OQTAL: Optimal Quaternion Tracking Using Attitude Error Linearization

PABLO GHIGLINO
JASON L. FORSHAW
VAIOS J. LAPPAS
University of Surrey, Guildford
Surrey, United Kingdom

The use of quaternions or quaternion error attitude control strategies for unmanned aerial vehicles (UAVs) is commonplace. Quaternion tracking error control is usually presented in rather theoretical works, where the control algorithm is almost exclusively chosen to be a suboptimal one. However, the application of optimal control techniques is usually associated to simplified attitude models frequently aimed at solving real-life problems. The work presented in this paper aims to formally merge the development of a complete theoretical quaternion error model with an optimal control strategy. Moreover, the application of optimal control algorithms to a fully defined quaternion error state-space model and the validation of the same in a real-time experimental setup is the focus of this research. The result is a novel controller named Optimal Quaternion Tracking of Attitude Error Linearization (OQTAL). The paper provides a comprehensive proof of stability, full simulation validation for a planetary landing gravity turn trajectory, and evidence of repeatable experimental work for a real-time quadrotor UAV on a motion capture testbed. OQTAL is compared with proven optimal forms of (PID) proportional-integral-derivative control and linear quadratic regulator control and is shown to have a 10%–20% reduction in error for the experimental setup trajectory tracking trials and an even larger tracking error reduction in the gravity turn simulation trials. Furthermore, for close tracking conditions, OQTAL behaves almost like a linear and time-invariant system, therefore requiring limited computation time for performing the trajectory tracking.

Manuscript received March 21, 2014; revised December 9, 2014; released for publication March 29, 2015.

DOI. No. 10.1109/TAES.2015.140219.

Refereeing of this contribution was handled by J. Li.

Authors' address: Surrey Space Centre, Department of Electronic Engineering, University of Surrey, BA Building, Guildford, Surrey GU2 7XH, United Kingdom, E-mail: (pablo.ghiglino@gmail.com).

I. INTRODUCTION

Attitude tracking and attitude control are topics that have been largely discussed in the available literature. On the one hand, several of these papers deal with the attitude selection of the attitude representation. Starting with [1] and [2], they provide a proportional-integral-derivative (PID) control implementation that relies on quaternion error regulation. They have been cited countless times, and this PID can be considered a de facto standard in attitude control. An extremely useful discussion is provided in [3] on different attitude and attitude error representations, their stability conditions, and the application of PID and adaptive control. Two more papers [4] and [5] deal further with the main properties of quaternion and rotation matrices: singularities, special orthogonal SO(3) space, and stability. In addition, [4] includes some advanced mathematical developments of the quaternion error, while [5] does a similar exercise for the rotation matrices. However, paramount for this research is [6], which provides a comprehensive development of the attitude error equations for both the quaternion and the rotation matrices that form the base of the OQTAL linearized model.

On the other hand, a large number of papers provide practical attitude control implementation using optimal control techniques for perhaps simpler attitude equation models. In terms of controlling vehicles, there is a vast amount of literature on linear-quadratic regulator (LQR) control applied to almost every type of conceivable vehicle, including helicopters [7], quadrotors [8], spacecraft [9], and other experimental testbeds. In many cases with these optimal control implementations, there is a limited range of operation because these simplified models have to be linear time-invariant (LTI) systems that require some particular state values to be able to work. A typical example of this is what are called hovering conditions, where the unmanned aerial vehicle (UAV) has a rotation state close to the origin (low values for rotation angles) and negligible linear velocity [7, 10].

The reason for classifying the existing literature into the preceding two categories is that each of them effectively tackles the attitude control problem in a different way. The first set deals with finding an adequate, accurate, and stable representation of the attitude equation model and usually applies to it an algorithm belonging to the suboptimal control family (PID, adaptive, etc.). The second set is a more practical approach to solving real-life UAV attitude control, where perhaps the underlying plant is more rudimentary but the control algorithm belongs to the optimal control family (LQR, H_∞, and variations of the same).

There have also been some research efforts [11, 12] that use more complex tracking models combined with some sort of optimal control.

The work presented in this paper is an effort to formally combine the two primary approaches to attitude control in such a way that the best features of each of them are included in the novel algorithm proposed here. In other words, we have created an optimal control implementation for the more complex and accurate quaternion error-based model, with the theoretical advantages of having a range of operation, and being feasible for a real-life experimental setup. The product of this research has been named Optimal Quaternion Tracking of Attitude Error Linearization (OQTAL); this name is used throughout this paper.

OQTAL has two important features that are explored in detail in this paper. First, the adapting process of this controller becomes the tuning of cost matrices, which is a priori a simpler task than adapting a more complex controller algorithm to pinpoint landing. Second, the OQTAL algorithm is based on exact linearization underlying models that, in close tracking conditions, behave effectively like LTI systems.

The new OQTAL control is compared with one specific implementation of each of the categories of papers presented earlier. One of them is the attitude PID controller for quaternion error, which is probably the most commonly used attitude controller [2]. The other is an attitude LQR controller applied to the complete Euler angle model, commonly used (in its basic form or variations of it) in [7–9] and [13–17].

The structure of the rest of this paper is as follows. Section II introduces the fundamental concepts of attitude control: Euler angle kinematics, dynamics and quaternion representation, and PID and LQR fundamentals. Section III formally presents the novel OQTAL controller, including a discussion on its stability and other formal characteristics. Sections IV and V present the simulation scenario and the results analysis, respectively. Sections VI and VII deal with the experimental setup and experimental performance analysis. Finally, Section VIII concludes the research.

II. ATTITUDE CONTROL FUNDAMENTALS

A. Attitude Kinematics and Dynamics

This section describes the attitude equations. These are the equations for rotation and rotation rate along each axis (x, y, z) and their corresponding dynamics. There are several ways of representing attitude equations; however, we only use Euler angles and quaternions. The former is more intuitive to understand, because the equations are more meaningful, while the latter was introduced as a solution to overcome the singularity limitations of the Euler angles.

1) *Euler Angle Representation*: The most basic way of representing the attitude state is by using the following two vectors: the Euler angle vector $\Theta = [\phi\ \theta\ \psi]^t$ and the angular velocity vector $\omega = [p\ q\ r]^t$. These two vectors, plus the corresponding kinematics and dynamics equations, compose the Euler representation.

The angular velocity vector can be presented in the inertial frame as $\dot{\Theta}$ (known as angular rates) or in the body frame as ω (known as angular velocity). Attitude kinematics, or the transformation of the angular velocity from body to inertial frame, is expressed as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{H}^{-1}(\mathbf{\Theta}) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{1}$$

where the matrix $\mathbf{H}^{-1}(\mathbf{\Theta})$ is the so-called angular velocity rotation matrix and is defined as follows:

$$\mathbf{H}^{-1}(\mathbf{\Theta}) \triangleq \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \tag{2}$$

The dynamics attitude equation—the relation between torque and angular acceleration—is defined traditionally as follows:

$$\mathbf{J}\dot{\omega} = \omega^{\times}\mathbf{J}\omega + \tau \tag{3}$$

where $\omega^{\times}$ is the skew symmetric of ω, $\mathbf{J}$ is the moment of inertia of the UAV, and τ is the applied input (torque). The first term on the right side of the equations is known as the dynamic coupling.

Finally, there is a more convenient way of representing them in a state-space form:

$$\begin{bmatrix} \dot{\omega} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{-1}\omega^{\times}\mathbf{J} & \mathbf{0} \\ \mathbf{H}^{-1}(\mathbf{\Theta}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \omega \\ \Theta \end{bmatrix} + \begin{bmatrix} \mathbf{J}^{-1} \\ \mathbf{0} \end{bmatrix} \tau \tag{4}$$

Many of the equations presented in this section may have singularities, for example, when $\theta = 90^\circ$ in the matrix $\mathbf{H}^{-1}(\mathbf{\Theta})$, which makes the attitude model not asymptotically stable. As explained in [3], these singularities are specific to the representation of the UAV rotation because real, solid bodies can rotate through any angle in any direction without experiencing them. This limitation is what has prompted the development of the quaternion representation.

2) *Quaternion Representation*: Equation (5) is the form or representation of the quaternion vector:

$$\mathbf{q} = \left[q_1\ q_2\ q_3\ \eta \right]^t \tag{5}$$

Equation (6) defines quaternion based on Euler angles:

$$\mathbf{q} = \begin{bmatrix} c\left(\frac{\psi}{2}\right) c\left(\frac{\phi}{2}\right) s\left(\frac{\theta}{2}\right) - s\left(\frac{\psi}{2}\right) s\left(\frac{\phi}{2}\right) c\left(\frac{\theta}{2}\right) \\ c\left(\frac{\psi}{2}\right) s\left(\frac{\phi}{2}\right) c\left(\frac{\theta}{2}\right) + s\left(\frac{\psi}{2}\right) c\left(\frac{\phi}{2}\right) s\left(\frac{\theta}{2}\right) \\ s\left(\frac{\psi}{2}\right) c\left(\frac{\phi}{2}\right) c\left(\frac{\theta}{2}\right) - c\left(\frac{\psi}{2}\right) s\left(\frac{\phi}{2}\right) s\left(\frac{\theta}{2}\right) \\ c\left(\frac{\psi}{2}\right) c\left(\frac{\phi}{2}\right) c\left(\frac{\theta}{2}\right) + s\left(\frac{\psi}{2}\right) s\left(\frac{\phi}{2}\right) s\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{6}$$

It can be easily proven that the quaternion vector has module $\|\mathbf{q}\| = (\mathbf{q}^t\mathbf{q})^{1/2} = 1$.

Similar to the Euler angle representation, there is a kinematics equation for quaternion:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{\eta} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & r & -q & p \\ -r & 0 & p & q \\ q & -p & 0 & -r \\ -p & -q & -r & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \eta \end{bmatrix} \tag{7}$$

The attitude dynamics do not change for quaternion representation, so we can use (3).

B. Quaternion Error Definition

Once the quaternion has been defined, we can proceed to the definition of the most relevant concept of this research effort: the error between commanded and actual attitude of a UAV expressed in the quaternion representation. This definition was introduced in [18], and it has been repeatedly used (e.g., in [1, 2, 4, 6]).

Given the following quaternion matrix definition

$$\mathbf{Q}^t \triangleq \begin{bmatrix} \eta & q_3 & -q_2 & -q_1 \\ -q_3 & \eta & q_1 & -q_2 \\ q_2 & -q_1 & \eta & -q_3 \\ q_1 & q_2 & q_3 & \eta \end{bmatrix} \tag{8}$$

the quaternion error $\mathbf{q}_e$ is then defined as

$$\mathbf{q}_e = \mathbf{Q_d}^t\mathbf{q} = \begin{bmatrix} \eta_d q_1 + q_{3d} q_2 - q_{2d} q_3 - q_{1d}\eta \\ -q_{3d} q_1 + \eta_d q_2 + q_{1d} q_3 - q_{2d}\eta \\ q_{2d} q_1 - q_{1d} q_2 + \eta_d q_3 - q_{3d}\eta \\ \lambda \end{bmatrix} \tag{9}$$

where $\mathbf{q}_d$ is the commanded or desired quaternion ($\mathbf{Q_d}$ is its matrix form) and λ is defined as follows:

$$\lambda = \mathbf{q}_d^t \mathbf{q}_q = q_{1d} q_1 + q_{2d} q_2 + q_{3d} q_3 + \eta_d \eta \tag{10}$$

This has two main properties. First, if both commanded and actual attitude positions are the same, then the quaternion error is as follows:

$$\mathbf{q}_e = [0 \quad 0 \quad 0 \quad \pm 1]^t \tag{11}$$

Second, if the commanded rotation is zero (i.e., all Euler angles are 0), then $\mathbf{q}_e = \mathbf{q}$.

