratic regulator provides the optimal state space controller gains for an LTI system given by

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{27}$$

assuming full-state feedback. We define the cost-to-go for the infinite-time solution as

$$J(\mathbf{x}, \mathbf{u}) = \int_0^\infty \left(\mathbf{x}^\top Q\mathbf{x} + \mathbf{u}R\mathbf{u}\right) dt \tag{28}$$

with Q and R being positive definite matrices that define the costs associated with the state and the input. The cost function given in (28) is minimized by the control input

$$\mathbf{u} = -K\mathbf{x}, \tag{29}$$

where K is given by

$$K = R^{-1} B^\top P, \tag{30}$$

and P is the solution to the Continuous-time Algebraic Riccati Equation (CARE),

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0. \tag{31}$$

It should be noted that in its basic form, an LQR controller is simply a regulator and the control input $\mathbf{u}$ will only attempt to drive the state to zero in an optimal way. If the desire is for the system to reach a desired state, $\check{\mathbf{x}}$, one can start by defining the error-state as

$$\tilde{\mathbf{x}} = \mathbf{x} - \check{\mathbf{x}} \tag{32}$$

and redefining the control input as

$$\tilde{\mathbf{u}} = -K\tilde{\mathbf{x}}. \tag{33}$$

This technique, however, will generally result in steady-state error between the state $\mathbf{x}$ and the reference trajectory $\check{\mathbf{x}}$. The steady-state error can be removed by augmenting the state with an integrator or by applying a model-based feed-forward control input,

$$\mathbf{u} = \tilde{\mathbf{u}} + \check{\mathbf{u}} = -K\tilde{\mathbf{x}} + \check{\mathbf{u}}. \tag{34}$$

A direct application of (32) in our case is not defined because the multirotor state is not a vector quantity. To compensate for this, we propose to compute control based on the error-state dynamics of the system, where the error-state is purely a vector quantity.

B. Error-State LQR

We can apply the same LQR approach to the error-state dynamics from (25). Since LQR control is a regulator, it will drive the error-state to zero, or our current state to our desired state. Since LQR is only defined for an LTI system, we can approximate the error-state system as an LTI system by linearizing about the current state at each time step. This gives us the system

$$\dot{\tilde{\mathbf{x}}} = A\tilde{\mathbf{x}} + B\tilde{\mathbf{u}} \tag{35}$$

with the matrices A and B given by

$$A(\mathbf{x}, \mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{x}}} f(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}) \tag{36}$$

$$B(\mathbf{x}, \mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{u}}} f(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}). \tag{37}$$

Using the error-state dynamics in (25) and dropping the subscripts and superscripts for compactness it can be seen that

$$A(\mathbf{x}, \mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{x}}} f(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}) \tag{38}$$

$$= \begin{bmatrix} \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{v}}} & \frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{r}}} \\ \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{v}}} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{r}}} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\mathbf{r}}} \end{bmatrix} \tag{39}$$

with the individual components given by

$$\frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{v}}} = \left(R_I^b\right)^\mathsf{T} \tag{40}$$

$$\frac{\partial \dot{\tilde{\mathbf{p}}}}{\partial \tilde{\mathbf{r}}} = -\left(R_I^b\right)^\mathsf{T} \left[\mathbf{v}_{b/I}^b\right]_\times \tag{41}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{v}}} = -c_d \left(I - \mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right) - \left[\boldsymbol{\omega}_{b/I}^b\right]_\times \tag{42}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\mathbf{r}}} = \left[gR_I^b\mathbf{e}_3\right]_\times \tag{43}$$

$$\frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\mathbf{r}}} = -\left[\boldsymbol{\omega}_{b/I}^b\right]_\times . \tag{44}$$

It can similarly be seen that

$$B(\mathbf{x}, \mathbf{u}) = \frac{\partial}{\partial \tilde{\mathbf{u}}} f(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{u}, \tilde{\mathbf{u}}) \tag{45}$$

$$= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{s}} & \frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\omega}} \\ \mathbf{0} & \frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\omega}} \end{bmatrix} \tag{46}$$

with the individual components given by

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{s}} = -\frac{g}{s_e}\mathbf{e}_3 \tag{47}$$

$$\frac{\partial \dot{\tilde{\mathbf{v}}}}{\partial \tilde{\omega}} = \left[\mathbf{v}_{b/I}^b\right]_\times \tag{48}$$

$$\frac{\partial \dot{\tilde{\mathbf{r}}}}{\partial \tilde{\omega}} = I_{3\times 3}. \tag{49}$$

By linearizing at every time step, the CARE must be solved at each time step with the current A and B matrices. We use the closed form, Schur decomposition method described in [15]. This method allows us to relinearize and recompute the optimal control at full rate in our experiments.

