16.31 Final Project Paper: Minimum-Snap Trajectory Generator with Error-State LQR Control of a Quadrotor MAV

Andrew Torgesen

***Abstract*—In an attempt to increase the agility of the Parrot Mambo quadrotor platform in tracking smooth, continuously varying position trajectories, the Simulink flight control system is augmented with a full-state trajectory generator, error-state LQR controller, and an updated attitude controller. The trajectory generator, which takes advantage of the differential flatness of multirotor dynamics, is able to generate a full-state trajectory from position, velocity, acceleration, and jerk commands. The error-state LQR and attitude controllers allow the quadrotor to follow the generated reference trajectory with greater accuracy than the default Simulink flight control system for the Parrot Mambo. Explanations and derivations for the Lie derivatives used for the error-state LQR are given. Simulation and hardware results are used to validate the performance of the augmented flight control system.**

I. Introduction

Experience has shown that quadrotor flight routines such as waypoint tracking can be satisfactorily executed with position and orientation-based successive loop closure. However, more acrobatic flight maneuvers which track continuous reference trajectories are better served with a control scheme that goes beyond purely position-based control.

Quadrotor flight performance for continuously varying trajectories is highly dependent on "supplementary" reference commands which leverage the quadrotor dynamics themselves. When reference trajectories are generated which take acceleration and jerk into account, for example, more agility is afforded as the time derivative of position encodes a sort of "anticipation" of the subsequent commanded change in position in the next step of the trajectory. Moreover, it will be shown in this paper that a set of commanded positions, velocities, accelerations, and jerks can be translated into a full-state trajectory command that, when tracked by a quadrotor, achieves the desired time derivatives of position while minimizing snap.

Full-state trajectory commands naturally lend themselves to tracking with a full-state feedback controller. Linear quadratic regulation, or LQR, is the optimal full-state feedback controller for linear systems. Quadrotor dynamics are inherently nonlinear, though they can be linearized and treated as a linear system if the state does not stray too far from the linearization point. This project considers an alternative formulation for the dynamics by expressing them in terms of the error-state. Interestingly, error-state dynamics are linear, and can thus be more soundly controlled by LQR. In order to derive an error-state LQR controller, however, some background in Lie theory is required. This paper delves into the relevant Lie theory and error-state LQR controller derivation.

This paper leverages the concepts of full-state trajectory generation and error-state LQR to augment the Parrot Mambo Simulink control system for more agile smooth trajectory tracking. The three main augmentations to the flight control system are visualized in Figure 1. Together, they consist what will be referred to as a full-state **T**rajectory **G**enerator and **E**rror-state **LQR** controller (**TG-ELQR**).

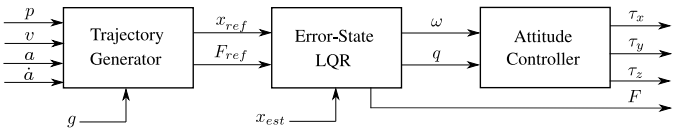

Figure 1: TG-ELQR architecture. The flight control system takes as input commands a reference position, velocity, acceleration, and jerk, and outputs commanded body torques and thrust for more acrobatic flight.

Section II gives a brief overview of related work whose results pertain to the various flight control system augmentations presented in this paper. Section III derives the equations for the minimum-snap trajectory generator. Subsequently, Section IV derives the linear error-state dynamics after reviewing the minimum requisite Lie theory. Section V discusses the modifications made to the representation and control methods for attitude in Simulink, and Sections VI and VII communicate the results and conclusions to be drawn from simulation and hardware testing of the TG-ELQR control system.

II. Related Work

The algorithms that are synthesized together in this paper come from the work of many different sources in the robotics community. A summary of those sources and their context within control theory is given in this section.

The concept of differential flatness is both explained and utilized for full-state trajectory generation in [9], [6]. It is a well-known phenomenon that is often used in multirotor control. The minimum-snap trajectory generator used in this paper is inspired by the work laid out in [6], [2].

