CEAS-GNC-2024-083

EuroGNC Bristol, UK June 11th-13th 2024

Rank one update-based efficient adaptive control allocation for multicopter

Hangxu Li Research Associate, Technical University of Munich, Institute of Flight System Dynamics, 85748, Garching, Germany. hangxu.li@tum.de

Stephan Myschik Professor, Universität der Bundeswehr München, Institute for Aeronautical Engineering, 85577, Neubiberg, Germany. stephan.myschik@unibw.de

Haichao Hong Associate Professor, Shanghai Jiao Tong University, School of Aeronautics and Astronautics, 200240, Shanghai, China. haichao.hong@sjtu.edu.cn

Florian Holzapfel Professor, Technical University of Munich, Institute of Flight System Dynamics, 85748, Garching, Germany. florian.holzapfel@tum.de

ABSTRACT

This paper proposes an efficient rank one update-based adaptive control allocation (CA) that shows satisfactory computational efficiency. A typical adaptive CA module includes parameter estimation and CA parts. In this paper, the parameter estimation part uses a recursive least square (RLS) filter to obtain an estimated CA matrix, and the CA module is designed based on the pseudo-inverse method. Because of using the pseudo-inverse method to allocate pseudo commands from the upstream controller, the singular value decomposition (SVD) of the CA matrix is required in every time step. Completely decomposing the matrix by using SVD methods will waste too much computational resources. Noticing that the estimated matrix is only perturbed by a rank one matrix in the parameter update step of RLS, a rank one update method is integrated into the RLS filter in this paper for updating SVD results of the estimated matrix without using SVD methods. Benefited from the low computational complexity of the rank one update method, the computational efficiency of the whole adaptive CA part is therefore enhanced. In the meantime, the control performance will not be deteriorated by using the rank one update method. The proposed rank one update-based adaptive CA is validated on a complex high-fidelity multicopter model and shows satisfactory computation and control performance.

Keywords: Adaptive control allocation; Flight control; Rank one update; Multicopter controller design; Computational efficiency

Nomenclature

G	=	center of gravity
R	=	reference point
B	=	body-fixed frame
C	=	control frame
O	=	north-east-down frame

E	=	earth-centered-earth-fixed frame
$\boldsymbol{\omega}$	=	angular rate
p, q, r	=	roll, pitch, and yaw rate
$\boldsymbol{I}$	=	moment tensor
$\boldsymbol{M}_p$	=	moment generated by rotors
$\boldsymbol{\Phi}$	=	Euler angle
Φ, Θ, Ψ	=	roll, pitch, and yaw angle
$\boldsymbol{V}$	=	velocity
$\boldsymbol{F}$	=	force
$\boldsymbol{\Omega}$	=	rotor speed
ω_n	=	natural frequency
ζ	=	damping ratio
$\boldsymbol{f}$	=	specific force
$\boldsymbol{g}$	=	gravitational acceleration
$\boldsymbol{M}$	=	rotation matrix
T	=	thrust
$\boldsymbol{\nu}$	=	pseudo inputs
$\boldsymbol{B}$	=	control effectiveness matrix
$\boldsymbol{B}_{ca}$	=	control allocation matrix
$\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}$	=	singular value decomposition
$\boldsymbol{\Theta}$	=	parameters to be estimated
J	=	cost function
$\boldsymbol{D}$	=	diagonal matrix

1 Introduction

Multicopter is becoming popular in many application scenarios because of its many advantages, like vertical take-off and landing capability, etc. For example, in Refs. [1], it is used to help save wounded soldiers in combat environments. Depending on different applications, multicopter controller design has different requirements for tracking performance, gust alleviation capability, etc. In order to meet these challenging requirements, researchers developed and implemented many advanced and powerful techniques to augment baseline controllers, for example, dynamic inversion, gain scheduling, parameter estimators, etc. Among these techniques, parameter estimators are often used to construct an adaptive controller that is able to counter uncertainties [2–4].

An adaptive controller is a kind of controller with adjustable parameters and a mechanism for adjusting the parameters [5], which is designed based on feedback of signals in a controlled system for control adaptation to effectively handle system uncertainties [6, 7]. Along the development process, different design methods for adaptive control come out. Model reference adaptive control uses the Lyapunov function to derive adaptation law, which is then able to guarantee the stability of closed-loop control systems [8–10]. The artificial neural network is also commonly used in designing adaptive controllers [11–13]. Another way to design an adaptive controller is to use different kinds of adaptive filters, like recursive least square (RLS) filter, Kalman filter, and so on, for providing baseline controller required information. In Refs. [14–16], these filters are used to obtain estimated actuator effectiveness, which is then exploited in baseline controller. If only parameters used in control allocation (CA) are estimated, it can be called adaptive CA scheme [17]. This paper also employs such a scheme because of its satisfactory performance and easy implementation. The RLS filter is chosen in the paper as the CA matrix estimator because of its fast convergence and low computational complexity [18].

In adaptive CA, while parameter estimation does not waste too much computation resources, allocating pseudo commands usually contributes a lot to the workload of flight computer. There are mainly two kinds of CA methods. One way to solve CA problem is by using optimization-based methods, like the active-set method. When using these methods, the number of iterations required to reach termination conditions is not determined and guaranteed [19]. In order to avoid such cases, another kind of CA method, pseudo-inverse method, is chosen in this paper to design the CA module. In Refs. [20, 21], redistributed scaling pseudo inverse-based CA together with null space transition technique is used and therefore needs singular value decomposition (SVD) of CA matrix. Because of that, the CA part mainly contributes to the computational load as in Ref. [20]. This problem is then investigated in this paper, aiming to reduce calculation of designed adaptive CA.

The main contribution of this paper is a rank one update-based adaptive CA. The proposed adaptive CA is used with a designed incremental model-based nonlinear dynamic inversion (INDI) baseline controller for controlling the velocities of a multicopter. The adaptive CA includes mainly a parameter estimation part and a CA part. Data from sensors and the actuator model are synchronized and then filtered by a third-order filter, forming inputs for the RLS filter inputs, i.e., pseudo inputs and rotor speeds. When estimating parameters, a common RLS filter can only compute CA matrix provided aforementioned inputs. Then, decomposing the estimated CA matrix by using SVD methods is required before implementing CA, which costs much calculation. In the traditional RLS, it is noticed that the estimated CA matrix is updated every time step with a rank one perturbation, which is suitable for implementing rank one update method. Therefore, this paper addresses the issue above by utilizing the rank one update method, which offers a computationally much more efficient way to output the SVD of the CA matrix, directly usable for calculating pseudo inverse. This route also eliminates the need for an additional SVD solver. The SVD results can be employed for null space-related techniques like null-space transition and excitation [4, 21], without compromising on control performance thanks to the high numerical precision of the rank one update. CA part uses the pseudo-inverse method for its stability and efficiency in calculation. Overall, the proposed rank one update-based adaptive CA approach reduces computational complexity while maintaining high precision and performance.

2 Plant Model

A multicopter model is used in this paper for testing designed rank one update-based adaptive controller, as in Fig. 1. The over-actuated characteristic of this multicopter enables it to serve as a valuable testbed for validating CA related research.

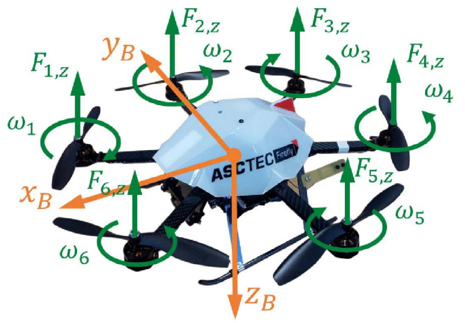

Fig. 1 Multicopter

2.1 Overview and coordinate system definition of the multicopter used as testbed

The structure of the over-actuated multicopter used in this paper is depicted in Fig. 1. A Flight computer and different types of sensors are installed in the middle of the airframe. The center of gravity (G) coincides with the reference point (R) which locates at the center of the airframe. Six arms of the airframe are designed with intervals of $60\,deg$ and have the same length of $0.215\,m$. Rotors with identical aerodynamics are mounted at the end of these arms with a height of $0.025\,m$. Rotors 1, 3, and 5 rotate clockwise, while rotors 2, 4, and 6 rotate counterclockwise. Except for those geometric parameters, the gross mass of multicopter is approximately $1.2\,kg$, and the moments of inertia along x-, y-, and z-axis are approximately 10^{-2}, 10^{-2}, and $1.5 \times 10^{-2}\ kg \cdot m^2$ respectively.