Although we relinearize and solve the CARE at each time step, the Q and R gain matrices are fixed. We choose these gains based on Bryson's rule [16]. In addition, we have found that better results are achieved by saturating the error-state in accordance with the maximum error terms used to choose the gains with Bryson's rule.

IV. EXPERIMENT

To test the proposed error-state LQR controller, we designed two experiments to be performed in simulation and hardware: (i) tracking step inputs to the desired position of the UAV and (ii) tracking time-dependent full-state trajectories. We first explain how we generate these time-dependent trajectories for the experiments and then detail the experimental setup for both simulation and hardware.

A. Trajectory Generation

One important consideration in high-performance control of quadrotors is the generation of smooth, feasible trajectories. The quadrotor has the benefit of being *differentially flat* which means that the required inputs to the quadrotor can be fully defined using derivatives of the outputs of the system, the desired position and heading [17]. If we are given some smooth, differentiable trajectory of our desired position and heading then we can compute the full state and required inputs as a function of time

$$\begin{pmatrix} \check{\mathbf{p}}^I_{b/I}(t) \\ \check{\mathbf{q}}^b_I(t) \\ \check{\mathbf{v}}^I_{b/I}(t) \\ \check{s}(t) \\ \check{\boldsymbol{\omega}}^I_{b/I}(t) \end{pmatrix} = \check{f}\begin{pmatrix} \check{\mathbf{p}}^I_{b/I}(t) \\ \dot{\check{\mathbf{p}}}^I_{b/I}(t) \\ \ddot{\check{\mathbf{p}}}^I_{b/I}(t) \\ \check{\psi}^I_{b/I}(t) \end{pmatrix}. \tag{50}$$

We derive the general case where the desired yaw angle of the multirotor UAV is a function of time. However, since it is well known that the yaw angle of a multirotor UAV is easily controllable independent of the other states [17], we simply command a constant zero yaw in our experiments.

We now derive the differentially flat outputs. First, desired position is given to us directly

$$\check{\mathbf{p}}^I_{b/I} = \check{\mathbf{p}}^I_{b/I}. \tag{51}$$

To derive desired attitude and throttle signal we start by applying Newton's second law, and consider the rotation from the heading-rotated, body-level coordinate frame, ℓ, to the body frame, b. Note that to avoid the need for an iterative solution, we neglect the forces due to drag that are accounted for in the quadrotor dynamics in (14). Newton's second law is given by

$$\sum F^I = m\ddot{\check{\mathbf{p}}}^I_{b/I} \tag{52}$$

$$-T\left(\check{R}^b_\ell\right)^\top + mg\mathbf{e}_3 = m\ddot{\check{\mathbf{p}}}^I_{b/I} \tag{53}$$

$$\frac{T}{m}\left(\check{R}^b_\ell\right)^\top \mathbf{e}_3 = g\mathbf{e}_3 - \ddot{\check{\mathbf{p}}}^I_{b/I}. \tag{54}$$

If we then define $\check{\mathbf{a}} = g\mathbf{e}_3 - \ddot{\check{\mathbf{p}}}^I_{b/I}$, then we get the following expression

$$\frac{T}{m}\left(\check{R}^b_\ell\right)^\top \mathbf{e}_3 = \check{\mathbf{a}} \tag{55}$$

where T and $\check{R}^b_\ell$ must satisfy the following conditions:

$$\frac{T}{m} = \|\check{\mathbf{a}}\| \tag{56}$$

$$\check{R}^I_\ell \mathbf{e}_3 = \frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}. \tag{57}$$

Because $T = \frac{s}{s_e}gm$, then

$$\check{s} = \frac{s_e}{g}\|\check{\mathbf{a}}\| \tag{58}$$

and (57) can be solved with

$$\check{\mathbf{q}}^b_\ell = \exp_{\mathbf{q}}(\theta\boldsymbol{\delta}) \tag{59}$$

where

$$\theta = \cos^{-1}\left(\mathbf{e}_3^\top \frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}\right) \tag{60}$$

$$\boldsymbol{\delta} = [\mathbf{e}_3]_\times \frac{\check{\mathbf{a}}}{\|\check{\mathbf{a}}\|}. \tag{61}$$

