Robotica (2022), **40**, pp. 3120–3135
doi:10.1017/S0263574722000091

CAMBRIDGE
UNIVERSITY PRESS

RESEARCH ARTICLE

Quaternion-based state-dependent differential Riccati equation for quadrotor drones: Regulation control problem in aerobatic flight

Saeed Rafee Nekoo*, José Ángel Acosta and Anibal Ollero

GRVC Robotics Lab., Departamento de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Seville, Spain
*Corresponding author. E-mail: saerafee@yahoo.com

Received: 7 September 2021; **Revised:** 24 November 2021; **Accepted:** 10 January 2022;
First published online: 17 February 2022

Keywords: SDDRE, Riccati, optimal control, quaternion, quadrotor, Cobra maneuver, aerobatic maneuver

Abstract

The quaternion is a powerful and common tool to avoid singularity in rotational dynamics in three-dimensional (3D) space. Here it has been particularly used as an alternative to Euler angles and rotation matrix. The application of the quaternion is exercised in quadrotor modeling and control. It changes the dynamics and represents a singularity-free attitude model. Here for the first time (for the best knowledge of authors), the state-dependent differential Riccati equation (SDDRE) control has been implemented on the quaternion-based model of a quadcopter. The proposed control structure is capable of aerobatic flight, and the Pugachev's Cobra maneuver is chosen to assess the capability of the quaternion-based SDDRE approach. The introduced control simulator is validated by comparison with conventional dynamics based on Euler angles, controlled using a proportional-derivative (PD) controller on a normal regulation flight. The simulator successfully performed the Cobra maneuver and also validated the proposed structure. The more precision in regulation along with lower energy consumption demonstrated the superiority of the introduced approach.

1. Introduction

The Euler angles and rotation matrix in three-dimensional (3D) space are vulnerable to singularities, considering dynamics, especially in an aerobatic flight. Here we focus, particularly on multirotor drones and quadcopters. The dynamics of the multirotor drones are usually subjected to hovering assumptions to guarantee a stable flight [1]. To perform agile maneuvers with sudden changes in attitude, geometric control was introduced that avoids singularities and computes the rotation matrix in another manifold [2]. Krishna et al. used the geometric control for helicopter trajectory tracking in agile flight regarding the attitude of the system [3]. The sliding mode was employed as the controller and the error function in the geometric domain constructed the sliding condition to prove the stability. The geometric control was also applied on quadrotor drones subject to wind disturbance and sudden changes in attitude dynamics [4]. A flip is representative of a challenging maneuver that includes passing through a singularity that was addressed by the geometric approach [5]. The π (rad) flip in roll angle was considered, sudden motion between two stable roll angles. The change in the manifold and solving singularity cost the user more complicated integration methods [6]. Variational integration is an effective method [7]. Improper treatment of the integration may cause numerical drift in the results, especially in big orientation changes.

The quaternion is another representation of the complex numbers in mathematics, with a wide range of usage in theory and application. The focus of this work is to implement the alternative representation of the Euler angles and rotation matrix in 3D space. The application of quaternion in control was reported

on aircraft [8, 9], orientation and translation control of manipulators [10, 11], control of autonomous underwater vehicles [12–15], helicopter attitude control [16], etc. Terze et al. used quaternion representation of the rotational dynamics for aircraft simulators and introduced shifting update process to ensure precise integration in long flight simulations [8]. Kinematics control of a robotic arm was presented using dual quaternion, aimed to present a robust controller without decoupling of the translational and orientation dynamics [10]. Cooke et al. presented a thorough implementation of the quaternion dynamics for flight simulation [17]. The performance of the quaternion-based control was validated by an omnidirectional intelligent navigator for an underwater platform [12]. Suzuki et al. presented the simulation and experimental studies on Lepton-Ex unmanned helicopter using quaternion feedback in control [16]. Quadrotor and multicopters were also employed quaternions [18–23]. A study compared three controllers, linear quadratic regulator (LQR), proportional-derivative (PD), and model predictive controller (MPC) for trajectory tracking that revealed the PD and the LQR gained better results in an ideal condition, and the MPC was more accurate in presence of disturbance [18]. Quaternion variables in the attitude controller provided an advantage, employment of low-cost sensors due to the high-speed capability of the singularity-free controller [20]. Palomo et al. presented an observer-based method based on position-yaw for the ellipsoid method [21]. Linear matrix inequalities were used to optimize the feedback of the observer, and experimental results showed successful tracking in high speed maneuver. Euler angles are popular in UAV modeling though they suffer from gimbal lock when two orthogonal axes align and lock together [24]. Another disadvantage of Euler angles is the computational cost by computing so many trigonometric functions [25], on the contrary, quaternion mathematics only involves algebraic computations [26]. Sanchez et al. used quaternion dynamics for quadrotor control based on receiving gesture commands [27], where it was important to cope with the unpredictability of the gesture and quaternion made it more reliable due to insensitivity to singularity.

Performing experiment on aggressive maneuvers requires more safety concerns in addition to solving singularities. Gillula et al. used the Hamilton–Jacobi differential game approach for finding a reachable set for aerobatic maneuvers to guarantee safety [28]. Considering the constraint of the actuators in the flight experiment is an important issue since, during the flip, it is highly probable to put the rotors in saturation [29]. Here in this work, the simulation of the Cobra maneuver is done and for experimentation, actuator limits and speed of the maneuver must be considered to avoid undesired saturations in the middle of trajectory that could be dangerous.

The state-dependent Riccati equation (SDRE) is a closed-loop optimal nonlinear controller introduced in the 1960s [30]. The utilization of the quaternion in SDRE was explored in several platforms such as satellite [31, 32], spacecraft [33–36], spacecraft in proximity operations [37], attitude control of a rigid body [38], and remote sensing CubeSats [39]. Here we present the state-dependent differential Riccati equation (SDDRE) control based on quaternion for quadcopter dynamic systems. For the best knowledge of authors, a quaternion-based SDDRE controller has not been used for quadrotors in the literature so far which makes the first contribution of this work. The SDDRE is a finite time controller that penalizes the states (error variables) by a final boundary condition [40].

The singularity-free attitude control is the principal advantage of the quaternion. The Cobra is a challenging aerobatic flight maneuver, performed by a jet aircraft [41, 42]. The motion turns the aircraft vertical (even for pitch angle $\theta > \pi/2$) to perform the maneuver along with sudden deceleration; the thrust of the jet engine helps to avoid the system from falling.

A tail-sitter drone is a good choice, with a vertical thrust option, to perform the Cobra maneuver [43]. Xu et al. presented iterative learning control for a tail sitter unmanned aerial vertical-take-off-and-landing system for Pugachev's Cobra maneuver [43]. They used the acceleration model that resulted in simple dynamics without system identification. The Cobra was exercised by a quadrotor using adaptive control [44]. The quadrotor possessed 28(g), a small lightweight platform. An adaptive backstepping controller with a modified recursive least-square was employed to control the system.

Contributions: (1) presenting a quaternion-based SDDRE control peculiar to a quadrotor. (2) Implementing Cobra maneuver in pitch angle in a forward flight. The performed Cobra in ref. [44] was done in vertical ascending flight. In a forward flight, conducting a Cobra maneuver, the system will

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

lose its thrust ($\theta \approx \pi/2$) and is subjected to fall. Here in this work, using SDDRE, the Cobra in forward flight is done.

Section 2 describes the preliminaries in quaternion mathematics. Section 3 states the system dynamics details. Section 4 presents the SDDRE control structure. Section 5 expresses the approximate closed-form solution to the SDDRE. Section 6 presents the implementation and method and cascade design. Simulations are reported in Section 7 which includes validation and aerobatic flight, and concluding remarks are summarized in Section 8.

Notations: $\mathbb{R}^n$ denotes the n-dimensional Euclidean space, $\mathbb{H}^n$ denotes the n-dimensional Hamilton space, and $(\cdot)^*$ performs conjugate transpose. $\mathbb{R}^{n\times m}$ is the set of $n \times m$ real matrices; $(\cdot)^T$ is the transpose of a matrix or a vector; $\bigotimes$ denotes Kronecker product; diag $(\cdot)$ means a diagonal matrix; $\mathbf{I}_{n\times n}$ and $\mathbf{0}_{n\times n}$ denote $n \times n$ identity and zero matrices.