Several coordinate systems are defined and used in this paper for developing controllers. A body-fixed frame (B) is defined as in Fig. 1, whose x-axis points in the direction of the angle bisector between the first and sixth arms. The y-axis of this frame is aligned with the second arm and the z-axis is perpendicular to the xy-plane. Besides, a control frame (C) is obtained through a rotation of the north-east-down (O) frame around its z_O axis with the yaw angle Ψ [20]. Origins of B-, C-, and O-frame are all located at the reference point. In this paper, an earth-centered-earth-fixed frame (E) is used as an inertial frame.

2.2 Dynamics and kinematics of the multicopter

The dynamics of the aforementioned multicopter can be described by several differential equations. In following equations, notation scheme like $\left.\left(\boxed{1}_{\boxed{2}}^{\boxed{3}}\right)^{\boxed{4}}_{\boxed{5}}\right|_{\boxed{6}}$ is used, where block 1 is the physical quantity, block 2 is index, block 3 denotes the reference, block 4 means the reference frame, block 5 is the notation frame, and block 6 is the type of this physical quantity. For example, $\left.\left(\boldsymbol{V}_K^R\right)_C^E\right|_{meas}$ means **meas**ured translational **K**inematic velocity at **R**eference point, differentiated with respect to **E**arth-centered-earth-fixed frame, and denoted in **C**ontrol frame. The rotational equation of motion can be expressed as:

$$\left(\dot{\omega}^{EB}\right)_B^B = \boldsymbol{I}_{BB}^{-1} \cdot \left[\boldsymbol{M}_P - \left(\omega^{EB}\right)_B \times \left(\boldsymbol{I}_{BB} \cdot \left(\omega^{EB}\right)_B\right)\right] \tag{1}$$

where $\left(\omega^{EB}\right)_B = [\ p \quad q \quad r\ ]^T$ is body angular rate written in B-frame, $\boldsymbol{I}_{BB}$ specifies moment tensor in B-frame, and $\boldsymbol{M}_P$ denotes moment generated by rotors. A strapdown equation can be used to obtain multicopter attitude represented by Euler angle, as below:

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{G}\left(\boldsymbol{\Phi}\right) \cdot \left(\omega^{EB}\right)_B \tag{2}$$

where $\boldsymbol{\Phi} = [\ \Phi \quad \Theta \quad \Psi\ ]^T$ is Euler angle, and $\boldsymbol{G}(\boldsymbol{\Phi}) = \begin{bmatrix} 1 & \sin\Phi \cdot \tan\Theta & \cos\Phi \cdot \tan\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi/\cos\Theta & \cos\Phi/\cos\Theta \end{bmatrix}$. Velocities in C-frame can be calculated via the translational equations of motion:

$$\left(\dot{\boldsymbol{V}}_K^R\right)_C^{EC} = \frac{1}{m} \cdot \left[\left(\boldsymbol{F}_P^R\right)_C + \left(\boldsymbol{F}_A^R\right)_C + \left(\boldsymbol{F}_G^R\right)_C\right] - \left(\boldsymbol{\omega}^{EC}\right)_C \times \left(\boldsymbol{V}_K^R\right)_C^E \tag{3}$$

where $\left(\boldsymbol{V}_K^R\right)_C^E = [\ u_C \quad v_C \quad w_C\ ]^T$ means kinematic speed in C-frame, m is mass of multicopter, $\boldsymbol{F}_P$ is the force generated by rotors and is used to control three-axis velocities by changing its magnitude and direction, $\boldsymbol{F}_A$ is aerodynamic drag force, $\boldsymbol{F}_G$ stands for gravity, and $\left(\boldsymbol{\omega}^{EC}\right)_C = [\ 0 \quad 0 \quad \dot{\Psi}\ ]^T$ is angular rate between inertial frame and C-frame. In this paper, rotor dynamics are simplified into first-order transfer functions when designing controllers in this paper:

$$\dot{\Omega}_i = K_{act,i} \cdot \left(\Omega_{i,cmd} - \Omega_i\right) \tag{4}$$

where Ω_i stands for the rotational speed of ith rotor, and $K_{act,i}$ is the bandwidth of ith rotor. The minimum and maximum rotational speeds of each rotor are $1,260\,rpm$ and $10,306\,rpm$ respectively.

3 Nominal controller design with adaptive CA

A nominal controller with an adaptive CA part is developed in this section for controlling three-axis velocities in C-frame and the time derivative of the yaw angle with an estimated CA matrix, as in Fig. 2. Details about this controller can be found in Ref. [4]. The controller is mainly composed of a control law part which is designed with the INDI method, a pseudo inverse-based CA part, and an estimation part which is used to provide a CA matrix. In the estimation part, a common RLS filter is used to estimate the CA matrix. And then this matrix is later decomposed by the SVD method for further use in a pseudo inverse-based CA module. Due to the utilization of SVD, the estimation module significantly contributes to the computational resource consumption of the adaptive controller. Besides, there are also some auxiliary blocks used to assist the controller with getting required signals, like the signal processing block. Details about these blocks can be found in Ref. [4].

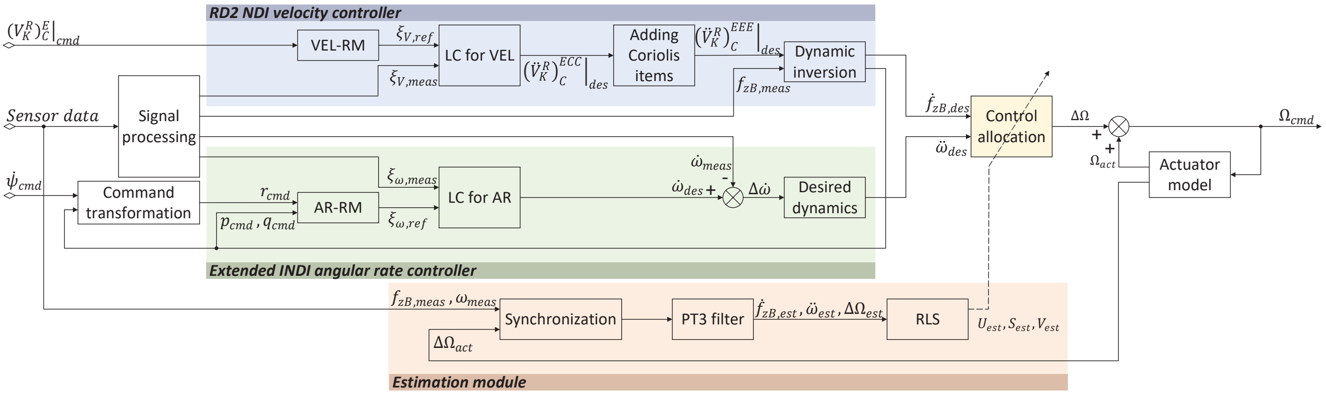

Fig. 2 Adaptive controller structure (VEL, velocity; AR, angular rate; RM, reference model; LC, linear controller; RLS, recursive least square)

3.1 NDI method-based control law

The designed NDI-based controller in this paper is composed of two parts, an outer loop velocity controller and an inner loop angular rate controller, as in Fig. 2.