The heading portion of attitude can now be applied to give us our full desired attitude

$$\check{\mathbf{q}}^b_I = \exp_{\mathbf{q}}\left(\check{\psi}\mathbf{e}_3\right) \otimes \check{\mathbf{q}}^b_\ell. \tag{62}$$

Desired velocity can be found using the desired attitude

$$\check{\mathbf{v}}^b_{b/I} = \check{R}^b_I \dot{\check{\mathbf{p}}}^I_{b/I}, \tag{63}$$

and the required angular rate is found by taking the time derivative of our desired attitude

$$\check{\boldsymbol{\omega}}^b_{b/I} = \frac{d}{dt}\check{\mathbf{q}}^b_I. \tag{64}$$

In our implementation, we do this numerically with central differencing on the manifold.

In summary, the differentially flat output of a quadrotor is given as follows:

$$\check{\mathbf{p}}^I_{b/I} = \check{\mathbf{p}}^I_{b/I} \tag{65}$$

$$\check{\mathbf{q}}^b_I = \exp_{\mathbf{q}}\left(\check{\psi}\mathbf{e}_3\right) \otimes \check{\mathbf{q}}^b_\ell \tag{66}$$

$$\check{\mathbf{v}}^b_{b/I} = \check{R}^b_I \dot{\check{\mathbf{p}}}^I_{b/I} \tag{67}$$

$$\check{s} = \frac{s_e}{g}\left\|g\mathbf{e}_3 - \ddot{\check{\mathbf{p}}}^I_{b/I}\right\| \tag{68}$$

$$\check{\boldsymbol{\omega}}^b_{b/I} = \frac{d}{dt}\check{\mathbf{q}}^b_I. \tag{69}$$

The examples in this work all reference the same figure-eight trajectory defined by

$$\check{\mathbf{p}}^I_{b/I}(t) = \mathbf{p}^I_{b/I}(t_0) + \begin{bmatrix} \delta_x \sin\left(\frac{T}{2\pi}t\right) \\ \delta_y \sin\left(\frac{T}{\pi}t\right) \\ \delta_z \sin\left(\frac{T}{2\pi}t\right) \end{bmatrix} \tag{70}$$

$$\check{\psi}^I_{b/I}(t) = 0 \tag{71}$$

where the $\delta_{(\cdot)}$ parameters define the dimensions of the trajectory, and T defines the period. While a trajectory defined by periodic functions is useful for simple demonstrations such as what we perform in this work, we direct the reader to more sophisticated methods of differentiable trajectory generation such as [17] for practical application.

B. Simulation

For simulation, we use Gazebo[1] and ROS[2] with the ROSflight software-in-the-loop (SIL) simulation [18]. This simulation setup allows us to test the exact code that also runs in hardware.

C. Hardware

A custom multirotor UAV built on a DJI 450 Flamewheel frame with an STM32F1 microcontroller running the ROSflight [18] flight control firmware was flown for the hardware experiments. The algorithm was implemented and run in real time onboard on an NVIDIA Jetson TX2. Though the TX2 has a GPU, all computation is done using only the ARM CPU, showing that this algorithm can also run at full rate on a variety of popular onboard computers. The multirotor UAV was flown in a small motion capture room with feedback from an OptiTrack[3] motion tracking system. The global position and attitude measurements from the motion capture system are fused in real time with the onboard IMU of the

[1]Gazebo: www.gazebosim.org
[2]Robot Operating System: www.ros.org
[3]OptiTrack: www.optitrack.com

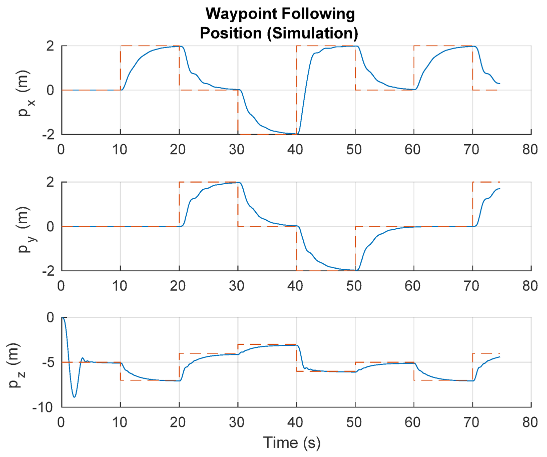

Fig. 2. Simulation results for the position of the multirotor UAV given step inputs to position. The red dotted line is the desired position and the blue solid line is the estimated position.