Lie theory, which is used in the derivation of the error-state LQR, constitutes an incredibly vast body of work, and is utilized in physics and general nonlinear theory [8]. In the realm of robotics, a small subset of the theory is increasingly used for state estimation–especially in conjunction with visual-inertial odometry (VIO) applications [7], [3]. This can be

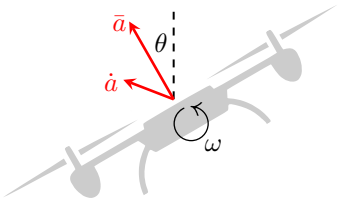

Figure 2: Illustration of the key state quantities involved in using differential flatness to derive commanded orientation and angular velocity from acceleration and jerk, respectively. Here, $\bar{a}$ is defined as the difference between gravity and the commanded acceleration: $g - a$.

attributed to the theory's ability to seamlessly integrate and calculate derivatives of affine transformations [1], [5].

Despite the increasing use of Lie theory in state estimation, there are more limited examples of its application to Control theory. While there are examples of theoretical exercises in classical and optimal control using Lie algebras, as in [10], to the author's knowledge there is only one example of Lie theory being used to formulate an error-state LQR, found in [2].

There is a vast body of literature on different methods for effective and robust attitude control on aerial platforms; this paper utilizes the approach for quaternion-based attitude control laid out in [4].

III. Trajectory Generator Derivation

As can be seen from Figure 1, the trajectory generator must take as inputs the commanded position p, velocity v, acceleration a, and jerk $\dot{a}$, all expressed in the inertial frame of reference. These specific inputs are chosen because of the concept of differential flatness that applies to quadrotors, which states that given the four "differentially flat" variables of three-dimensional position and yaw (and their derivatives), *all other states* can be algebraically derived. In this paper, yaw is commanded to be zero throughout the entire trajectory, so it is not considered.

Given the inputs to the trajectory generator, the quantities of commanded thrust F, attitude q_I^b, and angular rate $\omega_{b/I}^b = \begin{bmatrix} p & q & r \end{bmatrix}^T$ must be derived. The calculation of thrust is relatively straightforward, given the difference between the gravity vector and the commanded acceleration, $\bar{a} \triangleq g - a$:

$$F = m||\bar{a}|| \tag{1}$$

which can be intuited as relating thrust to the required force to achieve the magnitude of the desired acceleration, which at a minimum must counteract freefall.

Figure 2 gives some intuition for the calculation of attitude and angular rate. Essentially, because a quadrotor's thrust can only be applied straight out of the top of the vehicle body, any resultant acceleration vector will be aligned with the desired vehicle attitude. As a logical extension, the change in the acceleration vector (encoded by jerk) is directly related to the change in vehicle attitude (encoded by the commanded angular velocity). Given the elementary basis vectors e_1, e_2, e_3, these relationships are expressed as:

$$\theta = \cos^{-1}(e_3^T \frac{\bar{a}}{||\bar{a}||}) \tag{2}$$

$$q_I^b = Exp_q(\theta [e_3]_\times \frac{a}{||a||}) \tag{3}$$

for the relationship between $\bar{a}$ and $\bar{q}_I^b$ (note the use of the geodesic exponential map operator $\text{Exp}_q(\cdot)$–that will be expanded on in the section on Lie derivatives), and

$$h_\omega = \frac{\dot{a} - (((R_I^b)^T e_3)^T \dot{a})}{||g - \bar{a}||}(R_I^b)^T e_3 \tag{4}$$

$$p = h_\omega^T (R_I^b)^T e_2 \tag{5}$$

$$q = -h_\omega^T (R_I^b)^T e_1 \tag{6}$$

$$r = 0 \tag{7}$$

for the relationship between $\dot{a}$ and $\omega_{b/I}^b$. Equations 1-7 constitute the trajectory generation algorithm, feeding a reference state and reference thrust into the error-state LQR.

IV. Controller Derivation

The error-state LQR controller is akin to normal LQR, with a few quirks. First, the state vector is defined as $\tilde{x} = x \ominus \hat{x}$, such that $\tilde{x}$ (or the error-state) exists in the tangent space of the manifold that defines the state x. Additionally, the A and B state space matrices come from the Jacobians of the *error-state dynamics*, rather than the nominal dynamics. Finally, Jacobians are calculated from the standard definition of the derivative, substituting the plus and minus operators with $\oplus$ and $\ominus$, respectively. The following sections will walk through the requisite steps to arrive at the final controller.

