

Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation

Mancinelli, Alessandro; Remes, Bart D.W.; De Croon, Guido C.H.E.; Smeur, Ewoud J.J.

DOI

[10.1007/s10846-023-01865-8](https://doi.org/10.1007/s10846-023-01865-8)

Publication date

2023

Document Version

Final published version

Published in

Journal of Intelligent and Robotic Systems: Theory and Applications

Citation (APA)

Mancinelli, A., Remes, B. D. W., De Croon, G. C. H. E., & Smeur, E. J. J. (2023). Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 108(4), Article 67. <https://doi.org/10.1007/s10846-023-01865-8>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.



Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation

Alessandro Mancinelli¹ · Bart D. W. Remes¹ · Guido C. H. E. De Croon¹ · Ewoud J. J. Smeur¹

Received: 18 November 2022 / Accepted: 30 March 2023
© The Author(s) 2023

Abstract

Hybrid Unmanned Aerial Vehicles UAV are vehicles capable of take-off and landing vertically like helicopters while maintaining the long-range efficiency of fixed-wing aircraft. Unfortunately, due to their wing area, these vehicles are sensitive to wind gusts when hovering. One way to increase the hovering wind-rejection capabilities of hybrid UAV is through the addition of extra actuators capable of directing the thrust of the rotors. Nevertheless, the ability to control UAVs with many actuators is strictly related to how well the Control Allocation problem is solved. Generally, to reduce the problem complexity, conventional (CA) methods make use of linearized control effectiveness in order to optimize the inputs that achieve a certain control objective. We show that this simplification can lead to oscillations if it is applied to thrust vectoring vehicles, with pronounced non-linear actuator effectiveness. When large control objectives are requested or actuators saturate, the linearized effectiveness based CA methods tend to compute a solution far away from the initial actuator state, invalidating the linearization. A potential solution could be to impose limits on the solution domain of the linearized CA algorithm. However, this solution only reduces the oscillations at the expense of a lag in the vehicle acceleration response. To overcome this limitation, we present a fully nonlinear CA method, which uses an Sequential Quadratic Programming (SQP) algorithm to solve the CA problem. The method is tested and implemented on a single board computer that computes the actuator solution in real time onboard a dual axis tilting rotor quad-plane. Flight test experiments confirm the problem of severe oscillations in the linearized effectiveness CA algorithms and show how the only algorithm able to optimally solve the CA problem is the presented Nonlinear method.

Keywords UAV · VTOL · Control allocation · Nonlinear programming · INDI · Nonlinear actuators · Tilt rotor · Quad-plane · Fully actuated vehicles · Hybrid MAV · Weighted least squares · Quadratic programming

1 Introduction

In the last twenty years, a boom in the (UAV) industry has been experienced all over the world. UAV prove to be flexible vehicles for several different tasks ranging from Mapping &

Surveying to medical delivery. Of particular interest are the so-called 'hybrid' vehicles, able to combine the cruise efficiency of conventional fixed wing planes with the hovering capability typical of helicopters. This category of vehicles is characterized by the presence of a wing surface used to generate lift in the forward part of the flight. However, the presence of a wing brings significant disadvantages during the vertical take-off and landing phase. During vertical take-off and landing, the wing surface interacts with the wind, generating strong and unpredictable forces and moments on the aircraft. If a precision landing has to be performed, these forces and moments need to be opposed by the vehicle.

A possible way to increase a hybrid air vehicle's gust disturbance capabilities could be through the addition of extra actuators on the vehicle. Using tilting mechanisms can enable the vehicle to direct the thrust and moments generated by the motors to oppose those induced by the wind. An example of

✉ Alessandro Mancinelli
a.mancinelli@tudelft.nl

Bart D. W. Remes
b.d.w.remes@tudelft.nl

Guido C. H. E. De Croon
g.c.h.e.decroon@tudelft.nl

Ewoud J. J. Smeur
e.j.j.smeur@tudelft.nl

¹ Department of Control and Simulation,
Delft University of Technology,
Kluyverweg 1, 2629HS Delft, The Netherlands

this was proposed in [1], where the gust rejection capabilities of a conventional quad-plane were compared to the one of a dual-axis tilting rotor quad-plane, showing how effective the thrust vectoring capability is to minimize the influence of the wind on the vehicle in the hovering configuration.

Unfortunately, the large number of actuators increases the complexity of the Control Allocation (CA) problem for these vehicles. CA is the problem of distributing control effort over a set of actuators to achieve certain desired control forces and moments. These vehicles must solve the CA problem optimally to fully exploit their capabilities.

Several CA problem statements and resolution methods have been proposed over time. A good overview of the different CA methods is available in [2–4]. The unconstrained CA problem is most commonly solved with the pseudo inverse method, which inverts the mapping from control commands to the desired motion effects [5]. As for the constrained CA problem, redistributed pseudo-inverse [6] and direct allocation [7] are a few CA strategies based on the pseudo-inverse method, able to optimize the computed solution in case of actuator saturation. More elaborate iterative constrained CA methods, formulate the problem as a constrained linear or quadratic programming optimization problem [8–10]. Note that all these problem formulations presented so far still share the same linearized control effectiveness assumption.

Regrettably, when vehicles with strong nonlinear behavior are employed, linearized effectiveness approaches tend to fail. This happens especially when a large control objective needs to be realized with ineffective non-linear actuators.

A much more effective method, potentially capable of dealing with such a problem, is the nonlinear programming method. The very first attempt to approach the CA problem with a nonlinear programming method was presented in [11], with the aim of solving the allocation problem for reentry vehicles. During reentry, these vehicles are exposed to very high angle of attack, where the nonlinear dynamics of the effectors are too evident that the CA problem cannot be properly solved with a linearized approach. Simulation results confirmed the weaknesses of the linearized control effectiveness approaches in solving the CA problem for these kinds of vehicles, highlighting the potential of the nonlinear programming method.

Following the same idea, the nonlinear resolution of the CA problem was also applied to overactuated ground vehicles [12–14], gaining more and more popularity over time. Recently, learning-based approaches for the resolution of the nonlinear CA problem were proposed with the aim of reducing the computational load [15, 16]. Overall, although computationally intensive, the nonlinear programming approach for the resolution of the CA problem proved to be an extremely flexible and powerful method to determine control effectors solution for highly nonlinear and

overactuated vehicles. However, so far, limited computational power of embedded computers, and the necessity of solving the problem in real time, prevented this CA method to be implemented and tested on a flying vehicle.

In this paper, we address the CA problem of vehicles with highly nonlinear control effectors, such as thrust vectoring UAV. Within the paper we initially show the ineffectiveness of state-of-the-art linearized effectiveness approaches for these type of vehicles. Then, we propose a method to solve in real time the nonlinear CA problem using a Sequential Quadratic Programming (SQP) algorithm. The proposed method uses the nonlinear Equations Of Motion (EOM) inversion to compute the actuator solution, thus overcoming the control effectiveness linearization limitation. The control framework used to implement such a CA method is a modified Incremental Nonlinear Dynamic Inversion (INDI), adapted to include the incremental law through nonlinear EOM evaluation.

To prove the applicability of such a control strategy on a real time and computationally constrained platform, the control framework was implemented and tested on the dual-axis tilting rotor quad-plane depicted in Fig. 1. The vehicle has a 2.4 kg take-off mass and features 8 servos and 4 motors for a total of 12 actuators controlling the 6 degrees of freedom. A commercial Raspberry Pi 4B Single Board Computer (SBC) was used to solve the Nonlinear CA problem in real time onboard the drone. Thanks to hardware and software optimization, it was possible to run the algorithm computing the actuator solution at an average refresh rate of 350 hz, while using just one of the four available cores of the SBC. Another of the available cores was dedicated to the serial communication with the main Flight Control Board (FCB), leaving a total of 2 free cores available for running algorithms supporting additional tasks.

To summarize, the research highlights of the work are:

- We demonstrated the inapplicability of state-of-the-art linearized effectiveness Control Allocation algorithms on overactuated UAVs with thrust vectoring capability.



Fig. 1 A picture of the dual-axis tilting rotor quad-plane used as test bench for the CA methods

- We proposed a Control Allocation algorithm based on Nonlinear EOM inversion able to adequately control overactuated UAVs with thrust vectoring capability.
- We demonstrated the applicability of the proposed Nonlinear Control Allocation method on a flying vehicle by implementing the algorithm on a commercial SBC onboard of a dual-axis tilting rotor quad-plane.

The paper is structured as follows: In Section 2, the nonlinear system is presented and the INDI control framework is introduced. Within this section, the linear and quadratic programming formulation of the CA problem are also derived. In Section 3, the Nonlinear CA algorithm and the nonlinear INDI control framework are presented. In Section 4, the Nonlinear CA module is firstly implemented in simulation and then tested on the flying double axis tilting rotor quad-plane. In Section 5, some limitations of the Nonlinear CA methods are discussed. Finally, in Section 6 conclusions are drawn.

2 Preliminaries

2.1 System Description and INDI Derivation

Considering the generic nonlinear system:

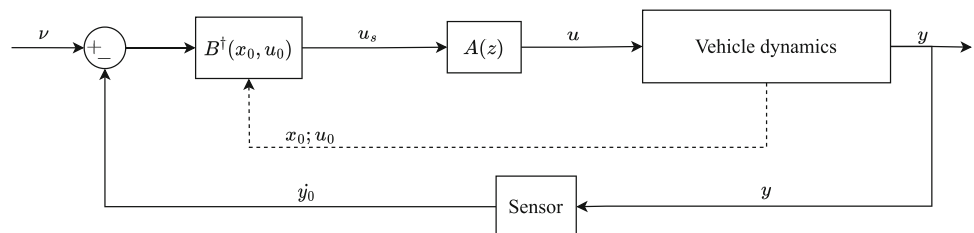
$$\begin{aligned} \dot{x} &= f(x(t), u(t)) \\ y &= h(x(t)) \end{aligned} \tag{1}$$

Where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input vector and $y(t) \in \mathbb{R}^l$ is the output vector. $f_{nx1} : D_{x,u} \rightarrow \mathbb{R}^n$ is a nonlinear function representing the vehicle dynamics while $h_{lx1} : D_x \rightarrow \mathbb{R}^l$ is a nonlinear function representing the output dynamics.

For the derivation of the linearized CA methods, let's now consider the first order Taylor expansion of the first term in Eq. 1, centered around the current state x_0 and control input u_0 :

$$\begin{aligned} \dot{x} \simeq & f(x_0, u_0) + \left. \frac{\partial f(x, u)}{\partial x} \right|_{x=x_0, u=u_0} (x - x_0) \\ & + \left. \frac{\partial f(x, u)}{\partial u} \right|_{x=x_0, u=u_0} (u - u_0) \end{aligned} \tag{2}$$

Fig. 2 Scheme of the INDI control framework. In the diagram, the dashed lines represent variables estimated through sensor (for the vehicle state) or actuator model (for the current actuator state). The term $A(z)$ represents the actuator dynamics transfer function



$$= \dot{x}_0 + \underbrace{F(x_0, u_0)(x - x_0)}_{F_{\Delta x}} + \underbrace{B(x_0, u_0)(u - u_0)}_{F_{\Delta u}},$$

where $F_{\Delta x}$ is the state dependent term and $F_{\Delta u}$ is the control input dependent term.

Under the assumption of small controller sampling time and very fast actuators, it is possible to apply the time scale separation principle, letting us neglect the state dependent term Δx [17]. Based on the aforementioned assumptions, Eq. 2 can then be rewritten as follows:

$$\dot{x} \simeq \dot{x}_0 + B(x_0, u_0)(u - u_0) = \dot{x}_0 + B(x_0, u_0)\Delta u. \tag{3}$$

From Eq. 3, it is now possible to derive the associated INDI control law for the vehicle. Assuming a direct expression between the state vector and the output vector (i.e. $y = h(x) = x$) and replacing \dot{x} with the desired state derivative \dot{x}_d , the control law becomes:

$$\begin{aligned} \Delta u &= B^\dagger(x_0, u_0)(\dot{y}_d - \dot{y}_0) \\ u_s &= u_0 + \Delta u \end{aligned} \tag{4}$$

where $B^\dagger(x_0, u_0)$ represents the generalized inverse of the effectiveness matrix, linearized around the current state x_0 and current control input u_0 . $\dot{y}_d - \dot{y}_0$ is the output derivative error \dot{y}_e , representing in this case the vehicle acceleration. The desired output derivative \dot{y}_d is also known in literature as pseudo-control v . The term u_s represents the control input solution of the CA problem.

Concerning the current output derivative value \dot{y}_0 , it represents the vehicle linear and angular accelerations and a sensor based estimation can be employed [18]. A scheme of the INDI control framework is depicted in Fig. 2.