Going forward, the quaternion error vector is presented as

$$\mathbf{q}_e = [\epsilon \ \lambda]^t \tag{12}$$

where $\epsilon \in \mathbb{R}^3$ corresponds to the first three elements of the error vector.

C. Translation Kinematics and Dynamics

The set of equations for basic spacecraft consists of the already-explained attitude equations in (1), (3), and (7) plus the so-called translational equations that describe the UAV position and linear velocity. The main translational vectors are $\mathbf{x}^I \in \mathbb{R}^{3\times 3}$ (i.e., the space vector) and $\mathbf{v}^I \in \mathbb{R}^3$ (the linear speed vector). Both are defined with respect to the inertial frame. Going forward, the superindexes I and B identify the inertial and body frame, respectively. The translation kinematics are as follows:

$$\mathbf{x}^I = [x, y, z] \tag{13}$$

$$\mathbf{v}^I = \dot{\mathbf{x}}^I \tag{14}$$

The main six degrees of freedom (6DoF) dynamics equation as defined in [19] is

$$m\dot{\mathbf{v}}^{\mathbf{B}} + m\omega^{\times}\mathbf{v}^I = \mathbf{f} + m\mathbf{R}\mathbf{g}^I \tag{15}$$

where m is the mass of the UAV, $\mathbf{g}^I$ is the gravity vector in the inertial frame, ω denotes the angular velocity vector in body coordinates, and $\mathbf{R}^t(\mathbf{\Theta})$ is the direction cosine matrix as defined for Euler representation (c and s represent sin and cos, respectively):

$$\boldsymbol{R}(\mathbf{\Theta}) = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ c\psi s\theta s\phi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & c\theta s\phi \\ c\psi s\theta c\phi + s\psi s\phi & s\psi s\theta c\phi - c\psi s\phi & c\theta c\phi \end{bmatrix} \tag{16}$$

D. Control Algorithms

Throughout this paper, the novel attitude control algorithm is compared with two other attitude controllers. First, the PID control implementation in [1] is a highly regarded and commonly used method for attitude control based on quaternions. Second, the traditional LQR controller (from now on called classic LQR) for attitude control, which is based on the state-space attitude equation model, is the most common LQR approach for UAVs in the literature [7–9, 13–17]. In this section, both baseline controllers are presented in detail.

E. Attitude PID Control

References [1] and [2] propose a PID attitude control based on the previously seen quaternion error. This proposed PID implementation includes torque and velocity (slew-rate limit) saturation and antiwindup extensions. The control law is defined in (17) and (19):

$$\begin{bmatrix} \delta_r \\ \delta_p \\ \delta_y \end{bmatrix} = \text{sat}_U \left[\boldsymbol{J} \left\{ 2k\text{sat}_L \left(\epsilon + \frac{1}{T} \int \epsilon\, dt \right) + c\boldsymbol{\omega} \right\} \right] \tag{17}$$

$$\text{sat}_L = \frac{c}{2k} \min \left\{ \sqrt{4a|\epsilon|}, |\boldsymbol{\omega}_{\max}| \right\} \tag{18}$$

$$\text{sat}_U(\boldsymbol{x}) = \min \{ U_{\max}, |\boldsymbol{x}| \} \tag{19}$$

where $\epsilon \in \mathbb{R}^{3\times 1}$ is the attitude error as per (12). In addition, c, T, and k are the PID gains of the system, a is the maximum angular acceleration value, and sat_x is the saturation function for the variable x. Within the attitude controller, the inner saturation (L) is a slew-rate limit for attitude, where $\boldsymbol{\omega}_{\max}$ is the maximum angular rate and $U_{\max}$ is the maximum total torque that can be applied.

This PID implementation, as presented in [2], is aimed at controlling the model defined in [19], which is presented in (7) and (3). However, it can be adapted to other UAVs like our quadrotor testbed, as explained in Section VII.

F. LQR Control Theory

The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case in which the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic functional is called the linear-quadratic problem. One of the main results of this theory is that the solution is provided by the LQR, a feedback controller whose equations are given in (20) and (21).

LQR is, therefore, a feedback controller whose main goal is to get an optimized feedback matrix based on two criteria: cost and accuracy. Given a linear system,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{20}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{21}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the variable under control, $\mathbf{u} \in \mathbb{R}^m$ is the input to the system, and $\mathbf{A} \in \mathbb{R}^n \times n$ and $\mathbf{B} \in \mathbb{R}^m \times n$ are matrices representing the controlled plant or system. Then, the LQR control becomes

$$u = -\mathbf{K} \cdot x \tag{22}$$

where $\mathbf{K} \in \mathbb{R}^m \times n$ is the LQR optimal gain matrix. The calculation of $\mathbf{K}$ is based on the Ricatti equations and, more importantly, its cost function, which is defined as follows:

$$f_{\text{cost}} = \int \left(\mathbf{x}^t\mathbf{Q}\mathbf{x} + \mathbf{u}^t\mathbf{R}\mathbf{u}\right)dt \tag{23}$$

where $\mathbf{Q}$ is the weight assigned to the accuracy of the control (minimize error) and $\mathbf{R}$ is the weight assigned to the applied input (minimize effort).

1) *Set-Point Tracking LQR*: The LQR's goal is to regulate; i.e., $\mathbf{x} \to 0$. However, it can be extended to track a time-varying set point. This is shown in (24), which is a modified version of (22):

$$\begin{aligned}\mathbf{u} &= -\mathbf{K}(\mathbf{x} - \mathbf{x}_{ss}) + \mathbf{u}_{ss}\\ &= -\mathbf{K}(\mathbf{x} - \mathbf{N}_x\mathbf{r}) + \mathbf{N}_u\mathbf{r}\\ &= -\mathbf{K}\mathbf{x} + (\mathbf{k}\mathbf{N}_x + \mathbf{N}_u)\,\mathbf{r}\end{aligned}$$

Defining $\bar{\mathbf{N}} \triangleq (\mathbf{k}\mathbf{N}_x + \mathbf{N}_u)$,

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \bar{\mathbf{N}}\mathbf{r} \tag{24}$$

where $\mathbf{r}$ is the set-point function and $\mathbf{u}_{ss}$ and $\mathbf{x}_{ss}$ are the steady-state input and state, respectively. $\mathbf{N}_x$ and $\mathbf{N}_u$ are defined in (25), where † is the Moore-Penrose pseudoinversion. This is used instead of the vanilla inversion because of the nonsquare nature of the matrices in the system:

$$\begin{bmatrix}\mathbf{N}_x\\ \mathbf{N}_u\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{\dagger}\begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix} \tag{25}$$

Substituting the modified input in (24) into the original state differential equation yields the new state-space equations for simulation in (26) and (27), where the set point is now the system input:

$$\dot{\mathbf{x}} = [\mathbf{A} - \mathbf{B}\mathbf{k}]\,\mathbf{x} + \left[\mathbf{B}\bar{\mathbf{N}}\right]\mathbf{r} \tag{26}$$

$$\mathbf{y} = [\mathbf{C} - \mathbf{D}\mathbf{k}]\,\mathbf{x} + \left[\mathbf{D}\bar{\mathbf{N}}\right]\mathbf{r} \tag{27}$$

2) *Integral LQR*: The integral action [20] is implemented by adding an extra state variable, i.e., the integral of the regulated variable. Given a state-space system of the form of (20), the integral of the regulated state is given as follows:

$$\mathbf{x}_I = \int (-\mathbf{y})dt = \int (-\mathbf{C}\mathbf{x})dt \tag{28}$$

Therefore,

$$\dot{\mathbf{x}}_I = -\mathbf{C}\mathbf{x} \tag{29}$$

and the corresponding integral action system becomes as follows:

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{x}}_{\mathbf{I}}\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{0}\\ -C & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{x}_{\mathbf{I}}\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\ \mathbf{0}\end{bmatrix}\mathbf{u} \tag{30}$$

3) *Linearization and Gain Scheduling*: LQR can only be applied as it is to linear-time-invariant (LTI) systems. However, in the real world, there are few cases in which phenomena can be modeled like LTI systems; in most cases—including aerial vehicles—nonlinearities are always present. Therefore, to apply LQR in these situations, some adapting process is required, such as treating the controlled system like an LTI system for certain conditions of operation.

First, Jacobian linearization—i.e., the first term of Taylor's series—is used to perform linearization. Gain scheduling is another technique that adapts nonlinear systems to make them controllable by LQR. It consists of transforming a single linear time-variant system into a set of LTI systems. In this paper, a simple algorithm is used for gain scheduling. It works as follows.

Given a stepwise set of ranges with a global step δ, each single variable ξ in the state vector $\mathbf{x}$ falls within a particular range of operation:

$$n\delta \le \xi < (n+1)\,\delta \equiv \xi \in [n\delta, (n+1)\,\delta]$$

For clarity, let us define each range as follows:

$$[n\delta, (n+1)\,\delta] \triangleq n\delta$$

This takes us to the current operation vector:

$$\xi_{\text{operation}} = \begin{bmatrix}\cdots\\ n_i\delta\\ \cdots\end{bmatrix}$$

While all variables within the vector are within the operation vector, the system behaves like an LTI. However, if any of these variables changes in a way that its value is no longer within its current range of operation, the operation vector is changed to the following:

$$\xi'_{\text{operation}} = \begin{bmatrix}\cdots\\ n'_i\delta\\ \cdots\end{bmatrix}$$

Therefore, the LTI matrix $\mathbf{A}(\mathbf{x}'_{\text{operation}})$ changes accordingly and the optimal gain $\mathbf{K}$ has to be solved again.

This gain-scheduling algorithm is rather rudimentary compared to more sophisticated fuzzy-logic implementations [17]. Furthermore, model switching might introduce input jumps and therefore potentially undesirable behavior of the system. However, an empirical selection of a small threshold δ makes the LQR operate stably, and it is able to produce a smooth input signal for both simulated and experimental tests.

G. LQR Attitude Control

This final section describes the implementation of the LQR technique introduced earlier to the attitude control problem. By applying this equation format to the state-space system presented in (4) and using the Euler angles as the control variable, we obtain the following:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\omega} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{-1}\omega^{\times}\mathbf{J} & \mathbf{0} \\ \mathbf{H}^{-1}(\boldsymbol{\Theta}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \omega \\ \Theta \end{bmatrix} + \begin{bmatrix} \mathbf{J}^{-1} \\ \mathbf{0} \end{bmatrix} \tau \quad (31)$$

$$\mathbf{y} = \begin{bmatrix} \Theta \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \omega \\ \Theta \end{bmatrix} \quad (32)$$

The application of LQR to the preceding set of equations is what we call classic QR. In this system, the matrix **A** is not linear or time invariant; therefore, this LQR requires gain scheduling. In this paper, we used the previously explained gain-scheduling implementation.

In the experiments with the quadrotor, set-point tracking is used. When applying this technique to classic LQR, the Moore-Penrose pseudoinversion might become singular for certain angle values. However, for the particular conditions of the UAV we are using, this problem has a very low probability of happening. This is clarified in Section VII.

