Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Hierarchical Control of a Quadcopter under Stuck Actuator Fault

Ngoc Thinh Nguyen * **Ionela Prodan** ** **Felix Petzke** ***
Stefan Streif* **Laurent Lefèvre** **

* *Univ. of Luebeck, Institute for Robotics and Cognitive Systems, 23562 Luebeck, Germany. Email:* nguyen@rob.uni-luebeck.de.
** *Univ. Grenoble Alpes, Grenoble INP* †*, LCIS, Valence, France. Email:* {ionela.prodan,laurent.lefevre}@ lcis.grenoble-inp.fr.
† *Institute of Engineering and Management Univ. Grenoble Alpes.*
*** *Technische Universität Chemnitz, Automatic Control and System Dynamics (ACSD) Lab, 09107 Chemnitz, Germany. Email:* {felix.petzke, stefan.streif}@etit.tu-chemnitz.de.

Abstract: We propose a hierarchical FTC (Fault Tolerant Control) scheme for trajectory tracking by a quadcopter system under stuck actuator fault and actuator saturation. Both the FDI (Fault Detection and Isolation) and control reconfiguration modules are implemented at the low-level associated with the rotation dynamics through a NMPC (Nonlinear Model Predictive Control) strategy. The uncontrolled (when under fault) yaw torque is predicted and compensated by the NMPC. It is shown that the overall control scheme succeeds in maintaining trajectory tracking for various fault events (both in the sense of having various stuck values and in the sense of changing the actuator under fault).

Keywords: FTC, Stuck actuator fault, Trajectory tracking, Hierarchical control, Feedback linearization, NMPC, Quadcopter.

1. INTRODUCTION

Recently, quadcopter unmanned aerial vehicles have elicited increasing interest in the research community as a typical system for studying control algorithms (Nascimento and Saska, 2019) as well as sensors/actuators fault diagnosis (Freddi et al., 2011; Shao et al., 2018). Among them, the FTC (Fault Tolerant Control) hierarchical control design (Stoican et al., 2018) is usually employed to take into account the decoupling between the translation and rotation dynamics and to counteract various types of faults, e.g., loss of rotor(s) (Freddi et al., 2011). One particular fault for aerial vehicles is the *stuck rotor* fault (Shao et al., 2018; Nguyen et al., 2017). Once stuck, the faulty actuators keep rotating at a constant speed regardless of the actual control inputs. Thus, under a unique stuck rotor fault, the quadcopter system not only loses one degree of freedom in its control ability but also gains persistent disturbances (Chen and Jiang, 2005).

This work extends previous results of the authors (Nguyen et al., 2017) where an FTC scheme for controlling a quadcopter system was developed by using FL (feedback linearization) but without providing an FDI module and considering actuator saturation. This shortcoming was due to the complexity of the whole controlled system under input constraints but can, arguably, lead to bad behavior such as loss of stability and decrease in performance. In the presence paper, these drawbacks have been overcome by designing a fault diagnosis module particularized for the stuck rotor fault (Hasan and Johansen, 2018). Furthermore, the saturation constraints on the rotor speeds are fulfilled by employing the NMPC (Nonlinear Model Predictive Control) strategy. More precisely, this paper provides several contributions which, to the best of our knowledge, are new to state of the art (w.r.t. similar works in FTC designs for multicopters as in Freddi et al. (2011); Nguyen et al. (2017)):

- A hierarchical control scheme for the trajectory tracking of a quadcopter system under actuator saturation and a stuck fault occurrence is provided. The high level employs a FL controller while the low level switches between two different NMPC schemes according to the system's functioning states (healthy or under fault). The two NMPC designs guarantee the satisfaction of the system's constraints.
- A model of the uncontrolled yaw motion under fault is proposed. This allows the prediction dynamics employed within the NMPC controller to be more realistic than keeping the yaw value as a constant feedback within the whole prediction horizon would have been.
- A fault diagnosis module for detecting the faults of a unique rotor being stuck at varying speeds is designed. The residue is constructed based on the differences between the estimated and reference normalized torques (given in the unit of the rotor speed).