A noncascaded NDI controller with a relative degree of 2 is developed for tracking velocity commands. It is mainly composed of a reference model, a linear controller, and a dynamic inversion part. A second-order dynamic equation is used as the reference model, whose natural frequency is 2 rad/s and damping ratio is 1. The reference states are used in the linear control law design:

$$\begin{aligned}\left(\ddot{\boldsymbol{V}}_K^R\right)_C^{ECC}\Big|_{des} =& \boldsymbol{K}_{F,V}\cdot\left(\ddot{\boldsymbol{V}}_K^R\right)_C^{ECC}\Big|_{ref} + \boldsymbol{K}_{P,V}\cdot\left(\left(\boldsymbol{V}_K^R\right)_C^{E}\Big|_{ref} - \left(\boldsymbol{V}_K^R\right)_C^{E}\Big|_{est}\right)\\ &+ \boldsymbol{K}_{D,V}\cdot\left(\left(\dot{\boldsymbol{V}}_K^R\right)_C^{EC}\Big|_{ref} - \left(\dot{\boldsymbol{V}}_K^R\right)_C^{EC}\Big|_{est}\right)\end{aligned} \tag{5}$$

where subscript est stands for estimated values. Desired jerk commands with respect to the C-frame are transferred into absolute jerk commands with respect to the inertial frame:

$$\left(\ddot{\boldsymbol{V}}_K^R\right)_C^{EEE}\Big|_{des} = \left(\ddot{\boldsymbol{V}}_K^R\right)_C^{ECC}\Big|_{des} + \left(\dot{\boldsymbol{\omega}}^{EC}\right)_C^{C}\times\left(\boldsymbol{V}_K^R\right)_C^{E} + \left(\boldsymbol{\omega}^{EC}\right)_C\times\left(\dot{\boldsymbol{V}}_K^R\right)_C^{EC} + \left(\boldsymbol{\omega}^{EC}\right)_C\times\left(\boldsymbol{f}_C+\boldsymbol{g}_C\right) \tag{6}$$

where $\boldsymbol{f}$ stands for specific force, $\boldsymbol{g}$ means gravitational acceleration. A dynamic inversion is used to get inner loop angular rate commands from the jerk commands:

$$\begin{bmatrix} p \\ q \\ \dot{f}_{zB}\end{bmatrix}_{des} = \begin{bmatrix} 0 & -1/f_{zB,meas} & 0 \\ 1/f_{zB,meas} & 0 & 0 \\ 0 & 0 & 1\end{bmatrix}\cdot\boldsymbol{M}_{BC}\cdot\left(\ddot{\boldsymbol{V}}_K^R\right)_C^{EEE}\Big|_{des} \tag{7}$$

where $\boldsymbol{M}_{BC}$ is rotation matrix from C-frame to B-frame, and $f_{zB,meas}$ denotes measured specific force along z-axis of B-frame.

In order to track angular rate commands, the extended INDI method considering actuator dynamics is used to design inner loop controller [22]. Similar to velocity controller, it is consisted of a reference model, a linear controller, and a dynamics inversion function. The reference model used in the angular rate controller is of the first order and has bandwidths of $10 rad/s$, $10 rad/s$, and $5 rad/s$ for roll, pitch, and yaw channels respectively. With referenced states and measured states, the linear controller is implemented to generate desired angular acceleration, as follows:

$$\left(\dot{\boldsymbol{\omega}}^{EB}\right)_B^{B}\Big|_{des} = \boldsymbol{K}_{F,\omega}\cdot\left(\dot{\boldsymbol{\omega}}^{EB}\right)_B^{B}\Big|_{ref} + \boldsymbol{K}_{P,\omega}\cdot\left(\left(\boldsymbol{\omega}^{EB}\right)_B\Big|_{ref} - \left(\boldsymbol{\omega}^{EB}\right)_B\Big|_{est}\right) \tag{8}$$

According to Ref. [22], the desired dynamics of angular acceleration need to be designed based on the bandwidth of actuators:

$$\left(\ddot{\boldsymbol{\omega}}^{EB}\right)_B^{BB}\Big|_{des} = \boldsymbol{K}_{\dot{\omega}}\cdot\left(\left(\dot{\boldsymbol{\omega}}^{EB}\right)_B^{B}\Big|_{des} - \left(\dot{\boldsymbol{\omega}}^{EB}\right)_B^{B}\Big|_{est}\right) \tag{9}$$

where $\boldsymbol{K}_{\dot{\omega}}$ is the bandwidth of desired dynamics and is chosen to be $diag\,(50, 50, 10)$.

Developed velocity controller and angular rate controller constitute a complete control law module and generate desired time derivative commands of pseudo control inputs which are defined as $\nu = \left[f_{zB}|_{des} \quad \left(\dot{\omega}^{EB}\right)_B^{B}\Big|_{des}\right]^T$ in this paper. These rate commands of pseudo inputs, i.e., $\dot{\boldsymbol{\nu}}_{des} = \left[\dot{f}_{zB} \quad \left(\ddot{\omega}^{EB}\right)_B^{BB}\right]^T$, will then be allocated into incremental rotor speed commands by a CA part. The incremental commands will be added to estimated rotor speeds for obtaining absolute rotor speed commands which are fed into motors, as in Fig. 2.

3.2 Pseudo inverse-based CA module

The CA problem can be established by calculating derivatives of pseudo inputs. The specific force along z-axis of B-frame can be expressed as follows:

$$f_{zB} = \frac{1}{m} \cdot \left[-T + \left(F_A^R\right)_{zB}\right] \tag{10}$$

Combining Eq. (1) with Eq. (10) and neglecting slow dynamics, the rates of pseudo inputs can be derived as in following equation:

$$\dot{\boldsymbol{\nu}} = \begin{bmatrix} \dot{f}_{zB} \\ \left(\ddot{\omega}^{EB}\right)_B^{BB} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial f_{zB}}{\partial T} \cdot \frac{\partial T}{\partial \boldsymbol{\Omega}} \\ \frac{\partial\left(\left(\dot{\omega}^{EB}\right)_B^B\right)}{\partial \boldsymbol{M}_P} \cdot \frac{\partial \boldsymbol{M}_P}{\partial \boldsymbol{\Omega}} \end{bmatrix}}_{\boldsymbol{B}} \cdot \dot{\boldsymbol{\Omega}} \tag{11}$$

where $\boldsymbol{B}$ is defined as control effectiveness matrix in this paper. The matrix stands for the capability by which rotor acceleration influences the rate of pseudo inputs. Assuming the actuator dynamics are of the first order as in Eq. (4), the expression of pseudo inputs rates in Eq. (11) can be further derived as follows:

$$\dot{\boldsymbol{\nu}} = \underbrace{\boldsymbol{B} \cdot \boldsymbol{K}_{act}}_{\boldsymbol{B}_{ca}} \cdot \Delta\boldsymbol{\Omega} \tag{12}$$

where $\boldsymbol{B}_{ca} \in \mathcal{R}^{4\times6}$ is the CA matrix in this paper, and $\Delta\boldsymbol{\Omega} = \boldsymbol{\Omega}_{cmd} - \boldsymbol{\Omega}$. Given a desired pseudo input rate command $\dot{\boldsymbol{\nu}}_{des}$, the CA problem is to find the incremental control vector $\Delta\boldsymbol{\Omega}$ satisfying Eq. (12). The equation coincides with the standard CA problem in Refs. [19, 23] and can be then solved by many different CA methods.

In this paper, pseudo inverse-based method is chosen to design CA part because of its simplicity and computational efficiency [24]. In order to use this method and also some other null space-based techniques, the pseudo inverse of the CA matrix $\boldsymbol{B}_{ca}^{+}$ is obtained by implementing SVD on $\boldsymbol{B}_{ca}$, as follows:

$$\boldsymbol{B}_{ca} = \boldsymbol{U} \cdot \underbrace{\begin{bmatrix} \boldsymbol{S} & \boldsymbol{0}_{4\times2} \end{bmatrix}}_{\boldsymbol{\Sigma}} \cdot \boldsymbol{V}^T \tag{13}$$

where $\boldsymbol{U} \in \mathcal{R}^{4\times4}$ and $\boldsymbol{V} \in \mathcal{R}^{6\times6}$ are orthogonal matrices, $\boldsymbol{S} \in \mathcal{R}^{4\times4}$ is a matrix whose diagonal elements are the singular values of $\boldsymbol{B}_{ca}$ matrix, and $\boldsymbol{\Sigma} = [\ \boldsymbol{S} \quad \boldsymbol{0}_{4\times2}\ ] \in \mathcal{R}^{4\times6}$ is a rectangular diagonal matrix. With the SVD results, the solution to the aforementioned CA problem in Eq. (12) can be expressed as follows:

$$\Delta\boldsymbol{\Omega} = \underbrace{\boldsymbol{V} \cdot \begin{bmatrix} \boldsymbol{S}^{-1} & \boldsymbol{0}_{4\times2} \end{bmatrix} \cdot \boldsymbol{U}^T}_{\boldsymbol{B}_{ca}^{+}} \cdot \dot{\boldsymbol{\nu}}_{des} \tag{14}$$

The absolute commands fed into actuators can then be obtained by adding estimated current rotor speeds on these allocated incremental commands.