Finally, applying integral action to classic LQR, let us define $\boldsymbol{\Theta}_I$ as follows:

$$\boldsymbol{\Theta}_I = \int (-\mathbf{y})dt = \int (-\boldsymbol{\Theta})dt \quad (33)$$

Then, the integral set of equations can be seen in the following:

$$\begin{bmatrix} \dot{\omega} \\ \dot{\Theta} \\ \Theta \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{-1}\omega^{\times}\mathbf{J} & \mathbf{0} & \\ \mathbf{H}^{-1}(\boldsymbol{\Theta}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \omega \\ \Theta \\ \Theta_I \end{bmatrix} + \begin{bmatrix} \mathbf{J}^{-1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tau \quad (34)$$

$$\begin{bmatrix} \Theta \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \omega \\ \Theta \\ \Theta_I \end{bmatrix} \quad (35)$$

III. OQTAL: NOVEL ATTITUDE CONTROLLER

OQTAL is founded on two pillars—an accurate attitude error model and optimal control. This section is structured accordingly. First, we explain the rationale behind through the mathematical process of producing an adequate linear model out the dynamic model provided in [6]. This is followed by a formal stability study of both the nonlinear model and the linearized version of the same. After that, the focus moves to the decision of which optimal control to use and its features. Finally, a summary of the three controllers is presented.

A. Quaternion Error Dynamics

The same definition of quaternion error is used in [1] and [6]. It provides an extension to the angular velocity error based on the quaternion error and the corresponding algebraic calculation of the error dynamics or derivatives.

First, ω_e is defined as the angular velocity error in body coordinates:

$$\omega_e \triangleq \omega - \mathbf{C_{BB}}\omega_d \quad (36)$$

Here, $\mathbf{C_{BB}}$ is defined as follows:

$$\mathbf{C_{BB}} \triangleq \mathbf{Q_e}^{-1}\mathbf{Q_e}^t$$

and

$$\mathbf{Q_e} \triangleq \mathbf{Q_d}\mathbf{Q}^t$$

Then, by applying some simple mathematical manipulations, the following are obtained (where $\mathbf{I}_3$ is the 3×3 identity matrix):

$$\dot{\omega}_e = 2\left(\lambda\mathbf{I}_3 - \epsilon^{\times} + \frac{\epsilon\epsilon^t}{\lambda}\right)\left[\ddot{\epsilon} + \frac{{\omega_e}^t\omega_e}{4}\epsilon\right] \quad (37)$$

$$\dot{\epsilon} = \left(\lambda\mathbf{I}_3 + \epsilon^{\times}\right)\omega_e/2 \quad (38)$$

$$\dot{\lambda} = -\epsilon^t\omega_e/2 \quad (39)$$

$$\tau = J\dot{\omega}_e + \omega^{\times}J\omega \quad (40)$$

This highly nonlinear set of equations is the base of the underlying plant for OQTAL. It has several relevant properties. First, (40) contains the term $\omega^{\times}$ J ω. This, as discussed in [3], is a model-dependent control law. This is a key concept that is discussed further in Section VII. The second relevant property of this model is that (37) to (40) are not only comprehensive (including equations for the state variables and for their derivatives) but also exclusively dependent on the attitude error state or derivatives. This makes this system of equations adequate for linearization and therefore a viable candidate for optimal control, as discussed in the next subsection.

B. OQTAL Underlying Model

As explained in the previous subsection, the system of (37) to (40) is a highly nonlinear one, meaning that it requires the application of a linearization technique convert it into a state-space system.

We considered Jacobian linearization [21] as an adequate way of obtaining a linearized state-space system of equations out of (37) to (40). The resulting linear system has, at a high level, the following structure:

$$\begin{bmatrix} \dot{\boldsymbol{\omega}}_e \\ \dot{\boldsymbol{q}}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{\dot{\omega}_e\omega_e} & \boldsymbol{A}_{\dot{\omega}_e\boldsymbol{q}_e} \\ \boldsymbol{A}_{\dot{\boldsymbol{q}}_e\omega_e} & \boldsymbol{A}_{\dot{\boldsymbol{q}}_e\boldsymbol{q}_e} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_e \\ \boldsymbol{q}_e \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{\dot{\omega}_e\boldsymbol{\tau}} \\ \boldsymbol{B}_{\dot{q}_e\boldsymbol{\tau}} \end{bmatrix} \boldsymbol{\tau} \quad (41)$$

$$\left[\, \boldsymbol{q}_e \,\right] = \left[\, \mathbf{0}\ \boldsymbol{I}_3 \,\right] \begin{bmatrix} \boldsymbol{\omega}_e \\ \boldsymbol{q}_e \end{bmatrix} \tag{42}$$

where $\mathbf{A}_{xy} \triangleq \frac{\partial x}{\partial y}$ is the Jacobian linearization matrix for the variable x with respect to y. The matrices obtained after the corresponding algebraic development are as follows (see Appendix A for their development in full):

$$\mathbf{A}_{\dot{\epsilon}\epsilon} = -\frac{1}{2}\omega_{\mathbf{e}}{}^{\times} \tag{43}$$

$$\mathbf{A}_{\dot{\epsilon}\omega_e} = \frac{1}{2}\left(\epsilon^{\times} + \lambda\mathbf{I}_3\right) \tag{44}$$

$$\begin{aligned}\mathbf{A}_{\dot{\omega}_e\epsilon} = 2\Bigg\{ & \ddot{\epsilon}^{\times} + \frac{1}{\lambda}\left(\epsilon\ddot{\epsilon}^t + \epsilon^t\ddot{\epsilon}\mathbf{I_3}\right) \\ & + \frac{\omega_e{}^t\omega_e}{4}\Bigg[\lambda\mathbf{I}_3 - 2\epsilon^{\times} + \frac{\epsilon\epsilon^t}{\lambda} \\ & + \frac{1}{\lambda}\left(\epsilon\epsilon^t + \epsilon^t\epsilon\mathbf{I_3}\right)\Bigg]\Bigg\}\end{aligned} \tag{45}$$

$$\mathbf{A}_{\dot{\omega}_e\omega_e} = \left(\lambda\mathbf{I}_3 - \epsilon^{\times} + \frac{\epsilon\epsilon^t}{\lambda}\right)\frac{\epsilon}{2}\omega_e{}^t \tag{46}$$

$$\mathbf{B}_{\tau\dot{\omega}_e} = J^{-1} \tag{47}$$

$$\mathbf{B}_{\tau\dot{\epsilon}} = 0 \tag{48}$$

This yields a new form of the state-space system that is linear (but not time invariant) and is based on accurate quaternion attitude error. It describes the calculation of the torque as a function of the angular velocity error and how to obtain the error vector (i.e., quaternion and angular velocity) derivative with respect to the error vector. The main difference between this linear system and those used for optimal control is that there is no dependency with respect to the UAV attitude state, i.e., the rotation angle or angular rate.

The next step is to verify the reliability of this set of equations by means of proving its stability.

C. Quaternion Error Uniqueness and Stability

The quaternion error vector has the same properties as the attitude quaternion vector in that its module is $\|\mathbf{q}_e\| = 1$ and it has the same nonuniqueness limitation that quaternions have: zero attitude error might produce $\lambda = \pm 1$. Leaving this limitation aside for a moment, there is another property of paramount importance for proving the stability of the system, which can be found in (38) and (39). These equations are algebraically identical to the ones used in [4] for proving the stability of the quaternion attitude equations. However, here they are applied to quaternion and angular velocity error rather than actual attitude and angular velocity. Therefore, we are going to follow the same approach. First, the use of direct Lyapunov methods shows that our model is not stable, because for every stable equilibrium point, there is an opposite unstable one. Given the following Lyapunov function and feedback, we have the following:

$$\hat{V}_1(\mathbf{q}_e) = 2(1-\lambda)$$
$$\boldsymbol{\omega}_e = -\epsilon$$

Replacing the feedback function (39) results in the following:

$$\dot{\lambda} = -\frac{1}{2}\epsilon^t(-\epsilon) = \frac{\epsilon^t\epsilon}{2} \tag{49}$$

The derivative of the Lyapunov function is

$$\dot{\hat{V}}_1(\mathbf{q}_e) = -2\dot{\lambda} = -\epsilon^t\epsilon \tag{50}$$

and this is always negative except for $\lambda = \pm 1$, which are precisely the two equilibrium points explained earlier. Therefore, this does not prove the stability of the system of equations. However, by choosing the Lipschitz-Lyapunov function and feedback, the stability of quaternion attitude error dynamics can be proved as follows:

$$\hat{V}_2(\mathbf{q}_e) = 2(1-|\lambda|)$$
$$\omega_e = -\mathrm{sgn}(\lambda)\,\epsilon$$

Replacing the feedback function (39) gives the following:

$$\dot{\lambda} = -\frac{1}{2}\mathrm{sgn}(\lambda)\,\epsilon^t(-\epsilon) = \frac{\mathrm{sgn}(\lambda)\,\epsilon^t\epsilon}{2} \tag{51}$$

The derivative of the Lyapunov function is as follows:

$$\dot{\hat{V}}_2(\mathbf{q}_e) = -2\mathrm{sgn}(\lambda)\,\dot{\lambda} = -\epsilon^t\epsilon \tag{52}$$

The authors of [4] go one step further for proof of stability under measurement noise. Their demonstration also applies identically to our system of quaternion error equations.

The linearized version can be also proved stable following a similar approach, using the same Lipschitz-Lyapunov function and the same feedback. Given the following differential equation for the quaternion error, we have the following:

$$\dot{\epsilon} = \left[\mathbf{A}_{\epsilon\epsilon} \quad \mathbf{A}_{\epsilon\omega_e}\right]\begin{bmatrix}\epsilon \\ \omega_e\end{bmatrix} \tag{53}$$

$$= \left[-\tfrac{1}{2}\epsilon^{\times} \ \ \tfrac{1}{2}\left(\epsilon^{\times} + \lambda\mathbf{I}_3\right)\right]\begin{bmatrix}\epsilon \\ \omega_e\end{bmatrix} = \frac{1}{2}\left(\epsilon^{\times}\omega_e + \lambda\omega_e\right) \tag{54}$$

$$\dot{\lambda} = -\frac{1}{2}\left(\epsilon^t\epsilon + \omega_e^t\omega_e\right) \tag{55}$$

Applying the previous Lipschitz-Lyapunov function and feedback, the same result as per (52) is obtained.

In summary, (37) to (40) have been formally proved to be stable by the use of a Lipschitz-Lyapunov function. Furthermore, the linearized version of the same equations, i.e., OQTAL's underlying plant, is proved to be stable. The next step is to choose the optimal control algorithm for the underlying devised plant.

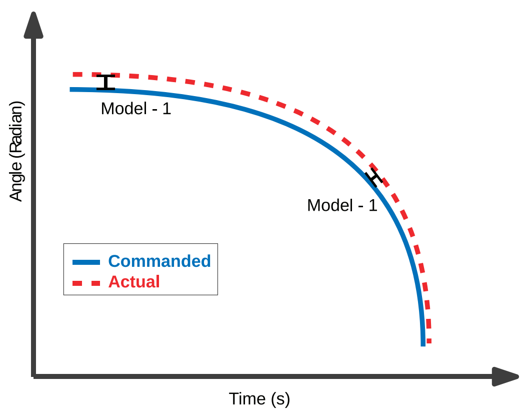


Fig. 1. Close-tracking OQTAL needs only one linear model instance.

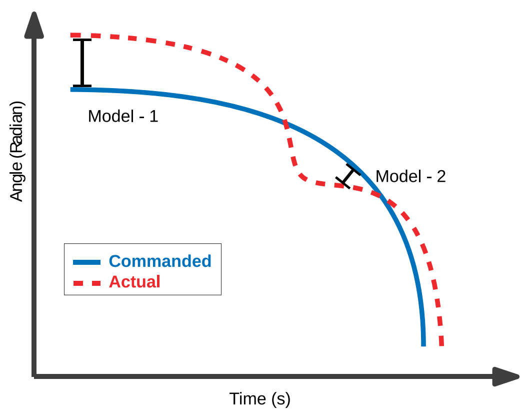


Fig. 2. Variable-tracking OQTAL needs several model instances.