A. Quadrotor Dynamics

The quadrotor state at each time step is defined to consist the vehicle's position, translational velocity, and orientation:

$$x = \begin{bmatrix} p_{b/I}^I & v_{b/I}^b & q_I^b \end{bmatrix}^T \in \mathbb{R}^6 \times \mathbb{S}^3, \tag{8}$$

and the input to the system consists the commanded thrust and angular rates:

$$u = \begin{bmatrix} F & p & q & r \end{bmatrix}^T \in \mathbb{R}^4. \tag{9}$$

The quadrotor dynamic model from input u to state evolution $\dot{x}$ used in the derivation of the error-state LQR controller is based on various simplifying assumptions:

- The dynamics between the actual and commanded angular rate $\omega_{b/I}^b$, governed principally by motor dynamics, are effectively instantaneous. This allows for $\omega_{b/I}^b$ to be considered as part of the input to the quadrotor plant.
- Aerodynamic drag is non-existent. This assumption is admissible for small aircraft at low relative air speeds.
- Thrust is linearly related to the throttle command.

Further, while the attitude portion of the state vector is represented by the quaternion $q_I^b \in \mathbb{S}^3$, the rotation matrix version of the attitude $R_I^b \in SO(3)$ will be used in the formulation of the dynamics when the attitude is being used to transform vectors, for simplicity.

The evolution of the MAV state vector x given the input u, then, can be expressed as

$$\dot{x} = \begin{bmatrix} \dot{p}_{b/I}^I \\ \dot{v}_{b/I}^b \\ \dot{q}_I^b \end{bmatrix} = \begin{bmatrix} (R_I^b)^T v_{b/I}^b \\ gR_I^b e_3 - \frac{F}{m} e_3 - [\omega_{b/I}^b]_\times v_{b/I}^b \\ q_I^b \otimes \begin{pmatrix} 0 \\ \frac{1}{2}\omega_{b/I}^b \end{pmatrix} \end{bmatrix} = f(x,u). \tag{10}$$

The above formulation for $f(x,u)$ will be very useful in deriving the error-state dynamics, whose state the LQR controller will attempt to drive to zero.

B. Lie Derivatives

Prior to continuing with deriving the error-state version of the quadrotor dynamics, it is necessary to give some brief background on fundamental topics from Lie theory. Lie theory deals with the mathematics of doing calculus on manifolds, which do not adhere to the rules of vector spaces yet *locally* look like vector spaces. The theory is relevant to the TG-ELQR controller because of the fact that the attitude of rigid bodies (represented by the quaternion q_I^b in this paper) *cannot be represented as a vector*. This fact is intuitive once one considers the non-commutativity of three-dimensional rotations, as well as the observation that attitude cannot be uniquely expressed as a weighted sum of three basis vectors. In fact, attitude exists and evolves on a manifold. Because the state vector of the dynamics includes attitude, the need arises to do calculus on manifolds in order to define the error state and to calculate Jacobians of the error-state dynamics.

Lie theory facilitates calculus on manifolds by providing the following:

- A **bi-directional geodesic mapping** between a manifold, $\mathcal{M}$, and a space tangent to it, $T_\mathcal{X}\mathcal{M}$, which *is* a vector space. The map from the tangent space to the manifold is called the **exponential map**, and the map from the manifold to the tangent space is called the **logarithmic map**.
- A method for **adding** a tangent-space vector to a manifold quantity with the $\oplus$ operator to obtain a new manifold quantity.
- A method for **subtracting** manifold quantities with the $\ominus$ operator to compute a tangent-space vector representing their difference.