3.3 Adaptive augmentation by using RLS filter

The CA matrix is obtained in this paper by estimation, which drives the developed controller into an adaptive one [4]. Assuming pseudo inputs rates $\dot{\boldsymbol{\nu}}$ in Eq. (12) is measurable and the incremental rotor speed vector $\Delta\boldsymbol{\Omega}$ can be obtained from actuator model, the standard observation model of pseudo inputs rates can be written as follows:

$$\boldsymbol{y} = \boldsymbol{\Theta} \cdot \boldsymbol{x} + \boldsymbol{v} \tag{15}$$

where $\boldsymbol{y} = \dot{\boldsymbol{v}} = \left[\dot{f}_{zB}\big|_{est} \quad \left(\ddot{\omega}^{EB}\right)_{B}^{BB}\Big|_{est} \right]^{T}$ denotes measurements, $\boldsymbol{\Theta} = \boldsymbol{B}_{ca}$ is parameters to be estimated, $\boldsymbol{x} = \Delta\boldsymbol{\Omega}$ stands for model inputs, and $\boldsymbol{v}$ is measurement noise. Based on Eq. (15), the estimation problem in this paper is formulated to find the unknown CA matrix which minimizes the cost function:

$$J_{RLS} = \sum\nolimits_{i=1}^{n} \lambda^{n-i} \cdot \left(\boldsymbol{y} - \hat{\boldsymbol{\Theta}} \cdot \boldsymbol{x}\right) \tag{16}$$

where $\hat{\boldsymbol{\Theta}}$ is estimated parameters, and λ is forgetting factor. In this paper, RLS method is used to solve the problem defined with Eqs. (15-16) and to develop major filter part of parameter estimation module.

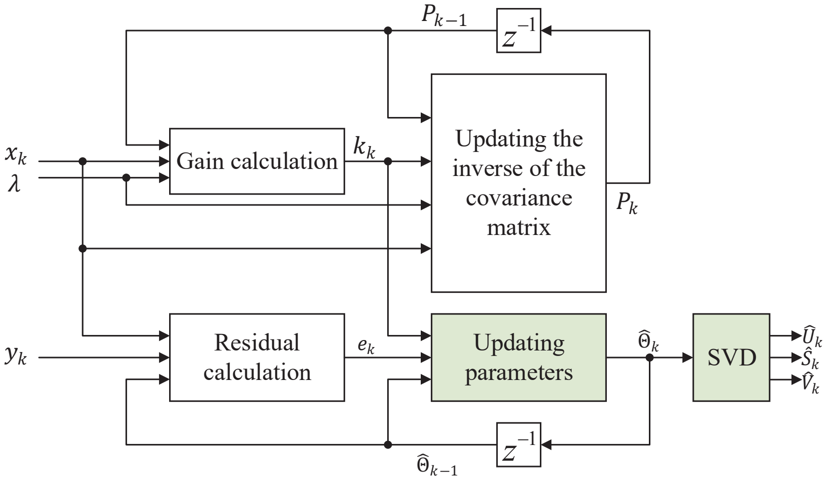

Fig. 3 Common RLS filter

As in Fig. 2, the parameter estimation module receives data, i.e., $f_{zB,meas}$, $\boldsymbol{\omega}_{meas}$, and $\Delta\boldsymbol{\Omega}_{act}$, from both the sensor and actuator model and outputs estimated SVD results of the CA matrix. Received data are firstly synchronized to exclude adverse effects caused by differences among pure time delays existed in sensors. Then a third-order filter is used to reduce noise in sensor data and estimate data fed into RLS, whose natural frequency is 25 rad/s. After obtaining required data, i.e., $\dot{f}_{zB}\big|_{est}$, $\left(\ddot{\omega}^{EB}\right)_{B}^{BB}\Big|_{est}$, and $\Delta\boldsymbol{\Omega}_{est}$, the RLS filter is able to provide the unknown CA matrix $\hat{\boldsymbol{\Theta}}$ online, as in Fig. 3. Its algorithm is as follows:

$$\boldsymbol{e}_t = \boldsymbol{y}_t^T - \boldsymbol{x}_t^T \hat{\boldsymbol{\Theta}}_{t-1}^T \tag{17}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_{t-1}\boldsymbol{x}_t \left(\lambda + \boldsymbol{x}_t^T \boldsymbol{P}_{t-1}\boldsymbol{x}_t\right)^{-1} \tag{18}$$

$$\hat{\boldsymbol{\Theta}}_t^T = \hat{\boldsymbol{\Theta}}_{t-1}^T + \boldsymbol{K}_t \boldsymbol{e}_t \tag{19}$$

$$\boldsymbol{P}_t = \frac{1}{\lambda}\left(\boldsymbol{P}_{t-1} - \boldsymbol{K}_t \boldsymbol{x}_t^T \boldsymbol{P}_{t-1}\right) \tag{20}$$

where $\boldsymbol{e}_t \in \mathcal{R}^{1\times 4}$ is a priori error, $\boldsymbol{P}_t \in \mathcal{R}^{6\times 6}$ is an approximation of the inverse correlation matrix, $\boldsymbol{K}_t \in \mathcal{R}^{6\times 1}$ is the gain matrix, and λ is the forgetting factor set to 0.999 in this paper. The initial value of $\boldsymbol{P}_t$ is chosen as $50 \times \boldsymbol{I}_{6\times 6}$. Subsequently, choosing an SVD method, the estimated matrix can be decomposed into singular values and singular vectors, which provides convenience for further calculating pseudo inverse of the matrix in CA module, as in green parts of Fig. 3.

4 Rank one update-based efficient RLS filter

The rank one update method is chosen to be integrated with common RLS and to decrease computational resources consumed by designed adaptive CA. In section 3.3, the SVD computation of the estimated CA matrix, which has a computational complexity of $O\left(n^3\right)$ for a $m \times n$ matrix [25], constitutes a significant portion of the overall CA computational resources. Noticing that the correction factor $\boldsymbol{K}_t\boldsymbol{e}_t$ in Eq. (19) has a rank of 1, the rank one update method can be implemented to iteratively update the SVD results of the estimated CA matrix, which has a computational complexity of $O\left(n^2\right)$ for a $m \times n$ matrix with rank one perturbation [26, 27]. Compared to decomposing the matrix by SVD methods, updating decomposition results with a rank one perturbation part saves computational resources.

4.1 Introduction of rank one update method

The rank one update method can be used to update the SVD of a matrix that is perturbed by a rank one matrix [26–28]. In this paper, the method is used to modify the parameter update step in RLS, i.e., Eq. (19), for iteratively updating SVD results of the estimated matrix in current step.

The parameter update step in RLS can be expressed into a standard rank one update problem by transposing Eq. (19) and expressing estimated matrix into SVD form:

$$\begin{aligned} \hat{\boldsymbol{U}}_t\hat{\boldsymbol{\Sigma}}_t\hat{\boldsymbol{V}}_t^T &= \hat{\boldsymbol{U}}_{t-1}\hat{\boldsymbol{\Sigma}}_{t-1}\hat{\boldsymbol{V}}_{t-1}^T + \boldsymbol{a}\boldsymbol{b}^T \\ &= \hat{\boldsymbol{U}}_{t-1}\hat{\boldsymbol{\Sigma}}_{t-1}\hat{\boldsymbol{V}}_{t-1}^T + \boldsymbol{e}_t^T\boldsymbol{K}_t^T \end{aligned} \tag{21}$$

where $\boldsymbol{a} = \boldsymbol{e}_t^T \in \mathcal{R}^{4\times1}$, and $\boldsymbol{b} = \boldsymbol{K}_t \in \mathcal{R}^{6\times1}$. Eq. (21) is consistent with the rank one problem described in Ref. [26] and can be solved step by step with the SVD result obtained from last time step.