IV. OQTAL CONTROL ALGORITHM

The second part of this research is the decision of an optimal control algorithm to apply to this system. The authors chose the LQR algorithm because of its simplicity. The proposed control law for OQTAL is

$$\tau = \boldsymbol{K}\left[\,\epsilon \;\, \boldsymbol{\omega}_e\,\right]^t + \omega^{\times}\boldsymbol{J}\boldsymbol{\omega} \tag{56}$$

where ω is the actual angular velocity as defined in Section II. This term makes OQTAL a model-dependent system. Therefore, OQTAL comprises the previously described linear model and the application of LQR to control it. There are, however, several aspects to be considered to determine the validity of OQTAL as a practical implementation for attitude tracking control problems. These include range of operation, gain scheduling, and adaptability of the model to real-life scenarios. All of them are covered in detail in this section.

A. Range of Operation

OQTAL's linear matrix depends only on the state values of the attitude error, and given that this matrix is a linearized version of (37) to (40), it means that the operation range is effectively the attitude error range.

To clarify this concept, an example of a trajectory where the rotation tracking is accurate is chosen; e.g., the controller closely follows the commanded attitude, as shown in Fig. 1. In such a case, the error vector is constantly small. Therefore, just one concrete linear matrix (with low error initial values) for the OQTAL state-space model probably suffices to perform the tracking.

At the other end of the scale, variable tracking with both large and small attitude errors, such as in Fig. 2, might require several concrete models and therefore some kind of model-switching technique like the gain-scheduling approach discussed previously.

This is a particular and fundamental property of OQTAL: For accurate tracking, complexity is reduced because model switching seldom happens. Therefore, the number of LQR Ricatti equation calculations is reduced.

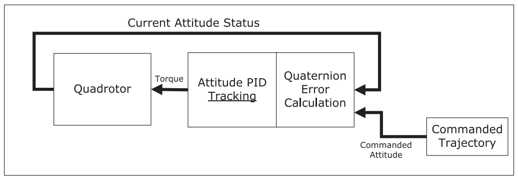


Fig. 3. Architectural diagram of PID controller.

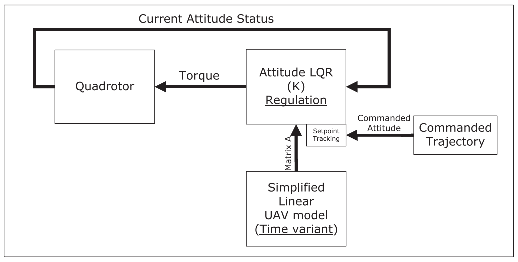


Fig. 4. Architectural diagram of classic LQR.

B. OQTAL With Integral Action

In some real-life scenarios, integral action might be required to compensate for unexpected forces, such as nonmodeled nonlinearities of the system and undesired drift in the arrangement of rotors. Given that OQTAL uses standard LQR techniques, the application of integral action is trivial:

$$\begin{bmatrix} \dot{\omega}_e \\ \dot{\mathbf{q}}_e \\ \dot{\mathbf{q}}_{e\,I} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\dot{\omega}_e\omega_e} & \mathbf{A}_{\dot{\omega}_e\mathbf{q}_e} & \mathbf{0} \\ \mathbf{A}_{\dot{\mathbf{q}}_e\omega_e} & \mathbf{A}_{\dot{\mathbf{q}}_e\mathbf{q}_e} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \omega_e \\ \mathbf{q}_e \\ \mathbf{q}_{e\,I} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{\tau\dot{\omega}_e} \\ \mathbf{B}_{\tau\dot{\mathbf{q}}_e} \\ \mathbf{0} \end{bmatrix} \boldsymbol{\tau} \tag{57}$$

C. OQTAL Control Architecture

Figs. 3–5 show the control architecture of OQTAL, along with the two controllers that we used for comparison. This gives a visual idea of what we meant by OQTAL combining the best features of the two approaches seen in Section I.

First, Fig. 3 shows how the PID controller in [1] works. It receives both the commanded and the actual

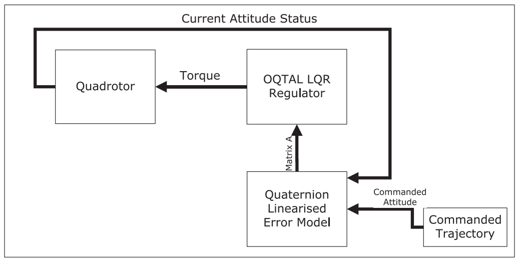


Fig. 5. Architectural diagram of OQTAL LQR.

attitude states. Then, it proceeds to calculate the quaternion error from these two inputs and apply standard PID techniques to the error vector.

Second, the traditional LQR proposal (Fig. 4) appears much like a standard LQR regulation, which means that it tries to bring the rotation state to rest (zero). The integral action and the set-point tracking make this controller an attitude tracking implementation.

Finally, OQTAL in Fig. 5, seems like a combination of the two earlier forms; i.e., it requires the quaternion error calculation and does the regulation over that vector. In other words, the LQR regulation is in effect the tracking in this case; therefore, there is no need for set-point tracking.

In summary, OQTAL consists of a combination of two things: the application of a standard LQR control to a set of linearized attitude error dynamics. This is what we meant by adapting the best of both streams: optimal control for an accurate representation of the attitude error dynamics. Furthermore, OQTAL is now fully defined and proved to be stable, and it has a range of operation that depends on attitude error only. The next step is thus testing OQTAL in different scenarios.

V. SIMULATION SETUP

The first phase of testing of OQTAL has been done in a simulation environment. The details of this simulation framework are explored in this section; these are the UAV model, trajectory model, control laws, and control optimization.

A. Simulation UAV Models

Within this research, two UAV models have been simulated and OQTAL has been applied to them. They are the spacecraft vanilla equations, as per Section II.C, and a quadrotor UAV model developed in Matlab/Simulink at Surrey Space Centre (SSC) [22]. The reason for choosing the former is its simplicity, which is ideal for this first testing phase. The latter was selected as a preparation step for the proper real-time quadrotor device.

B. Gravity Turn Trajectory

The second design decision is the type of trajectory to be tested. During this research, three trajectories have been used: 1) single step, which refers to the single step from point A to point B for a single Euler angle variable (ϕ, θ, or ψ); 2) gravity turn, alluding to the classic spacecraft descent maneuver where the UAV passes from the horizontal high-speed flight mode to a hovering, vertically oriented state; and 3) square trajectory, as described in Section VII.

TABLE I
Gravity Turn, UAV, and Simulation Specifications

Parameter name	Parameter value
Initial velocity	70 m/s
Initial angle of attack	$\pi/2$ rad
Initial altitude	2000 m
Mars gravity	$3.7N$
UAV N factor	1.5
UAV mass	0.432
UAV moment of inertia	$0.5\ \mathbf{I}_3$
Simulation time	23 s
Simulation sample time	1/60 s

Several possible combinations of the preceding have been tested during the research. However, for brevity, only the gravity turn using the vanilla spacecraft equations is presented in Section VI. The reason for choosing this trajectory is that the gravity turn is a trajectory purely driven by attitude control. Therefore, the better the control is, the closer to the perfect gravity turn curve the UAV will be.

First, let us describe the classic or vacuum gravity turn as presented in [23], so named because it means that quadratic air drag or other external forces have not been taken into consideration. The goal of any gravity turn trajectory is for the UAV pitch angle to match the angle of attack. The curve produced by this trajectory is described by

$$\dot{\boldsymbol{v}} = \boldsymbol{g}(n - \cos(\theta)) \tag{58}$$

$$\dot{\theta}\boldsymbol{v} = \sin(\theta) \tag{59}$$

where $\boldsymbol{v}$ is the linear velocity module, θ is the angle of attach, and $n = \boldsymbol{F}/(mg)$ is the thruster ratio. In the case of n being constant, these equations cannot be analytically resolved.

The actual parameters of the trajectory are as per the technical note [24], which specifies a thrust power-based gravity turn. Table I shows the gravity turn parameters and UAV specifications.

The spacecraft enters at the speed specified later and in the x-axis direction. Therefore, the trajectory occurs in the xz plane. The initial pitch angle and the angle of attach are both $\pi/2$.

The gravity turn control algorithm implemented here consists of two parallel processes. On the one hand, maximum thrust is applied constantly. This thrust here is Nmg, where N is as per Table I. On the other hand, the attitude control aims at keeping the UAV angle of attack opposite to the linear velocity.

This kind of control does not aim to achieve pinpoint landing accuracy; its only objective is to bring the UAV to a hovering state. However, two papers have been found that attempt to increase landing accuracy in gravity turn trajectories [25, 26]. No further work has been done in this

TABLE II
LQR Tuning Parameters

Criterion	Definition	PID	LQR
Quaternion error	$\int \Vert \boldsymbol{Q}_d \boldsymbol{q} \Vert$	Yes	Yes
Position error	$\int \Vert \boldsymbol{X}_d^I - \boldsymbol{Z}^I \Vert$	Yes	Yes
Effort	$\int \Vert \tau \Vert$	Yes	Yes
Number of calculations	Gain-scheduling switches	No	Yes
Cost function	$\int (q\epsilon^t\epsilon + r\tau^t\tau)$	No	Yes

research on this particular area, and it remains a subject for future work.

VI. GRAVITY TURN SIMULATION RESULTS

A. Testbed Comparison Criteria

We used three criteria for validating the performance of OQTAL and the other two controllers. First, the attitude error is understood as the tracking error between the commanded and the actual attitude states and is expressed in quaternion representation. Second, the position error is the difference between the ideal trajectory and the actual position with respect to the inertial frame for any given time *t*. Third, the effort of the controller is the torque input commanded by the control algorithm on the UAV.

The other indices that are used for evaluation of the performance are LQR specific. First, the number of calculations is the number of gain-scheduling model switches, i.e., how many times the Ricatti equation has to be solved (for PID, this number is zero, which makes it the best controller in terms of computation time). Finally, the cost function is defined in (23), with the tuning matrices defined in the form $\mathbf{Q} = q\mathbf{I}$ and $\mathbf{R} = r\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Table II summarizes the meanings of these criteria and the controllers for which they are valid.

B. Controller Optimization

To make the comparison among controllers as fair as possible, each of the tested controllers has been tuned for the simulated gravity turn. The next two subsections focus on this tuning process. After that, we take the best-tuned version of each control and compare them. The results are presented in the last part of this section.