2. Preliminaries: Quaternion mathematics

Here we consider the quaternion definition with real-scalar part q_0 and vector-imaginary part $\mathbf{q}_v = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, set all together

$$\mathbf{q} = \begin{bmatrix} q_0 \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} \cos\dfrac{\vartheta}{2} \\ \mathbf{r}\sin\dfrac{\vartheta}{2} \end{bmatrix}, \tag{1}$$

where $\mathbf{r} \in \mathbb{R}^3$ is a unit rotation vector and ϑ is the corresponding rotation angle about that ref. [20]. To solve the ambiguity of the direction of quaternions, $0 \le q_0 \le 1$ is chosen. A conjugate transpose of the quaternion (1) is presented as

$$\mathbf{q}^* = \begin{bmatrix} q_0 \\ -\mathbf{q}_v \end{bmatrix}. \tag{2}$$

The multiplication product of two arbitrary quaternions $\mathbf{q}$ and $\mathbf{p}$ is defined through Kronecker product

$$\mathbf{q} \otimes \mathbf{p} = \mathbf{Q}(\mathbf{q})\mathbf{p} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}. \tag{3}$$

A unit quaternion can build a rotation transformation by two multiplications by Kronecker product that could rotate an arbitrary vector $\mathbf{v}$ from the global coordinate to a moving coordinate $\mathbf{q}$ as in the form of $\mathbf{q} \otimes \left[0, \mathbf{v}^T\right]^T \otimes \mathbf{q}^*$. So, using definitions (2) and (3), the rotation matrix is built by replacing x, y, z in turn into $\mathbf{v}$:

$$\mathbf{R}_x(\mathbf{q}) = \mathbf{q} \otimes [0,1,0,0]^T \otimes \mathbf{q}^*,$$
$$\mathbf{R}_y(\mathbf{q}) = \mathbf{q} \otimes [0,0,1,0]^T \otimes \mathbf{q}^*,$$
$$\mathbf{R}_z(\mathbf{q}) = \mathbf{q} \otimes [0,0,0,1]^T \otimes \mathbf{q}^*,$$

which form [25]:

$$\mathbf{R}(\mathbf{q}) = \left[\mathbf{R}_x(2\colon 4), \mathbf{R}_y(2\colon 4), \mathbf{R}_z(2\colon 4)\right] = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_0q_1 + 2q_2q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}.$$

where that is $\mathbf{R}_x(2\colon 4)$ selects three last components of $\mathbf{R}_x$.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

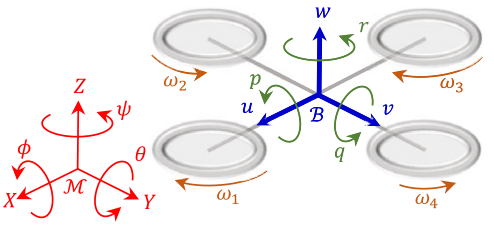

Fig. 1. *The definition of the reference coordinates for a sample quadrotor drone.*

The angular velocity vector of the quaternion is accessible (supposedly) in local moving coordinate; it is presented by the vector $\boldsymbol{\omega}(\mathbf{q}, \dot{\mathbf{q}})$:$\mathbb{H} \times \mathbb{R}^4 \to \mathbb{R}^3$ where $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T \left(\frac{\text{rad}}{\text{s}}\right)$. Then the derivative of the quaternion is found [45]:

$$\dot{\mathbf{q}}_{\omega} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} = \frac{1}{2}\mathbf{Q}(\mathbf{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{4}$$

in which $\dot{\mathbf{q}}_{\omega}(\mathbf{q}, \boldsymbol{\omega})$: $\mathbb{H} \times \mathbb{R}^3 \to \mathbb{R}^4$ and $\mathbf{Q}(\mathbf{q})$ has been introduced in Eq. (3).

The relation between the Euler angles (φ, θ, ψ) and quaternions is also needed for the control design [25]:

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\varphi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\varphi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \sin\frac{\varphi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\varphi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\varphi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\varphi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\varphi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\varphi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix}, \tag{5}$$

and also with inverse mapping, one could find:

$$\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(2\left(q_0q_1 + q_2q_3\right), q_0^2 - q_1^2 - q_2^2 + q_3^2\right) \\ \text{asin}\left(2\left(q_0q_2 - q_1q_3\right)\right) \\ \text{atan2}\left(2\left(q_0q_3 + q_1q_2\right), q_0^2 + q_1^2 - q_2^2 - q_3^2\right) \end{bmatrix}.$$

3. Dynamics of the system

Consider a "plus-shaped" quadrotor drone with two moving and fixed reference coordinates, body frame denoted by $\mathcal{B}$, and global frame marked with $\mathcal{M} = \{X, Y, Z\}$, with respect, see Fig. 1. The position of the moving coordinate is defined through the vector $\boldsymbol{\xi}_1(t) = \left[x_c(t), y_c(t), z_c(t)\right]^T$ (m) . The kinematics equation is

$$\dot{\boldsymbol{\xi}}_1(t) = \mathbf{R}(\mathbf{q}(t))\boldsymbol{v}_1(t), \tag{6}$$

where $\boldsymbol{v}_1(t) = [u(t), v(t), w(t)]^T \left(\frac{\text{m}}{\text{s}}\right)$, and $\mathbf{R}(\mathbf{q})$:$\mathbb{H} \to \mathbb{R}^{3\times3}$ is the quaternion-based rotation matrix; and $\{\varphi, \theta, \psi\}$ (rad) are Euler angles set in global coordinate. The local angular velocity vector set on the body frame is also named $\boldsymbol{v}_2(t) = \left[p(t), q(t), r(t)\right]^T \left(\frac{\text{rad}}{\text{s}}\right)$.

To find the dynamics of the system, Newton–Euler equation could be used that results in two sets of dynamic equations, the first set is translational:

$$m\ddot{\boldsymbol{\xi}}_1(t) = \mathbf{R}_3(\mathbf{q}(t))T_B(t) - mg\mathbf{e}_3, \tag{7}$$

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

where $\mathbf{R}_3(\mathbf{q})$ is the last column of $\mathbf{R}(\mathbf{q})$, $g = 9.81\ \left(\frac{\mathrm{m}}{\mathrm{s}^2}\right)$ is the gravity constant, m (kg) is the total mass of the drone, and $\mathbf{e}_3 = [0,0,1]^T$. The second set is rotational dynamic:

$$\mathbf{J}\dot{\boldsymbol{\upsilon}}_2(t) = \boldsymbol{\tau}_{\mathrm{B}}(t) - \boldsymbol{\upsilon}_2(t) \times \mathbf{J}\boldsymbol{\upsilon}_2(t), \tag{8}$$

where $\boldsymbol{\tau}_{\mathrm{B}}(t) = \left[\tau_x(t), \tau_y(t), \tau_z(t)\right]^T$ (Nm) is the input torque vector and $\mathbf{J} = \mathrm{diag}\left(I_{xx}, I_{yy}, I_{zz}\right)\ \left(\mathrm{kgm}^2\right)$ is the inertia matrix assigned in the body frame.

The state-vector is

$$[\mathbf{x}(t)]_{13\times 1} = \left[\boldsymbol{\xi}_1^T(t), \mathbf{q}^T(t), \dot{\boldsymbol{\xi}}_1^T(t), \dot{\boldsymbol{\upsilon}}_2^T(t)\right]^T,$$

which provides the state-space representation of the multi-copter using Eqs. (4)–(8) and setting $\boldsymbol{\omega}(t) = \boldsymbol{\upsilon}_2(t) = \left[p, q, r\right]^T$ in Eq. (4):

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{R}(\mathbf{q}(t))\boldsymbol{\upsilon}_1(t) \\ \frac{1}{2}\mathbf{Q}(\mathbf{q}(t)) \begin{bmatrix} 0 \\ \boldsymbol{\upsilon}_2(t) \end{bmatrix} \\ \frac{1}{m}\left(\mathbf{R}_3(\mathbf{q}(t))T_{\mathrm{B}}(t) - mg\mathbf{e}_3 - \mathbf{D}\dot{\boldsymbol{\xi}}_1(t)\right) \\ \mathbf{J}^{-1}(\boldsymbol{\tau}_{\mathrm{B}}(t) - \boldsymbol{\upsilon}_2(t) \times \mathbf{J}\boldsymbol{\upsilon}_2(t)) \end{bmatrix}. \tag{9}$$

In state-space Eq. (9), $\mathbf{D} \in \mathbb{R}^{3\times 3}$ collects drag and aerodynamics parameters of the quadcopter model and has been added to complete the model.

4. The state-dependent differential Riccati equation controller design

Consider the time-invariant affine-in-control nonlinear system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}\left(\mathbf{x}(t), \mathbf{u}(t)\right), \tag{10}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^m$ is the input vector, $\mathbf{f}(\mathbf{x}(t))$: $\mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))$: $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the dynamics and they satisfy local Lipschitz condition. n is the dimension of the state vector and m is the total number of actuators. The system Eq. (10) is transformed into state-dependent coefficient (SDC) parameterization [46]:

$$\mathbf{f}(\mathbf{x}(t)) = \mathbf{A}(\mathbf{x}(t))\mathbf{x}(t), \tag{11}$$

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{B}(\mathbf{x}(t))\mathbf{u}(t), \tag{12}$$

in which $\mathbf{B}(\mathbf{x}(t))$: $\mathbb{R}^n \to \mathbb{R}^{n\times m}$ and $\mathbf{A}(\mathbf{x}(t))$: $\mathbb{R}^n \to \mathbb{R}^{n\times n}$ are held. The pair of $\{\mathbf{A}(\mathbf{x}(t)), \mathbf{B}(\mathbf{x}(t))\}$ is a controllable parameterization of system (10) [47].