Solving a rank one update problem such as Eq. (21) can be split into several steps. The hat symbol $\hat{\square}$ used to stand for estimation in Eq. (21) is ignored in following rank one update method introduction part for simplicity. Details of rank one update method can be found in Refs. [26, 27, 29]. In this paper, most derivation steps are ignored, and only conclusive equations of the rank one method are introduced:

0) The rank one update method is used to get the SVD result of a rank one perturbed matrix:

$$\underbrace{\boldsymbol{U}_t\boldsymbol{\Sigma}_t\boldsymbol{V}_t^T}_{\boldsymbol{\Theta}_t} = \underbrace{\boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{V}_{t-1}^T}_{\boldsymbol{\Theta}_{t-1}} + \boldsymbol{a}\boldsymbol{b}^T \tag{22}$$

1) As in Ref. [26], Eq. (22) should be transformed into two rank one update problem:

$$\begin{aligned} \boldsymbol{\Theta}_t\boldsymbol{\Theta}_t^T &= \boldsymbol{U}_t \underbrace{\boldsymbol{\Sigma}_t\boldsymbol{\Sigma}_t^T}_{\boldsymbol{D}_t} \boldsymbol{U}_t^T = \left(\boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{V}_{t-1}^T + \boldsymbol{a}\boldsymbol{b}^T\right)\left(\boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{V}_{t-1}^T + \boldsymbol{a}\boldsymbol{b}^T\right)^T \\ &= \boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{V}_{t-1}^T\boldsymbol{V}_{t-1}\boldsymbol{\Sigma}_{t-1}^T\boldsymbol{U}_{t-1}^T + \underbrace{\boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{V}_{t-1}^T\boldsymbol{b}}_{\bar{\boldsymbol{b}}}\,\boldsymbol{a}^T + \boldsymbol{a}\,\underbrace{\boldsymbol{b}^T\boldsymbol{V}_{t-1}\boldsymbol{\Sigma}_{t-1}^T\boldsymbol{U}_{t-1}^T}_{\bar{\boldsymbol{b}}^T} + \boldsymbol{a}\,\underbrace{\boldsymbol{b}^T\boldsymbol{b}}_{\beta}\,\boldsymbol{a}^T \\ &= \boldsymbol{U}_{t-1}\underbrace{\boldsymbol{\Sigma}_{t-1}\boldsymbol{\Sigma}_{t-1}^T}_{\boldsymbol{D}_{t-1}}\boldsymbol{U}_{t-1}^T + \bar{\boldsymbol{b}}\boldsymbol{a}^T + \boldsymbol{a}\bar{\boldsymbol{b}}^T + \beta\boldsymbol{a}\boldsymbol{a}^T \end{aligned} \tag{23}$$

where $\boldsymbol{D}$ is actually the eigenvalue matrix of $\boldsymbol{\Theta}\boldsymbol{\Theta}^T$. The perturbation part can be further derived:

$$\begin{aligned} \bar{\boldsymbol{b}}\boldsymbol{a}^T + \boldsymbol{a}\bar{\boldsymbol{b}}^T + \beta\boldsymbol{a}\boldsymbol{a}^T &= \begin{bmatrix} \boldsymbol{a} & \bar{\boldsymbol{b}} \end{bmatrix} \begin{bmatrix} \beta & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{a}^T \\ \bar{\boldsymbol{b}}^T \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} \boldsymbol{a} & \bar{\boldsymbol{b}} \end{bmatrix} \boldsymbol{Q}}_{\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{b}_1 \end{bmatrix}} \begin{bmatrix} \rho_1 & 0 \\ 0 & \rho_2 \end{bmatrix} \underbrace{\boldsymbol{Q}^T \begin{bmatrix} \boldsymbol{a}^T \\ \bar{\boldsymbol{b}}^T \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{b}_1 \end{bmatrix}^T} \\ &= \rho_1\boldsymbol{a}_1\boldsymbol{a}_1^T + \rho_2\boldsymbol{b}_1\boldsymbol{b}_1^T \end{aligned} \tag{24}$$

where $\begin{bmatrix} \beta & 1 \\ 1 & 0 \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \rho_1 & 0 \\ 0 & \rho_2 \end{bmatrix} \boldsymbol{Q}^T$ is obtained by implementing Schur-decomposition. Then:

$$\boldsymbol{U}_t \boldsymbol{D}_t \boldsymbol{U}_t^T = \underbrace{\boldsymbol{U}_{t-1} \boldsymbol{D}_{t-1} \boldsymbol{U}_{t-1}^T + \rho_1 \boldsymbol{a}_1 \boldsymbol{a}_1^T}_{\tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{U}}^T} + \rho_2 \boldsymbol{b}_1 \boldsymbol{b}_1^T \tag{25}$$

2) The Eq. (25) can be viewed as two consecutive rank one modifications of symmetric eigenproblem:

$$\tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{U}}^T = \boldsymbol{U}_{t-1} \boldsymbol{D}_{t-1} \boldsymbol{U}_{t-1}^T + \rho_1 \boldsymbol{a}_1 \boldsymbol{a}_1^T = \boldsymbol{U}_{t-1} \left(\boldsymbol{D}_{t-1} + \rho_1 \bar{\boldsymbol{a}} \bar{\boldsymbol{a}}^T \right) \boldsymbol{U}_{t-1}^T \tag{26}$$

$$\boldsymbol{U}_t \boldsymbol{D}_t \boldsymbol{U}_t^T = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{U}}^T + \rho_2 \boldsymbol{b}_1 \boldsymbol{b}_1^T = \tilde{\boldsymbol{U}} \left(\tilde{\boldsymbol{D}} + \rho_2 \bar{\boldsymbol{b}} \bar{\boldsymbol{b}}^T \right) \tilde{\boldsymbol{U}}^T \tag{27}$$

where $\bar{\boldsymbol{a}} = \boldsymbol{U}_{t-1}^T \boldsymbol{a}_1$ and $\bar{\boldsymbol{b}} = \tilde{\boldsymbol{U}}^T \boldsymbol{b}_1$.

3) The key to solve aforementioned Eqs. (26-27) is calculating the eigendecomposition of both $\boldsymbol{D}_{t-1} + \rho_1 \bar{\boldsymbol{a}} \bar{\boldsymbol{a}}^T$ and $\tilde{\boldsymbol{D}} + \rho_2 \bar{\boldsymbol{b}} \bar{\boldsymbol{b}}^T$, i.e., to get eigenvalues and eigenvectors of a known diagonal matrix $\boldsymbol{D}$ with rank one perturbation $\rho \boldsymbol{z}\boldsymbol{z}^T$, as follows:

$$\tilde{\boldsymbol{C}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{C}}^T = \boldsymbol{D} + \rho \boldsymbol{z}\boldsymbol{z}^T \tag{28}$$

Solutions to the problem can be found in Ref. [29] and simply introduced here:

3.1) An initial deflation should be done to simplify the given problem in Eq. (28), like multiple eigenvalues [29].

3.2) Based on a theorem proven by Golub [30], the eigenvalues of $\boldsymbol{D} + \rho \boldsymbol{z}\boldsymbol{z}^T$ can be obtained by solving following secular equation with iterative approximation method:

$$w(\lambda) \equiv 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{(d_j - \lambda)} = 0 \tag{29}$$

where ζ_j is the jth element in normalized $\boldsymbol{z}$, and d_j is the jth eigenvalue of $\boldsymbol{D}$.