1) *Optimization of PID*: In the case of the PID, we used an offline genetic algorithm named Artificial Bee Colony (ABC) as per the implementation in [27]. The reason for choosing this algorithm is that ABC is considered able to find PID gains that are close to the optimal solution [28]. The optimization was performed for a fitness function consisting of the weighted sum of the accumulated attitude quaternion error and the accumulated torque:

$$f_{\text{fitness}} = \int \left[q\left(\epsilon^t \epsilon\right) + r\left(\tau^t \tau\right) \right] \tag{60}$$

TABLE III
PID Tuning Summary

Criterion [q, r]	[1, 1]	[10, 1]	[100, 1]	[1, 0]
Position error	80.3	29.78	29.8	49.1
Quaternion error	0.1280	0.0349	0.0328	0.0754
Effort	0.2609	0.8252	0.9476	0.3955
Fitness value	0.3889	1.1744	4.2286	0.4709

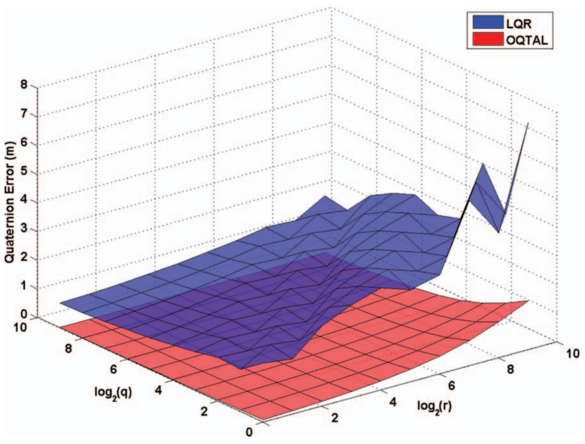


Fig. 6. Quaternion error optimization comparison surface. It shows same plot surfaces for quaternion error for classic LQR and OQTAL for all [q, r] pairs.

For the PID ABC optimization, the gains *c*, *k*, and *T* have been tuned for the following weight values: [q, r] ∈ [1, 1], [10, 1], [100, 1], [1, 0].

Table III shows the result of running the ABC tuning process for each of the previously specified weight values.

Each value corresponds to the integral of the instantaneous values for a complete simulated trajectory; e.g., the position error corresponds to the integral of the position error for the whole trajectory execution.

2) *Optimization of Classic and OQTAL LQR*: For the LQR case, a brute force approach has been used. The optimization was performed by varying factors q and r as defined. The brute force algorithm was performed by using a geometric progression with a factor of 2 for q and r and testing all possible combinations of them. The range of $q \in 2^i$, $0 <= i < 20$, and $r \in 2^i$, $0 <= i < 20$, comes about for two reasons. First, this range contains the optimal q, r pairs for both classic and OQTAL LQR instances. Second, this range is where classic LQR works stably, whereas larger values of q make classic LQR start to show unexpected behavior. Although OQTAL works for larger values of q, but it also shows unstable behavior past $q > 2^{20}$. The chosen gain-scheduling threshold of $\delta = 0.2$ was empirically found to be optimal for classic LQR for this particular trajectory and UAV.

Figs. 6–9 show the results of this tuning exercise, one for each of the previously mentioned criteria. These graphics are surface plots that depict both OQTAL and classic LQR surfaces for the logarithmic scaled [q, r] range. Comparison between the two LQR controllers

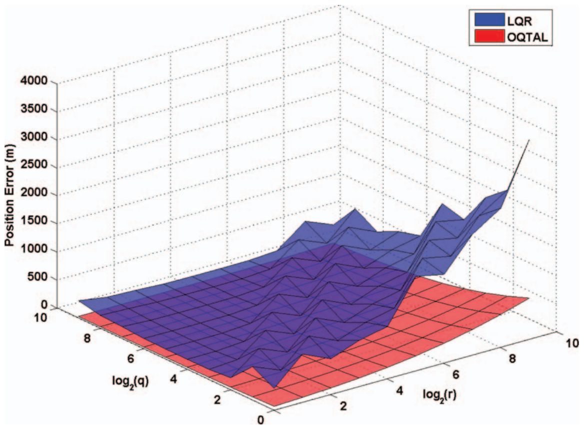


Fig. 7. Position error optimization comparison surface. Plotted on same graph are surfaces for position error for classic LQR and OQTAL for all $[q, r]$ pairs.

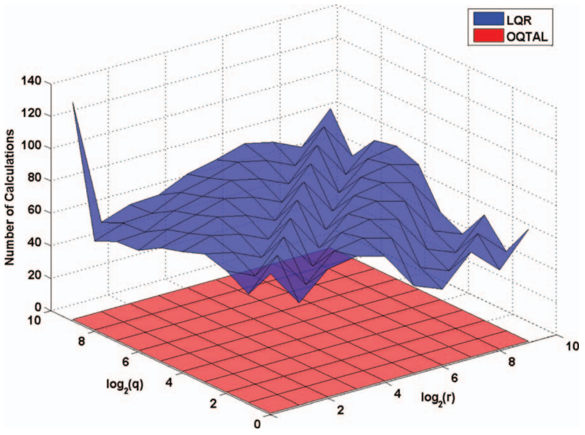


Fig. 8. Number of calculations optimization comparison surface. Plotted on same graph are surfaces for number of calculations for classic LQR and OQTAL for all $[q, r]$ pairs.

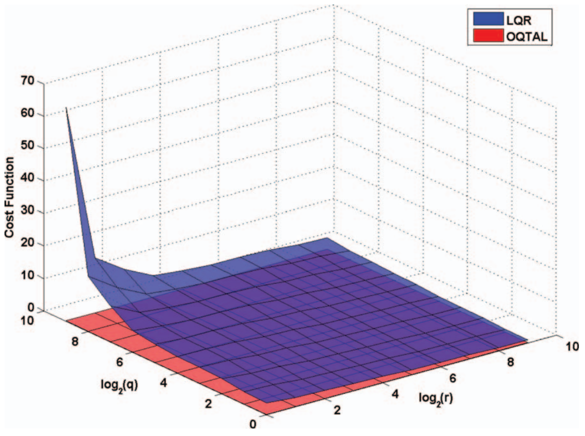


Fig. 9. Effort optimization comparison surface. Plotted on same graph are surfaces for torque input for classic LQR and OQTAL for all $[q, r]$ pairs.

shows that OQTAL clearly performs better than classic LQR in each of these criteria, with significantly lower errors and costs across the entire testing spectrum.

TABLE IV
Best-Tuned Controllers Comparison

	PID	Classic LQR	OQTAL
Gains	$c = 50$	$q = 128$	$q = 256$
	$k = 5.95$	$r = 1$	$r = 1$
	$T = 1.15$		
Position error	29.78	298.1	26.13
Quaternion error	0.0349	0.8256	0.0052
Number of calculations	N/A	23	1
Torque	1.1744	7.13	0.2045

N/A = not applicable.

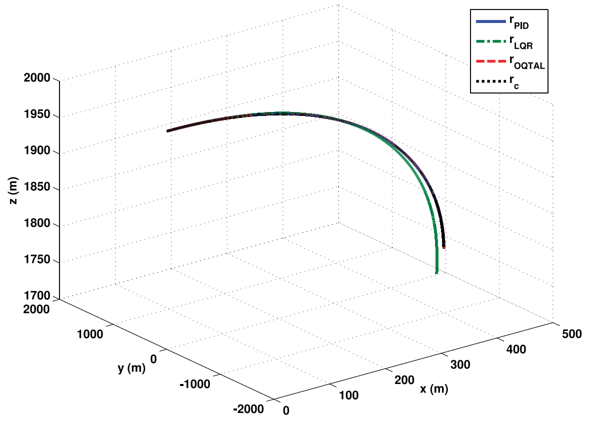


Fig. 10. Simulated gravity turn in 3D, showing 3D position for PID, classic LQR, and OQTAL.

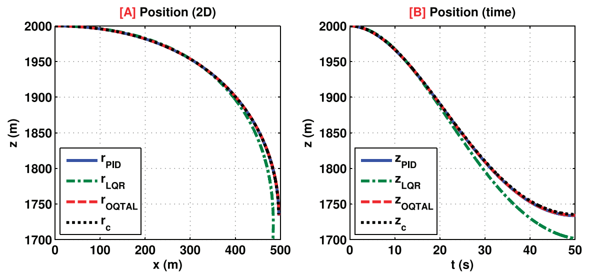


Fig. 11. Simulated gravity turn position, showing (A) 2D position (top-down view), (B) 2D position (side view), (C) x-position time history, and (D) z-position time history.

C. Performance Analysis

We now take the best-tuned version of each of the three controllers and summarize the results obtained in Table IV.

These statistics are also shown in Figs. 10–14, where the three controllers are compared against the ideal classic or vacuum gravity turn as per the equations described [23]. That means that each of the graphics shows four trajectories: one for each of the three control instances plus the ideal vacuum trajectory.

It can be seen that they all run accurate tracking, including Fig. 12, which represents ϕ (pitch angle) with respect to altitude—probably the most important function when benchmarking gravity turns [29].

Clearly, OQTAL leads in every performance criterion in Table IV. OQTAL appears closer to the $z = 0$ plane compared to classic LQR. An interesting result is the one

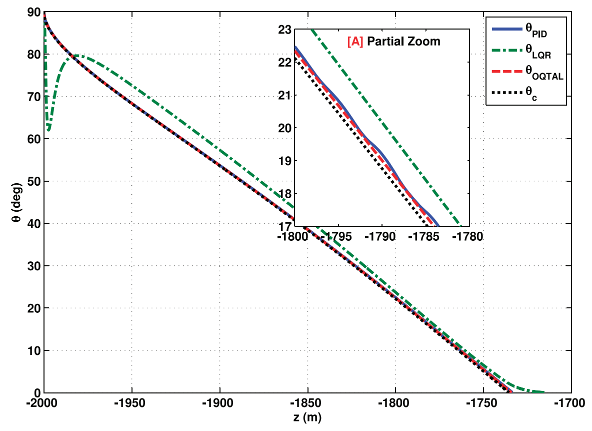


Fig. 12. Simulated gravity turn pitch attitude, showing pitch angle for PID, classic LQR, and OQTAL.

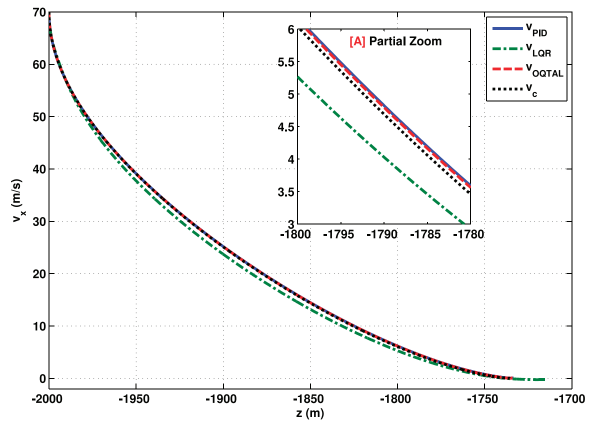


Fig. 13. Simulated gravity turn X velocity, showing velocity in x-axis for PID, classic LQR, and OQTAL.

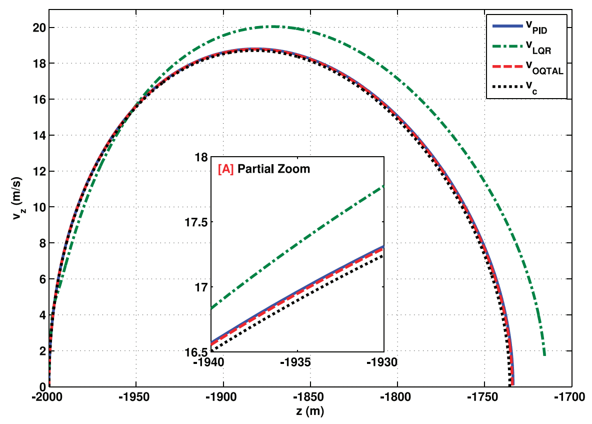


Fig. 14. Simulated gravity turn Z velocity, showing velocity in z-axis for PID, classic LQR, and OQTAL.

for number of calculations for this gravity turn trajectory, which expresses one of the main features of OQTAL. As explained in the previous section, OQTAL model switching depends on the tracking error, not the actual vehicle state, while the provided classic LQR depends on the rotation state value. Hence, there is the need for model switching: there is a variation of 90° from the beginning to the end of the trajectory.