The remainder of this paper is organized as follows. Section 2 presents first mode of a standard quadcopter system and of the stuck actuator fault. Next, Section 3 introduces the hierarchical FTC scheme and the fault diagnosis module. The simulation results are given and discussed in Section 4. Finally, Section 5 draws the conclusions and presents the future directions.

2. MODEL DYNAMICS

• *System modeling*

This section briefly recapitulates the mathematical model of a standard quadcopter as found in (Freddi et al., 2011; Nguyen et al., 2017). Firstly, the translation dynamics are given in the global fixed frame as follows:

$$\dot{\xi} = v, \tag{1a}$$
$$\dot{v} = -g\boldsymbol{e}_z + R_3 T/m, \tag{1b}$$

with $\xi = [x\ y\ z]^\top$ the position and $v = [v_x\ v_y\ v_z]^\top$ the translation velocity of the quadcopter, $\boldsymbol{e}_z = [0\ 0\ 1]^\top$, m the system mass and g the gravity consant. $T \in \mathbb{R}_{\geqslant 0}$ is the input thrust. R_3 stands for the 3^{rd} column of the roll–pitch–yaw rotation matrix (Nguyen et al., 2017, 2018):

$$R_3 = [\mathrm{c}\,\phi\,\mathrm{s}\,\theta\,\mathrm{c}\,\psi + \mathrm{s}\,\phi s\psi,\ c\phi s\theta s\psi - s\phi\,\mathrm{c}\,\psi,\ \mathrm{c}\,\phi\,\mathrm{c}\,\theta]^\top, \tag{2}$$

with "c" the $\cos(\cdot)$ and "s" the $\sin(\cdot)$ functions. (ϕ, θ, ψ) are the roll, pitch and yaw angles which subject to the rotation dynamics given as follows:

$$\dot{\eta} = W(\eta)\omega, \tag{3a}$$
$$\dot{\omega} = J^{-1}\left(-\omega \times (J\omega) + \tau\right), \tag{3b}$$

with $\eta = [\phi\ \theta\ \psi]^\top$ the angle vector, $\omega = [\omega_x\ \omega_y\ \omega_z]^\top$ the angle rate vector, $\tau = [\tau_\phi\ \tau_\theta\ \tau_\psi]^\top$ gathering the roll, pitch, yaw torques, and $J = \mathrm{diag}\{J_x, J_y, J_z\}$ the inertia matrix. The matrix $W(\eta) \in \mathbb{R}^{3\times 3}$ depends on the roll and pitch angles as follows (with "t" the $\tan(\cdot)$ function):

$$W(\eta) = \begin{bmatrix} 1 & \mathrm{s}\,\phi\,\mathrm{t}\,\theta & \mathrm{c}\,\phi\,\mathrm{t}\,\theta \\ 0 & \mathrm{c}\,\phi & -\mathrm{s}\,\phi \\ 0 & \mathrm{s}\,\phi/\mathrm{c}\,\theta & \mathrm{c}\,\phi/\mathrm{c}\,\theta \end{bmatrix}. \tag{4}$$

The thrust and the three torques are calculated in terms of the four rotor speeds (denoted by Ω_i for the i^{th} rotor):

$$\mathsf{u} = M\boldsymbol{\Omega}^2, \tag{5}$$

with $\mathsf{u} = [T\ \tau_\phi\ \tau_\theta\ \tau_\psi]^\top$ and $\boldsymbol{\Omega} = [\Omega_1 \dots \Omega_4]^\top$ the rotor speed vector. Note that, the four rotor can only rotate in their predefined directions, i.e., $\Omega_i \in \mathbb{R}_{\geqslant 0}, \forall i \in \{1 \dots 4\}$. In (5), the configuration matrix M is given by:

$$M = \begin{bmatrix} K_T & K_T & K_T & K_T \\ 0 & -LK_T & 0 & LK_T \\ -LK_T & 0 & LK_T & 0 \\ -b & b & -b & b \end{bmatrix}, \tag{6}$$

with K_T the thrust coefficient, b the drag coefficient and L the arm length, all assumed known.