3.3) The explicit computation of the updated eigenvectors $\tilde{\boldsymbol{C}}$ can then be done as follows:

$$\boldsymbol{c}_i = \frac{\boldsymbol{D}_i^{-1} \boldsymbol{z}}{\left\| \boldsymbol{D}_i^{-1} \boldsymbol{z} \right\|_2} \tag{30}$$

where $\boldsymbol{D}_i = \boldsymbol{D} - \lambda_i \boldsymbol{I}$.

4) Substituting the eigendecomposition into Eqs. (26-27) leads to following equations:

$$\tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\tilde{\boldsymbol{U}}^T = \boldsymbol{U}_{t-1} \left(\boldsymbol{D}_{t-1} + \rho_1 \bar{\boldsymbol{a}} \bar{\boldsymbol{a}}^T \right) \boldsymbol{U}_{t-1}^T = \underbrace{\boldsymbol{U}_{t-1} \tilde{\boldsymbol{C}}}_{\tilde{\boldsymbol{U}}} \tilde{\boldsymbol{D}} \tilde{\boldsymbol{C}}^T \boldsymbol{U}_{t-1}^T \tag{31}$$

$$\boldsymbol{U}_t \boldsymbol{D}_t \boldsymbol{U}_t^T = \tilde{\boldsymbol{U}} \left(\tilde{\boldsymbol{D}} + \rho_2 \bar{\boldsymbol{b}} \bar{\boldsymbol{b}}^T \right) \tilde{\boldsymbol{U}}^T = \underbrace{\tilde{\boldsymbol{U}} \boldsymbol{C}_t}_{\boldsymbol{U}_t} \boldsymbol{D}_t \boldsymbol{C}_t^T \tilde{\boldsymbol{U}}^T \tag{32}$$

Until now, the left singular vector $\boldsymbol{U}_t$ is successfully updated, and it is also easy to obtain singular values $\boldsymbol{\Sigma}_t$ from the eigenvalue matrix $\boldsymbol{D}_t$ of $\boldsymbol{\Theta}_t \boldsymbol{\Theta}_t^T$ by taking square root.

5) With $\boldsymbol{U}_t$ and $\boldsymbol{\Sigma}_t$, a set of right singular vectors in column space can be obtained, as follows:

$$\boldsymbol{\Theta}_t \boldsymbol{v}_i = \rho_i \boldsymbol{u}_i, \quad for\ i\ from\ 1\ to\ r \tag{33}$$

where r is the rank of $\boldsymbol{\Theta}_t$.

6) If required, null space basis can be calculated by solving following equation:

$$\boldsymbol{V}_r^T \boldsymbol{x} = 0 \tag{34}$$

where $\boldsymbol{V}_r = [\ \boldsymbol{v}_1 \quad \cdots \quad \boldsymbol{v}_r\ ]$ is the column space basis obtained in step 5). Solutions to Eq. (34) constitute null space vectors $\boldsymbol{V}_n$. For guaranteeing the orthogonality of right singular vectors, Gram-Schmidt process can be implemented on the full right singular matrix $\boldsymbol{V} = [\ \boldsymbol{V}_r \quad \boldsymbol{V}_n\ ]$.

So far, the rank one update method for SVD is introduced. It is obvious that Eq. (19) can be solved by the method described by Eqs. (22-34) when the SVD of the estimated CA matrix $\hat{\boldsymbol{\Theta}}$ is required.

4.2 Updating singular value decomposition of CA matrix in RLS filter

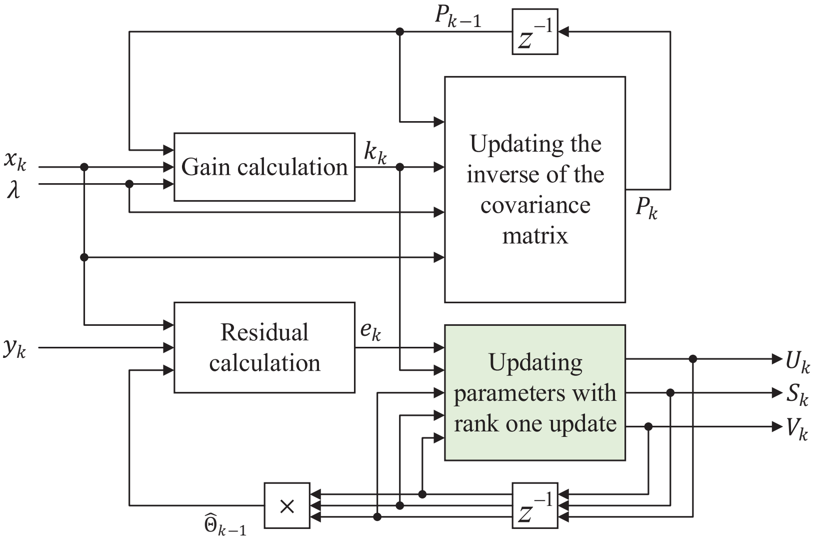

Fig. 4 RLS filter with rank one update

In designed INDI controller with an adaptive CA part, the complete SVD process of the CA matrix in every time step costs most of the computational resources. A rank one update method described in section 4.1 is integrated with RLS in this paper for iteratively updating SVD, as in Fig. 4. In this way, this modified efficient RLS can directly output SVD results of estimated matrix which can be easily utilized in CA module. Benefited from efficient computation of rank one update, modified adaptive CA module saves calculation and time.

By updating SVD of estimated CA matrix with perturbation in Eq. (19), rank one update is integrated with RLS filter, as shown in Fig. 4. The parameter updating part, i.e., the green part of Fig. 4, provides SVD results of estimated matrix for downstream CA part. Rank one update-based parameter update step in modified RLS filter can be expressed as follows:

$$\left\{\hat{\boldsymbol{U}}_t, \hat{\boldsymbol{S}}_t, \hat{\boldsymbol{V}}_t^T\right\} = f_{Rank1}\left(\hat{\boldsymbol{U}}_{t-1}, \hat{\boldsymbol{S}}_{t-1}, \hat{\boldsymbol{V}}_{t-1}^T, \boldsymbol{e}_t^T, \boldsymbol{K}_t\right) \tag{35}$$

where f_{Rank1} stands for rank one update algorithm described by Eqs. (22-34). The other equations of modified RLS are the same as in Eqs. (17-18,20).

The improvement in computational efficiency comes from using rank one update method in parameter update step. Comparing Fig. 3 with Fig. 4, changes are mainly in green blocks. An SVD module is required in Fig. 3 to calculate SVD results, while proposed efficient RLS can directly output SVD of estimated matrix in update step. Therefore, a comparison of computational efficiency between common RLS and proposed efficient RLS can be simplified to compare parameter update step in both filters.

5 Results analysis

Proposed rank one update-based adaptive CA is tested on a complex high-fidelity multicopter model described as in section 2. The INDI controller with rank one update-based RLS filter is compared against the one with common RLS to show that the tracking performance is not deteriorated by rank one update method. Besides, data fed into parameter update step are recorded for further computational tests. With these data, numerical calculation tests are done on parameter update step of both common and proposed RLS. Results show that rank one update-based parameter update performs substantially better than using SVD after parameter update in terms of saving computational resources.

5.1 Tracking performance of the INDI controller with proposed adaptive CA

Designed INDI controller with rank one update-based adaptive CA is tested and compared with the one using SVD to get pseudo inverse of CA matrix. Firstly, the SVD results obtained from common RLS and proposed RLS are compared in the same simulation test to show that there is no large numerical difference caused by using rank one update method. Then, command tracking performance is shown to have no large change after replacing common RLS with proposed one.