For completeness, it is possible to manipulate the terms in Eq. 3 to directly compute the control input solution, rather than the control input increment:

$$u_s - u_0 = B^\dagger(x_0, u_0)(\dot{y}_d - \dot{y}_0) \tag{5}$$

Then, by bringing the current actuator state on the right side and including it into the inversion law it becomes:

$$u_s = B^\dagger(x_0, u_0)(v - \dot{y}_0 + B(x_0, u_0)u_0), \tag{6}$$

where the desired acceleration vector \dot{y}_d was replaced by the pseudo-control vector v .

Instead of minimizing the actuator solution norm, it is also possible to minimize the solution norm with respect to a desired control input state u_d [3]:

$$u_g = B^\dagger(x_0, u_0)v_g \quad (7)$$

with

$$\begin{aligned} v_g &= v - \dot{y}_0 + B(x_0, u_0)(u_0 - u_d) \\ u_g &= u_s + u_d \end{aligned}$$

The problem formulation in Eq. 7 is not yet a constrained problem and the solution can't be optimized in case of actuator saturation. Within the paper, we will refer to the CA strategy in Eq. 7 as the Pseudo Inverse Unconstrained (PIU) CA algorithm.

2.2 Quadratic Programming Formulation of the CA Problem

Equation 3 can be reformulated as quadratic programming problem [10]. This brings two main benefits. The first benefit is the optimization of the control input solution in case of actuator saturation. The second benefit is the possibility to include multiple objectives in the cost function to be minimized during the resolution process.

Following the formulation presented by [18], the CA problem becomes:

$$\begin{aligned} C(u) &= \|W_u(u - u_d)\|^2 + \gamma_v \|W_v(B(x_0, u_0)u - v_w)\|^2 \\ &= \left\| \begin{pmatrix} \gamma_v^{\frac{1}{2}} W_v B \\ W_u \end{pmatrix} u - \begin{pmatrix} \gamma_v^{\frac{1}{2}} W_v v_w \\ W_u u_d \end{pmatrix} \right\|^2; \\ u_s &= \arg \min C(u) \\ &\text{subject to} \\ u_{\min} &< u < u_{\max} \end{aligned} \quad (8)$$

with

$$v_w = v - \dot{y}_0 + B(x_0, u_0)u_0$$

where u_s is the control input solution. The two diagonal matrices W_u and W_v represent respectively the weight for the control input and the pseudo-control. The role of these matrices will be more clear later on, when we will apply the CA methods on the flying vehicle. The term γ_v is the scale factor assigning the preference on the two objectives included in the cost function. In order to prioritize the minimization of the control objective v over the energy efficiency of the solution, this term should be chosen to be very high

($\gamma_v \gg 1$). The solution of the least squares problem presented in Eq. 8 can be efficiently determined with the active set method presented in [18].

Within this paper, we will refer to this CA strategy as the Weighted Least Squares (WLS) CA algorithm.

3 Nonlinear Formulation of the CA Problem

So far, the methods presented to solve the CA problem associated with the INDI control framework make use of a linearized approximation of the actuator effectiveness. However, in some occasions, the vehicle effectors behavior cannot be fully represented by a linear system. For those types of platforms, it is possible to formulate and solve the CA problem with an iterative nonlinear optimization process [11]. If we still assume very fast actuators, the Nonlinear CA problem can be reformulated as follows:

$$\begin{aligned} C(u) &= \|\gamma_u^{\frac{1}{2}} W_u(u - u_d)\|^2 + \|W_v(f(x_0, u) - v_n)\|^2; \\ u_s &= \arg \min C(u) \\ &\text{subject to} \\ u_{\min} &< u < u_{\max} \end{aligned} \quad (9)$$

with

$$v_n = v - \dot{y}_0 + f(x_0, u_0)$$

where γ_u is the control input optimality scale factor. In contrast to Eq. 8, in this cost function, the scale factor is assigned to the control input rather than to the control objective. It is therefore suggested to assign a very small value ($\gamma_u \ll 1$) to keep the problem priority on the primary control objective.

The main noticeable difference between the WLS CA problem formulated in Eq. 8 and the Nonlinear CA problem formulated in Eq. 9 is the replacement of the linearized control effectiveness matrix with the nonlinear vehicle dynamics expression.

Concerning the two diagonal matrices W_u and W_v , as in the WLS CA, they penalize respectively the associated actuator command and control objective. Within the paper, we will refer to this CA strategy as the Nonlinear CA algorithm.

3.1 Resolution Method of the Nonlinear CA Problem

The problem presented in Eq. 9 is a constrained nonlinear optimization problem whose solution can be determined using the nonlinear programming approach. Among the different nonlinear programming algorithms proposed in literature [19], SQP has been shown to be the most efficient and accurate algorithm in almost all circumstances [20]. We

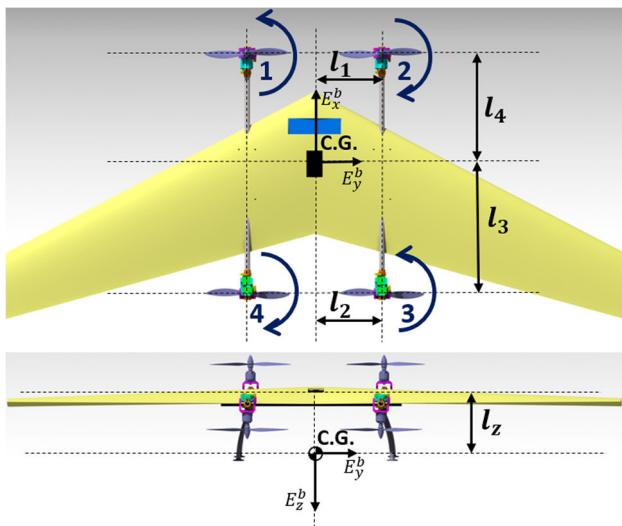


Fig. 3 Motor spinning direction and vehicle geometry with respect to the Center of Gravity (CG)

therefore decided to employ this algorithm to solve the non-linear problem presented in Eq. 9.

Among all the different SQP implementations available, we decided to employ the line search based SQP algorithm implemented in the MATLAB *fmincon* function. A detailed derivation of the SQP method for the resolution of the Non-linear CA problem is beyond the scope of this paper. For the interested reader, [19] explains in detail how the optimization process works and how it can be implemented.

In brief, the strategy of the SQP optimization process is to reduce the nonlinear problem complexity by the formulation of a series of unconstrained sequential quadratic programming sub problems. Each sub problem solution determines the searching direction while the step size is determined by evaluating a merit function containing both the cost function and the problem constraints. Specifically, two main characteristics of the chosen SQP optimization strategy are particularly beneficial for the implementation of our CA problem:

- Strict feasibility of bounds: at every iteration, the implemented SQP algorithm never presents a solution that is out of bounds.
- Robustness to NaN or Inf results: In case a step is taken which results in an invalid evaluation of the cost function, the algorithm tries to take a smaller step with the aim of computing a valid solution.

4 Implementation of the CA Methods on the Dual-Axis Tilting Rotor Quad-Plane

In this section we will derive the INDI control framework for the dual-axis tilting rotor quad-plane in Fig. 1. The vehicle

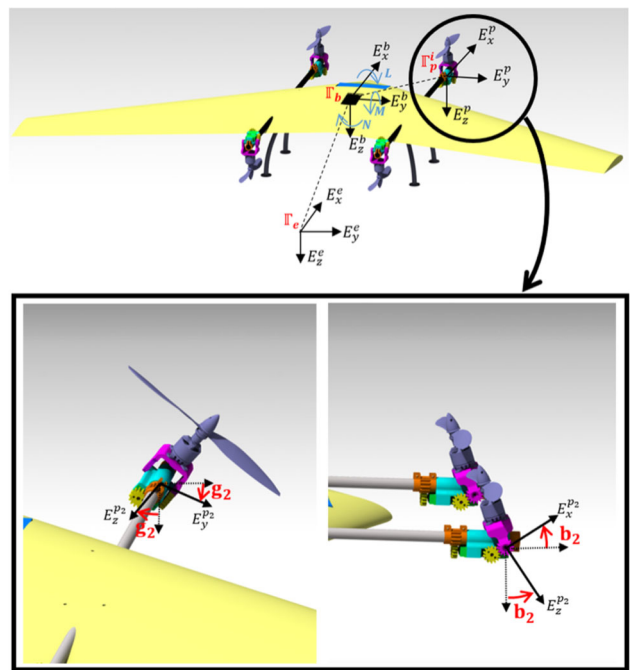


Fig. 4 Overview of the Earth, Body and Propeller reference frames and rotor tilting angles

was presented in our previous work [1] and has a total of 12 actuators including 8 servo-rotors. The disposition and notation of the actuators can be seen in Figs. 3 and 4. The number actuators gives the vehicle a full 6 Degrees Of Freedom (DOF) and makes it overactuated. Furthermore, the presence of tiltable rotors gives the aircraft highly non-linear actuator dynamics. All these characteristics give the aircraft a high potential maneuverability but at the same time make the CA problem particularly complex to solve. We will use the vehicle platform as a test bed to compare the different previously presented linear and non-linear CA methods (PIU, WLS and Nonlinear).

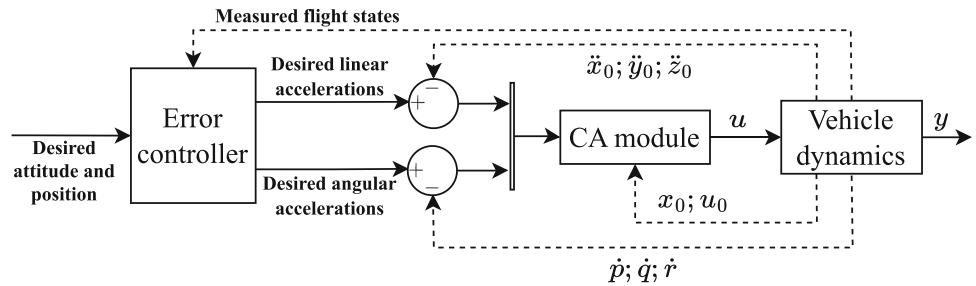
The first step for the implementation of the presented CA methods on the flying vehicle is the derivation of the vehicle EOM. Once derived, the EOM can be replaced by the generic nonlinear system considered in Eq. 2.

4.1 Equation of Motion Derivation

For the sake of brevity, we defer to the appendix for the complete derivation of the equations of motion, the definition of the reference systems and the assumptions made. The equations of motion for the dual axis tilt rotor quad-plane are as follows:

$$\begin{cases} \ddot{P}_e = \frac{1}{m} (F^p + F^a) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1} (-\omega \times I_b \omega + M^t + M^d + M^i + M^p + M^a + M^r + M^{tilt}) \end{cases}, \quad (10)$$

Fig. 5 INDI control diagram for the dual-axis tilting rotor quad-plane in hovering configuration. In the diagram, the dashed lines represent variables estimated through sensor (for the vehicle state) or actuator model (for the current actuator state)



where \ddot{P}_e are the linear acceleration in the earth reference frame and $\dot{\omega}$ represent the body rates derivative. The forces vectors F^a and F^p represent respectively the aerodynamics forces and the thrust produced by the propellers, expressed in the earth reference frame. As for the moments equilibrium, the moment term M^t represents the torque generated by the rotors due to the propeller thrust, M^d represents the torque generated by the rotors due to the propeller drag, M^i represents the torque induced by the i -th propeller inertia due to the motor rotational rate change $\dot{\Omega}_i$. The term M^p represents the torque generated by the rotor precession term due to the tilting rotation, M^r models the inertial term of the rotors due to the vehicle rates and M^{ilt} represents the torque generated by the rotor inertia due to the tilting rotation. A more detailed explanation of each element of Eq. 10 is covered in the [Appendix](#).

linear and angular acceleration. A representation of the error controller scheme is depicted in Fig. 6.