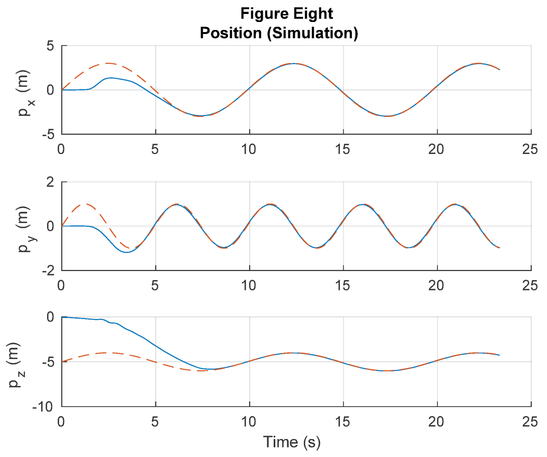

Fig. 3. Simulation results of a multirotor UAV tracking a figure eight trajectory. The red dotted line is the desired position and the blue solid line is the estimated position.

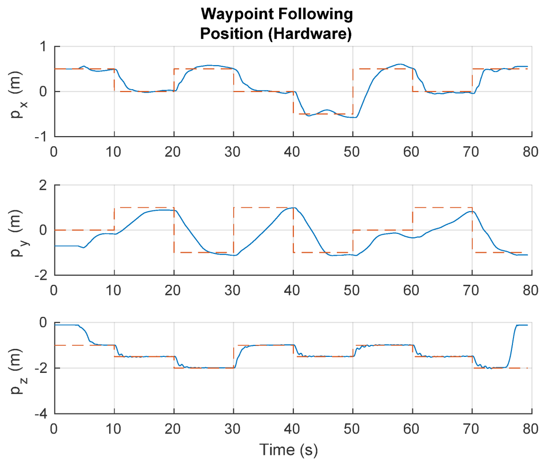

Fig. 4. Hardware results for the position of the multirotor UAV given step inputs to position. The red dotted line is the desired position and the blue solid line is the estimated position.

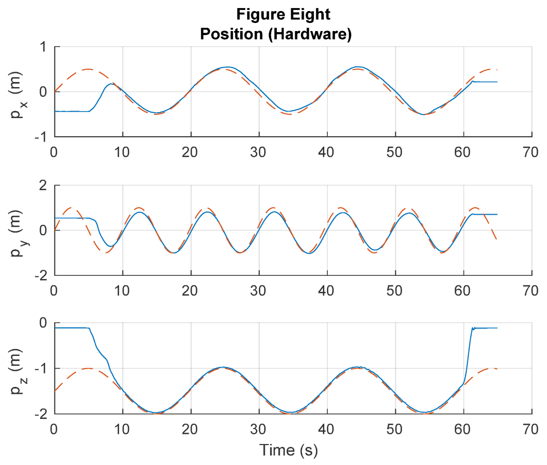

Fig. 5. Hardware results of a multirotor UAV tracking a figure eight trajectory. The red dotted line is the desired position and the blue solid line is the estimated position.

UAV using an extended Kalman filter (EKF) to produce full state estimates.

For added safety, the computed control inputs are saturated before being sent to the flight controller. The throttle signal, s, was saturated to a maximum value of 0.85 and the angular rate commands, $\boldsymbol{\omega}_{b/I}^{b}$, were saturated such that each component $|\omega| \leq 2\,\mathrm{rad}/\mathrm{s}$.

V. Results

A. Simulation

Figs. 2 and 3 show simulation results of the controller following desired position step inputs (waypoints) and a figure-eight trajectory. The multirotor begins the simulation at rest on the ground and converges to the desired trajectory within a few seconds. Note that the figure-eight trajectory tracking is near perfect even though the feed forward inputs computed from (65) - (69) do not account for the force due to drag and the controller does not model thrust dynamics nor torque dynamics.

B. Hardware

During the hardware experiments, the entire control algorithm was run at the full streaming rate of the onboard IMU, which was set to 250 Hz. The computation time of the algorithm was shown to have a mean of 274.6 µs and a standard deviation of 43.84 µs. This shows that the proposed LQR formulation can run at full rate even on computationally constrained platforms.