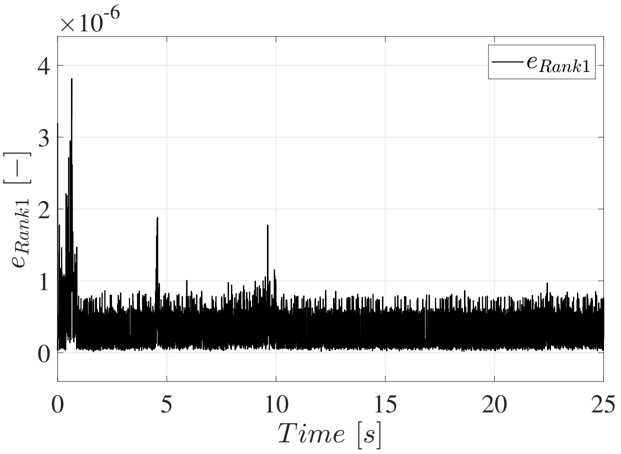

Fig. 5 Difference between rank one update and SVD

Rank one update method as described in section 4.1 will produce numerical differences with SVD methods [26, 27]. When implementing the method, some thresholds are required for terminating iteration process of the approximation method used to solve Eq. (29). Different thresholds may cause different levels of numerical deviation between SVD results obtained from rank one update method and SVD methods. In this paper, the threshold is set according to suggestions from Ref. [29]. A test is done that rank one update-based RLS filter runs in parallel with common RLS in close loop control. The SVD result difference is quantified in this paper as follows:

$$e_{Rank1} = \frac{\|\boldsymbol{U}_{Rank1} - \boldsymbol{U}_{SVD}\|_F}{m} + \frac{\|\boldsymbol{S}_{Rank1} - \boldsymbol{S}_{SVD}\|_F}{\sqrt{r}} + \frac{\left\|\boldsymbol{V}_{r,Rank1} - \boldsymbol{V}_{r,SVD}\right\|_F}{\sqrt{mn}} \tag{36}$$

where $\boldsymbol{V}_r$ is the column space basis of estimated matrix $\boldsymbol{\Theta}$, subscript $\square_{Rank1}$ stands for parameters obtained from rank one update, subscript $\square_{SVD}$ means parameters from SVD method, $\|\cdot\|_F$ is the Frobenius norm of a matrix, m, n, r are the row number, column number, and rank of $\boldsymbol{\Theta}$, respectively. The simulation result shows that the numerical difference is quite small with a magnitude of 10^{-6}, as in Fig. 5. It can be seen from Fig. 5 that the differences at certain time steps are larger than in peacetime, which is caused by large rank one perturbation on previous matrix. Sudden changes of $\dot{\boldsymbol{v}}$ in Eq. (11) and $\Delta\boldsymbol{\Omega}$ occur every time when there is a step command, as in Fig. 6, which give a strong excitation and more information to RLS filter. The rank one perturbation obtained from vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ in Eq. (22) are positively influenced by the measurement of $\dot{\boldsymbol{v}}$ and $\Delta\boldsymbol{\Omega}$, and therefore are also enlarged. Due to the rank one update method used in this paper, the difference in singular values is stable even though there is a large perturbation. The differences in singular vectors are increased because of large rank one perturbation, and are proven in Ref. [27, 29] that it is bounded by thresholds set in rank one update method.

As in aforementioned discussion, rank one update method may cause little numerical error and then influence estimation results. The estimated matrix with this error will further affect the control of the plant. However, due to the very small magnitude of the error, i.e., about 10^{-6}, it will not cause significant changes in plant states and the estimation error can also be corrected and eliminated by measurements fed back from plant. Therefore, the tracking performance of the controller with rank one update-based adaptive CA remains nearly the same as the one executing complete SVD in every time step, as in Fig. 6. The black line in Fig. 6 stands for tracking performance of designed controller with SVD used in adaptive CA, and the blue line corresponds to the controller using rank one update. These two lines coincide well,

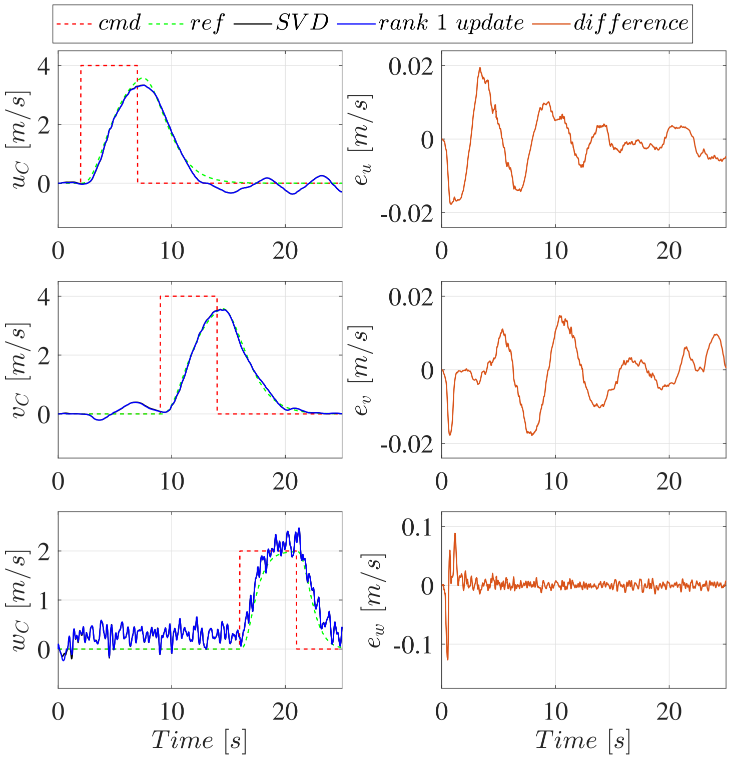

Fig. 6 Difference in terms of tracking performance between controllers using rank one update and SVD

which means there is not too large deviation in tracking performance between using SVD and rank one update in developed controller. The right part of Fig. 6 shows that the deviations in three velocity channels are in the magnitude of 10^{-2}. At the beginning of this simulation, RLS filter will converge quickly because of strong excitation, and the estimated matrix will also have rapid changes. In this stage, the numerical differences mentioned above can cause differences in estimation and tracking performance between putting SVD and rank one update in adaptive CA. Therefore, the difference in *w* channel is a little bit larger with a magnitude of 10^{-1}. Generally speaking, using rank one update-based adaptive CA will not adversely affect the tracking performance.

5.2 Computational efficiency analysis of rank one update-based RLS filter

Rank one update method, which can update SVD result of a rank one perturbed matrix [26, 27], is integrated into the designed adaptive CA module in this paper for reducing calculation. The SVD of CA matrix is usually required for implementing pseudo inverse-based CA method and null space-based techniques. In common adaptive CA module as in Ref. [4], estimated CA matrix obtained from RLS filter is decomposed completely by SVD methods in every time step. The decomposition uses much computational resources because of its complexity. In order to reduce calculation, rank one update is used in parameter update step of RLS algorithm for iteratively updating SVD of estimated CA matrix, as in section 4. In this way, rank one update connects RLS filter with CA part, and complete SVD in every time step is no more required.

Several computational efficiency tests are done in this paper to verify the proposed RLS filter. As in section 4.2, only the parameter update steps in common RLS and modified RLS are different. So, tests are only implemented to compare computational efficiency of this step. Data used in the tests are obtained from simulation experiment in section 5.1, which have 25001 data points. The data will be calculated 100 times in single test, which means 100 × 25001 iterations for both SVD-based and rank one update-based parameter update. The SVD used in common RLS is the MATLAB built-in SVD function with MATLAB version as R2023a. The parameter update step of common RLS and the SVD function are transferred into C code by MATLAB Coder. Besides, rank one update-based parameter update step as in Eq. (35) is also transferred into C code in the same way. The toolchain used in MATLAB Coder is 'Microsoft Visual Studio Project 2019 | CMake (64-bit Windows)'. These two sets of C Code run in Visual Studio 2019 and are analyzed by Performance Profiler. When running C code, the configuration is set to 'Release', and the platform is set to 'x64'. The computer used here has an AMD Ryzen 7 PRO 4750U CPU and 2 × 8 GB DDR4 memory.

The aforementioned computational efficiency test results are as in Tab. 1. It is obvious that rank one update-based parameter update step is faster than SVD-based one, which benefits from the efficient computation of rank one update method. For processing 100 × 25001 data points in x64 platform, rank one update only needs about 28195.55 ms while SVD takes roughly 43210.50 ms. By using rank one

Table 1 Computational time results of SVD-based and rank one update-based parameter update step

Methods	SVD	Rank One Update
Computational Time (ms)	41904.75	28087.75
	50538.59	25261.37
	43359.43	30515.53
	41706.36	30987.84
	38543.35	26125.26
Average Computational Time (ms)	43210.50	28195.55

update to connect RLS with CA part rather than SVD, the computational efficiency is enhanced by 34.75%. It is shown in Tab. 1 that there are some differences among computational times in each test. It is caused by different environments, computer workloads, CPU usage, etc. In each experiment, diligent efforts are made to minimize these discrepancies to ensure fair comparisons.