As for the pseudo-control vector, it is composed of the following desired linear and angular accelerations, expressed in the earth reference frame Γ_e :

$$v = (\ddot{x}_d \ \ddot{y}_d \ \ddot{z}_d \ \dot{p}_d \ \dot{q}_d \ \dot{r}_d)^T \tag{11}$$

For the control input vector, it is constituted of a total of 12 elements containing both the motor rotational speed and the rotor tilting angles:

$$u = (\Omega_1 \ \Omega_2 \ \Omega_3 \ \Omega_4 \ b_1 \ b_2 \ b_3 \ b_4 \ g_1 \ g_2 \ g_3 \ g_4)^T \tag{12}$$

4.2 INDI Framework for the Dual-Axis Tilting Rotor Quad-Plane

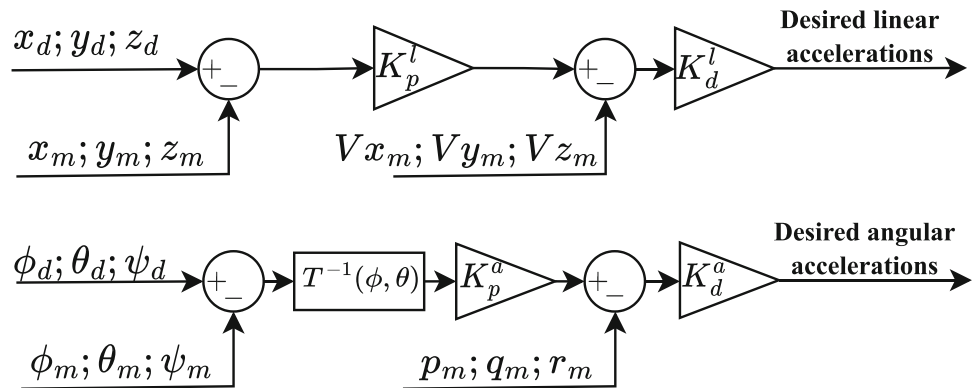
For a vehicle such as the dual-axis tilting rotor quad-plane, the control module scheme is slightly different from the one of a conventional under-actuated vehicle. In the hovering configuration, the dual-axis tilting rotor quad-plane has full 6 DOF control. Therefore, in this flight configuration, an outer loop is not required and a single control loop can be employed to directly control attitude and position. The resulting INDI control scheme is presented in Fig. 5, where a linear error controller is employed for the generation of the desired

4.2.1 Derivation of the Effectiveness Matrix for the Linearized CA Methods

For the PIU and the WLS control allocation methods, an estimate of the linear actuator effectiveness of the vehicle is required. As described in the previous section, the estimation of the linear control effectiveness matrix will be evaluated in the current state x_0 and current control input u_0 and then inverted to determine the control input set.

The effectiveness matrix is composed of the partial derivative of the EOM in Eq. 10 with respect to the control

Fig. 6 Error Controller for the 6 DOF INDI control scheme. The subscript m indicates measured values while the subscript d represents the desired variables



input array defined in Eq. 12, for a total matrix size of 6x12:

$$B = \frac{\partial(\ddot{P}_e ; \dot{\omega})}{\partial u} = \begin{pmatrix} \frac{\partial \ddot{x}}{\partial \Omega_1} & \frac{\partial \ddot{x}}{\partial \Omega_2} & \frac{\partial \ddot{x}}{\partial \Omega_3} & \frac{\partial \ddot{x}}{\partial \Omega_4} & \frac{\partial \ddot{x}}{\partial b_1} & \frac{\partial \ddot{x}}{\partial b_2} & \frac{\partial \ddot{x}}{\partial b_3} & \frac{\partial \ddot{x}}{\partial b_4} & \frac{\partial \ddot{x}}{\partial g_1} & \frac{\partial \ddot{x}}{\partial g_2} & \frac{\partial \ddot{x}}{\partial g_3} & \frac{\partial \ddot{x}}{\partial g_4} \\ \frac{\partial \ddot{y}}{\partial \Omega_1} & \frac{\partial \ddot{y}}{\partial \Omega_2} & \frac{\partial \ddot{y}}{\partial \Omega_3} & \frac{\partial \ddot{y}}{\partial \Omega_4} & \frac{\partial \ddot{y}}{\partial b_1} & \frac{\partial \ddot{y}}{\partial b_2} & \frac{\partial \ddot{y}}{\partial b_3} & \frac{\partial \ddot{y}}{\partial b_4} & \frac{\partial \ddot{y}}{\partial g_1} & \frac{\partial \ddot{y}}{\partial g_2} & \frac{\partial \ddot{y}}{\partial g_3} & \frac{\partial \ddot{y}}{\partial g_4} \\ \frac{\partial \ddot{z}}{\partial \Omega_1} & \frac{\partial \ddot{z}}{\partial \Omega_2} & \frac{\partial \ddot{z}}{\partial \Omega_3} & \frac{\partial \ddot{z}}{\partial \Omega_4} & \frac{\partial \ddot{z}}{\partial b_1} & \frac{\partial \ddot{z}}{\partial b_2} & \frac{\partial \ddot{z}}{\partial b_3} & \frac{\partial \ddot{z}}{\partial b_4} & \frac{\partial \ddot{z}}{\partial g_1} & \frac{\partial \ddot{z}}{\partial g_2} & \frac{\partial \ddot{z}}{\partial g_3} & \frac{\partial \ddot{z}}{\partial g_4} \\ \frac{\partial \dot{\rho}}{\partial \Omega_1} & \frac{\partial \dot{\rho}}{\partial \Omega_2} & \frac{\partial \dot{\rho}}{\partial \Omega_3} & \frac{\partial \dot{\rho}}{\partial \Omega_4} & \frac{\partial \dot{\rho}}{\partial b_1} & \frac{\partial \dot{\rho}}{\partial b_2} & \frac{\partial \dot{\rho}}{\partial b_3} & \frac{\partial \dot{\rho}}{\partial b_4} & \frac{\partial \dot{\rho}}{\partial g_1} & \frac{\partial \dot{\rho}}{\partial g_2} & \frac{\partial \dot{\rho}}{\partial g_3} & \frac{\partial \dot{\rho}}{\partial g_4} \\ \frac{\partial \dot{q}}{\partial \Omega_1} & \frac{\partial \dot{q}}{\partial \Omega_2} & \frac{\partial \dot{q}}{\partial \Omega_3} & \frac{\partial \dot{q}}{\partial \Omega_4} & \frac{\partial \dot{q}}{\partial b_1} & \frac{\partial \dot{q}}{\partial b_2} & \frac{\partial \dot{q}}{\partial b_3} & \frac{\partial \dot{q}}{\partial b_4} & \frac{\partial \dot{q}}{\partial g_1} & \frac{\partial \dot{q}}{\partial g_2} & \frac{\partial \dot{q}}{\partial g_3} & \frac{\partial \dot{q}}{\partial g_4} \\ \frac{\partial \dot{r}}{\partial \Omega_1} & \frac{\partial \dot{r}}{\partial \Omega_2} & \frac{\partial \dot{r}}{\partial \Omega_3} & \frac{\partial \dot{r}}{\partial \Omega_4} & \frac{\partial \dot{r}}{\partial b_1} & \frac{\partial \dot{r}}{\partial b_2} & \frac{\partial \dot{r}}{\partial b_3} & \frac{\partial \dot{r}}{\partial b_4} & \frac{\partial \dot{r}}{\partial g_1} & \frac{\partial \dot{r}}{\partial g_2} & \frac{\partial \dot{r}}{\partial g_3} & \frac{\partial \dot{r}}{\partial g_4} \end{pmatrix} \quad (13)$$

For the determination of the solution, both the WLS and the PIU have to invert the constant effectiveness matrix in Eq. 13. Unfortunately, for some current actuator values and states, the effectiveness matrix may contain small elements. This means that the associated actuator solution will be far away from the current actuator state, invalidating the linearization. The PIU and WLS CA algorithms have no information about the system’s nonlinearities and therefore have to wait for the actuator movement to realize that the computed solution does not give the desired result.

This behavior is clearly visible by running the WLS CA algorithm with a desired vertical acceleration and all motors slightly tilted outward, as depicted in Fig. 7:

$$\begin{aligned} v &= (0 \ 0 \ -10 \ 0 \ 0 \ 0)^T \\ u_0 &= (700 \ 700 \ 700 \ 700 \\ &\quad 0 \ 0 \ 0 \ 0 \\ &\quad -.1 \ .1 \ .1 \ -.1)^T. \end{aligned} \quad (14)$$

Due to the linearization, the effectiveness matrix calculated with this actuator state contains some non-zero elements for the terms $\frac{\partial \ddot{z}}{\partial g_i}$. This means that the CA algorithm may use the lateral tilting to control the vertical acceleration without increasing the RPM of the motors, which would lead to an increase in the cost according to Eq. 8. Unfortunately, the effectiveness matrix only contains a coefficient and does not have any information of the nonlinearities of the platform actuators. This means that the WLS CA algorithm will

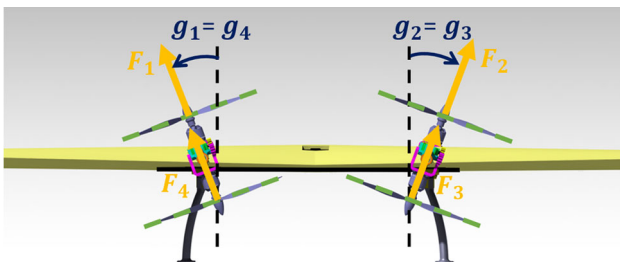


Fig. 7 Initial actuator state defined in Eq. 14

eventually compute a big lateral tilting angular change as an alternative to increasing the motor power. The actuator solution computed by the WLS CA algorithm, for the problem conditions in Eq. 14 is reported in Eq. 15 and represented in Fig. 8.

$$u_s = (950 \ 950 \ 950 \ 950 \\ 0 \ 0 \ 0 \ 0 \\ .78 \ -.78 \ -.78 \ .78)^T. \quad (15)$$

By comparing the computed solution with the saturation points of the actuators in Table 2, it is visible that the proposed WLS actuator solution saturates both the motor rotational speed and the lateral tilting angle in an attempt to generate a vertical acceleration. This condition is clearly non-optimal, as a zero tilting angle would yield more vertical acceleration, and the controller will detect the ineffectiveness of the solution only once the tilting actuator will move. This will eventually trigger permanent oscillations in the angular solution computed by the WLS CA algorithm.

For the same scenario, the PIU CA algorithm computes an unfeasible actuator solution, with a motor rotational speed 1.2 times the saturation point and with the lateral tilt angle on a non-neutral condition:

$$u_s = (1161 \ 1161 \ 1161 \ 1161 \\ 0 \ 0 \ 0 \ 0 \\ .11 \ -.11 \ -.11 \ .11)^T. \quad (16)$$

We can compare the results computed by WLS and PIU with the one computed by the Nonlinear CA algorithm:

$$u_s = (950 \ 950 \ 950 \ 950 \\ 0 \ 0 \ 0 \ 0 \\ 0 \ 0 \ 0 \ 0)^T. \quad (17)$$

In this case, the CA algorithm only saturates the motors, with both azimuth and elevation tilting angles equal to zero.

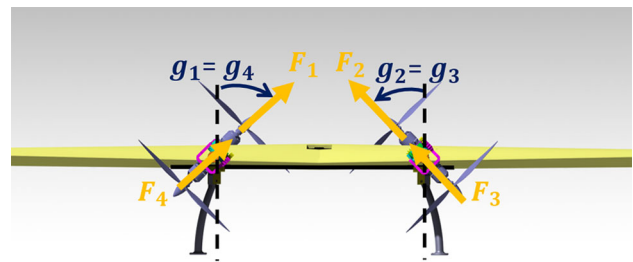


Fig. 8 Actuator solution of the problem defined in Eq. 14 determined using the WLS CA algorithm

4.2.2 Simplification of the EOM for the Nonlinear CA Method Formulation

To decrease the computational load of the Nonlinear CA method, only the main terms of the EOM of Eq. 10 are considered. The vehicle dynamics considered for the nonlinear optimization process in Eq. 9 is the following:

$$\begin{cases} \ddot{P}_e = \frac{1}{m} (F_p + F_a^{wb}) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1} (-\omega \times I_b \omega + M_p^T + M_p^D + M_a^{wb}) \end{cases} \quad (18)$$

The secondary terms of the EOM not considered for the inversion will be treated as unmodeled dynamics and handled by the error controller.

Note that this formulation of the EOM also considers the aerodynamic effects not considered in the linearized actuator effectiveness matrix formulated in Eq. 13. It is now possible to insert the simplified EOM expression of Eq. 18 into the cost function derived in Eq. 9 to complete the Nonlinear CA problem formulation.

4.2.3 CA Parameters Choice

For the implementation of the previously introduced CA methods, a few parameters have to be set and described. The parameter in common between all the CA algorithms is the choice of the desired actuator state. For the tilting system, during the hovering test, the desired state of the tilting angles is set to the vertical condition ($b_i = g_i = 0$). Regarding the motor desired state, it was set to 100 rad/s , equivalent to the minimum applicable value.

For the WLS and Nonlinear CA algorithms, the weighting matrices W_u , W_v and the penalty parameters γ_v and γ_u have to be chosen. Starting with the control input weighting matrix W_u , it is a square diagonal matrix of the same size as the number of actuators (12x12 in that case). Each diagonal element of the matrix assigns a penalty on the usage of the respective actuator and it is located on the secondary objective of the cost function. Ideally, we prefer to prioritize the usage of the tilting system over the motor for a more energy efficient solution. Therefore, we decided to apply a unitary penalty coefficient to the tilting system and a penalty coefficient equal to 3 to the motors. The weighting matrix W_u used for the vehicle then becomes:

$$W_u = \text{diag} (3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1). \quad (19)$$

As for the pseudo-control weighting matrix W_v , as already explained in [18], it should be set in such a way for the problem to firstly prioritize the pitch and roll changes, then the

linear acceleration changes and the yaw angle. This leads to a pseudo-control weighting matrix W_v of this form:

$$W_v = \text{diag} (0.01, 0.01, 0.02, 0.2, 0.2, 0.01). \quad (20)$$

Concerning the penalty parameter γ , we set the value of $1e5$ to γ_v and $1e-5$ to γ_u while for both the Nonlinear and WLS problem formulations, a maximum number of 60 iterations was set and the current actuator state u_0 was used as starting point for the optimization process.