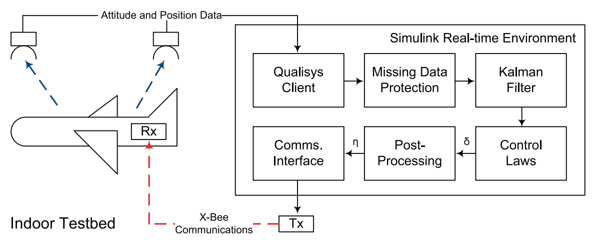


Fig. 15. Indoor testbed, showing different elements present in developed indoor testbed. UAV data enter computer via motion capture system, where they are processed and filtered. Control laws calculate necessary commands (δ) that need to be sent to UAV; these are postprocessed (η) and transmitted out to UAV.

Having seen that OQTAL performs the best among the tested controllers in a simulated environment, we move on to a real-life experimental setup.

VII. EXPERIMENTAL SETUP

The testbed we used for the experimental testing has been employed in previous research efforts [22, 27, 30] and is shown in Fig. 15. The testbed includes a Qualisys motion capture system, which comprises six cameras positioned around the SSC lab room that track markers on the quadrotor and transmit the position and rotation vectors (X^I, Θ) via Ethernet. The Simulink Real-time Environment on the computer orchestrates the testbed flow and contains all modules described earlier: UAV, control, and commanded trajectory. The quadrotor is an Ascending Technologies Aztec Hummingbird X3D-BL, and the communications have been set up via an X-Bee link (wireless). All these communications between the quadrotor and the Simulink environment are undertaken with embedded C MEX files via a universal serial bus communication port. A complete explanation of the quadrotors is included in the next section.

The advantage of using a closed-loop indoor testbed such as this is that it is a highly controlled environment. This allows the results for different control laws to be compared accurately and with assurance that the results are not influenced by real-world factors, such as environmental disturbances (e.g., wind), sensor noise, transport or real-time delays, or insufficient processing power.

A. The Quadrotor UAV

A quadrotor, also known as a quadcopter, is a UAV that is lifted and propelled by four rotors. Quadrotors are classified as rotorcraft, as opposed to fixed-wing aircraft, because their lift is generated by a set of revolving narrow-chord airfoils.

The control law inputs, $[\delta_r, \delta_p, \delta_y, \delta_c]$, correspond to roll, pitch, thrust, and yaw respectively; the range of each input is $[-1\ 1]$. As in (61), the angular velocities are linearly proportional to the attitude inputs. Thrust initially is modeled as linear but is postprocessed with a nonlinear

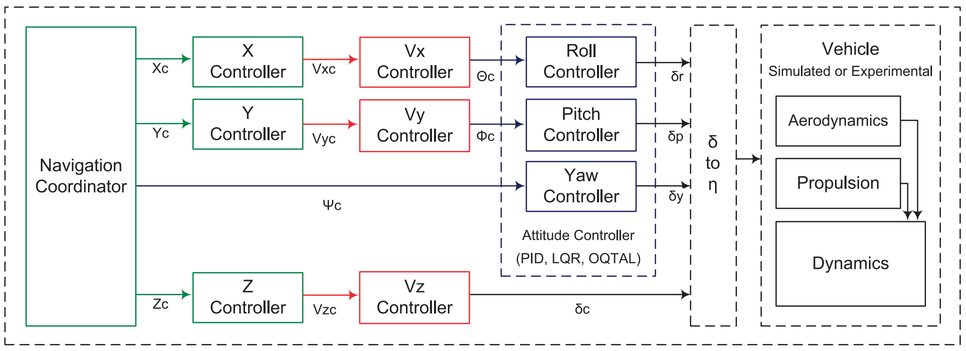


Fig. 16. Control architecture, showing entire control architecture for simulated and experimental work. Outer loops are PID, and internal attitude control loop is one of the following: PID, LQR, or OQTAL.

characteristic:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} K_\phi \cdot \delta_r \\ K_\theta \cdot \delta_p \\ K_\psi \cdot \delta_y \end{bmatrix} T = K_T \cdot \delta_c \qquad (61)$$

Here, $\boldsymbol{u} \in \mathbb{R}^3$ is the input command vector, and K_ϕ, K_θ, K_ψ, and K_T are empirically determined constants. This set of equations, along with the 6DoF presented in Section II, comprises the quadrotor model fully.

B. Control Architecture

Given that all three controllers are for attitude only, an outer PID control architecture has been setup. Fig. 16 shows this architecture in detail. One can see in this figure that there is one PID for each variable (X, Y, Z, V_X, V_Y, V_Z). The reason for this particular design is that the chosen PID optimization technique [27] presents an online tuning algorithm based on the ABC genetic algorithm, which works best in a single input–single output setup. See [27] for exact details. The control architecture and, more importantly, the gains for the outer PIDs remains the same for all three controllers under study. The rest of this section focuses on the practical details required for performing the experimental trajectories.

C. LQR Adaptation

First, the presence of the quadrotor inner attitude controller requires a change in both LQR algorithms, which corresponds to nullify the angular velocity or the angular velocity error, respectively, from the state vector. The reason for this nullification can be found in (61): it shows that our quadrotor has a built-in attitude controller that has the effect of making the quadrotor input proportional to the angular velocity and not to the angular acceleration as expected; therefore, the angular velocity cannot play any role in any feedback attitude controller, and only the rotation state (i.e., rotation angles) can be used. A similar approach was followed for the PID controller, where the term $c\omega$ in (17) has also been nullified to adapt PID control to the quadrotor UAV. The results show that even under this strong change in the attitude error model, OQTAL is still able to perform not only correctly but also better than the other tested controllers.

Second, Qualisys introduces measurement noise that, if not treated, has an impact on OQTAL terms depending on the second derivative of the quaternion error $\ddot{\epsilon}$. This is because the second derivative of a signal with noise might have unpredictably large values. To solve this problem, it is suggested to either apply some kind of noise cancellation technique like Kalman filters or assume $\ddot{\epsilon} \approx 0$ and remove the term; the latter has been selected. The elimination of this term has the positive effect of increasing stability in noisy conditions, because this term has a strong effect even in close tracking situations. On the negative side, the quaternion error model is theoretically less accurate. However, in close tracking conditions or even in a smooth trajectory tracking situation, the second derivative of the quaternion error should be small, but in a noise-exposed scenario, this is not case and the second derivative is high in any case (unless noise reduction techniques are applied). Therefore, the elimination of this term is not necessarily a loss in the quality of the model, and it might even increase its accuracy.

Third, in this real-life setup, integral action was used in both LQR controllers. The q factor is similarly defined as in (23), except that the integral part is multiplied by a factor of 0.01; therefore, the $\mathbf{Q}$ matrix is defined as follows:

$$\mathbf{Q} = q \begin{bmatrix} \mathbf{I}_6 & \mathbf{0} \\ \mathbf{0} & 0.01\mathbf{I}_3 \end{bmatrix} \qquad (62)$$

Finally, it was mentioned in the quadrotor model section that the attitude range of operation of this is small ($\approx [0 - 15^\circ]$). While this is irrelevant to OQTAL, it noticeably benefits classic LQR. This is because this small range of operation makes the linearized model more accurate for the corresponding rotational range and reduces the number of model switches required. This is visually seen in the next section, where classic LQR performs closer to OQTAL in this particular setup. Furthermore, the underlying classic LQR model becomes singularity free for this range of operation.

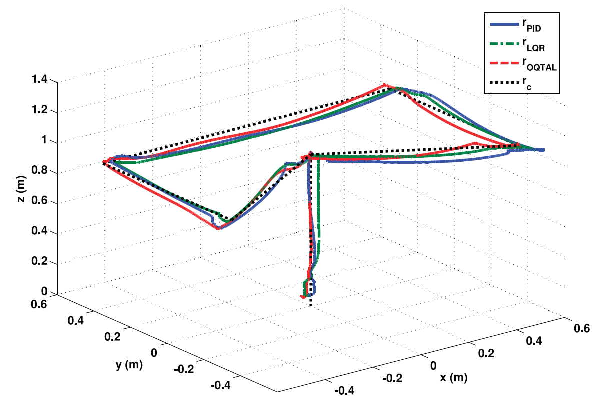


Fig. 17. Experimental sample square pattern in 3D, showing 3D position for PID and OQTAL plus integral.

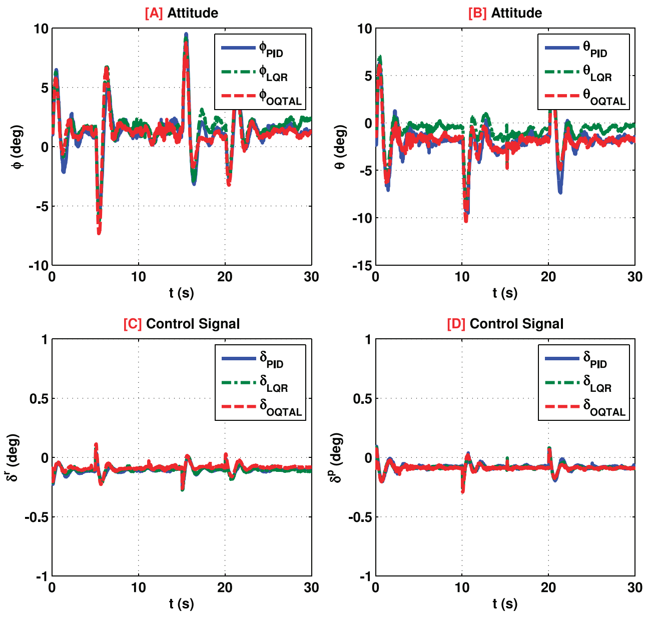


Fig. 19. Experimental sample square pattern attitude and control signal, showing (A) attitude (roll), (B) control signal (roll), (C) attitude (pitch), and (D) control signal (pitch).

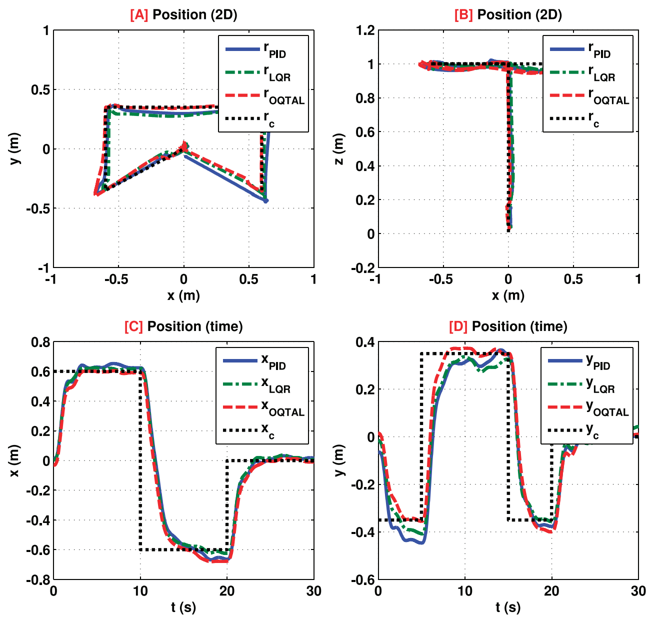


Fig. 18. Experimental sample square pattern position, showing (A) 2D position (top-down view), (B) 2D position (side view), (C) *x*-position time history, and (D) *y*-position time history.