These three concepts can be intuitively understood by considering the scenarios given in Figures 3 and 4. For each scenario, the goal is to define the addition operation to incrementally evolve an object on a manifold. In the non-trivial case where the manifold curves away from its tangent space as one moves away from the tangent point (as is the case for quaternions and other attitude representations), it is apparent that the addition operation requires a geodesic mapping from the tangent space to the manifold. Without getting into the details of the geodesic mapping, a new, more general addition operation is defined, $\oplus$:

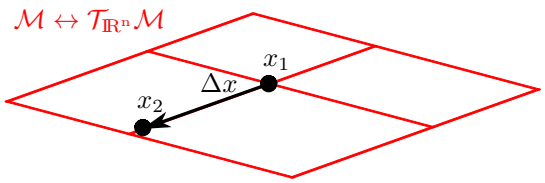

Figure 3: Illustration of the operation of adding a "tangent space" element $\Delta x \in \mathcal{T}_{\mathbb{R}^n}\mathcal{M}$ to the "manifold" quantity $x_1 \in \mathcal{M}$ to obtain the new quantity x_2 on the "manifold." The words "tangent space" and "manifold" are expressed in quotes here because the manifold in question is really just a vector space, making x_1 and x_2 vectors along with Δx. When the manifold is a vector space, its corresponding tangent space is the same everywhere, and is also a vector space (the same vector space as the one x_1 and x_2 belong to, in fact). Thus, the composition of x_1 and Δx simplifies down to the familiar operation of vector addition.

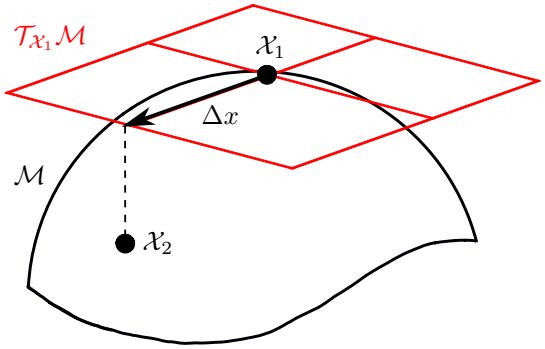

Figure 4: Illustration of the operation of adding the tangent space vector $\Delta x \in \mathcal{T}_{\mathcal{X}_1}\mathcal{M}$ to the manifold quantity $\mathcal{X}_1$ to obtain the new quantity $\mathcal{X}_2$ on the manifold. Contrasted with the scenario in Figure 3, this time $\mathcal{X}_1$ does *not* live in a vector space, and thus is *not* a vector. The incremental quantity Δx, however, lives in a tangent vector space which is tangent to $\mathcal{M}$ precisely at $\mathcal{X}_1$, and thus is still a vector. There exists an operation for composing $\mathcal{X}_1$ and Δx to obtain $\mathcal{X}_2$, but this time it is not as simple as vector addition.

$$\mathcal{X}_1 \oplus \Delta x = \mathcal{X}_2 \tag{11}$$

which encompasses the operation expressed in Figure 4 by composing a manifold quantity with a vector before mapping the result back onto the manifold. There exists a corresponding generalized subtraction operation, $\ominus$:

$$\mathcal{X}_2 \ominus \mathcal{X}_1 = \Delta x \tag{12}$$

which undoes the $\oplus$ operation by expressing the difference between the two manifold quantities $\mathcal{X}_1$ and $\mathcal{X}_2$ as the vector Δx, which lives in the tangent space to the manifold at $\mathcal{X}_1$, or $\mathcal{T}_{\mathcal{X}_1}\mathcal{M}$.

With the $\oplus$ and $\ominus$ operations defined, one can redefine the definition of the derivative to a more general form (referred to

as the Lie derivative) that works for functions of both vector and manifold quantities:

$$\frac{\partial f}{\partial \mathcal{X}} = \lim_{\Delta x \to 0} \frac{f(\mathcal{X} \oplus \Delta x) \ominus f(\mathcal{X})}{\Delta x} \tag{13}$$

Equations 11-13 are utilized subsequently to define the error-state, the error-state dynamics, and the Jacobians of the error-state.

C. Error-State Dynamics

Because the objective is to control off the error between the current and commanded states directly, it is necessary to define both the quadrotor error-state as well as the dynamics associated with the error-state. In turn, the state space model used to calculate the LQR gains will be derived by linearizing the error-state dynamic model, not the nominal dynamics defined in Equation 10.