The last parameter that needs to be discussed for the constrained CA problems is the constraint definition. Ideally, the problem constraints are constant at every iteration and equal to the physical actuator limits. However, one may claim that with a limit on the maximum change in control input, the error that would be made by linearizing the system would become smaller. To investigate this, when using the WLS CA algorithm for the resolution of the CA problem, a different choice of the problem constraints will also be tested. The modified constraint, denoted as ‘‘local tilting constraint’’ later in the paper, is of the following form:

$$\begin{pmatrix} \Omega^{\min} \\ \Omega^{\min} \\ \Omega^{\min} \\ \Omega^{\min} \\ \max(b_1^{\min}, b_1^0 - \delta_c) \\ \max(b_2^{\min}, b_2^0 - \delta_c) \\ \max(b_3^{\min}, b_3^0 - \delta_c) \\ \max(b_4^{\min}, b_4^0 - \delta_c) \\ \max(g_1^{\min}, g_1^0 - \delta_c) \\ \max(g_2^{\min}, g_2^0 - \delta_c) \\ \max(g_3^{\min}, g_3^0 - \delta_c) \\ \max(g_4^{\min}, g_4^0 - \delta_c) \end{pmatrix} \leq u \leq \begin{pmatrix} \Omega^{\max} \\ \Omega^{\max} \\ \Omega^{\max} \\ \Omega^{\max} \\ \min(b_1^{\max}, b_1^0 + \delta_c) \\ \min(b_2^{\max}, b_2^0 + \delta_c) \\ \min(b_3^{\max}, b_3^0 + \delta_c) \\ \min(b_4^{\max}, b_4^0 + \delta_c) \\ \min(g_1^{\max}, g_1^0 + \delta_c) \\ \min(g_2^{\max}, g_2^0 + \delta_c) \\ \min(g_3^{\max}, g_3^0 + \delta_c) \\ \min(g_4^{\max}, g_4^0 + \delta_c) \end{pmatrix} \quad (21)$$

Where the superscripts ‘‘0’’ indicates the current value, the superscripts max and min indicate respectively the maximum and minimum physical value of the actuator and δ_c represents the constant angular region of constraint. The maximum and minimum actuator values for the dual-axis tilting rotor quadplane are reported in Table 2.

4.2.4 Actuator Model Identification

The previously proposed CA methods all rely on an incremental control law and therefore require an estimate of the current actuator state to work. Accurate modeling of the actuator dynamics is also needed for the vehicle simulation.

For the motor propeller actuation system, we determined its dynamics through bench tests using an external RPM sensor. The result of a step test campaign on the motor propeller system is shown in Fig. 9A. In the same plot, the motor rotational speed evolution was also fitted with a first and a

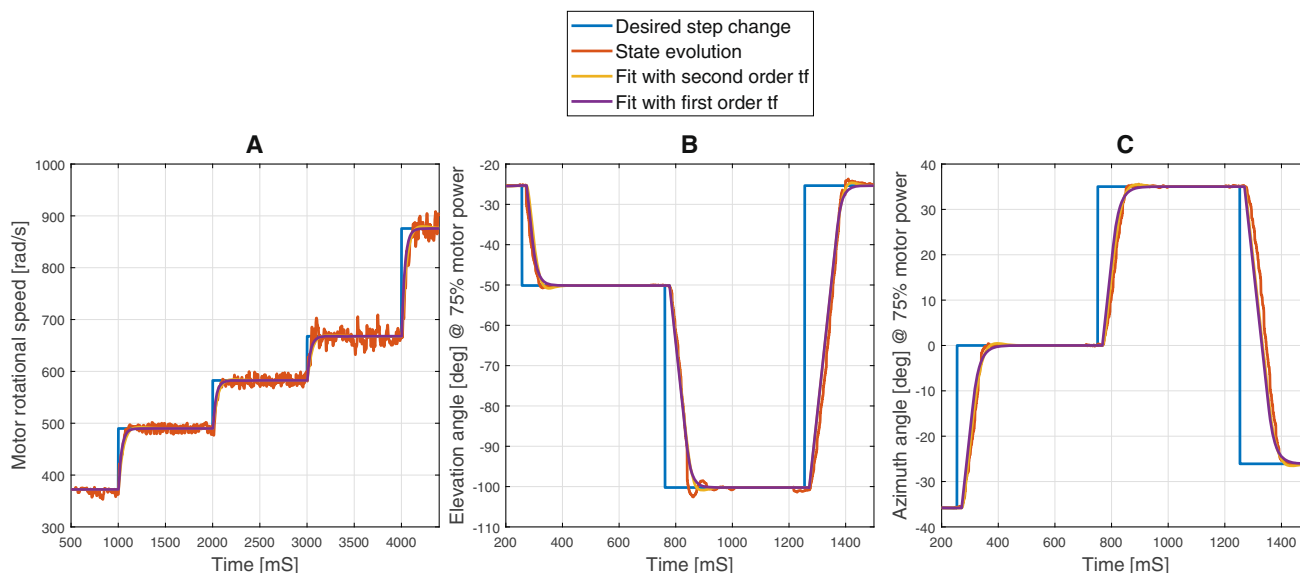


Fig. 9 Step test actuator response for the identification of the dynamics. The left plot (A) is the motor step test, the central plot (B) is the azimuth tilting angle step test and the right plot (C) is the elevation tilting angle step test. The blue curve represents the desired step change and the red

curve reports the variable evolution. The purple and yellow curves are a fit of the variable evolution using respectively a first and a second order discrete transfer function

second order discrete transfer function. As visible, the motor-propeller system can be modeled using a first order discrete transfer function. The analytical expression of the first order discrete transfer function associated with the motor dynamics is the following:

$$H_{motor}(z) = \frac{0.05824z^{-1}}{1 - 0.9418z^{-1}} \tag{22}$$

The sampling frequency used to determine the transfer function in Eq. 22 is 500 Hz. This is the update frequency of the main flight computer running onboard the vehicle.

Concerning the identification of the tilting dynamics, the approach of [1] was used. An IMU was mounted solidly to the tilting structure and angle step changes were induced on the tilting servos. The dynamics is then recorded and analyzed to determine the tilting rotor dynamics at different motor speeds.

The motor spinning power chosen to identify the motor dynamics discrete transfer function was 75%. The tilting evolution for the azimuth and elevation rotor angle step test is displayed in Fig. 9B and C. The tilting angle evolution was fitted with a first and second order discrete transfer function featuring also a rate saturation limit. In this case, the second order transfer function gives a slightly better representa-

tion of the tilting dynamics. The analytical expressions of second order discrete transfer function used for the azimuth and elevation tilting angle dynamics are the following:

$$H_{az}(z) = (z^{-6}) \frac{0.00386z^{-1} + 0.003679z^{-2}}{1 - 1.858z^{-1} + 0.8659z^{-2}} \tag{23}$$

$$R_l^{az} = 9.95 \frac{rad}{s}$$

$$H_{el}(z) = (z^{-6}) \frac{0.006779z^{-1} + 0.006384z^{-2}}{1 - 1.822z^{-1} + 0.8353z^{-2}} \tag{24}$$

$$R_l^{el} = 11.34 \frac{rad}{s}$$

Where R_l represents the rate saturation limit value of the tilting system. The same sampling frequency of 500 Hz was used for the determination of the discrete transfer functions in Eqs. 23 and 24. For completeness, in Table 1, the characteristics of the equivalent continuous time system associated to the actuators are displayed.

4.2.5 Actuator Constraint and Problem Normalization

For a complete CA problem formulation, two more steps are required. The first step is the determination of the physical

Table 1 Characteristics of the equivalent continuous time system associated to the actuators

Actuator	Corner frequency ω_c	Damping ratio ζ	Rate limit
Motor	30 rad/s	–	–
Tilt elevation	60 rad/s	1.5	11.34 rad/s
Tilt azimuth	45 rad/s	1.6	9.95 rad/s

Table 2 Actuators physical limit for the CA problem

	Max value	Min value
Motor rotational speed Ω_i	950 rad/s	100 rad/s
Azimuth tilting angle g_i	45 deg	-45 deg
Elevation tilting angle b_i	25 deg	-90 deg

limits of the actuators on the vehicle. The second step is to use the total travel of the actuators to normalize the CA problem, in such a way to equally compare the actuator effects during the CA resolution process. The physical limits of the actuators are displayed in Table 2.

With the maximum and minimum actuator limits determined, it is possible to scale the control input vector presented in Eq. 12 for the normalization of the CA problem:

$$u_n = \begin{pmatrix} \frac{\Omega_1}{G_m} & \frac{\Omega_2}{G_m} & \frac{\Omega_3}{G_m} & \frac{\Omega_4}{G_m} \\ \frac{b_1}{G_{el}} & \frac{b_2}{G_{el}} & \frac{b_3}{G_{el}} & \frac{b_4}{G_{el}} \\ \frac{g_1}{G_{az}} & \frac{g_2}{G_{az}} & \frac{g_3}{G_{az}} & \frac{g_4}{G_{az}} \end{pmatrix}. \quad (25)$$

Where the scaling factors G_m , G_{el} and G_{az} are displayed in Table 3. Using the normalized control input array u_n presented in Eq. 25, the overall actuator travel will be the same for all the actuators and will be equal to 2. For a complete normalization of the CA problem, the maximum, minimum and desired actuator value, the effectiveness matrix B and the EOM were normalized as well.

4.3 Experimental Setup

The implementation of the previously presented CA methods on a real-time setup requires a reasonable amount of computational power that a microcontroller alone cannot provide. In order to solve the computational deficit problem, we decided to distribute the computational load among two boards. The main FCB used is a Pixhawk 4 flight controller running the open-source paparazzi software¹ [21]. The task of the Pixhawk 4 is to interact with the vehicle sensors and instruments, control the actuators and to generate the pseudo-control vector. Once the pseudo-control vector is generated, it is communicated over a UART interface to the Raspberry Pi 4B,² whose task it is to solve the CA problem and communicate the actuator solution back to the Pixhawk 4.

The development of the CA modules operating on the SBC was characterized by a few steps. Initially, the PIU,

¹ <https://wiki.paparazziuav.org/>

² <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>

Table 3 Scaling factor for the normalization of the control input vector

Motor scaling factor G_m	425 rad/s
Azimuth tilting scaling factor G_{az}	0.785 rad
Elevation tilting scaling factor G_{el}	2.705 rad

WLS and Nonlinear CA algorithms were implemented and tested in Simulink. Secondly, using the Simulink Coder support package,³ the C functions implementing the CA methods were generated. Finally, the generated C code of the CA routines were manually optimized, compiled and then deployed on the Raspberry Pi 4B SBC.

For the communication between the Raspberry Pi 4 and the Pixhawk, a serial communication was employed. In order to maximize the computational speed of the CA solver, the serial communication on the raspberry Pi was implemented on a different thread. As for the rest of the components employed for the vehicle hardware such as the type of tilting servos and motors used, the reader can refer to [1].

4.4 Simulation Results

With the knowledge of the vehicle dynamics, the actuator characteristics, and a proper identification of the control framework, it was possible to set up a simulation environment in Simulink. The simulation outcome could then be compared to the flight test result for the model validation.

A mixed maneuver engaging attitude, altitude and position changes was chosen. Initially, a pitch and roll angle of 20 degrees are commanded. Then, the vehicle is asked to track a desired altitude and position step change.

As a reference for the model validation, only the Nonlinear CA algorithm was used to compare the simulation outcome with the flight test results. In Fig. 10 the attitude and position evolution between flight test and simulation is compared. As for the actuator evolution, for the sake of simplicity, only the tilting angles and motor power of the rotor number 2 are compared in Fig. 11.

4.4.1 Comments

The flight test results and the simulation outcome are matching, thus validating the modeling of the vehicle and actuator dynamics. The only visible mismatch happens around time 20 seconds, during the climbing phase of the maneuver. As also confirmed by the logs, this mismatch may have been caused by a sudden voltage drop related to a spike in power demand of the motors. This condition also influences the motor rotational speed response of the flight test which presents a damped oscillation not present in the simulation.