The cost function of the SDDRE is structured as

$$J(\cdot) = \frac{1}{2}\mathbf{x}^T(t_{\mathrm{f}})\mathbf{F}\mathbf{x}(t_{\mathrm{f}}) + \frac{1}{2}\int_0^{t_{\mathrm{f}}} \left[\mathbf{x}^T(t)\mathbf{Q}(\mathbf{x}(t))\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}(\mathbf{x}(t))\mathbf{u}(t)\right] \mathrm{d}t, \tag{13}$$

where $t_{\mathrm{f}} \in \mathbb{R}^+$ is the final time of the control task, $\mathbf{Q}(\mathbf{x}(t))$: $\mathbb{R}^n \to \mathbb{R}^{n\times n}$, $\mathbf{R}(\mathbf{x}(t))\mathbb{R}^n \to \mathbb{R}^{m\times m}$, and $\mathbf{F} \in \mathbb{R}^{n\times n}$ are weighting matrices, for states and inputs in $t \in [0, t_{\mathrm{f}})$ and states at the final time t_{f} with respect. The pair of $\left\{\mathbf{A}(\mathbf{x}(t)), \mathbf{Q}^{\frac{1}{2}}(\mathbf{x}(t))\right\}$ is an observable parameterization of system (10) where $\mathbf{Q}^{\frac{1}{2}}(\mathbf{x}(t))$ is the Cholesky decomposition of weighting matrix in (13).

The control law is defined by applying $\frac{\partial \mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t))}{\partial \mathbf{u}(t)} = \mathbf{0}$, stationary condition, on Hamiltonian function $\mathcal{H}\left(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t)\right) = J(\cdot) + \boldsymbol{\lambda}^T(t)\dot{\mathbf{x}}(t)$:

$$\mathbf{u}(t) = -\mathbf{R}^{-1}(\mathbf{x}(t))\mathbf{B}^T(\mathbf{x}(t))\mathbf{K}(\mathbf{x}(t))\mathbf{x}(t), \tag{14}$$

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

where $\boldsymbol{\lambda}(t) = \mathbf{K}(\mathbf{x}(t))\mathbf{x}(t)$ is the co-state vector and $\mathbf{K}(\mathbf{x}(t))$: $\mathbb{R}^n \rightarrow \mathbb{R}^{n\times n}$ is the control gain of the control equation, SDDRE.

Using the necessary condition for optimality, $\frac{\partial \mathcal{H}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t))}{\partial \mathbf{x}(t)} = -\dot{\boldsymbol{\lambda}}(t)$ and the derivative of the co-state vector $\dot{\boldsymbol{\lambda}} = \dot{\mathbf{K}}\mathbf{x} + \mathbf{K}\dot{\mathbf{x}}$, one could find:

$$\dot{\mathbf{K}}(\mathbf{x}) + \mathbf{K}(\mathbf{x})\mathbf{A}(\mathbf{x}) - \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) + \mathbf{A}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) + \mathbf{Q}(\mathbf{x}) + \boldsymbol{\Pi}(\mathbf{x}) = \mathbf{0},$$

where $\boldsymbol{\Pi}(\mathbf{x})$ collects the derivative terms (see the complete equation in ref. [46]), then the SDDRE is represented as [48]:

$$-\dot{\mathbf{K}}(\mathbf{x}) = \mathbf{Q}(\mathbf{x}) - \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) + \mathbf{K}(\mathbf{x})\mathbf{A}(\mathbf{x}) + \mathbf{A}^T(\mathbf{x})\mathbf{K}(\mathbf{x}), \tag{15}$$

with $\mathbf{K}\,(\mathbf{x}\,(t_{\mathrm{f}})) = \mathbf{F}$, a final boundary condition.

5. An approximate closed-form solution to the SDDRE

The quadrotor dynamics in Eq. (9) must be controlled by the SDDRE approach. The extended linearization model of (9) is so-called state-dependent coefficient parameterization (or apparent linearization) [49]. To solve the SDDRE (15) and to find the control gain, several methods could be used such as backward integration (BI) [50], state-transition matrix [51], and Lyapunov-based approach [52]. The BI imposes a two-round solution that might not be proper for systems that need frequent changes in initial and final conditions, nonrepetitive systems. The STM approach is not computationally robust when the final time is long [53], then the Lyapunov-based approach has been selected for this work. It works with both positive and negative solutions to the Riccati equation and delivers an approximate closed-form answer, in just one round. This work uses Lyapunov-based method via negative root to the related SDRE, $\mathbf{K}_{\mathrm{ss}}^{-}(\mathbf{x})$ to find the symmetric positive-definite solution to the SDDRE, $\mathbf{K}(\mathbf{x}(t))$ [50]. Subtracting (15) from

$$\mathbf{A}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}) + \mathbf{K}_{\mathrm{ss}}(\mathbf{x})\mathbf{A}(\mathbf{x}) + \mathbf{Q}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}) = \mathbf{0}, \tag{16}$$

generates

$$\begin{aligned}-\dot{\mathbf{K}}(\mathbf{x}) = {} & [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{A}(\mathbf{x}) + \mathbf{A}^T(\mathbf{x})\,[\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})] \\ & - \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) \\ & + \mathbf{K}_{\mathrm{ss}}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}).\end{aligned} \tag{17}$$

Adding and subtracting $\mathbf{KBR}^{-1}\mathbf{B}^T\mathbf{K}_{\mathrm{ss}}$, $\mathbf{K}_{\mathrm{ss}}\mathbf{BR}^{-1}\mathbf{B}^T\mathbf{K}$ and $\mathbf{K}_{\mathrm{ss}}\mathbf{BR}^{-1}\mathbf{B}^T\mathbf{K}_{\mathrm{ss}}$ to (17) result in:

$$\begin{aligned}-\dot{\mathbf{K}}(\mathbf{x}) = {} & \mathbf{A}^T(\mathbf{x})\,[\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})] + [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{A}(\mathbf{x}) \\ & - [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}) \\ & - \left[\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x})\right]^T\,[\mathbf{K}(\,\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\,\mathbf{x}\,)] \\ & - \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) \\ & + \mathbf{K}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}) \\ & + \mathbf{K}_{\mathrm{ss}}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}(\mathbf{x}) \\ & - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}).\end{aligned} \tag{18}$$

Rewriting (18):

$$\begin{aligned}-\dot{\mathbf{K}}(\mathbf{x}) = {} & [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{A}(\mathbf{x}) + \mathbf{A}^T(\mathbf{x})\,[\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})] \\ & - [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x}) \\ & - \left[\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\mathbf{K}_{\mathrm{ss}}(\mathbf{x})\right]^T\,[\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})] \\ & - [\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})]\,\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^T(\mathbf{x})\,[\mathbf{K}(\mathbf{x}) - \mathbf{K}_{\mathrm{ss}}(\mathbf{x})],\end{aligned}$$

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

and introducing a new variable

$$\mathbf{P}^{-1}(\mathbf{x}) = \mathbf{K}(\mathbf{x}) - \mathbf{K}_{ss}(\mathbf{x}),$$

and expressing the closed-loop matrix of the system

$$\mathbf{A}_{cl}(\mathbf{x}) = \mathbf{A}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x})\mathbf{K}_{ss}(\mathbf{x}),$$

one could present

$$-\dot{\mathbf{K}}(\mathbf{x}) = \mathbf{P}^{-1}(\mathbf{x})\mathbf{A}_{cl}(\mathbf{x}) + \mathbf{A}_{cl}^{T}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}) - \mathbf{P}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}) \tag{19}$$

Regarding that $\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{P}^{-1}(\mathbf{x})\right) = -\mathbf{P}^{-1}(\mathbf{x})\dot{\mathbf{P}}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x})$, Eq. (19) should be changed to

$$\mathbf{P}^{-1}(\mathbf{x})\dot{\mathbf{P}}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}) = \mathbf{P}^{-1}(\mathbf{x})\mathbf{A}_{cl}(\mathbf{x}) + \mathbf{A}_{cl}^{T}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}) - \mathbf{P}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}),$$

and consequently, results in a state-dependent differential Lyapunov Eq. [50]:

$$\dot{\mathbf{P}}(\mathbf{x}) = \mathbf{A}_{cl}(\mathbf{x})\mathbf{P}(\mathbf{x}) + \mathbf{P}(\mathbf{x})\mathbf{A}_{cl}^{T}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x}), \tag{20}$$

with a final boundary condition $\mathbf{P}(t_f) = [\mathbf{F} - \mathbf{K}_{ss}(\mathbf{x}(t))]^{-1}$. A solution to (20) is

$$\mathbf{P}(\mathbf{x}) = \mathbf{E}(\mathbf{x}) + \exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}[\mathbf{P}(t_f) - \mathbf{E}(\mathbf{x})]\exp\left\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\right\}, \tag{21}$$

in which $\mathbf{E}(\mathbf{x}(t))$ is an answer to a state-dependent algebraic Lyapunov equation:

$$\mathbf{E}(\mathbf{x})\mathbf{A}_{cl}^{T}(\mathbf{x}) + \mathbf{A}_{cl}(\mathbf{x})\mathbf{E}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x}) = \mathbf{0}. \tag{22}$$