The error-state is nothing more than the (generalized) difference between the nominal state, x, and the desired state, $\breve{x}$:

$$\tilde{x} = x \ominus \breve{x} \tag{14}$$

It is important to note that $\tilde{x}$ (and even its attitude component, $\tilde{q}_I^b$) is a proper vector, due to the aforementioned properties of the $\ominus$ operator. Using Equations 14 and 10, the error-state dynamics can be derived by careful handling of the $\oplus$ and $\ominus$ operations:

$$\begin{aligned}\dot{\tilde{x}} &= \begin{bmatrix} (R_I^b)^T \tilde{v}_{b/I}^b - (R_I^b)^T [v_{b/I}^b]_\times \tilde{r}_I^b \\ g[R_I^b e_3]_\times \tilde{r}_I^b - \frac{\tilde{F}}{m} e_3 - [\omega_{b/I}^b]_\times \tilde{v}_{b/I}^b + [v_{b/I}^b]_\times \tilde{\omega}_{b/I}^b \\ \dot{\tilde{r}}_I^b = \tilde{\omega}_{b/I}^b - [\omega_{b/I}^b]_\times \tilde{r}_I^b \end{bmatrix} \\ &= F(\tilde{x}, x, u).\end{aligned} \tag{15}$$

From Equation 15, the Jacobian matrices A and B can be calculated for the derivation of the LQR gain matrix, K.

D. LQR Formulation

As implied in Equation 13, the calculation of Jacobians for the error-state dynamics $F(\tilde{x}, x, u)$ involves careful handling of the generalized addition and subtraction operators. A brief example of taking such a derivative (which contributes to the Jacobian $\partial F / \partial \tilde{q} \in A$) is shown below:

$$\frac{\partial (R_I^b)^T \tilde{v}_{b/I}^b}{\partial \tilde{q}_I^b} = \lim_{\tilde{q}_I^b \to 0} \frac{((R_I^b)^T \oplus \tilde{q}_I^b)\tilde{v}_{b/I}^b - (R_I^b)^T \tilde{v}_{b/I}^b}{\tilde{q}_I^b} \tag{16}$$

$$= \lim_{\tilde{q}_I^b \to 0} \frac{((R_I^b)^T \text{Exp}(J_r(q_b^I)\tilde{q}_I^b) - (R_I^b)^T)\tilde{v}_{b/I}^b}{\tilde{q}_I^b} \tag{17}$$

$$= \lim_{\tilde{q}_I^b \to 0} \frac{((R_I^b)^T (I + [J_r(q_b^I)\tilde{q}_I^b]_\times) - (R_I^b)^T)\tilde{v}_{b/I}^b}{\tilde{q}_I^b} \tag{18}$$

$$= \lim_{\tilde{q}_I^b \to 0} \frac{(R_I^b)^T [J_r(q_b^I)\tilde{q}_I^b]_\times \tilde{v}_{b/I}^b}{\tilde{q}_I^b} \tag{19}$$

$$= \lim_{\tilde{q}_I^b \to 0} -\frac{(R_I^b)^T [\tilde{v}_{b/I}^b]_\times J_r(q_b^I)\tilde{q}_I^b}{\tilde{q}_I^b} \tag{20}$$

$$= -(R_I^b)^T [\tilde{v}_{b/I}^b]_\times J_r(q_b^I), \tag{21}$$

where $J_r(q_b^I) \triangleq \partial (R_b^I)^T / \partial \tilde{q}_I^b$ is the Lie derivative of a rotation matrix with respect to the error quaternion, which is approximately the identity matrix when q_b^I is close to the identity rotation. This example shows how, due to the nature of the $\oplus$ operator, the exponential map $\text{Exp}(\cdot)$ (which is the geodesic mapping between the tangent space and the manifold mentioned earlier) must be utilized. A full derivation of all Jacobian terms will not be given in this paper. Instead, the final results will be given:

$$A = \begin{bmatrix} 0 & \frac{\partial \dot{\tilde{p}}}{\partial \tilde{v}} & \frac{\partial \dot{\tilde{p}}}{\partial \tilde{q}} \\ 0 & \frac{\partial \dot{\tilde{v}}}{\partial \tilde{v}} & \frac{\partial \dot{\tilde{v}}}{\partial \tilde{q}} \\ 0 & 0 & \frac{\partial \dot{\tilde{q}}}{\partial \tilde{q}} \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ \frac{\partial \dot{\tilde{v}}}{\partial \tilde{F}} & \frac{\partial \dot{\tilde{v}}}{\partial \tilde{\omega}} \\ 0 & \frac{\partial \dot{\tilde{q}}}{\partial \tilde{\omega}} \end{bmatrix} \tag{22}$$