³ <https://www.mathworks.com/products/simulink-coder.html>

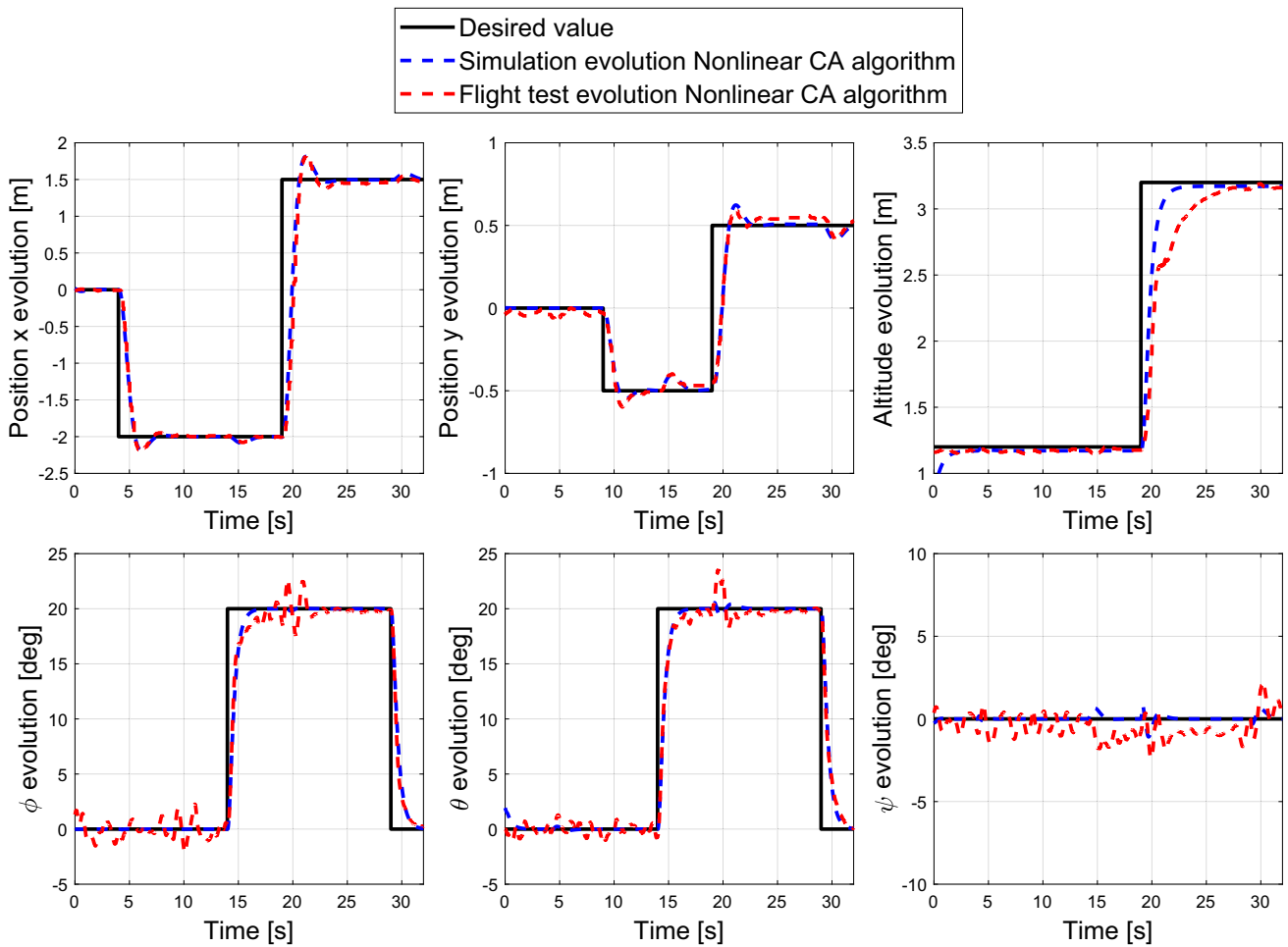


Fig. 10 Position and attitude evolution comparison between simulation and flight test using the Nonlinear CA algorithm

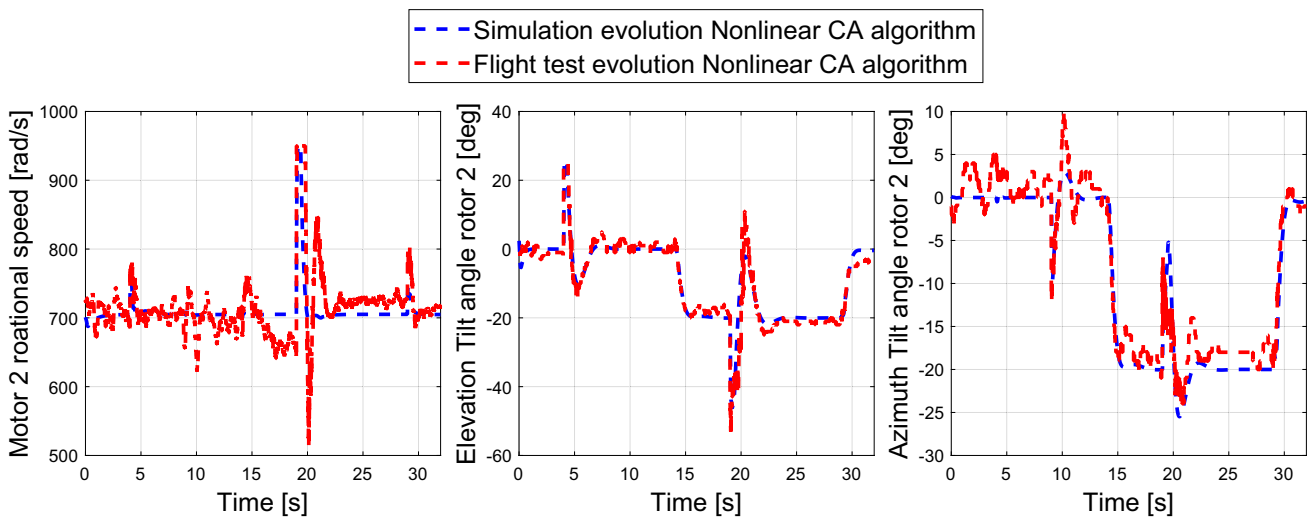


Fig. 11 Tilting angles and motor rotational speed evolution of rotor number 2 computed by the Nonlinear CA algorithm. The dashed blue curve is the simulated actuator evolution while the dashed red curve is the actuator evolution experienced during the flight test

4.5 Flight Test Results

To assess the behavior of the different CA methods on the flying vehicle, the same maneuver proposed for the simulation was programmed on board the drone. The flight test was carried out in the TUDelft Cyberzoo flight arena and the flying vehicle was always secured with a rope from the arena ceiling. The rope was manually handled in such a way to minimize the interference with the flight.

For an accurate estimation of the vehicle position and orientation, an Inertial Navigation System (INS) composed of an OptiTrack optical motion system, a 3-axis accelerometer, a 3-axis magnetometer and a 3-axis gyroscope was employed. Since the controller is designed to track accelerations in the earth reference frame Γ_e , an estimation of the inertial accelerations of the vehicle in the earth reference frame is needed. This estimation is obtained by projecting the inertial body accelerations recorded by the accelerometer to the earth reference frame through Eq. 26.


An overview of the different CA algorithms tested on the flying vehicle are summarized in Table 4 while in Fig. 12, the attitude and position tracking for the successful and

partially successful flights are displayed. The partially successful flight curves are truncated at the time the flight test was stopped. In Fig. 13, the actuators evolution of the rotor number 2 are shown. As done for the analysis of the simulation results, the rotor number 2 was chosen as reference for the analysis of the solution computed by the different CA algorithms during the flight tests. Finally, in Fig. 14, the plot containing the run-time of the three main CA algorithms is displayed. The run-time was defined as the time needed by the CA algorithm running on the Raspberry pi to compute the actuator solution. This value defines the actual update period of the desired actuator value and it is generally not constant.

4.5.1 Comments

Several interesting observations can be made by looking at the flight test results for the tested CA methods. Both the Nonlinear and WLS CA methods provide a valid actuator solution at every time step. These methods are constrained optimization processes and therefore always provide an actuator solution within the actuator limits. On the other hand,

Table 4 Overview of the different CA algorithms tested on the flying vehicle

Algorithm tested	Flight test outcome	Comments	Average run-time	Video
Nonlinear CA	Successful	Smooth actuator response, best position and attitude tracking.	2959 uS	
WLS CA	Failed	Permanent and uncontrolled oscillations in the actuator solution. Not able to track neither the desired position nor the attitude.	–	
WLS CA with identity control input weighting matrix W_u	Partially Successful	Motor saturation due to yaw tracking. Oscillations in the actuator solution once the rapid position change is commanded.	1711 uS	
WLS CA with local tilting constraint and $\delta_c = 10$ deg	Successful	Permanent bounded oscillations in the tilting actuator solution. Position and attitude tracking comparable to the Nonlinear CA.	4690 uS	
WLS CA with local tilting constraint and $\delta_c = 5$ deg	Partially successful	Permanent bounded oscillations in the tilting actuator solution. Overshoot in the position tracking.	5066 uS	
PIU CA	Partially successful	Motor saturation due to yaw tracking. Oscillations in the actuator solution once the rapid position change is commanded.	11 uS	

Video link: https://youtu.be/OwoZQ9u_5EE

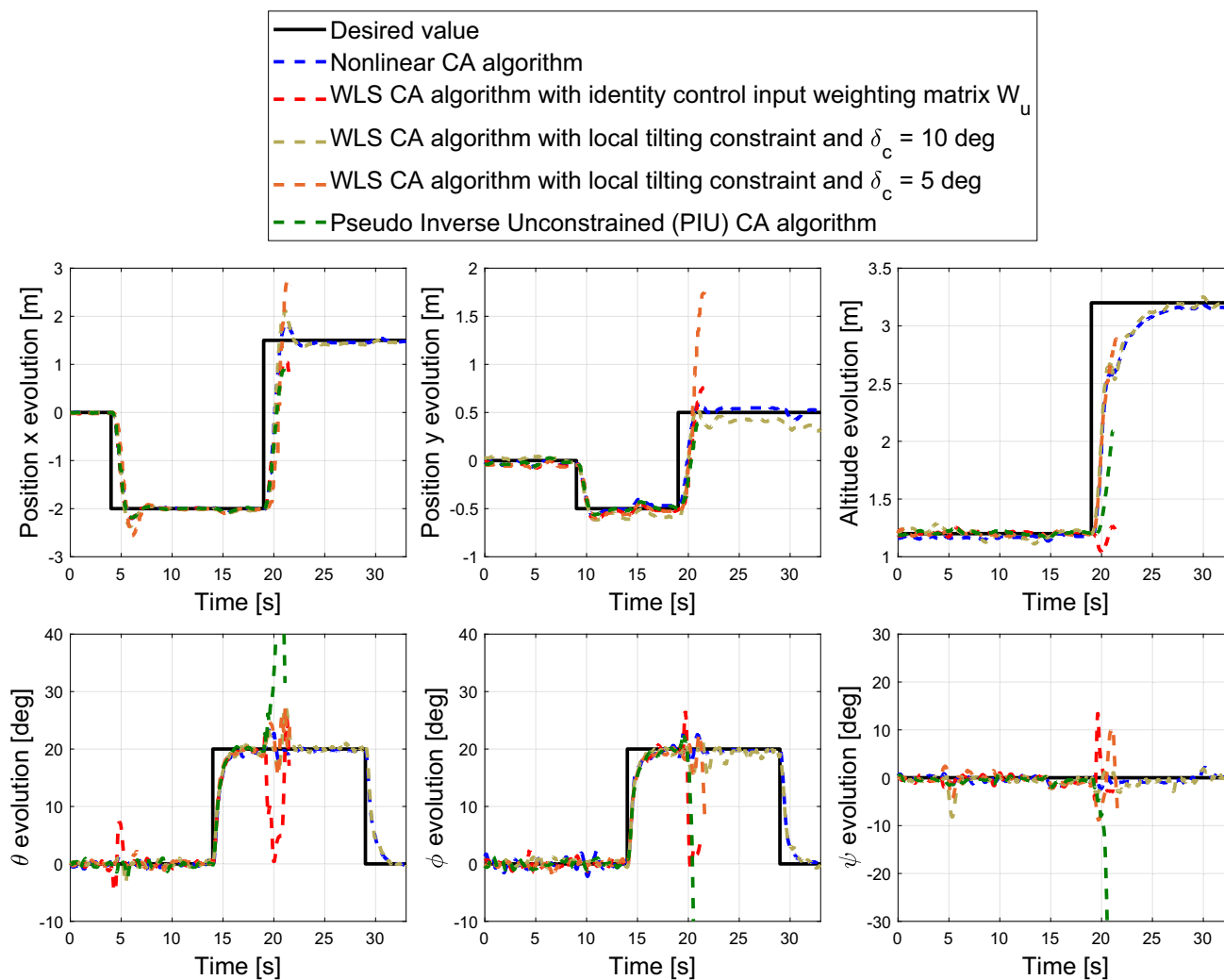


Fig. 12 Position and attitude tracking using different CA algorithms for the same flight test maneuver

the PIU CA method, being an unconstrained optimization problem, does not account for actuator saturation, providing a solution that is sometimes out of the available actuator range. This behavior is clearly visible at time 19 seconds, where the step change in altitude and position is requested. During that rapid maneuver, the PIU CA algorithm computes a motor speed solution way above the saturation point. This means that the controller tries to achieve the desired acceleration set by clipping a value to an already saturated actuator state, leading to a visible loss of attitude tracking.