Proof of (21) could be checked by the substitution of Eq. (21) into (20):

$$\begin{aligned}
&\dot{\mathbf{E}}(\mathbf{x}) + \frac{\mathrm{d}\left(\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\right)}{\mathrm{d}t}\exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}[\mathbf{P}(t_f) - \mathbf{E}(\mathbf{x})]\exp\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\} \\
&\quad - \exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}\dot{\mathbf{E}}(\mathbf{x})\exp\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\} \\
&\quad + \exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}[\mathbf{P}(t_f) - \mathbf{E}(\mathbf{x})]\exp\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\}\frac{\mathrm{d}\left(\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\right)}{\mathrm{d}t} \\
&= \mathbf{E}(\mathbf{x})\mathbf{A}_{cl}^{T}(\mathbf{x}) + \exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}[\mathbf{P}(t_f) - \mathbf{E}(\mathbf{x})]\exp\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\}\mathbf{A}_{cl}^{T}(\mathbf{x}) + \mathbf{A}_{cl}(\mathbf{x})\mathbf{E}(\mathbf{x}) \\
&\quad + \mathbf{A}_{cl}(\mathbf{x})\exp\{\mathbf{A}_{cl}(\mathbf{x})(t - t_f)\}[\mathbf{P}(t_f) - \mathbf{E}(\mathbf{x})]\exp\{\mathbf{A}_{cl}^{T}(\mathbf{x})(t - t_f)\} - \mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x}).
\end{aligned} \tag{23}$$

From (22) we have

$$\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x}) = \mathbf{A}_{cl}(\mathbf{x})\mathbf{E}(\mathbf{x}) + \mathbf{E}(\mathbf{x})\mathbf{A}_{cl}^{T}(\mathbf{x}). \tag{24}$$

The algebraic Lyapunov Eq. (22) results in $\dot{\mathbf{E}}(\mathbf{x}) = \mathbf{0}$. Regarding frozen computation at each simulation time-step, we neglect the time derivative of $\mathbf{A}_{cl}(\mathbf{x})$ and as a result, $\frac{\mathrm{d}(\mathbf{A}_{cl}(\mathbf{x})(t-t_f))}{\mathrm{d}t} = \mathbf{A}_{cl}(\mathbf{x}) + \underbrace{\dot{\mathbf{A}}_{cl}(\mathbf{x})(t - t_f)}_{\mathbf{0}}$. Substituting (24) into (23), mathematical manipulation cancels all terms and shows that the solution (21) holds for Eq. (20). Since we neglected $\dot{\mathbf{A}}_{cl}(\mathbf{x})(t - t_f) \approx \mathbf{0}$ in the derivative $\frac{\mathrm{d}(\mathbf{A}_{cl}(\mathbf{x})(t-t_f))}{\mathrm{d}t}$ and considered $\dot{\mathbf{E}}(\mathbf{x}) = \mathbf{0}$, this approach is so-called a closed-form approximate solution.

The positive gain of the SDDRE (15) is regarded as $\mathbf{K}(\mathbf{x}(t)) = \mathbf{K}_{ss}(\mathbf{x}(t)) + \mathbf{P}^{-1}(\mathbf{x}(t))$, in which $\mathbf{K}_{ss}(\mathbf{x}(t))$ could be a negative definite $\mathbf{K}_{ss}^{-}(\mathbf{x}(t))$ or positive definite $\mathbf{K}_{ss}^{+}(\mathbf{x}(t))$ solution to the SDRE (16). It works with both of them.

The details of the positive and negative roots of the SDRE are reported in Sections 3.1.1 and 3.3.2 of Ref. [50]. The negative root is computationally more robust than the positive one, and it has been

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

used here in this work. Finally, notice that the negative definite solution to the SDRE (16) is the positive definite answer to

$$-\mathbf{K}_{\mathrm{n}}^{+}(\mathbf{x})\mathbf{A}(\mathbf{x}) - \mathbf{A}^{T}(\mathbf{x})\mathbf{K}_{\mathrm{n}}^{+}(\mathbf{x}) - \mathbf{K}_{\mathrm{n}}^{+}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{R}^{-1}(\mathbf{x})\mathbf{B}^{T}(\mathbf{x})\mathbf{K}_{\mathrm{n}}^{+}(\mathbf{x}) + \mathbf{Q}(\mathbf{x}) = \mathbf{0},$$

with consideration of $\mathbf{K}_{\mathrm{ss}}^{-}(\mathbf{x}(t)) = -\mathbf{K}_{\mathrm{n}}^{+}(\mathbf{x}(t))$.

6. Implementation of the SDDRE on quaternion-based dynamics

The state-space equation of the system (9) must be represented by SDC forms (11) and (12). On the one hand, the dimension of the state vector (13 states) is not compatible with the cascade approach, commonly used in quadrotor control (12 states) [5]. On the other hand, the separation of rotational and translational dynamics was reported helpful in the control design due to different speeds of them, slow and fast dynamic [54]. So, here we propose two subcontrollers for the translation and rotational dynamics, connected through cascade design. For the translation dynamic, the set of SDC parameterization is

$$\mathbf{A}_{\mathrm{t}}(\mathbf{x}(t)) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{R}(\mathbf{q}(t)) \\ \mathbf{0}_{3\times3} & -\dfrac{\mathbf{D}}{m} \end{bmatrix}, \mathbf{B}_{\mathrm{t}} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \dfrac{\mathbf{I}_{3\times3}}{m} \end{bmatrix},$$

in which hovering condition has been assumed to find $\dot{\boldsymbol{\xi}}_1 = \underbrace{\mathbf{R}(\mathbf{q})}_{\mathbf{I}}\boldsymbol{v}_1 \approx \boldsymbol{v}_1$, to make the factorization possible, and "t" stands for translation. For the orientation section, we neglect the scalar part of the quaternion, the first column, and the row of $\mathbf{Q}(\mathbf{q}(t))$ in (9), then the SDC parameterization is

$$\mathbf{A}_{\mathrm{o}}(\mathbf{x}(t)) = \begin{bmatrix} \mathbf{0}_{3\times3} & [\mathbf{Q}(2{:}\,4, 2{:}\,4)]_{3\times3} \\ \mathbf{0}_{3\times3} & -\mathbf{J}^{-1}[\boldsymbol{v}_2 \times \mathbf{I}]_{3\times3} \end{bmatrix}, \mathbf{B}_{\mathrm{o}} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{J}^{-1} \end{bmatrix},$$

where $\mathbf{Q}(2{:}\,4, 2{:}\,4)$ collects the second to fourth components of $\mathbf{Q}(\mathbf{q}(t))$ in columns and rows, and "o" stands for orientation. Based on (14), the translation control law for regulation to the desired condition rather than zero is

$$\mathbf{u}_{\mathrm{t}}(t) = -\mathbf{R}_{\mathrm{t}}^{-1}(\mathbf{x}(t))\,\mathbf{B}_{\mathrm{t}}^{T}\mathbf{K}_{\mathrm{t}}(\mathbf{x}(t)) \begin{bmatrix} \boldsymbol{\xi}_1(t) - \boldsymbol{\xi}_{1,\mathrm{des}} \\ \dot{\boldsymbol{\xi}}_1(t) - \dot{\boldsymbol{\xi}}_{1,\mathrm{des}} \end{bmatrix}, \tag{25}$$

where $\mathbf{R}$, $\mathbf{Q}_{\mathrm{t}}$, $\mathbf{F}_{\mathrm{t}}$, and $\mathbf{K}_{\mathrm{t}}$ are the corresponding matrices (with proper dimension) for the translation and Riccati equation and "des" defines the desired condition.

The computation of input law (25) is based on a fully actuated system, which is not possible in reality. So, to transform the $[\mathbf{u}_{\mathrm{t}}(t)]_{3\times1}$ to the scalar thrust input considering the gravity as well, we use cascade design [55]:

$$T_{\mathrm{B}}(t) = m\left(\left[\mathbf{R}_3(\mathbf{q}(t))\right]_1 u_{\mathrm{t},1}(t) + \left[\mathbf{R}_3(\mathbf{q}(t))\right]_2 u_{\mathrm{t},2}(t) + \left[\mathbf{R}_3(\mathbf{q}(t))\right]_3 \left(u_{\mathrm{t},3}(t) + g\right)\right),$$

where that is $\left[\mathbf{R}_3(\mathbf{q}(t))\right]_1$ is the first component of $\mathbf{R}_3(\mathbf{q}(t))$. The cascade design delivers the necessary desired Euler angles

$$\theta_{\mathrm{des}}(t) = \tan^{-1}\left(\frac{u_{\mathrm{t},1}\cos\psi_{\mathrm{des}} + u_{\mathrm{t},2}\sin\psi_{\mathrm{des}}}{u_{\mathrm{t},3} + g}\right), \tag{26}$$

$$\phi_{\mathrm{des}}(t) = \sin^{-1}\left(\frac{u_{\mathrm{t},1}\sin\psi_{\mathrm{des}} - u_{\mathrm{t},2}\cos\psi_{\mathrm{des}}}{\sqrt{u_{\mathrm{t},1}^2 + u_{\mathrm{t},2}^2 + \left(u_{\mathrm{t},3} + g\right)^2}}\right), \tag{27}$$

in which ψ_{des} could possess an arbitrary value.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

The orientation control law is also presented as

$$\mathbf{u}_{\mathrm{o}}(t) = -\mathbf{R}_{\mathrm{o}}^{-1}\left(\mathbf{x}(t)\right)\mathbf{B}_{\mathrm{o}}^{T}\mathbf{K}_{\mathrm{o}}\left(\mathbf{x}(t)\right)\begin{bmatrix}\mathcal{F}\mathbf{e}_{\mathbf{q}}(t)\\ \boldsymbol{v}_2(t)-\boldsymbol{v}_{2,\mathrm{des}}\end{bmatrix}, \tag{28}$$

where $\mathbf{e}_{\mathbf{q}}(t)$ is quaternion error and

$$\mathcal{F} = \begin{bmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.$$

The quaternion error in (28) is defined as

$$\mathbf{e}_{\mathbf{q}}(t) = -\mathbf{q}_{\mathrm{des}}(t)\otimes\mathbf{q}^{*}(t) = -\mathbf{Q}\left(\mathbf{q}_{\mathrm{des}}(t)\right)\mathbf{q}^{*}(t),$$

where $\mathbf{q}_{\mathrm{des}}(t)$ is defined by substituting (26), (27), and ψ_{des} into Eq. (5).