Fig. 4 shows the multirotor position along with the commanded positions for a waypoint path. The clean step response with minimal overshoot is notable considering we did not hand tune the LQR gains beyond an initial value derived from Bryson's rule. Fig. 5 shows how well the multirotor is able to follow the figure-eight trajectory, and Fig. 6 depicts the same flight plotted in two dimensions. The major deviations visible in this plot depict the take-off and landing portions of the flight.

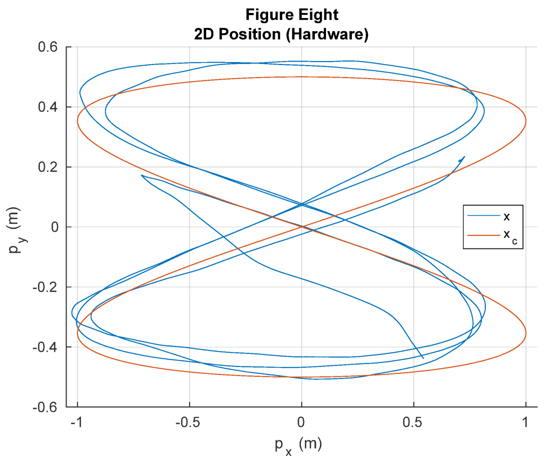

Fig. 6. Top down view of a multirotor UAV tracking a figure eight trajectory in hardware. The major deviation in position indicates the time during take-off when the multirotor must get to the trajectory before following it. The red solid line is the desired position and the blue solid line is the estimated position.

VI. Conclusion

In this work we propose a novel LQR formulation derived from Lie theory. We show that by using the error-state dynamics, we can properly compute control using linear algebra techniques only on appropriate vector quantities. Our implementation is efficient enough to relinearize the system and update the LQR gains in less than one millisecond. Simulation and hardware experiments show the effectiveness and simplicity of this control approach.

Appendix

To derive the error state dynamics of the multirotor UAV we first establish a few identities and approximations. As it is convenient to work with rotation matrices, we first write (20) in terms of rotation matrices. Since rotation matrices concatenate in an order opposite to quaterions, we have

$$R_I^b = R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\check{R}_I^b. \tag{72}$$

With this, we can express the desired attitude as

$$\check{R}_I^b = R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)^\top R_I^b. \tag{73}$$

and when $\tilde{\mathbf{r}}_I^b$ is small, we employ (11) to get

$$\check{R}_I^b \approx R\left(\begin{bmatrix} 1 \\ \frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right)^\top R_I^b \tag{74}$$

$$\approx R\left(\begin{bmatrix} 1 \\ -\frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right) R_I^b. \tag{75}$$

Now, we can substitute $R\left(\begin{bmatrix} 1 \\ -\frac{1}{2}\tilde{\mathbf{r}}_I^b \end{bmatrix}\right)$ into eq. (8) to obtain

$$\check{R}_I^b \approx \left(I + \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) R_I^b. \tag{76}$$

We also note that the transpose is given by

$$\left(\check{R}_I^b\right)^\top = \left(R_I^b\right)^\top \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\right) \tag{77}$$

and can be similarly approximated as

$$\left(\check{R}_I^b\right)^\top \approx \left(R_I^b\right)^\top \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right). \tag{78}$$

We also employ the skew symmetric identity that

$$[\mathbf{a}]_\times \mathbf{b} = -[\mathbf{b}]_\times \mathbf{a}. \tag{79}$$

1) Position: Differentiating the error state for the position term given by (17) we get

$$\dot{\tilde{\mathbf{p}}}_{b/I}^I = \dot{\mathbf{p}}_{b/I}^I - \dot{\check{\mathbf{p}}}_{b/I}^I. \tag{80}$$