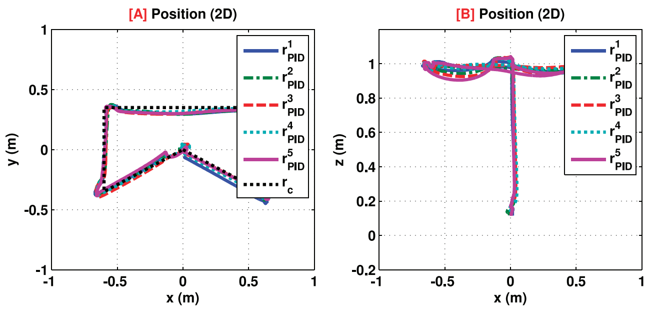


Fig. 20. Experimental repeated square pattern PID, showing (A) 2D position (top-down view) and (B) 2D position (side view).

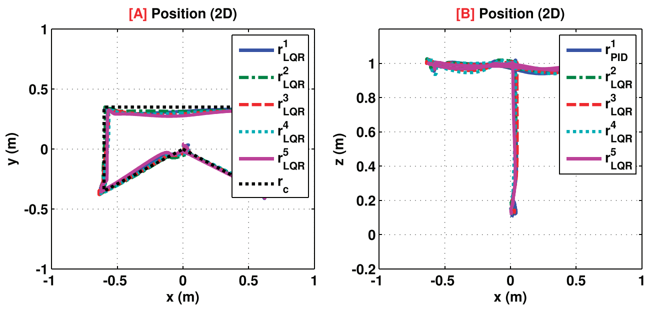


Fig. 21. Experimental repeated square pattern LQR, showing (A) 2D position (top-down view) and (B) 2D position (side view).

VIII. EXPERIMENTAL RESULTS ANALYSIS

The campaign of tests performed consists of at least 25 runs per controller. This was done for two reasons. First, we wanted to have the certainty that all controllers repeatedly work correctly and stably. Second, to check the variability of them as a small standard deviation means more accurate tracking occurs.

Three sample runs, one for each controller, are presented in Figs. 17–19. Fig. 17 shows the whole trajectory in a three-dimensional (3D) graphic; this is to give an idea of the designed trajectory pattern. Fig. 18 shows two-dimensional (2D) versions of the same. Within this set of four graphics, the first one (X vs. Y) shows more clearly that OQTAL tracking is visually better than the rest of them and able produce very small position errors in some segments. Finally, Fig. 19 shows the attitude tracking for roll and pitch. This shows also how close all controllers are to one another; therefore, all three controllers are well tuned.

Moreover, Figs. 20–22 show five repeated executions placed together in the same graphics for each controller. Again, all runs for all controllers are similar, showing that all controllers, including the outer PIDs, have been well tuned.

Finally, Table V shows a summary of the performances. From all runs, the best five trajectories

TABLE V
Repeated Experimental Results

Controller	Pos error (m) Average	Pos error (m) SD	Vel error (m/s) Average	Vel error (m/s) SD	Att error (°) Average	Att error (°) SD	Max Q	Calcs
PID	0.330	**0.005**	0.147	0.004	1.192	0.098	—	**0**
LQR	0.323	0.005	0.143	0.002	1.575	0.047	≤20	89
OQTAL	**0.306**	0.012	**0.139**	**0.001**	**1.184**	**0.039**	**≤200**	**1**

Pos = position, Vel = velocity, Att = attitude, Max = maximum, Calcs = calculations, SD = standard deviation.

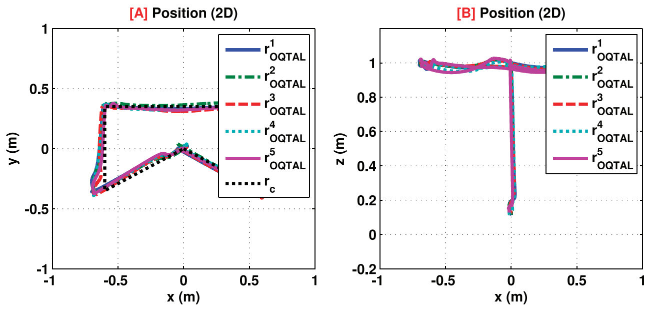


Fig. 22. Experimental repeated square pattern OQTAL plus integral, showing (A) 2D position (top-down view) and (B) 2D position (side view).

(these with the least amount of position error) of each controller were selected. For each set of five runs, the accumulated position, linear velocity, and attitude error were used to obtain average and standard deviation. Another two columns were added, both only applicable to LQR. The first is the range of the matrix $\mathbf{Q}$ given a matrix $\mathbf{R} = \mathbf{I}_3$. This matrix was incremented iteratively for each controller until the system started to lose stability. The second one is the number of calculations required for a threshold of $\delta = 0.5$.

Bold items in Table V signify the controller that yields the best performance for the respective metric. OQTAL plus integral performs better in all measured metrics except the standard deviation of the position error. More importantly, the number of calculations and the quaternion error are better in OQTAL.

IX. CONCLUSION

This paper has shown the research effort of formally combining an optimal control algorithm like LQR with more a accurate representation of the attitude tracking error than the ones usually chosen—in particular, a comprehensive quaternion error dynamic model. This optimal attitude tracking control has been named OQTAL. Its underlying mathematical model has been formally developed and proved, its stability has been proved, and the resulting optimal control has been fully described. Furthermore, a large campaign of tests has been performed on both a simulated environment and in a real experimental setup. In particular, the exhaustive optimization process carried for classic LQR and OQTAL in the gravity turn experiment demonstrated the potential of OQTAL as a powerful attitude tracking control. In all these scenarios, OQTAL has not only shown better performance than the other standard controller implementations used for comparison but also required a very low computation time, behaving sometimes like an LTI system. Moreover, OQTAL has responded extremely well to the important attitude model change required by the SSC quadrotor. However, the cost matrices have been empirically shown to depend on both the UAV model and the particular scenario in which OQTAL is used. OQTAL is still advantageous as the optimal attitude tracking problem becomes a cost matrix optimization problem. It remains for future work to find out how each nonlinear factor affects these matrices. Nevertheless, all features that the authors were expecting for this combined control named OQTAL are present in it, making this controller potentially an excellent practical alternative for UAV attitude control.

APPENDIX A. VECTOR CALCULUS IDENTITIES

The following vector calculus identities involve cross products and cross-product differentiation:

$$\mathbf{x} \times \mathbf{y} = -\mathbf{y} \times \mathbf{x} \tag{63}$$

$$\frac{\partial \mathbf{x} \times \mathbf{y}}{\partial \mathbf{x}} = \mathbf{x} \times \frac{\partial \mathbf{y}}{\partial \mathbf{x}} + \frac{\partial \mathbf{x}}{\partial \mathbf{x}} \times \mathbf{y} \tag{64}$$

The next results follow from the previous properties:

$$\frac{\partial\left(\mathbf{x} \times \mathbf{y}\right)}{\partial \mathbf{y}} = \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \times \mathbf{y} + \mathbf{x} \times \frac{\partial \mathbf{y}}{\partial \mathbf{y}} = \mathbf{x} \times \mathbf{I}_{n\times n} = \mathbf{x}^{\times} \tag{65}$$

$$\frac{\partial\left(\mathbf{x} \times \mathbf{y}\right)}{\partial \mathbf{x}} = -\frac{\partial\left(\mathbf{y} \times \mathbf{x}\right)}{\partial \mathbf{x}} = -\mathbf{y}^{\times} \tag{66}$$

The following properties were derived by the author. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the following applies:

$$\frac{\partial\left(\mathbf{x}\mathbf{x}^{*}\mathbf{y}\right)}{\partial \mathbf{x}} = \mathbf{x}\mathbf{y}^{*} + \mathbf{x}^{*}\mathbf{y}\mathbf{I}_{n\times n} \tag{67}$$

$$\frac{\partial\left(\mathbf{x}^{*}\mathbf{x}\mathbf{y}\right)}{\partial \mathbf{x}} = 2\mathbf{y}\mathbf{x}^{*} \tag{68}$$

APPENDIX B. FULL JACOBIAN LINEARIZATION OF THE QUATERNION ERROR DYNAMICS

In the calculation of $\mathbf{A}_{\dot{\epsilon}\epsilon}$,

$$\mathbf{A}_{\dot{\epsilon}\epsilon} \triangleq \frac{\partial \dot{\epsilon}}{\partial \epsilon} = \frac{\partial\left[\left(\lambda \mathbf{I}_3 + \epsilon^{\times}\right)\omega_e/2\right]}{\partial \epsilon} \tag{69}$$

and we keep only the terms that depend on ϵ and apply (66):

$$\frac{\partial \dot{\epsilon}}{\partial \epsilon} = \frac{\partial \left[\epsilon^{\times} \omega_e / 2\right]}{\partial \epsilon} = -\frac{1}{2} \boldsymbol{\omega}_{\mathbf{e}}^{\times} \tag{70}$$

In the calculation of $\mathbf{A}_{\dot{\epsilon}\omega_e}$,

$$\mathbf{A}_{\dot{\epsilon}\omega_e} \triangleq \frac{\partial \dot{\epsilon}}{\partial \omega_e} = \frac{\partial \left[\left(\lambda \mathbf{I}_3 + \epsilon^{\times}\right) \omega_e / 2\right]}{\partial \omega_e} \tag{71}$$

and we split the sum of partial derivatives and apply (65):

$$\frac{\partial \dot{\epsilon}}{\partial \omega_e} = \frac{\lambda \partial \omega_e}{2 \partial \omega_e} + \frac{\partial \left(\epsilon^{\times} \omega_e\right)}{2 \partial \omega_e} = \frac{1}{2}\left(\epsilon^{\times} + \lambda \mathbf{I}_3\right) \tag{72}$$

In the calculation of $\mathbf{A}_{\dot{\omega}_e \epsilon}$,

$$\mathbf{A}_{\dot{\omega}_e \epsilon} \triangleq \frac{\partial \dot{\omega}_e}{\partial \epsilon} = \frac{\partial \left[2\left(\lambda \mathbf{I}_3 - \epsilon^{\times} + \frac{\epsilon \epsilon^t}{\lambda}\right)\left[\ddot{\epsilon} + \frac{\omega_e{}^t \omega_e}{4} \epsilon\right]\right]}{\partial \epsilon} \tag{73}$$

and we expand the multiplication, remove the independent terms, extract scalars (λ and $\omega_e{}^t\, \omega_e$), and split the sum of partial derivatives:

$$\frac{\partial \dot{\omega}_e}{\partial \epsilon} = \frac{\lambda \omega_e{}^t \omega_e}{2} \frac{\partial \epsilon}{\partial \epsilon} - \frac{2 \partial \left(\epsilon^{\times} \ddot{\epsilon}\right)}{\partial \epsilon} - \frac{\omega_e{}^t \omega_e}{2} \frac{\partial \left(\epsilon^{\times} \epsilon\right)}{\partial \epsilon} + \frac{2}{\lambda} \frac{\partial \left(\epsilon \epsilon^t \ddot{\epsilon}\right)}{\partial \epsilon} + \frac{\omega_e{}^t \omega_e}{2\lambda} \frac{\partial \left(\epsilon \epsilon^t \epsilon\right)}{\partial \epsilon} \tag{74}$$

Now, applying the rules explained earlier and reorganizing the terms, we get the following:

$$\frac{\partial \dot{\omega}_e}{\partial \epsilon} = 2\left(\ddot{\epsilon}^{\times} + \frac{1}{\lambda}\left(\epsilon \ddot{\epsilon}^t + \epsilon^t \ddot{\epsilon} \mathbf{I}_3\right) + \frac{\omega_e{}^t \omega_e}{4}\left(\lambda \mathbf{I}_3 - 2\epsilon^{\times} + \frac{1}{\lambda}\left(\epsilon \epsilon^t + \epsilon^t \epsilon \mathbf{I}_3\right)\right)\right) \tag{75}$$