7. Simulations

7.1. Validation

To validate the proposed controller design and the simulator, a comparison has been done with the PD controller and a conventional sliding mode control (SMC). We consider a system based on the Euler angles in rotational dynamics, controlled by a simple PD to have it as a reference. Then the SDDRE controller is implemented on the quaternion-based dynamics to check the performance and also to validate the correctness of the implementation. The SMC has been frequently used in UAV control [56], in the context of robustness or combined with other techniques [57]. Based on that, the SMC is selected for comparison to add the more detailed result.

The mass of the system is $m = 0.468\,(\mathrm{kg})$, the drag coefficient matrix is $\mathbf{D} = \mathrm{diag}\,(0.25, 0.25, 0.25)\left(\frac{\mathrm{kg}}{\mathrm{s}}\right)$, the components of the inertia matrix are $I_{xx} = 4.856\times10^{-3}\left(\mathrm{kgm}^2\right)$, $I_{yy} = 4.856\times10^{-3}\left(\mathrm{kgm}^2\right)$, and $I_{zz} = 8.801\times10^{-3}\left(\mathrm{kgm}^2\right)$. The time of the simulation has been set $t_{\mathrm{f}} = 10\,(\mathrm{s})$, and the drone flies from zero coordinate to the desired position in Cartesian coordinate $(-2, 3, 1.5)\,(\mathrm{m})$. All of the initial conditions are set zero including position, velocity, Euler angles, and angular velocity. The initial condition of the quaternions is found by substituting $(\phi\,(0)\,,\theta\,(0)\,,\psi\,(0))$ into (5) to reach $\mathbf{q}\,(0) = [1, 0, 0, 0]^{T}$.

The PD control gains are set as $\mathbf{K}_{\mathrm{P,t}} = \mathbf{I}_{3\times3}$, $\mathbf{K}_{\mathrm{D,t}} = 2\times\mathbf{I}_{3\times3}$, $\mathbf{K}_{\mathrm{P,o}} = \mathbf{I}_{3\times3}$, and $\mathbf{K}_{\mathrm{D,o}} = 0.5\times\mathbf{I}_{3\times3}$. The SDDRE controller gains are selected as follows: $\mathbf{R}_{\mathrm{t}} = \mathbf{I}_{3\times3}$, $\mathbf{Q}_{\mathrm{t}} = \mathrm{diag}\,(0.1, 0.1, 0.1, 0.2, 0.2, 0.2)$, $\mathbf{F}_{\mathrm{t}} = 100\times\mathbf{Q}_{\mathrm{t}}$, $\mathbf{R}_{\mathrm{o}} = \mathbf{I}_{3\times3}$, $\mathbf{Q}_{\mathrm{o}} = \mathrm{diag}\,(2, 2, 2, 0, 0, 0)$, $\mathbf{F}_{\mathrm{o}} = 10\times\mathbf{Q}_{\mathrm{o}}$.

Two separate SMC controllers are considered for translation and orientation parts, consistent with Section 6. The sliding surface is $\mathbf{s}_i(\mathbf{X}) = \dot{\tilde{\mathbf{X}}}_i + \boldsymbol{\Lambda}_i\tilde{\mathbf{X}}_i$ for $i = \{\mathrm{t,o}\}$ (translation and orientation) where $\tilde{\mathbf{X}}_i = \boldsymbol{\xi}_i - \boldsymbol{\xi}_{i,\mathrm{des}}$ and $\boldsymbol{\Lambda}_i$ is a strictly positive constant matrix. The control law is also in the form of $\mathbf{u}_i = \mathbf{B}_{i,\mathrm{SMC}}^{-1}(\mathbf{x})\left(\ddot{\boldsymbol{\xi}}_{i,\mathrm{des}} - \mathbf{f}_{i,\mathrm{SMC}}(\mathbf{x}) - \mathbf{K}_{i,\mathrm{SMC}}\tanh\left(\frac{\mathbf{s}_i(\mathbf{x})}{\sigma}\right)\right)$, where $\mathbf{B}_{\mathrm{t,SMC}} = 1/m\mathbf{I}_{3\times3}$, $\mathbf{B}_{\mathrm{o,SMC}}(\mathbf{x}) = \mathbf{J}_{\mathrm{c}}^{-1}\left(\boldsymbol{\xi}_2\right)$, $\mathbf{f}_{\mathrm{t,SMC}}(\mathbf{x}) = -\frac{1}{m}\mathbf{D}\dot{\boldsymbol{\xi}}_1(t)$, $\mathbf{f}_{\mathrm{o,SMC}}(\mathbf{x}) = -\mathbf{J}^{-1}\left(\boldsymbol{\xi}_2\right)\mathbf{C}\left(\boldsymbol{\xi}_2, \dot{\boldsymbol{\xi}}_2\right)\dot{\boldsymbol{\xi}}_2$, and $\mathbf{K}_{i,\mathrm{SMC}}$ is the correction gain of the SMC. More details on conventional dynamics, such as $\mathbf{J}_{\mathrm{c}}\left(\boldsymbol{\xi}_2\right) = \mathbf{W}^{T}\left(\boldsymbol{\xi}_2\right)\mathbf{I}\mathbf{W}\left(\boldsymbol{\xi}_2\right)$, can be found in ref. [54]. To avoid chattering in SMC, $\tanh\left(\frac{\mathbf{s}_i(\mathbf{x})}{\sigma}\right)$ is used where $\sigma = 0.2$ in this simulation. The SMC control parameters are selected as $\boldsymbol{\Lambda}_{\mathrm{t}} = \mathrm{diag}\,(0.5, 0.5, 1)$, $\boldsymbol{\Lambda}_{\mathrm{o}} = 1.5\times\mathbf{I}_{3\times3}$, $\mathbf{K}_{\mathrm{t,SMC}} = \mathrm{diag}\,(5, 5, 1)$, $\mathbf{K}_{\mathrm{o,SMC}} = 2.5\times\mathbf{I}_{3\times3}$, and desired accelerations are set zero, $\ddot{\boldsymbol{\xi}}_{i,\mathrm{des}} = \mathbf{0}$.

Simulating the system, the results are found in the following. The position variables of the drone are illustrated in Fig. 2 to Fig. 4 with respect. The roll and pitch angles of the multi-copter are plotted in Fig. 5 and Fig. 6 with respect. The input norm of the quadrotor is illustrated in Fig. 7 and the configuration and trajectories of the drones are shown in Fig. 8.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

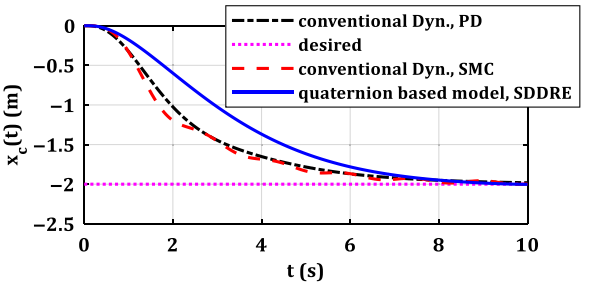

Fig. 2. *x-axis regulation of the system, comparison with PD and SMC.*

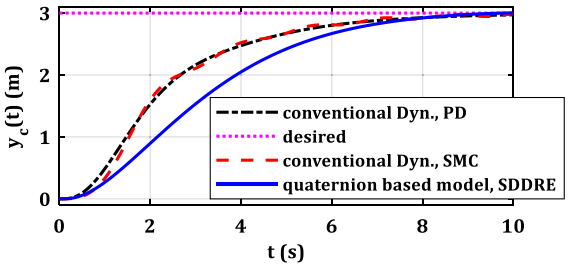

Fig. 3. *y-axis regulation of the system, comparison with PD and SMC.*

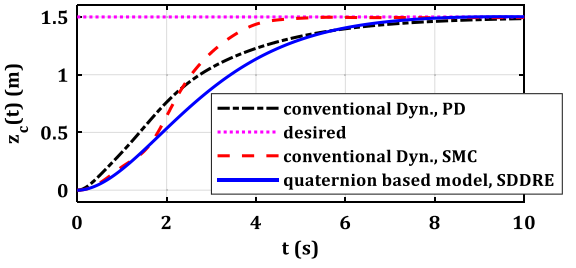

Fig. 4. *z-axis regulation of the system, comparison with PD and SMC.*

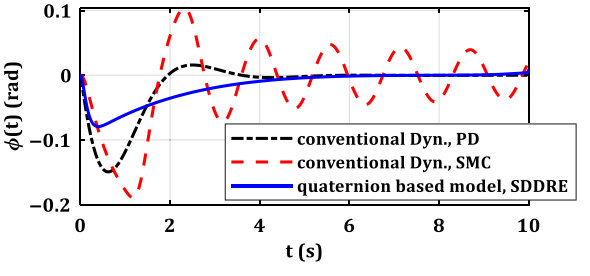

Fig. 5. *The roll angle of the system, comparison with PD and SMC.*

The error of the regulation with PD controller was gained higher than the other two and the error of the SDDRE was obtained the least, see Table I. Since the norm of the inputs (representative of the energy consumption) of the SDDRE is less than the PD and SMC, the performance of the proposed system is satisfactory. The results also confirm the validity of the quaternion-based dynamics and also the control implementation.