Substituting in the multirotor dynamics from (14) and simplifying we get

$$\dot{\tilde{\mathbf{p}}}_{b/I}^I = \left(R_I^b\right)^\top \mathbf{v}_{b/I}^b - \left(\check{R}_I^b\right)^\top \check{\mathbf{v}}_{b/I}^b \tag{81}$$

$$= \left(R_I^b\right)^\top \mathbf{v}_{b/I}^b - \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right) R_I^b\right)^\top \check{\mathbf{v}}_{b/I}^b \tag{82}$$

$$= \left(R_I^b\right)^\top \mathbf{v}_{b/I}^b - \left(R_I^b\right)^\top \left(R\left(\exp_{\mathbf{q}}\left(\tilde{\mathbf{r}}_I^b\right)\right)\right)^\top \left(\mathbf{v}_{b/I}^b - \tilde{\mathbf{v}}_{b/I}^b\right) \tag{83}$$

$$= \left(R_I^b\right)^\top \mathbf{v}_{b/I}^b - \left(R_I^b\right)^\top \left(I - \left[\tilde{\mathbf{r}}_I^b\right]_\times\right) \left(\mathbf{v}_{b/I}^b - \tilde{\mathbf{v}}_{b/I}^b\right). \tag{84}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\dot{\tilde{\mathbf{p}}}_{b/I}^I = \left(R_I^b\right)^\top \tilde{\mathbf{v}}_{b/I}^b - \left(R_I^b\right)^\top \left[\mathbf{v}_{b/I}^b\right]_\times \tilde{\mathbf{r}}_I^b. \tag{85}$$

2) Velocity: Differentiating the error state for the velocity term given by (18) we get

$$\dot{\tilde{\mathbf{v}}}_{b/I}^b = \dot{\mathbf{v}}_{b/I}^b - \dot{\check{\mathbf{v}}}_{b/I}^b. \tag{86}$$

Substituting in the multirotor dynamics from (14) and simplifying we get

$$\begin{aligned}\dot{\tilde{\mathbf{v}}}^b_{b/I} = &\left(gR^b_I\mathbf{e}_3 - g\frac{s}{s_e}\mathbf{e}_3\right.\\ &\left.-c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\mathbf{v}^b_{b/I} - \left[\boldsymbol{\omega}^b_{b/I}\right]_\times \mathbf{v}^b_{b/I}\right)\\ &-\left(g\check{R}^b_I\mathbf{e}_3 - g\frac{\check{s}}{s_e}\mathbf{e}_3\right.\\ &\left.-c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\check{\mathbf{v}}^b_{b/I} - \left[\check{\boldsymbol{\omega}}^b_{b/I}\right]_\times \check{\mathbf{v}}^b_{b/I}\right)\end{aligned} \tag{87}$$

$$\begin{aligned}= &\left(gR^b_I\mathbf{e}_3 - g\check{R}^b_I\mathbf{e}_3\right) - \left(g\frac{s}{s_e}\mathbf{e}_3 - g\frac{\check{s}}{s_e}\mathbf{e}_3\right)\\ &-\left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\mathbf{v}^b_{b/I}\right.\\ &\left.-c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\check{\mathbf{v}}^b_{b/I}\right)\\ &-\left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times\mathbf{v}^b_{b/I} - \left[\check{\boldsymbol{\omega}}^b_{b/I}\right]_\times\check{\mathbf{v}}^b_{b/I}\right)\end{aligned} \tag{88}$$

$$\begin{aligned}\approx &\left(gR^b_I\mathbf{e}_3 - g\left(I+\left[\tilde{\mathbf{r}}^b_I\right]_\times\right)R^b_I\mathbf{e}_3\right)\\ &-\left(g\frac{\tilde{s}}{s_e}\mathbf{e}_3\right) - \left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I}\right)\\ &-\left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times\mathbf{v}^b_{b/I}\right.\\ &\left.-\left[\left(\boldsymbol{\omega}^b_{b/I}-\tilde{\boldsymbol{\omega}}^b_{b/I}\right)\right]_\times\left(\mathbf{v}^b_{b/I}-\tilde{\mathbf{v}}^b_{b/I}\right)\right)\end{aligned} \tag{89}$$