Another major consideration has to be done when applying equal weight in the control input matrix W_u , as done for the PIU CA algorithm and the WLS CA algorithm with unitary W_u . With those two CA methods, there is no penalty on the usage of the motor over the tilting system. This means that it is mathematically more convenient to use motor power differential instead of rotor tilt to achieve yaw changes. This condition is observed around time 15 seconds, where the

vehicle corrects a yaw error by decreasing the motor power of rotor 2 and 3 while increasing the motor power of rotors 1 and 4. Unfortunately, this practice leads to the saturation of rotors 1 and 4, therefore reducing the maneuverability of the vehicle.

Being able to prioritize the usage of the tilting system over motor power is a very important feature for the determination of an energy efficient solution. However, while it was possible to fly the vehicle using the Nonlinear CA method with the control input weighting matrix W_u in Eq. 19, it was impossible to do so with the WLS CA algorithm. Assigning penalty on the motor power highlights the limitations of the linearized approximation of the actuator effectiveness. Under this condition, the WLS CA method tends to allocate the control objective to the tilting system as much as possible, leading to rapidly oscillating tilting angles solution coupled with an inaccurate motor power solution. To better understand the reasoning behind the tilting oscillations, the reader

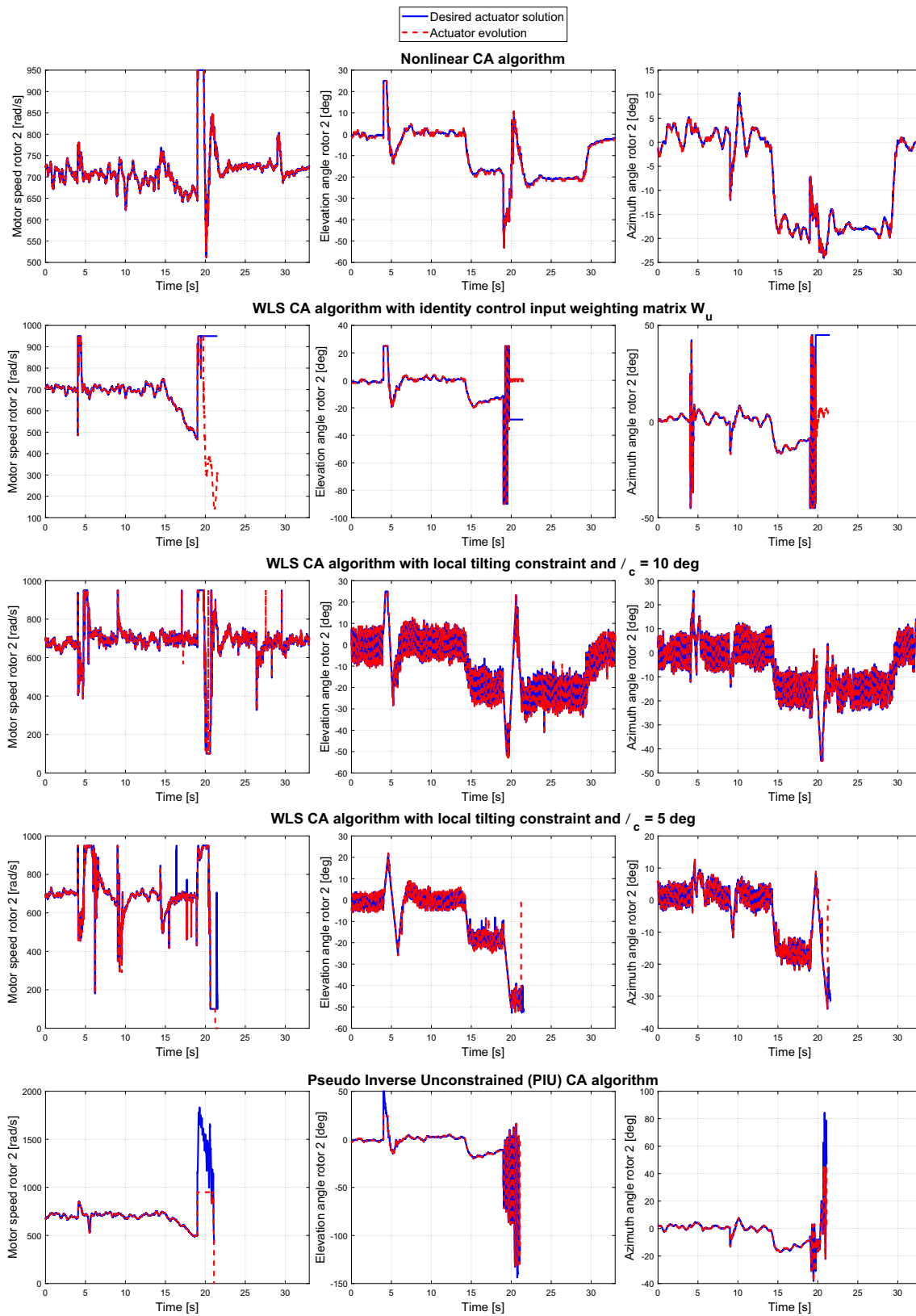


Fig. 13 Tilting angles and motor rotational speed evolution of rotor number 2 computed by the different CA algorithms for the same flight test maneuver

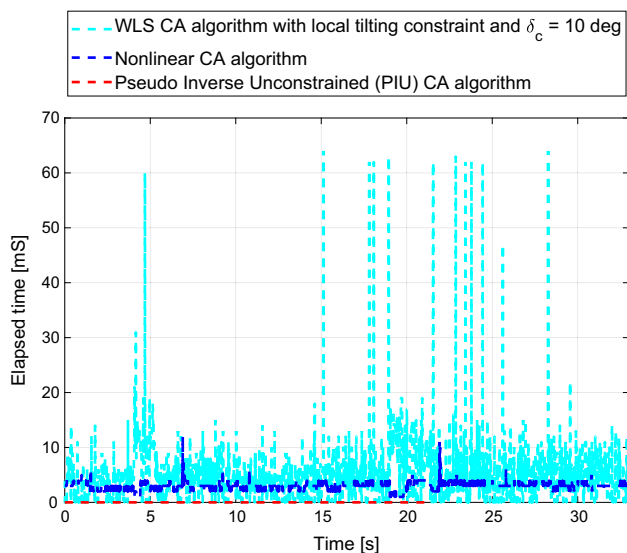


Fig. 14 Run-time of the three main CA algorithms during the flight test maneuver

can refer to Section 4.2.1, where this condition was analyzed in detail.

It is possible to partially bound the oscillatory actuator solution by applying a local constraint to the tilting solution in the WLS CA problem formulation, as presented in Eq. 21. However, this condition induces a delay in the acceleration tracking proportional to the speed of the actuators. The delay in the acceleration tracking leads to an overshoot in the position and attitude tracking of the vehicle. When a $\delta_c = 5$ deg is applied, the overshoot in the position tracking becomes so pronounced that the flight test had to be stopped prematurely to avoid a crash with the lateral net.

Concerning the proposed Nonlinear CA method, it always computes a smooth, valid and effective actuator solution, correctly prioritizing the desired control input and actuator use according to the provided weighting matrices W_v and W_u . Among the three analyzed algorithms, the proposed Nonlinear CA is the Control Allocation strategy providing the best tracking for both position and attitude, as visible from Fig. 12.

As for the computational load of the different algorithms, the run-time needed by the algorithms to compute the actuator solution is displayed in Fig. 14. The WLS CA algorithm is the most computational-intensive algorithm with an average of 4.7 milliseconds needed for the determination of the actuator solution. The large run-time needed by the WLS algorithm for the determination of the solution is mainly associated with the frequent actuator saturation condition experienced during the flight. In case of actuator saturation, the iterative active-set optimization process run by the WLS algorithm needs multiple iterations to determine the solution, leading to an increase in execution time. The second most computa-

tionally intensive algorithm is the Nonlinear CA algorithm, with an average run-time of 2.9 milliseconds. Interestingly, the Nonlinear CA algorithm, as opposed to the WLS CA, experiences a reduced run-time when a set of actuators is saturated. Finally, the computational time of the PIU CA algorithm is the smallest one, requiring an average of 11 microseconds to compute, and it is more or less constant during the whole flight.

5 Limitations of the Proposed Nonlinear CA Method

There are two main limitations concerning the applicability of the proposed Nonlinear CA on flying vehicles. The first limitation has to do with the computational power required by the Nonlinear CA algorithm to run. Depending on the number of actuators and the complexity of the EOM, the cost function formulation and its gradient might increase in complexity, leading to a longer run-time needed by the SQP algorithm for the computation of the actuator solution. A longer run-time will reduce the phase margin of the controller if it becomes excessive. A similar issue can be encountered if a less powerful SBC is used to run the Nonlinear CA algorithm. Note that the nonlinear CA algorithm determines the actuator solution iteratively. Therefore, there is a direct relation between the number of iterations and the accuracy of the computed actuator solution. At the same time, the number of iterations also affects the overall run-time and computational load of the algorithm. The choice of the number of iterations is therefore a key parameter that must be chosen as a compromise between the computational load and the accuracy of the solution.

The second limitation affecting the Nonlinear optimization process is to ensure to find an actuator solution which is not in a local minimum of the cost function. The easiest way to prove the absence of local minima in the cost function, is to ensure global convexity of the function. Unfortunately, the complexity of the proposed cost function makes it difficult to analytically prove the global convexity. Even if the convexity of the cost function cannot be analytically determined, it is still possible to investigate if starting the optimization process using a different initial condition would change the cost function's final value. This gives a rough idea about the shape of the cost function and the presence of local minima.

In order to check if the cost function contains local minima, a monte-carlo simulation was performed varying the initial actuator set of the optimizer u_{init} , for different acceleration set, actuator condition and vehicle states. A total of 7000 tests were performed each time using 300 different initial actuator sets, for a total of more than 2 million optimization runs. A uniform statistical distribution was used to determine states and control objective values at every itera-

tion. The constant and randomized variables used during the statistical analysis are displayed in Table 5. The choice of the variable ranges was set in such a way as to reproduce extreme conditions for the vehicle state, actuator state and for the control objective.

5.1 Statistical Analysis Results

Out of the 7000 tests performed during the simulation, 98.6% of the time, the cost function value determined using $u_{init} = u_0$ is within 10% of the minimum cost function value determined over the 300 runs with randomized u_{init} . Only 93 tests out of 7000 provided a final cost function mismatch of more than 10%, highlighting a potential local minimum condition.

In order to evaluate the primary objective term of the cost function, the residual norm of the solution determined using a different initial actuator set was compared with the solution providing the minimum residual norm for that test. The residual vector is defined as the difference between the desired accelerations and the accelerations achieved by the determined actuator solution. The norm of this vector will therefore be composed by a sum of angular and linear accelerations with units rad/s^2 and m/s^2 .

In Fig. 15, the residual norm of the "minimum norm actuator solution" is compared with the solution computed using different u_{init} . We define as "minimum norm actuator solution", the actuator solution providing minimum residual norm out of the 300 randomized u_{init} runs. The blue histogram in Fig. 15 represents the residual norm error between the "minimum norm actuator solution" and the "maximum norm actuator solution" of the 300 optimization runs. In the same figure, the red histogram is the residual error between the "minimum norm actuator solution" and the solution coming from an arbitrary initial actuator set. The arbitrary

actuator set was chosen as the value coming from the first of the 300 random optimization runs. Finally, the red histogram shows the residual norm error between the "minimum norm actuator solution" and the solution computed using the current actuator state as initial point for the optimizer.

Out of the 7000 tests performed, the maximum residual norm between the "minimum norm actuator solution" and the solution computed using $u_{init} = u_0$ is 3.5. This value increases to 7.2 when the arbitrary u_{init} is considered and reaches 10.4 when it is compared with the "maximum norm actuator solution". Interestingly, it was observed that in a total of 20 tests, the solution computed using $u_{init} = u_0$ leads to the lowest residual solution.