Validation with previous work: To confirm the correctness of the quaternion-based dynamics and the SDDRE controller, an existing model will be employed for comparison. Xiong and Zhang used a global fast terminal sliding mode controller (TSMC) for quadrotor regulation and also compared the results

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

Table I. *Comparison of PD, SMC, and SDDRE controller.*

Method	Position error (mm)	Overshoot	Convergence speed	Energy consumption: the area under the norm signals, Fig. 7
PD	35.99	Only in orientation	Second	Highest, 46.3504
SMC	12.26	In translation and orientation	First	Middle, 46.4512
SDDRE	4.69	None	Third	Lowest, 46.3102

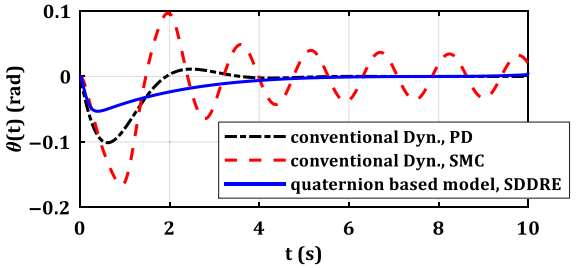

Fig. 6. *The pitch angle of the system, comparison with PD and SMC.*

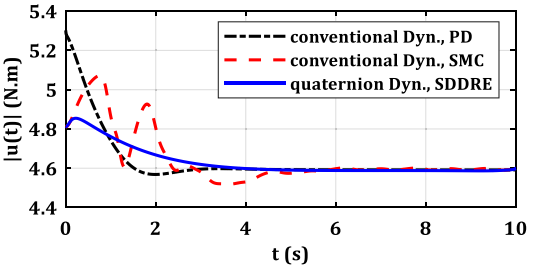

Fig. 7. *The input norm of the inputs of the system, comparison with PD and SMC.*

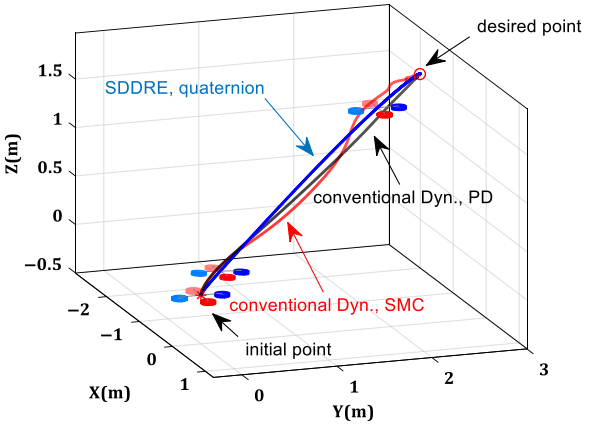

Fig. 8. *The configuration and trajectories of the quadrotor drones with PD, SMC, and SDDRE controllers.*

with conventional SMC [58]. Here the parameters of the system are substituted into the quaternion model and the SDDRE controls the model. The results are similar to the ones in Fig. 2 of ref. [58] presented in this section, in Fig. 9. The regulation of translation control is quite like the TSMC and regulated to desired values around 2s, without overshoot. The controller parameters are similar to Sections 7.2.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

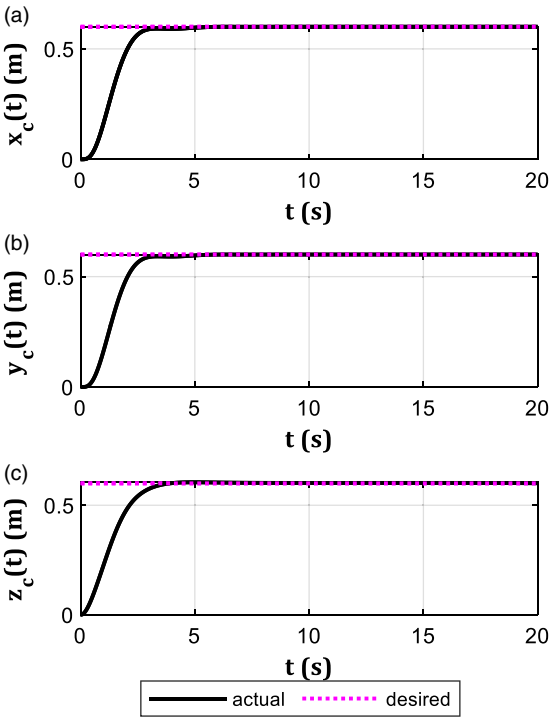

Fig. 9. *The validation results of the quaternion-based SDDRE with previous work in ref. [58].*

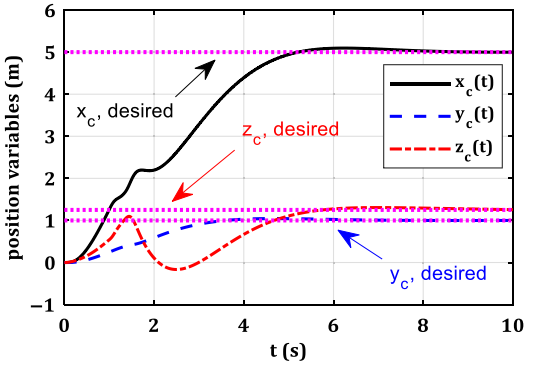

Fig. 10. *The position variables of the system in aerobatic maneuver.*

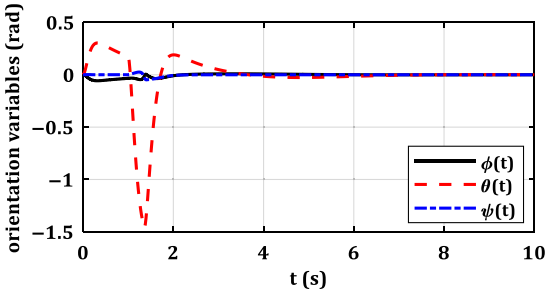

Fig. 11. *The Euler angles of the drone in aerobatic maneuver.*

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

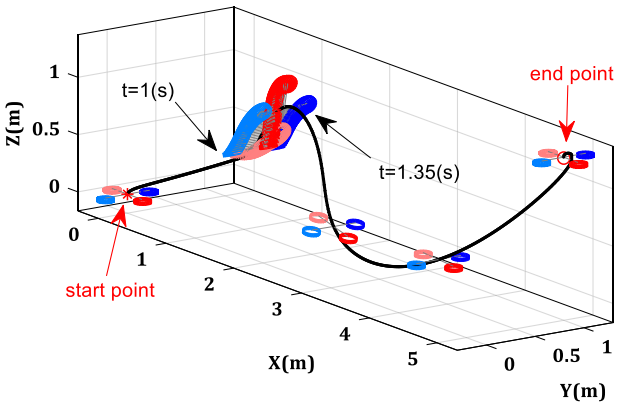

Fig. 12. *The Cobra maneuver via the SDDRE and quaternion dynamics.*

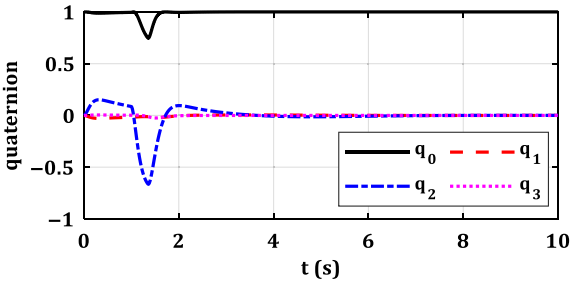

Fig. 13. *The quaternions.*

It should be noted that since the loop is closed, it cannot be said that the dynamics are validated; however, observing the behavior of the system in comparison with the SMC, the closed-loop system is validated.