$$\frac{\partial \dot{\tilde{p}}}{\partial \tilde{v}} = (R_I^b)^T \tag{23}$$

$$\frac{\partial \dot{\tilde{p}}}{\partial \tilde{q}} = -(R_I^b)^T [v_{b/I}^b]_\times \tag{24}$$

$$\frac{\partial \dot{\tilde{v}}}{\partial \tilde{v}} = -[\omega_{b/I}^b]_\times \tag{25}$$

$$\frac{\partial \dot{\tilde{v}}}{\partial \tilde{q}} = [g R_I^b e_3]_\times \tag{26}$$

$$\frac{\partial \dot{\tilde{q}}}{\partial \tilde{q}} = -[\omega_{b/I}^b]_\times \tag{27}$$

$$\frac{\partial \dot{\tilde{v}}}{\partial \tilde{F}} = -\frac{1}{m} e_3 \tag{28}$$

$$\frac{\partial \dot{\tilde{v}}}{\partial \tilde{\omega}} = [v_{b/I}^b]_\times \tag{29}$$

$$\frac{\partial \dot{\tilde{q}}}{\partial \tilde{\omega}} = I \tag{30}$$

Equations 22-30 are evaluated at hover conditions ($\bar{v} = 0$, $\bar{\omega} = 0$, $\bar{R} = I$) to compute the error-state space matrices. With the A and B matrices, the LQR controller gains can finally

be derived. For this paper, the components of the error-state and input are weighted according to Bryson's Rule, with the following maximum allowable errors:

- X-Position: 2.0
- Y-Position: 2.0
- Z-Position: 1.0
- u-Velocity: 100.0
- v-Velocity: 100.0
- w-Velocity: 2.4
- Angular: 2.0
- Angular Velocity: 2.0
- Throttle: 0.5

and the K gain matrix for the LQR control $u = \bar{u} - K\tilde{x}$ is calculated from (A, B, Q, R) using the Matlab *lqr* command.

V. Attitude Representation and Control

As is apparent from Sections III and IV, this paper not only uses quaternions to represent attitude, but also uses the manifold-specific $\oplus$ and $\ominus$ operators to increment and difference them. Matlab and Simulink have no built-in support for these operations, so a custom Matlab quaternion library was written for the TG-ELQR Simulink implementation. The library can be found at https://github.com/goromal/matlab-utils.

Additionally, by inspecting the *Attitude Controller* block in Figure 1, the Simulink attitude control for the Parrot Mambo was modified to control off of both a quaternion and the commanded angular rate coming from the error-state LQR. Because of how the TG-ELQR is formulated, the input angular rate ω to the attitude controller is dependent on the trajectory error, whereas the input quaternion q is not. This disparity, while unfortunate, becomes increasingly less relevant as the quadrotor gets closer to the desired trajectory. The modified attitude controller, from [4], is formulated as

$$q_e = q_I^b \otimes q \tag{31}$$

$$\begin{bmatrix}\tau_x & \tau_y & \tau_z\end{bmatrix}^T = -\text{sgn}(\bar{q}_e)K_P\vec{q}_e - K_D(\omega_{b/I}^b - \omega) \tag{32}$$

which is essentially PD control using a quaternion. In Equation 32, $\bar{q}_e$ is the real part of the error quaternion and $\vec{q}_e$ is the imaginary part. The use of the signum function ensures that there is no ambiguity in the direction of the applied control, just in case q_e is formulated differently across time steps (a phenomenon made possible by the quaternion group's double coverage of the 3D rotation group, as discussed in [3]).

VI. Results

A time-dependent lemniscate trajectory command which also varies sinusoidally in height is used as a benchmark on which to gauge the relative performance of the TG-ELQR controller. First, a comparison is made between the benchmark tracking performance of the TG-ELQR controller and the default Simulink controller for the Parrot Mambo in simulation. The stability and performance of the TG-ELQR controller is then validated with hardware testing. An in-depth analysis of the results is given in Section VII.