In the calculation of $\mathbf{A}_{\dot{\omega}_e \omega_e}$,

$$\mathbf{A}_{\dot{\omega}_e \omega_e} \triangleq \frac{\partial \dot{\omega}_e}{\partial \omega_e} = \frac{\partial \left[2\left(\lambda \mathbf{I}_3 - \epsilon^{\times} + \frac{\epsilon \epsilon^t}{\lambda}\right)\left[\ddot{\epsilon} + \frac{\omega_e{}^t \omega_e}{4} \epsilon\right]\right]}{\partial \epsilon} \tag{76}$$

and we expand the multiplication, remove the independent terms, and split the sum of partial derivatives:

$$\frac{\partial \dot{\omega}_e}{\partial \omega_e} = \frac{\lambda}{2} \frac{\partial \left(\omega_e{}^t \omega_e \epsilon\right)}{\partial \epsilon} - \frac{1}{2} \frac{\partial \left(\epsilon^{\times} \omega_e{}^t \omega_e \epsilon\right)}{\partial \epsilon} + \frac{1}{2\lambda} \frac{\partial \left(\epsilon \epsilon^t \omega_e{}^t \omega_e \epsilon\right)]}{\partial \epsilon} \tag{77}$$

Reorganizing the scalar $\omega_e{}^t\, \omega_e$ and using $\epsilon^{\times}\, \epsilon = 0$ for convenience, we get the following:

$$\frac{\partial \dot{\omega}_e}{\partial \omega_e} = \frac{\lambda}{2} \frac{\partial \left(\omega_e{}^t \omega_e \epsilon\right)}{\partial \epsilon} - \frac{1}{2\lambda} \frac{\partial \left(\omega_e{}^t \omega_e \left(\epsilon \epsilon^t \epsilon\right)\right)}{\partial \epsilon} \tag{78}$$

Then, applying the properties explained earlier, we get the following:

$$\frac{\partial \dot{\omega}_e}{\partial \omega_e} = \left(\lambda \mathbf{I}_3 + \frac{\epsilon \epsilon^t}{\lambda}\right) \frac{\epsilon}{2} \omega_e{}^t \tag{79}$$

REFERENCES

[1] Wie, B., Weiss, H., and Arapostathis, A.
A quaternion feedback regulator for spacecraft eigenaxis rotations.
AIAA Journal of Guidance, Control, and Dynamics, **12**, 3 (1989), 375–380.

[2] Wie, B., Weiss, H., and Arapostathis, A.
Rapid multitarget acquisition and pointing control of agile spacecraft.
AIAA Journal of Guidance, Control, and Dynamics, **25**, 1 (2002), 96–104.

[3] Wen, J.-Y., and Kreutz-Delgado, K.
The attitude control problem.
IEEE Transactions on Automatic Control, **36**, 10 (1991), 1148–1162.

[4] Mayhew, C., Sanfelice, R., and Teel, A.
Quaternion-based hybrid control for robust global attitude tracking.
IEEE Transactions on Automatic Control, **56**, 11 (2011), 2555–2566.

[5] Chaturvedi, N., Sanyal, A., and McClamroch, N.
Rigid-body attitude control.
IEEE Control Systems, **31**, 3 (2011), 30–51.

[6] Bach, R., and Paielli, R.
Linearization of attitude-control error dynamics.
IEEE Transactions on Automatic Control, **38**, 10 (1993), 1521–1525.

[7] Jiang, Z., Han, J., Wang, Y., and Song, Q.
Enhanced LQR control for unmanned helicopter in hover.
In *Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics*, 2006, Harbin, China, 1438–1443.

[8] Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J.
Quadrotor helicopter trajectory tracking control.
In *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, HI, 2008, 1–14.

[9] Beatty, S.
Comparison of PD and LQR methods for spacecraft attitude control using star trackers.
In *Proceedings of the World Automation Congress*, Budapest, Hungary, 2006, 1–6.

[10] Kim, B., Chang, Y., Keh, J., Ha, H., and Lee, M.
Design of 6-DOF attitude controller of hovering model helicopter.
In *Proceedings of the 30th Annual Conference of IEEE Industrial Electronics Society*, Busan, South Korea, 2004, Vol. 1, 104–110.

[11] Antunes, D., Silvestre, C., and Cunha, R.
On the design of multi-rate tracking controllers: An application to rotorcraft guidance and control.
In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, Aug. 2007, 1–23.

[12] Fjellstad, O.-E., and Fossen, T.
Singularity-free tracking of unmanned underwater vehicles in 6 DOF.
In *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, FL, 1994, Vol. 2, 1128–1133.

[13] Friis, J., Nielsen, E., Andersen, R. F., Bønding, J., Jochumsen, A., and Sørensen, A. F.
Autonomous landing on a moving platform.
Department of Electronic Systems, Aalborg University, Aalborg, Denmark, Tech Rep., 2009.

[14] Lai, G., Fregene, K., and Wang, D.
A control structure for autonomous model helicopter navigation.
In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Halifax, Nova Scotia, 2000, Vol. 1, 103–107.

[15] Bouabdallah, S.
Design and control of quadrotors with application to autonomous flying.
Ph.D. dissertation, Ecole Polytechnique Federale De Lausanne, Switzerland, 2007.

[16] Whang, I.-H., and Cho, S.
LQR gain-schedule controller for vertical line following.
Electronics Letters, **46**, 14 (2010), 991–993.

[17] Yu, G.-R.
Nonlinear optimal control of helicopter using fuzzy gain scheduling.
In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Waikoloa, HI, 2005, Vol. 4, 3313–3318.

[18] Mortensen, R. E.
A globally stable linear attitude regulator.
International Journal of Control. [Online]. **8**, 3 (1968), 297–302. Available: http://www.tandfonline.com/doi/abs/10.1080/00207176808905679.

[19] Zipfel, P.
Modeling and Simulation of Aerospace Vehicle Dynamics.
AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 2007.

[20] Adir, V. G., and Stoica, A. M.
Integral LQR control of a star-shaped octorotor.
INCAS Bulletin. [Online]. **4**, 2 (2012), 3–18. Available: http://www.dx.doi.org/10.13111/2066-8201.2012.4.2.1.

[21] Lewis, A. D., and Tyner, D. R.
Geometric Jacobian linearization and LQR theory.
Journal of Geometric Mechanics, **2**, 4 (2010), 397–440.

[22] Forshaw, J. L., Lappas, V. J., and Briggs, P.
Indoor experimentation and flight test results for a twin rotor tailsitter unmanned air vehicle.
In *Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference*, Boston, MA, Aug. 2013.

[23] McInnes, C. R.
Gravity turn descent with quadratic air drag.
Journal of Guidance, Control, and Dynamics, **20**, 2 (1997), 393–394.

[24] Test facility detailed requirements document.
In *Precision Landing GNC Test Facility.* Paris, France: European Space Agency, 2006.

[25] Smith, R.
Closed-loop aeromaneuvering for a mars precision landing.
University of California–Santa Barbara, Santa Barbara, CA, Tech Rep. CCEC-96-0205, 1996.

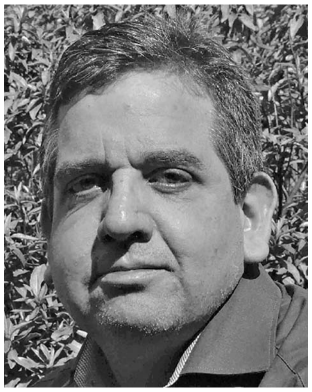
[26] Smith, R., Bayard, D., and Mease, K.
Mars precision landing: An integrated estimation, guidance and control simulation.
University of California–Santa Barbara, Santa Barbara, CA, Tech Rep. CCEC-98-0918, 1998.

[27] Ghiglino, P., Forshaw, J. L., and Lappas, V. J.
Online PID self-tuning using an evolutionary swarm algorithm with experimental quadrotor flight results.
In *Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference*, Boston, MA, Aug. 2013.

[28] Gao, F., Qi, Y., Yin, Q., and Xiao, J.
An novel optimal PID tuning and on-line tuning based on Artificial Bee Colony algorithm.
In *Proceedings of the International Conference on Computational Intelligence and Software Engineering*, Wuhan, China, 2010, 1–4.

[29] Shang, Y., Kristiansen, K. K. U., and Palmer, P.
Dynamic systems approach to the lander descent problem.
Journal of Guidance, Control, and Dynamics, **34** (2011), 911–915.

[30] Forshaw, J. L., and Lappas, V. J.
Transitional control architecture, methodology, and robustness for a twin helicopter rotor tailsitter.
In *Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference*, Minneapolis, MN, Aug. 2012.



Pablo Ghiglino was born in Chile in 1973. He undertook his first degree in electric and electronics engineering at the Universidad Politécnica de Madrid (Spain) in 1995 and completed his M.Sc. degree at the Institute of Automation and Software Engineering at the University of Stuttgart (Germany) in 1996. He has worked in the information technology (IT) development industry since 1997 and currently is a freelance IT specialist developer in the investment banking sector. Along with his full-time job, he is pursuing a Ph.D. degree at the SSC (United Kingdom) in the area that he feels most passionate about. Throughout his life, he has received several awards, including finalist in the national Chilean math Olympics (1990), inclusion among the all-time top students at the University of Stuttgart (1996), Banking Technologies Awards at Credit-Suisse (2010), and inclusion in the *Who's Who 2011* book.

Jason L. Forshaw was born in the United Kingdom in 1986. He received an M.Eng. (Hons.) degree in aerospace engineering from the University of Sheffield in 2008 (visiting student at Virginia Polytechnic Institute and State University in 2007), an M.S. degree in aeronautics and astronautics from Stanford University in 2010, and a Ph.D. degree in electronics engineering at the SSC (United Kingdom) in 2013. As a postdoctoral research fellow at the SSC, he provides technical direction and leadership as a control systems specialist on the following projects: Intrepid (Airbus DS), SME-Sat (FP7), Service Orientated Active Debris Removal (SSTL, ESA), and RemoveDebris (FP7). His other industrial experience includes QinetiQ, Rolls-Royce, and BAE Systems. Research interests and publications include modeling and simulation, controls, and testbed design for both UAV and space applications. Dr. Forshaw is a member of the IET, IEEE, and a senior member of the AIAA. He has a range of awards and scholarships from both university and professional institutions, including the Sir Basil Blackwell Award, the Sir Frederick Mappin Medal, and a Stanford University fellowship.

Vaios J. Lappas received his B.Sc. degree in aerospace engineering from Ryerson University (Canada) in 1998, an M.Sc. degree in space studies from the International Space University (France) in 1999, and a Ph.D. degree in electronics engineering from the SSC (United Kingdom) in 1999. He has been the principal investigator on many ESA, commercial, U.S. Air Force, and National Aeronautics and Space Administration projects on solar sails, attitude control, control momentum gyroscopes, CubeSats, and electric propulsion. He is a professor of space vehicle control at the SSC (United Kingdom) and coordinates the following projects: DeorbitSail, Deploytech, SpacePlan2020, SME-Sat, and RemoveDebris. He participates in the FP7 projects: NEOshield and QB50. Furthermore, Prof. Lappas holds research grants with Astrium, ESA, and the Technology Strategy Board/United Kingdom Space Agency on deorbiting systems, controls, and microelectric propulsion.