7.2. Cobra maneuver

The motivation of using quaternions and avoiding Euler angles in rotational dynamics are to gain a singularity-free controller, and as a result, obtain agile flight and aerobatic maneuvers. One of the hardest positions in quadrotor control in $\pi/2$ (rad) rotation either in roll or pitch angles. The Cobra maneuver is famous for a jet aircraft to perform aerobatic shows or in combat for sudden brake, etc. For the quadrotors, it is the first time that this motion is simulated in a forward flight; the Cobra in ascending motion was reported [44]. That is a challenge since, in $\phi = \pi/2$ or $\theta = \pi/2$, there is no thrust to compensate the gravity; for an aircraft, a jet engine supports the gravity. This might cause a crash or fall for the multirotor. To perform the Cobra maneuver, $\theta_{\text{des}} = \pi/2$ is imposed in $t \in [1, 1.35]$ (s) instead of Eq. (26). After that, the multirotor drone tries to recover the stability and regulate to final condition. The simulation time is $t_{\text{f}} = 10$ (s) , and the parameters of the control are $\mathbf{R_t} = \mathbf{I}_{3\times3}$, $\mathbf{Q_t} = \text{diag}\,(1, 1, 1, 0.5, 0.5, 0.5)$, $\mathbf{F_t} = 10 \times \mathbf{Q}$, $\mathbf{R_o} = \mathbf{I}_{3\times3}$, $\mathbf{Q_o} = \text{diag}\,(2, 2, 2, 0, 0, 0)$, $\mathbf{F_o} = 10 \times \mathbf{Q_o}$. The start point of the regulation is set at zero along with other initial conditions; the endpoint is chosen (5, 1, 1.25) (m). Simulating the drone, the error is found 7.75 (mm) and the system successfully performed the maneuver. The position variables and attitude ones are shown in Fig. 10 and Fig. 11 with respect. The trajectory and configuration of the system are demonstrated in Fig. 12. The quaternions are plotted in Fig. 13. The input thrust and input torque signals are presented in Fig. 14 and Fig. 15 with respect.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

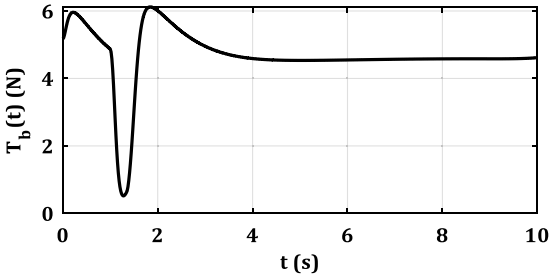

Fig. 14. *The input thrust of the quadrotor.*

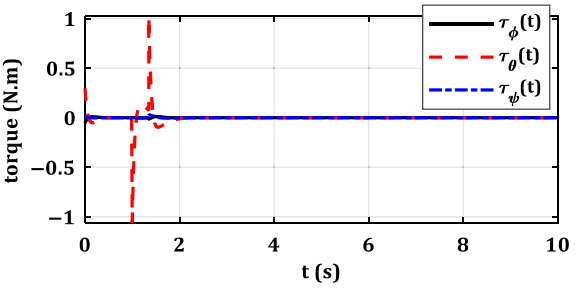

Fig. 15. *The input torque signals of the drone.*

8. Conclusions

This work investigated the quaternion-based control design using the SDDRE to control a quadrotor in aerobatic flight. The Euler angles are vulnerable to big changes in attitude and rotational dynamics, specifically, the rotation matrix. The diagonal components of $\mathbf{R}(\phi, \theta, \psi)$ become zero for either of ϕ, θ, ψ at $\pi/2$. This means the omission of thrust in flight and unstable conditions. Specifically, the controllability pair will be unsatisfied. To solve this problem, quaternion representation has been used to have a singular-free attitude control and rotation matrix. The introduced model has been validated through a comparison with the conventional Euler dynamics controlled by a PD input law. To show the application of the proposed method, a challenging maneuver, Cobra, has been simulated in the forward flight, successfully controlled. The Cobra maneuver has put the drone in a position without thrust to compensate for the gravity; however, this approach generated a stable motion to reduce the fall in that condition.

Acknowledgment. This work is supported by the HYFLIERS project (HYbrid FLying-rolling with-snakE-aRm robot for contact inspection) funded by the European Commission H2020 Programme under grant agreement ID: 779411 (https://cordis.europa.eu/project/rcn/213049). The work has also been partially funded by the European Commission H2020 Programme under AERIAL-CORE project 871479.

Author Contributions. SRN: writing initial draft, review and editing, conceptualization, methodology, validation, and investigation. JAA: review and editing, supervision, and investigation. AO: project administration, funding acquisition, review and editing, and investigation.

Supplementary Material. To view supplementary material for this article, please visit https://doi.org/10.1017/S0263574722000091.

References

[1] P. Bibik, J. Narkiewicz, M. Zasuwa and M. Żugaj, "Quadrotor Dynamics and Control for Precise Handling," **In:** *Innovative Simulation Systems, Studies in Systems, Decision and Control* (A. Nawrat and K. Jędrasiak, eds.) (Springer International Publishing, 2016) pp. 335–351. ISBN: 978-3-319-21118-3.

[2] T. Fernando, J. Chandiramani, T. Lee and H. Gutierrez, "Robust Adaptive Geometric Tracking Controls on SO (3) with an Application to the Attitude Dynamics of a Quadrotor UAV," *50th IEEE Conference on Decision and Control and European Control Conference,* Orlando, FL, USA (2011) pp. 7380–7385.
[3] A. B. Krishna, A. Sen and M. Kothari, "Robust Geometric Control of a Helicopter Using Sliding Mode Control," *International Conference on Unmanned Aircraft Systems*, Athens, Greece (2020) pp. 737–743.
[4] W. S. Craig, D. W. Yeo and D. A. Paley, "Geometric Control of a Quadrotor in Wind with Flow Sensing and Thrust Constraints: Attitude and Position Control," *AIAA Scitech 2019 Forum*, San Diego, CA (2019) p. 1192.
[5] S. R. Nekoo, J. Á. Acosta and A. Ollero, "Geometric control using the state-dependent Riccati equation: application to aerial-acrobatic maneuvers," *Int. J. Control,* 1–13 (2021). https://doi.org/10.1080/00207179.2021.1881165
[6] J. C. Monforte, *Geometric, Control and Numerical Aspects of Nonholonomic Systems* (Springer-Verlag, Berlin Heidelberg, 2004).
[7] T. Lee, M. Leok and N. H. McClamroch, *Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*, vol. **13** (Springer, Cham, 2017) p. 31.
[8] Z. Terze, D. Zlatar and V. Pandža, "Aircraft attitude reconstruction via novel quaternion-integration procedure," *Aerospace Sci. Technol.* **97**, 105617 (2020).
[9] Z. Zhou, Q. Zhang, Q. Liu, Q. Zeng and X. Tian, "Adaptive quaternion particle filter using generalized likelihood ratio test for aircraft attitude estimation in the presence of anomalous measurement," *Meas. Sci. Technol.* **32**(4), 045004 (2021).
[10] L. F. C. Figueredo, B. V. Adorno, J. Y. Ishihara and G. A. Borges, "Robust Kinematic Control of Manipulator Robots Using Dual Quaternion Representation," *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany (2013) pp. 1949–1955.
[11] H. J. Savino, L. C. A. Pimenta, J. A. Shah and B. V. Adorno, "Pose consensus based on dual quaternion algebra with application to decentralized formation control of mobile manipulators," *J. Franklin Inst.* **357**(1), 142–178 (2020).
[12] G. Antonelli, S. Chiaverini, N. Sarkar and M. West, "Adaptive Control of an Autonomous Underwater Vehicle: Experimental Results on ODIN," *IEEE Trans. Control Syst. Technol.* **9**(5), 756–765 (2001).
[13] X. Liu, M. Zhang, J. Chen and B. Yin, "Trajectory tracking with quaternion-based attitude representation for autonomous underwater vehicle based on terminal sliding mode control," *Appl. Ocean Res.* **104**, 102342 (2020).
[14] J. Rodriguez, H. Castañeda and J. L. Gordillo, "Lagrange modeling and navigation based on quaternion for controlling a micro AUV under perturbations," *Rob. Auton. Syst.* **124**, 103408 (2020).
[15] P. D. de Cerqueira Gava, V. A. M. Jorge, C. L. N. Júnior and G. J. Adabo, "AUV Cruising Auto Pilot for a Long Straight Confined Underwater Tunnel," *2020 IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada (2020) pp. 1–8.
[16] S. Suzuki, D. Nakazawa, K. Nonami and M. Tawara, "Attitude control of small electric helicopter by using quaternion feedback," *J. Syst. Des. Dyn.* **5**(2), 231–247 (2011).
[17] J. M. Cooke, M. J. Zyda, D. R. Pratt and R. B. McGhee, "NPSNET: Flight simulation dynamic modeling using quaternions," *Presence Teleoperators Virtual Environ.* **1**(4), 404–420 (1992).
[18] M. Islam and M. Okasha, "A Comparative Study of PD, LQR and MPC on Quadrotor Using Quaternion Approach," *2019 7th International Conference on Mechatronics Engineering (ICOM)*, Putrajaya, Malaysia (2019) pp. 1–6.
[19] M. Islam, M. Okasha and E. Sulaeman, "A model predictive control (MPC) approach on unit quaternion orientation based quadrotor for trajectory tracking," *Int. J. Control Autom. Syst.* **17**(19), 2819–2832 (2019).
[20] J.-W. Kang, N. Sadegh and C. Urschel, "Quaternion Based Nonlinear Trajectory Control of Quadrotors with Guaranteed Stability," *American Control Conference*, Denver, CO, USA (2020) pp. 3834–3839.
[21] F. Oliva-Palomo, A. Sanchez-Orta, H. Alazki, P. Castillo and A.-J. Muñoz-Vázquez, "Robust global observer position-yaw control based on ellipsoid method for quadrotors," *Mech. Syst. Signal Process.* **158**, 107721 (2021).
[22] E. Najafi, G. A. D. Lopes and R. Babuška, "Balancing a legged robot using state-dependent Riccati equation control," *IFAC Proc.* **47**(3), 2177–2182 (2014).
[23] H. T. Nguyen, N. T. Nguyen, I. Prodan and F. L. Pereira, "Trajectory tracking for a multicopter under a quaternion representation," *IFAC-PapersOnLine* **53**(2), 5731–5736 (2020).
[24] A. Chovancová, T. Fico, P. Hubinský and F. Duchoň, "Comparison of various quaternion-based control methods applied to quadrotor with disturbance observer and position estimator," *Rob. Auton. Syst.* **79**, 87–98 (2016).
[25] E. Fresk and G. Nikolakopoulos, "Full Quaternion Based Attitude Control for a Quadrotor," *European Control Conference*, Zürich, Switzerland (2013) pp. 3864–3869.
[26] C. Zha, X. Ding, Y. Yu, and X. Wang, "Quaternion-based nonlinear trajectory tracking control of a quadrotor unmanned aerial vehicle," *Chin. J. Mech. Eng.* **30**(1), 77–92 (2017).
[27] L. F. Sanchez, H. Abaunza and P. Castillo, "User–robot interaction for safe navigation of a quadrotor," *Robotica* **38**(12), 2189–2203 (2020).
[28] J. H. Gillula, H. Huang, M. P. Vitus and C. J. Tomlin, "Design of Guaranteed Safe Maneuvers Using Reachable Sets: Autonomous Quadrotor Aerobatics in Theory and Practice," 2010 *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA (2010) pp. 1649–1654.
[29] M. Cutler and J. How, "Actuator constrained Trajectory Generation and Control for Variable-Pitch Quadrotors," *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, MN (2012) p. 4777.
[30] J. D. Pearson, "Approximation methods in optimal control I. Sub-optimal control," *Int. J. Electron.* **13**(5), 453–469 (1962).
[31] D. K. Parrish and D. B. Ridgely, "Attitude Control of a Satellite Using the SDRE Method," *Proceedings of the American Control Conference*, Albuquerque, New Mexico (1997) pp. 942–946.