6 Conclusion

A rank one update-based adaptive CA is proposed in this paper for improving computational efficiency and in the same time keeping good control performance. The adaptive CA in this paper uses RLS to estimate CA matrix and pseudo inverse-based CA method to allocate desired pseudo commands. Therefore, the SVD result of the estimated matrix is required in every time step for calculating its pseudo inverse, which wastes computational resources in common SVD-based adaptive CA. In this paper, the rank one update method is used to update the SVD result and connect RLS with CA module. This rank one update-based adaptive CA is tested in a multicopter testbed where an INDI controller is designed to control three-axis velocities. Results show that the SVD results obtained from rank one update method and SVD method have a small difference of about 10^{-5}. With such a negligible difference, the control performance has not large difference between using rank one update and SVD methods. The maximum difference exists in vertical velocity control channel, which is about 0.1 m/s. When it comes to computational efficiency, rank one update-based adaptive CA shows significant improvement compared to the SVD-based one. The update step in RLS with rank one update needs only about 28195.55 ms for a 100×25001 data points test, while the update step with SVD takes almost 43210.50 ms. In other words, the rank one update method saves computational time by 34.75%. The future work would be testing the developed rank one update-based adaptive CA on the airborne computer of a real multicopter.

Acknowledgments

The first author is financially supported for his Ph.D. research by China Scholarship Council with the project reference number of 202006020037. The authors would like to acknowledge Jiannan Zhang and Venkata Sravan Akkinapalli for their support in providing useful and meaningful suggestions regarding the works of this paper.

References

[1] W Schmidbauer, C Jänig, E Vits, T Gruebl, S Sauer, N Weller, K Kehe, F Holzapfel, T Lüth, KG Kanz, et al. Ein neues rettungskonzept für schwerstverletzte in militärischen und zivilen großschadenslagen: Dronevac. *Notfall+ Rettungsmedizin*, pages 1–8, 2023. DOI: 10.1007/s10049-023-01190-5.

[2] Nhan Nguyen, Kalmanje Krishnakumar, John Kaneshige, and Pascal Nespeca. Flight dynamics and hybrid adaptive control of damaged aircraft. *Journal of guidance, control, and dynamics*, 31(3):751–764, 2008. DOI: 10.2514/1.28142.

[3] Venkata S Akkinapalli, Guillermo P Falconí, and Florian Holzapfel. Fault tolerant incremental attitude control using online parameter estimation for a multicopter system. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pages 454–460. IEEE, 2017. DOI: 10.1109/MED.2017.7984159.

[4] Hangxu Li, Stephan Myschik, and Florian Holzapfel. Null-space-excitation-based adaptive control for an overactuated hexacopter model. *Journal of Guidance, Control, and Dynamics*, 46(3):483–498, 2023. DOI: 10.2514/1.G006771.

[5] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[6] Gang Tao. *Adaptive control design and analysis*, volume 37. John Wiley & Sons, 2003.

[7] Marc Steinberg. Historical overview of research in reconfigurable flight control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 219(4):263–275, 2005. DOI: 10.1243/095441005X30379.

[8] AM Annaswamy, E Lavretsky, ZT Dydek, TE Gibson, and M Matsutani. Recent results in robust adaptive flight control systems. *International Journal of Adaptive Control and Signal Processing*, 27(1-2):4–21, 2013. DOI: 10.1002/acs.2341.

[9] Guillermo Paúl Falconí Salazar. *Adaptive Fault Tolerant Control for VTOL Aircraft with Actuator Redundancy*. PhD thesis, Technische Universität München, 2021.

[10] Pranav Bhardwaj, Venkata Sravan Akkinapalli, Jiannan Zhang, Saurabh Saboo, and Florian Holzapfel. Adaptive augmentation of incremental nonlinear dynamic inversion controller for an extended f-16 model. In *AIAA Scitech 2019 Forum*, page 1923, 2019. DOI: 10.2514/6.2019-1923.

[11] Byoung S Kim and Anthony J Calise. Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics*, 20(1):26–33, 1997. DOI: 10.2514/2.4029.

[12] Ye Zhou, Erik-Jan van Kampen, and QiPing Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. *Journal of Guidance, Control, and Dynamics*, 40(2):493–496, 2016. DOI: 10.2514/1.G001762.

[13] Hangxu Li, Liguo Sun, Wenqian Tan, Baoxu Jia, and Xiaoyu Liu. Switching flight control for incremental model-based dual heuristic dynamic programming. *Journal of Guidance, Control, and Dynamics*, 43(7):1352–1358, 2020. DOI: 10.2514/1.G004519.

[14] Ewoud JJ Smeur, Qiping Chu, and Guido CHE De Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016. DOI: 10.2514/1.G001490.

[15] Simon Hafner, Barzin Hosseini, Xiang Fang, and Florian Holzapfel. Online parameter estimation of the b-matrix of a quadcopter in time-and frequency-domain. In *International Conference on Guidance, Navigation and Control*, pages 2758–2769. Springer, 2022. DOI: 10.1007/978-981-19-6613-2_268.

[16] Simon F Hafner, Seyedbarzin Hosseini, and Florian Holzapfel. Excitation monitoring for online parameter estimation. In *AIAA SCITECH 2023 Forum*, page 0039, 2023. DOI: 10.2514/6.2023-0039.

[17] Yu Liu and Luis G Crespo. Adaptive control allocation in the presence of actuator failures. *Journal of Control Science and Engineering*, 2012:3–3, 2012. DOI: 10.1155/2012/502149.

[18] Simon S Haykin. *Adaptive filter theory*. Pearson Education India, 2002.

[19] Tor A Johansen and Thor I Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013. DOI: 10.1016/j.automatica.2013.01.035.

[20] Stefan A Raab, Jiannan Zhang, Pranav Bhardwaj, and Florian Holzapfel. Proposal of a unified control strategy for vertical take-off and landing transition aircraft configurations. In *2018 Applied Aerodynamics Conference*, page 3478, 2018. DOI: 10.2514/6.2018-3478.

[21] Jiannan Zhang, Pranav Bhardwaj, Stefan A Raab, Saurabh Saboo, and Florian Holzapfel. Control allocation framework for a tilt-rotor vertical take-off and landing transition aircraft configuration. In *2018 Applied Aerodynamics Conference*, page 3480, 2018. DOI: 10.2514/6.2018-3480.

[22] Stefan A Raab, Jiannan Zhang, Pranav Bhardwaj, and Florian Holzapfel. Consideration of control effector dynamics and saturations in an extended indi approach. In *AIAA Aviation 2019 Forum*, page 3267, 2019. DOI: 10.2514/6.2019-3267.

[23] Wayne Durham, Kenneth A Bordignon, and Roger Beck. *Aircraft control allocation*. John Wiley & Sons, 2017.

[24] Michael W Oppenheimer, David B Doman, and Michael A Bolender. Control allocation for over-actuated systems. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–6. IEEE, 2006. DOI: 10.1109/MED.2006.328750.

[25] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[26] Ratnik Gandhi and Amoli Rajgor. Updating singular value decomposition for rank one matrix perturbation. *arXiv preprint arXiv:1707.08369*, 2017. DOI: 10.48550/arXiv.1707.08369.

[27] James R Bunch and Christopher P Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31(2):111–129, 1978. DOI: 10.1007/BF01397471.

[28] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006. DOI: 10.1016/j.laa.2005.07.021.

[29] James R Bunch, Christopher P Nielsen, and Danny C Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978. DOI: 10.1007/BF01396012.

[30] Gene H Golub. Some modified matrix eigenvalue problems. *SIAM review*, 15(2):318–334, 1973. DOI: 10.1137/1015032.