A. Simulation

Figures 5 and 6 give representative tracking performances for the default Simulink controller and the TG-ELQR controller, respectively. The difficulty of the default controller in converging on the trajectory is notable–particular difficulty arises when all three positional degrees of freedom command an acceleration. The TG-ELQR controller, on the other hand, handles these acceleration commands well without appearing to diverge from the trajectory over time.

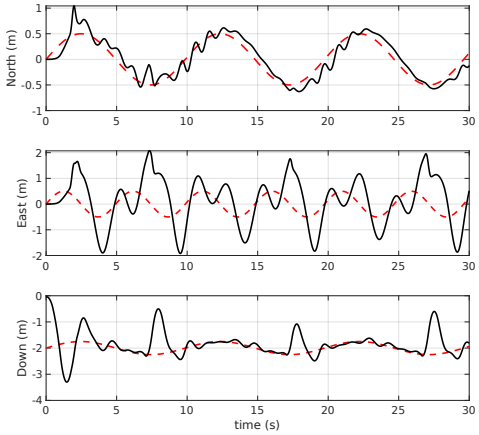

Figure 5: Default controller tracking in simulation.

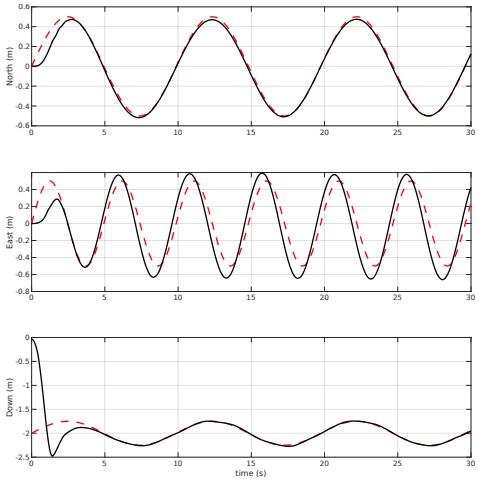

Figure 6: TG-ELQR controller tracking in simulation.

Figure 7 elucidates the gap in performance between the TG-ELQR and default controllers. Instead of simply trying to follow a position trajectory with non-zero acceleration and jerk based on position commands alone, the TG-ELQR controller calculates and follows a full-state command which is able to implicitly encode the acceleration and jerk in the commanded trajectory.

B. Hardware

To validate the stability and relative robustness of the TG-ELQR controller, the simulation benchmark scenario is replicated in a flight on the Parrot Mambo hardware platform itself. Due to the pre-calculation of the LQR gains and the

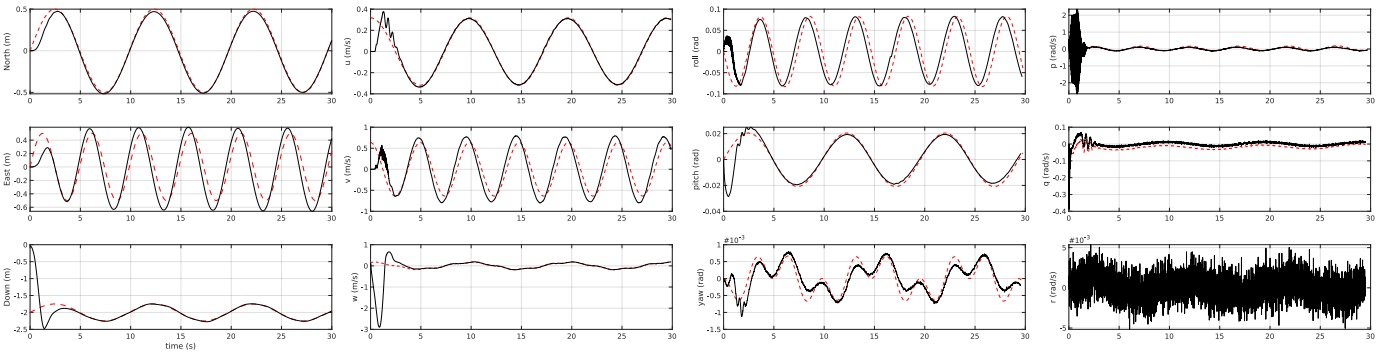

Figure 7: TG-ELQR tracking performance for all states generated by the minimum-snap trajectory generator. Pictured, in addition to the position tracking performance visible in Figure 6, are all of the commanded states that are *implicitly* necessary to track in order to achieve agile flight performance.