https://doi.org/10.1017/S0263574722000091 Published online by Cambridge University Press

[32] A. G. Romero, L. C. G. Souza and R. A. Chagas, "Application of the SDRE Technique in the Satellite Attitude and Orbit Control System with Nonlinear Dynamics," *2018 SpaceOps Conference*, Marseille, France (2018) p. 2536.
[33] D. T. Stansbery and J. R. Cloutier, "Position and Attitude Control of a Spacecraft Using The State-Dependent Riccati Equation Technique," *Proceedings of the American Control Conference*, Chicago, Illinois (2000) pp. 1867–1871.
[34] W. Luo and Y.-C. Chu, "Attitude Control Using the SDRE Technique," *7th International Conference on Control, Automation, Robotics and Vision*, Singapore (2002) pp. 1281–1286.
[35] C. Pukdeboon and A. S. Zinober, "Optimal sliding mode controllers for spacecraft attitude manoeuvres," *IFAC Proc. Vol.* **42**(6), 173–178 (2009).
[36] C. Pukdeboon, "Robust optimal output feedback sliding mode control for spacecraft attitude tracking maneuvers," *Int. J. Pure Appl. Math.* **79**(1), 11–27 (2012).
[37] P. Li, X. Yue, X. Chi and C. Du, "Optimal Relative Attitude Tracking Control for Spacecraft Proximity Operation," *25th Chinese Control and Decision Conference*, Taiyuan, China (2013) pp. 4582–4587.
[38] M. Safi, M. Mortazavi and S. M. Dibaji, "Global stabilization of attitude dynamics: SDRE-based control designs," *AUT J. Model. Simul.* **50**(2), 203–210 (2018).
[39] A. G. Romero and L. C. G. de Souza, "State-dependent Riccati equation controller using Java in remote sensing CubeSats," *J. Appl. Remote Sens.* **13**(3), 032509 (2019).
[40] T. Cimen, "Survey of state-dependent Riccati equation in nonlinear optimal feedback control synthesis," *J. Guidance Control Dyn.* **35**(4), 1025–1047 (2012).
[41] N. Wang, R. Ma, X. Chang and L. Zhang, "Numerical virtual flight simulation of quasi-cobra maneuver of a fighter aircraft," *J. Aircraft* **58**(1), 138–152 (2021).
[42] B. Gal-Or and D. D. Baumann, "Mathematical phenomenology for thrust-vectoring-induced agility comparisons," *J. Aircraft.* **30**(2), 248–255 (1993).
[43] W. Xu and F. Zhang, "Learning Pugachev's Cobra maneuver for tail-sitter UAVs using acceleration model," *IEEE Rob. Autom. Lett.* **5**(2), 3452–3459 (2020).
[44] Y. Chen and N. O. Pérez-Arancibia, "Adaptive Control of Aerobatic Quadrotor Maneuvers in the Presence of Propeller-Aerodynamic-Coefficient and Torque-Latency Time-Variations," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, Canada (2019) pp. 6447–6453.
[45] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix* **58**(15), 1–35 (2006).
[46] S. R. Nekoo, "Tutorial and review on the state-dependent Riccati equation," *J. Appl. Nonlinear Dyn.* **8**(2), 109–166 (2019).
[47] S. R. Nekoo, "A PDE breach to the SDRE," *Asian J. Control* **22**(2), 667–676 (2020).
[48] S. R. Nekoo, "Digital implementation of a continuous-time nonlinear optimal controller: An experimental study with real-time computations," *ISA Trans.* **101**, 346–357 (2020).
[49] A. Jayaram and M. Tadi, "Synchronization of chaotic systems based on SDRE method," *Chaos Solitons Fractals* **28**(3), 707–715 (2006).
[50] M. H. Korayem and S. R. Nekoo, "Finite-time state-dependent Riccati equation for time-varying nonaffine systems: Rigid and flexible joint manipulator control," *ISA Trans.* **54**, 125–144 (2015).
[51] S. Mathavaraj and R. Padhi, "Finite-Time LQR and SDRE for Satellite Formation Flying," **In:** *Satellite Formation Flying* (Springer, Singapore, 2021) pp. 99–109. ISBN: 978-981-15-9631-5.
[52] A. Heydari and S. N. Balakrishnan, "Approximate Closed-Form Solutions to Finite-Horizon Optimal Control of Nonlinear Systems," *American Control Conference*, Fairmont Queen Elizabeth, Montréal, Canada (2012) pp. 2657–2662.
[53] M. H. Korayem and S. R. Nekoo, "Nonlinear Optimal Control via Finite Time Horizon State-Dependent Riccati Equation," *Second RSI/ISM International Conference on Robotics and Mechatronics*, Tehran, Iran (2014) pp. 878–883.
[54] S. R. Nekoo, J. Á. Acosta, A. E. Gomez-Tamm and A. Ollero, "Optimized Thrust Allocation of Variable-Pitch Propellers Quadrotor Control: A Comparative Study on Flip Maneuver," *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, Cranfield, UK (2019) pp. 86–95.
[55] S. R. Nekoo, J. Á. Acosta and A. Ollero, "Fully Coupled Six-DoF Nonlinear Suboptimal Control of a Quadrotor: Application to Variable-Pitch Rotor Design," *Iberian Robotics Conference*, Porto, Portugal (2019) pp. 72–83.
[56] R. Xu and U. Ozguner, "Sliding Mode Control of a Quadrotor Helicopter," *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA (2006) pp. 4957–4962.
[57] G. E. M. Abro, S. A. Zulkifli, V. S. Asirvadam and Z. A. Ali, "Model-Free-based Single-Dimension Fuzzy SMC Design for Underactuated Quadrotor UAV," **In:** *Actuators* (H. Wang, ed.) (MDPI, Switzerland, 2021) p. 191.
[58] J.-J. Xiong and G.-B. Zhang, "Global fast dynamic terminal sliding mode control for a quadrotor UAV," *ISA Trans.* **66**, 233–240 (2017).

Cite this article: S. R. Nekoo, J. Á. Acosta and A. Ollero (2022). "Quaternion-based state-dependent differential Riccati equation for quadrotor drones: Regulation control problem in aerobatic flight", Robotica **40**, 3120–3135. https://doi.org/10.1017/S0263574722000091