$$\begin{aligned}= &\left(-g\left[\tilde{\mathbf{r}}^b_I\right]_\times R^b_I\mathbf{e}_3\right)\\ &-\left(g\frac{\tilde{s}}{s_e}\mathbf{e}_3\right) - \left(c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I}\right)\\ &-\left(\left[\boldsymbol{\omega}^b_{b/I}\right]_\times\mathbf{v}^b_{b/I}\right.\\ &\left.-\left[\left(\boldsymbol{\omega}^b_{b/I}-\tilde{\boldsymbol{\omega}}^b_{b/I}\right)\right]_\times\left(\mathbf{v}^b_{b/I}-\tilde{\mathbf{v}}^b_{b/I}\right)\right).\end{aligned} \tag{90}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\begin{aligned}\dot{\tilde{\mathbf{v}}}^b_{b/I} = &\, g\left[R^b_I\mathbf{e}_3\right]_\times\tilde{\mathbf{r}}^b_I - g\frac{\tilde{s}}{s_e}\mathbf{e}_3\\ &-c_d\left(I-\mathbf{e}_3\mathbf{e}_3^\mathsf{T}\right)\tilde{\mathbf{v}}^b_{b/I}\\ &-\left[\boldsymbol{\omega}^b_{b/I}\right]_\times\tilde{\mathbf{v}}^b_{b/I} + \left[\mathbf{v}^b_{b/I}\right]_\times\tilde{\boldsymbol{\omega}}^b_{b/I}.\end{aligned} \tag{91}$$

3) Attitude: To derive the error state dynamics corresponding to attitude, we start with (22). From (19) we see that $\dot{\mathbf{r}}^b_I = \boldsymbol{\omega}^b_{b/I}$. Substituting this defintion into (22) we get

$$\dot{\tilde{\mathbf{r}}}^b_I = \boldsymbol{\omega}^b_{b/I} - R^b_I\left(\check{R}^b_I\right)^\mathsf{T}\check{\boldsymbol{\omega}}^b_{b/I}. \tag{92}$$

We can simplify this expression and show that

$$\dot{\tilde{\mathbf{r}}}^b_I = \boldsymbol{\omega}^b_{b/I} - R^b_I\left(R^b_I\right)^\mathsf{T}\left(R\left(\exp_\mathbf{q}\left(\tilde{\mathbf{r}}^b_I\right)\right)\right)^\mathsf{T}\check{\boldsymbol{\omega}}^b_{b/I} \tag{93}$$

$$\approx \boldsymbol{\omega}^b_{b/I} - R^b_I\left(R^b_I\right)^\mathsf{T}\left(I-\left[\tilde{\mathbf{r}}^b_I\right]_\times\right)\check{\boldsymbol{\omega}}^b_{b/I} \tag{94}$$

$$= \boldsymbol{\omega}^b_{b/I} - \left(I-\left[\tilde{\mathbf{r}}^b_I\right]_\times\right)\left(\boldsymbol{\omega}^b_{b/I}-\tilde{\boldsymbol{\omega}}^b_{b/I}\right). \tag{95}$$

By simplifying and neglecting higher-order terms, we get the final expression

$$\dot{\tilde{\mathbf{r}}}^b_I = \tilde{\boldsymbol{\omega}}^b_{b/I} - \left[\boldsymbol{\omega}^b_{b/I}\right]_\times\tilde{\mathbf{r}}^b_I. \tag{96}$$

References

[1] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 4001–4008.

[2] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, "LQR control for a quadrotor using unit quaternions: Modeling and simulation," in *Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on*. IEEE, 2013, pp. 172–178.

[3] P. Foehn and D. Scaramuzza, "Onboard state dependent LQR for agile quadrotors," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6566–6572.

[4] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[5] J. Solà, "Quaternion kinematics for the error-state Kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[6] D. P. Koch, D. O. Wheeler, R. Beard, T. McLain, and K. M. Brink, "Relative multiplicative extended Kalman filter for observable GPS-denied navigation," 2017.

[7] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li, "High performance full attitude control of a quadrotor on SO (3)," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1698–1703.

[8] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[9] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," 2006.

[10] R. T. Casey, M. Karpenko, R. Curry, and G. Elkaim, "Attitude representation and kinematic propagation for low-cost UAVs," *AIAA Guidance, Navigation, and Control (GNC) Conference*, pp. 1–25, 2013. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2013-4615

[11] R. Leishman, J. Macdonald, R. Beard, and T. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems Magazine*, vol. 34, no. 1, pp. 28–41, Feb 2014.

[12] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," *arXiv e-prints*, p. arXiv:1812.04755, Dec 2018.

[13] F. L. Markley, "Multiplicative vs. additive filtering for spacecraft attitude determination," 2004.

[14] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[15] A. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Transactions on automatic control*, vol. 24, no. 6, pp. 913–921, 1979.

[16] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2018.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[18] J. Jackson, G. Ellingson, and T. McLain, "ROSflight: A lightweight, inexpensive MAV research and development tool," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 758–762.