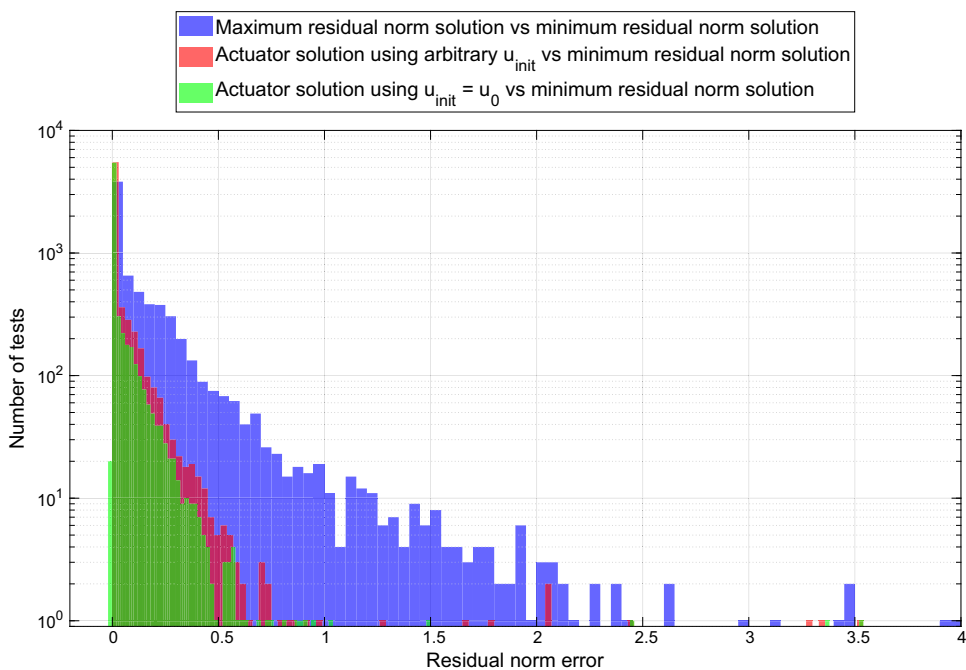
To evaluate the secondary objective term of the cost function, the actuator solution displacement from the current actuator state was also analyzed. A bigger actuator displacement in the solution involves a rapid and sudden change of the actuator state. In certain circumstances we would like to limit the actuator displacement in favor of a slightly bigger residual to avoid sudden jumps in the actuator solution and to achieve a smooth actuator evolution over time. This is particularly important in our case, where the actuators are mainly constituted of servo motors. To evaluate this parameter, the scaled actuator displacement of the solution computed using different u_{init} conditions is shown in Fig. 16. In this figure, the magenta histogram shows the scaled solution displacement from u_0 of the "minimum norm solution", the red histogram represents the scaled solution displacement using the arbitrary initial actuator set while the green histogram is the scaled solution displacement when using the current actuator state as initial value for the optimizer.

Starting the optimization from different initial actuator states also provides different actuator solutions and therefore different residuals. However, the initial actuator state leading to a solution with minimal residual norm is not known in

Table 5 Variables definition for the statistical analysis of the Nonlinear CA solution

	Min value	Max value
Randomized variables		
Motor rotational speed Ω_i	150 rad/s	950 rad/s
Elevation tilting angle b_i	-90 deg	25 deg
Azimuth tilting angle g_i	-45 deg	45 deg
Desired linear acceleration increment	$-5 \frac{m}{s^2}$	$5 \frac{m}{s^2}$
Desired angular acceleration increment	$-5 \frac{rad}{s^2}$	$5 \frac{rad}{s^2}$
Pitch angle θ	-20 deg	20 deg
Roll angle ϕ	-20 deg	20 deg
Forward speed V_x	0 m/s	3 m/s
Constant variables	Value	
Angular rates p, q, r	0	
Lateral and vertical speed V_y, V_z	0	
Yaw angle ψ	0	

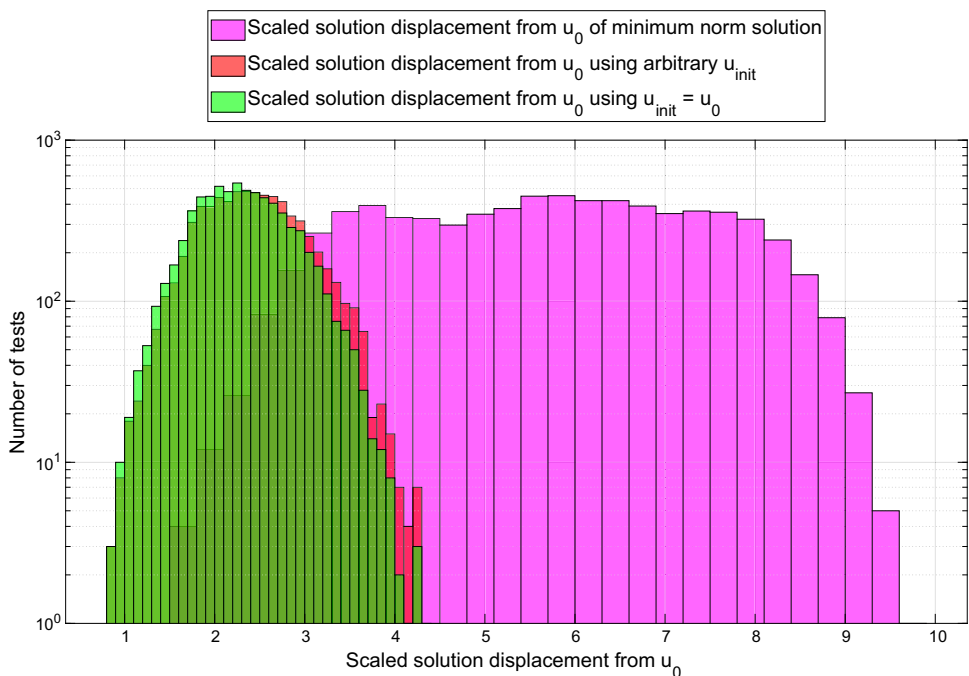
Fig. 15 Residual norm error between the minimum residual norm solution and the solution computed using different u_{init} conditions. Notice that the y-axis is logarithmic and some points are not fully represented. One point of the red histogram and four points of the blue histogram are above the residual norm error of 4. These points are mentioned in the Statistical analysis results Section



advance. The only two plausible options when running the Nonlinear optimization process in real time are either to use an arbitrary actuator state or to use the current actuator state as initial optimization point. The results in Fig. 15 clearly shows that starting the optimizer from the current actuator state provides a lower residual norm than starting it from an arbitrary actuator point.

At the same time, it is clear from Fig. 16 that initializing the Nonlinear CA algorithm from an initial actuator set different from the current one, leads to a solution with higher displacement from u_0 . Therefore, even if by starting the optimization process at the current actuator state we might not be able to find the solution with the lowest norm, we expect to find the sub-optimal solution with minimum actuator dis-

Fig. 16 Scaled displacement of the computed actuator solution from the current actuator value for different u_{init} conditions. Notice that the y-axis is logarithmic. The displacement is defined as the norm of the difference in normalized control input $|u_n - u_{n0}|$, with u_n as defined in Eq. 25



placement. This choice should mitigate sudden jumps in the actuator solution, guaranteeing a smooth actuator evolution over time.

6 Conclusions

In this paper we addressed the Control Allocation (CA) problem of hybrid Unmanned Aerial Vehicle (UAV) with nonlinear effectiveness actuators, such as tilting rotor vehicles.

Through flight tests and simulations we proved that state-of-the-art linearized effectiveness CA algorithms like the Pseudo Inverse Unconstrained (PIU) and Weighted Least Squares (WLS) are not capable of determining a smooth and effective actuator solution for tilting rotor vehicles. We proposed a Nonlinear CA algorithm capable of computing an optimal and smooth actuator solution for such vehicles. Furthermore, the vehicle performance in terms of attitude and position tracking obtained using the Nonlinear CA algorithm outperformed the one obtained using any other tested CA method. It was observed that under high control objective commands or when an uneven control input weighting matrix W_u is used, the solution computed by linearized effectiveness algorithms is characterized by uncontrolled and wide oscillatory tilting angles, coupled with an inaccurate motor power solution. In an attempt to mitigate the oscillatory behavior of the WLS CA algorithm, a local tilting constraint was included in the WLS problem structure. Although potentially beneficial, this condition only bounds the oscillations of the tilting solution at the expense of an added acceleration response delay, inversely proportional to the applied angular region of constraint δ_c .

Concerning possible future development, even if no criticalities were found during the flight tests, a statistical analysis on the cost function used in the Nonlinear CA algorithm revealed the presence of some local minimum points. Future work should focus on this aspect, trying to modify the cost function in such a way to eliminate the presence of local minima in it. Furthermore, the Nonlinear CA algorithm could be extended to the forward flight, with the aim of developing a unified control strategy for hovering, transitioning and forward flight of tilt rotor hybrid vehicles.

Acknowledgements The work was carried out within the Unmanned Valley Project. The authors would like to thank the "Europees Fonds voor Regionale Ontwikkeling(EFRO)" who is founding the Unmanned Valley project under grant code KvW-00168 for the South-Holland region.

Author Contributions all authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Alessandro Mancinelli. The first draft of the manuscript was written by Alessandro Mancinelli and Ewoud J.J. Smeur and all authors

commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability the datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics Approval the work does not require ethics approval since it does not involve human or animal subjects.

Financial interests authors Bart D.W. Remes, Guido C.H.E. De Croon and Ewoud J.J. Smeur declare they have no financial interests. Author Alessandro Mancinelli has received financial support under the form of PhD grant by the "Europees Fonds voor Regionale Ontwikkeling(EFRO)" with grant code KvW-00168.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

List of Acronyms

<i>CA</i>	Control Allocation
<i>VTOL</i>	Vertical Takeoff and Landing
<i>SBC</i>	Single Board Computer
<i>SQP</i>	Sequential Quadratic Programming
<i>EOM</i>	Equations Of Motion
<i>INDI</i>	Incremental Nonlinear Dynamic Inversion
<i>FCB</i>	Flight Control Board
<i>PIU</i>	Pseudo Inverse Unconstrained
<i>WLS</i>	Weighted Least Squares
<i>CG</i>	Center of Gravity
<i>INS</i>	Inertial Navigation System
<i>UAV</i>	Unmanned Aerial Vehicle
<i>IMU</i>	Inertial Measurement Unit
<i>DOF</i>	Degrees Of Freedom

Appendix: Mathematical model of the dual-axis tilting rotor quad-plane

In this section, we will report the EOM derivation for the dual-axis tilting rotor quad-plane as described in our previous paper [1].

A.1 Reference Frames and Notation

Firstly, the rotor disposition and spinning direction have to be characterized. In Fig. 3, a scheme containing the rotor disposition and the motors spinning direction is shown.

Secondly, it is important to define the different reference frames used for the characterization of the vehicle dynamics:

- Earth Frame Γ_e : Origin on the Earth surface, x_e aligned with Earth north, y_e axis aligned with Earth east and z_e axis pointing towards the center of Earth.
- Body Frame Γ_b : Origin in the airplane CG, x_b axis in the vehicle plane of symmetry and pointing to the nose, z_b axis in the vehicle plane of symmetry and perpendicular to x_b , y_b axis perpendicular to x_b and z_b , pointing to the right wing.
- Propeller Frame Γ_p^i : Origin in the center of rotation of the i -th rotor, axis direction aligned with the body frame Γ_b when the tilting angles g_i and b_i are zero.
- Wind Frame Γ_w : Origin in the airplane CG, x_w axis in the vehicle plane of symmetry pointing in the wind speed direction, z_w in the vehicle plane of symmetry and perpendicular to x_w , y_w perpendicular to x_w and z_w , pointing to the right wing.

An overview of the Earth, Body and Propeller frames and the identification of the rotor tilting angles is shown in Fig. 4.

The transformation matrices for the projection of a vector from one reference frame to another can be identified as follows:

For the coordinate transformation between the body reference frame Γ_b to earth reference frame Γ_e the following matrix is used:

$$R_{eb} = \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & -s_\phi c_\psi + c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

such that $\bar{x}_e = R_{eb} \cdot \bar{x}_b$ (26)

where c and s represent the abbreviation respectively of the cosine and sine function, while ϕ , θ and ψ are the Euler angles in the traditional ZYX order.

For the coordinate transformation between the propeller frame Γ_p to body reference frame Γ_b the following matrix is used:

$$R_{bp}^i = \begin{bmatrix} c(b^i) & 0 & s(b^i) \\ s(g^i)s(b^i) & c(g^i) - s(g^i)c(b^i) & \\ -c(g^i)s(b^i) & s(g^i) & c(g^i)c(b^i) \end{bmatrix}$$

such that $\bar{x}_b = R_{bp}^i \cdot \bar{x}_p^i$ (27)

where the angles b_i and g_i are the i -th rotor tilting angles. Conventionally, within the paper we will refer to b_i as ele-

vation tilting angle and to g_i as azimuth tilting angle. For a visual representation of the tilting angles, the reader can refer to Fig. 4.

Concerning the coordinate transformation between the wind frame Γ_w and the body frame Γ_b , the following matrix is used:

$$R_{bw} = \begin{bmatrix} c(\alpha)c(\beta) & -c(\alpha)s(\beta) & -s(\alpha) \\ s(\beta) & c(\beta) & 0 \\ s(\alpha)c(\beta) & -s(\alpha)s(\beta) & c(\alpha) \end{bmatrix}$$

such that $\bar{x}_b = R_{bw} \cdot \bar{x}_{wb}$ (28)

where α is the angle of attack and β is the sideslip angle.

Finally, we define the matrix T used to obtain the rate of change of the Euler angles from the body rates ω :

$$T = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix}$$

such that $\dot{\Theta} = T \cdot \omega$ (29)

where Θ represents the Euler angle vector composed by ϕ , θ and ψ .