building of the code base from the ground-up, the TG-ELQR augmentation is able to compile and transfer directly to the Mambo's onboard computer. The resulting flight tracking performance is given in Figure 8. Though there is a noticeable degradation in performance compared to in simulation, the controller is still able to follow the commanded trajectory without struggling with commanded accelerations or drifting over time.

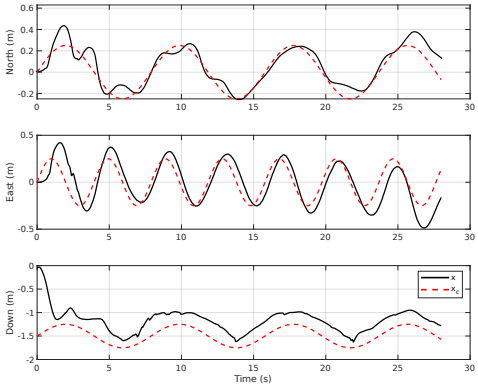

Figure 8: TG-ELQR tracking performance in hardware.

VII. Conclusions

The superior tracking performance of the TG-ELQR controller points to the fact that the default controller for the Parrot Mambo is designed to achieve good performance in tracking step commands or waypoints, and not a continuously varying position command. The TG-ELQR controller, on the other hand, is able to convert smoothly varying position (and yaw) commands to full-state commands that can be efficiently tracked with full-state feedback for more agile flight performance. Instead of constantly trying to drive the system to a level attitude, the error-state LQR is able to drive the state to commanded non-zero angular values by modulating the commanded thrust and angular rates. In a future iteration, a more advanced attitude control method (such as feedback linearization) could be used to attain still more agile performance.

Comparison between the simulation and hardware results clearly shows that hardware tracking, while still adequate, is still very much subject to disturbances and modeling errors in the absence of integral control–especially in altitude. Performance could be improved by adding an integrator term, as is often done will full-state feedback controllers. It is also possible that the hardware system suffers from insufficiently aggressive error-state LQR gains, though the process of finding a good balance of error-state LQR weights can be difficult given the inherent physical limitations of the Parrot Mambo platform. That being said, the author is impressed by the relative versatility and ease-of-use of the Mambo, and certainly plans to take on future projects (for amusement's sake) with the minidrone in the future.

References

[1] Tom Drummond and Roberto Cipolla. Visual tracking and control using lie algebras. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 652–657. IEEE, 1999.

[2] Michael Farrell, James Jackson, Jerel Nielsen, Craig Bidstrup, and Tim McLain. Error-state lqr control of a multirotor uav. pages 704–711, 06 2019.

[3] Wendelin Feiten, Pradeep Atwal, Robert Eidenberger, and Thilo Grundmann. 6d pose uncertainty in robotic perception. In *Advances in Robotics Research*, pages 89–98. Springer, 2009.

[4] Jonathan P. How, Emilio Frazzoli, and Girish Vinayak Chowdhary. *Linear Flight Control Techniques for Unmanned Aerial Vehicles*, pages 529–576. Springer Netherlands, Dordrecht, 2015.

[5] Venkatesh Madyastha, Vishal Ravindra, Srinath Mallikarjunan, and Anup Goyal. Extended kalman filter vs. error state kalman filter for aircraft attitude estimation. In *AIAA Guidance, Navigation, and Control Conference*, page 6615, 2011.

[6] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.

[7] Joan Solà. Quaternion kinematics for the error-state kalman filter. *CoRR*, abs/1711.02508, 2017.

[8] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *CoRR*, abs/1812.01537, 2018.

[9] E. Tal and S. Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4282–4288, Dec 2018.

[10] Zhifei Zhang, Alain Sarlette, and Zhihao Ling. Integral control on lie groups. *Systems & Control Letters*, 80:9–15, 2015.