A.2 Assumptions

In order to facilitate the EOM derivation, a few assumptions are made:

- Inflow into the propeller is assumed not to influence its performance.
- The thrust generated by the rotor is always perpendicular to the propeller disk and it is applied in the center of the propeller disk.
- The change in the body inertia due to the rotor tilting is negligible and x_b , y_b and z_b are vehicle principal axes.
- x_p , y_p and z_p are principal axes for the propeller, and the inertia terms I_{xx}^p and I_{yy}^p are negligible.
- The inertia tensor of the tilting mechanism in the propeller reference frame Γ_p is a diagonal matrix.

A.3 Equations Of Motion Derivation

With the reference frames and assumptions defined, it is possible to analyze all the forces and moments contributing to the system dynamics for the development of the EOM:

$$\begin{cases} \ddot{P}_e = \frac{1}{m} (F^p + F^a) + g \hat{z}_e \\ \dot{\omega} = I_b^{-1} (-\omega \times I_b \omega + M^t + M^d + M^i + M^p + M^a + M^r + M^{tilt}) \end{cases} (30)$$

where \ddot{P}_e are the linear acceleration in the earth reference frame and $\dot{\omega}$ represent the body rates derivative.

Each term of Eq. 30 refers to a specific contribution as follows:

- F^P : Forces produced by the propeller thrust rotated to the earth frame:

$$F^P = \sum_{i=1}^N R_{eb} R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ -K_p^T \Omega_i^2 \end{pmatrix}, \quad (31)$$

where K_p^T is the thrust coefficient of the motor and Ω_i is the rotational speed of the i -th motor.

- F^a : Aerodynamic forces produced by the vehicle in the earth frame:

$$F^a = R_{eb} R_{bw} \begin{pmatrix} -D^{wb} \\ Y^{wb} \\ -L^{wb} \end{pmatrix}, \quad (32)$$

where D_{wb} , Y_{wb} and L_{wb} are the aerodynamics forces acting on the vehicle and can be expressed as follows [22]:

$$\begin{pmatrix} D^{wb} \\ Y^{wb} \\ L^{wb} \end{pmatrix} = Q \begin{pmatrix} C_{D0} + k_{cd}(C_{L0} + C_{L\alpha})^2 \\ C_{Y\beta} \beta \\ C_{L0} + C_{L\alpha} \alpha \end{pmatrix}, \quad (33)$$

$$Q = \frac{1}{2} \rho S V_{tot}^2$$

where ρ is the air density, S is the wing surface and V_{tot} is the airspeed.

- M^a : Aerodynamic moments acting on the vehicle in the body reference frame:

$$M^a = \begin{pmatrix} M_L^a \\ M_M^a \\ M_N^a \end{pmatrix} = Q \begin{pmatrix} \bar{b}(C_{l0} + C_{l\beta} \beta + \frac{\bar{b}}{2V_{tot}}(C_{lp} p + C_{lr} r)) \\ \bar{c}(C_{m0} + C_{m\alpha} \alpha) \\ \bar{b}(C_{np} \frac{\bar{b}}{2V_{tot}} p + C_{nr} \frac{\bar{b}}{2V_{tot}} r) \end{pmatrix}, \quad (34)$$

where M_L^a , M_M^a and M_N^a represent respectively the aerodynamic roll, pitch and yaw moment acting on the vehicle.

The coefficients present in the second term of Eq. 34 and in the first term of Eq. 33 can be identified through test flights, CFD analysis or geometrical vehicle properties [23]. Within this paper, we will employ the aerodynamic coefficients identified in [1].

- M^t : Torque generated by the rotors due to the propeller thrust:

$$M^t = \sum_{i=1}^N \left(R_{bp}^i \cdot \begin{bmatrix} 0 \\ 0 \\ -K_p^T \Omega_i^2 \end{bmatrix} \right) \times (l_x^i \ l_y^i \ l_z^i), \quad (35)$$

where (l_x^i, l_y^i, l_z^i) are the coordinates of the i -th rotor in the body reference frame.

- M^d : Torque generated by the rotors due to the propeller drag:

$$M^d = \sum_{i=1}^N -R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ K_p^M \Omega_i^2 \end{pmatrix} (-1)^i, \quad (36)$$

where K_p^M is the torque coefficient of the motor and can be identified as previously described for K_p^T .

- M^i : Torque generated by the propeller inertia due to the rotational rate change:

$$M^i = \sum_{i=1}^N -J_p R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ \dot{\Omega}_i \end{pmatrix} (-1)^i, \quad (37)$$

where J_p is the propeller inertia.

- M^P : Torque generated by the rotor precession term due to the tilting rotation:

$$M^P = \sum_{i=1}^N R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ J_p \Omega_i \end{pmatrix} \times \begin{pmatrix} \dot{g}_i \\ \dot{b}_i \\ 0 \end{pmatrix} (-1)^i. \quad (38)$$

- M^{tilt} : Torque generated by the rotor inertial term due to the tilting rotation:

$$M^{tilt} = \sum_{i=1}^N R_{bp}^i \begin{pmatrix} \ddot{g}_i I_{xx_i}^{tilt} \\ \ddot{b}_i I_{yy_i}^{tilt} \\ 0 \end{pmatrix} (-1)^i, \quad (39)$$

where $I_{xx_i}^{tilt}$ and $I_{yy_i}^{tilt}$ are the i -th rotor tilting inertia in the propeller reference frame.

- M^r : Inertial term of the rotor due to the vehicle rates

$$M^r = \sum_{i=1}^N -\omega \times \left(R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ J_p \Omega_i \end{pmatrix} \right) (-1)^i. \quad (40)$$

References

1. Mancinelli, A., Smeur, E.J.J., Remes, B., Croon, G.D.: Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design. Int. Conf. Unmanned Aircraft Syst. (ICUAS) (2022). <https://doi.org/10.1109/ICUAS54217.2022.9836063>
2. Sadien, E., Roos, C., Birouche, A., Carton, M., Grimault, C., Romana, L.E., Basset, M.: A detailed comparison of control allocation techniques on a realistic on-ground aircraft benchmark. Amer. Cont. Conf. 2019 (2019). <https://doi.org/10.23919/ACC.2019.8814718>
3. Johansen, A., Fossen, I.: Control allocation - a survey. Automatica, 49(5), (2013). <https://doi.org/10.1016/j.automatica.2013.01.035>

4. Bodson, M.: Evaluation of optimization methods for control allocation. *J. Guid. Cont. Dynamics*, **25**(4), (2002). <https://doi.org/10.2514/2.4937>
5. Barata, J., Hussein, M.: The moore-penrose pseudoinverse: a tutorial review of the theory. *Brazilian J. Phys.* **42**, (2012). <https://doi.org/10.1007/s13538-011-0052-z>
6. Shi, J., Zhang, W., Li, G., Liu, X.: Research on allocation efficiency of the redistributed pseudo inverse algorithm. *Sci. China Inf. Sci.* **53**, (2010). <https://doi.org/10.1007/s11432-010-0032-x>
7. Durham, W.C.: Constrained control allocation. *J. Guidance Cont. Dynamics* **16**(4) (1993). <https://doi.org/10.2514/3.21072>
8. Bodson, M., Frost, S.A.: Load balancing in control allocation. *J. Guidance Cont. Dynamics* **34**(2), (2011). <https://doi.org/10.2514/1.51952>
9. Petersen, J.A.M., Bodson, M.: Constrained quadratic programming techniques for control allocation. *IEEE Trans. Cont. Syst. Technol.* **14**(1), (2006). <https://doi.org/10.1109/TCST.2005.860516>
10. Harkegard, O.: Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation. *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2 (2002). <https://doi.org/10.1109/CDC.2002.1184694>
11. Poonamallee, V.L., Yurkovich, S., Serrani, A., Doman, D.B.: A nonlinear programming approach for control allocation. *Proceedings of the 2004 American Control Conference*. vol. 2, (2004). <https://doi.org/10.23919/ACC.2004.1386822>
12. Wang, J., Longoria, R.G.: Coordinated and reconfigurable vehicle dynamics control. *Trans. Cont. Syst. Technol.* **17**(3), (2009). <https://doi.org/10.1109/TCST.2008.2002264>
13. Chen, Y., Wang, J.: Fast and global optimal energy-efficient control allocation with applications to over-actuated electric ground vehicles. *IEEE Trans. Cont. Syst. Technol.* **20**(5), (2012). <https://doi.org/10.1109/TCST.2011.2161989>
14. Chen, Y., Wang, J.: A global optimization algorithm for energy-efficient control allocation of over-actuated systems. *Proceedings of the 2011 American Control Conference*. (2011). <https://doi.org/10.1109/ACC.2011.5990899>
15. Huan, H., Wan, W., We, C., He, Y.: Constrained nonlinear control allocation based on deep auto-encoder neural networks. *European Control Conference (ECC)*. (2018). <https://doi.org/10.23919/ECC.2018.8550445>
16. Khan, H., Mobeen, S., Rajput, J., Riaz, J.: Nonlinear control allocation: A learning based approach. (2022). <https://doi.org/10.48550/arXiv.2201.06180>
17. Simplício, P., Pavel, M.D., VanKampen, E., Chu, Q.P.: An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. *Cont. Eng. Practice* **21**(8), (2013). <https://doi.org/10.1016/j.conengprac.2013.03.009>
18. Smeur, E.J.J., Chu, Q., De Croon, G.C.H.E.: Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *J. Guidance Cont. Dynamics* **39**(3), (2016). <https://doi.org/10.2514/1.G001490>
19. Wright, J.: Numerical optimization. Editor: Springer - ISBN: 978-0-38-730303-1 (1951)
20. Schittkowski, K.: NLPQL: A fortran subroutine solving constrained nonlinear programming problems. *Annal. Oper. Res.* **5**, (1986). <https://doi.org/10.1007/BF02739235>
21. Gati, B.: Open source autopilot for academic research - the paparazzi system. *2013 American Control Conference* (2013). <https://doi.org/10.1109/ACC.2013.6580045>
22. Klein, V., Morelli, E.: Aircraft system identification: theory and practice. Chapter 3. Editor: AIAA education (2006)
23. Smetana, F.O., Summey, D.C., Johnson, W.D.: Riding and handling qualities of light aircraft - A review and analysis. Editor: National aeronautics and space administration. (1972)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Alessandro Mancinelli obtained his BSc in aerospace engineering from Sapienza University of Rome in 2016. Subsequently, in 2018, he pursued his MSc in aeronautical engineering from the same university. Immediately after completing his MSc, he commenced his career as a Flight Laws control engineer for Airbus France, a role he held until 2020. Following this, in 2020, he successfully undertook a fully funded MBA program at the Collège des Ingénieurs. In 2021, he embarked on a PhD journey at Delft University of Technology, with his primary focus being the development of a UAV platform capable of autonomous landing on a moving platform.

Bart D. W. Remes is an Ir project manager and researcher in the MAVlab at TUD. He received his M.Sc. in Aerospace Engineering from the Technical University of Delft, focusing on Aerospace for Sustainable Engineering and Technology. In 2003, he founded the Micro Air Vehicles Laboratory (MAVlab) (<http://mavlab.tudelft.nl/>). Currently, he holds the position of researcher and project manager at the Micro Air Vehicle lab. Bart was involved in the first Dutch BVLOS flight and the design and construction of various Micro Air Vehicles, such as the www.delftAcopter.nl, the DelFly II, and the DelFly Micro (<http://www.delfly.nl/>). All of these Micro Air Vehicles (MAVs) have received multiple awards, including a mention in the Guinness Book of Records in 2009 for the "Smallest Camera Airplane, DelFly Micro," and the DelFly Nimble, which was published and featured on the cover of the September 14 issue of Science magazine in 2018. In addition to his work on flapping wing MAVs, Bart has initiated and worked on hybrid air vehicles (<http://www.nederdrone.nl/>, <http://www.atmosuav.com/>) and the design of the smallest open-source autopilot in the world, the "Lisa S." This 2x2 cm device is based on the Paparazzi open-source software.

Guido C. H. E. De Croon received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies with computationally efficient, bio-inspired algorithms for robot autonomy, with an emphasis on computer vision. Since 2008 he has worked on algorithms for achieving autonomous flight with small and light-weight flying robots, such as the DelFly flapping wing MAV. In 2011–2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow based control algorithms for extraterrestrial landing scenarios. After his return at TU Delft, his work has included fully autonomous flight of a 20-gram DelFly, a new theory on active distance perception with optical flow, and a swarm of tiny drones able to explore unknown environments. Currently, he is Full Professor at TU Delft and scientific lead of the Micro Air Vehicle lab (MAVLab) of Delft University of Technology.

Ewoud J. J. Smeur received his B.Sc. and M.Sc. degrees in Aerospace Engineering from Delft University of Technology. In 2018, he obtained a Ph.D. degree from the same university on the topic of incremental control of hybrid micro air vehicles. He worked for Alten Nederland in Eindhoven in 2018 and did a post-doc in Munich in 2019. Since 2020, he is assistant professor at Delft University of Technology. His research interests are in control and guidance of novel types of micro air vehicles.