Furthermore, the rotor speed Ω_i tracks its reference denoted by $\Omega_{i,r}$ which is actually the input sent to the quadcopter system (1)-(5). The tracking mechanism is subject to actuator saturation and faults as detailed hereinafter.

• *Actuator under saturation constraint and stuck fault*

In what follows we consider a *stuck actuator* fault (Chen and Jiang, 2005; Nguyen et al., 2017) for the quadcopter system. This type of fault forces the rotor speed Ω_i to remain stuck at a constant value $\Omega_{i,\alpha}$, regardless of the actual reference value $\Omega_{i,r}$. We neglect the internal dynamics of the rotors (which are usually considered as a low-pass filter (Shao et al., 2018)), hence, under saturation constraint and under fault of a unique stuck rotor, the rotating speed of the i^{th} rotor is modeled through:

$$\Omega_i = \begin{cases} \mathrm{sat}(\Omega_{i,r}|\Omega_{\max}), & \text{under nominal condition,} \\ \Omega_{i,\alpha}, & \text{under stuck fault,} \end{cases} \tag{7}$$

with $\Omega_{i,r} \in \mathbb{R}_{\geqslant 0}$ the speed reference of the i^{th} rotor, $\omega_{\max} > 0$ the maximum rotor speed. The saturation function $\mathrm{sat}(\cdot)$ is simply taken as:

$$\mathrm{sat}(\Omega_{i,r}|\Omega_{\max}) = \begin{cases} \Omega_{i,r}, & \text{if } |\Omega_{i,r}| \leqslant \Omega_{\max}, \\ \Omega_{\max}, & \text{if } |\Omega_{i,r}| > \Omega_{\max}. \end{cases} \tag{8}$$

For consistency, the constant stuck value $\Omega_{i,\alpha}$ is expressed in terms of $0 \leqslant \alpha \leqslant 1$ as a percentage of the maximum rotor speed, i.e.:

$$\Omega_{i,\alpha} = \alpha\Omega_{\max}. \tag{9}$$

Within this paper, we consider at most one rotor being stuck at a time, so that the quadcopter can still track a 3D reference trajectory (in the sense of tracking the position component of the trajectory while losing command over the yaw component). Having one stuck rotor means that one degree of freedom is lost in the control of the quadcopter system (1)-(5). Thus, considering the i^{th} rotor stuck at $\Omega_{i,f}$ from (9) leads to a rotor speed vector written as follows:

$$\boldsymbol{\Omega}_{i,\alpha} = [\Omega_1 \dots \Omega_{i,\alpha} \dots \Omega_4]^\top. \tag{10}$$

Consequently, (5) is rewritten as follows:

$$\mathsf{u} = \begin{cases} M\boldsymbol{\Omega}^2, & \text{under nominal condition,} \\ M\boldsymbol{\Omega}_{i,\alpha}^2, & \text{the } i^{th} \text{ rotor under fault (stuck).} \end{cases} \tag{11}$$

Let us introduce the notations of several variables employed within the paper to easily describe the system under fault of the i^{th} stuck rotor ($i \in \{1 \dots 4\}$):

- $\widehat{M_i}$ and $\widetilde{M_i}$, both in $\mathbb{R}^3$, gather the first and the last three elements of the i^{th} column of M in (6), respectively.
$$\widehat{M_1} = [K_T\ 0\ -LK_T]^\top,\ \widetilde{M_4} = [LK_T\ 0\ b]^\top. \tag{12}$$
- $\widehat{M}$ and $\widetilde{M}$, both in $\mathbb{R}^{3\times 4}$, gather the first and the last 3 rows of M in (6) respectively. $\widehat{M_{!i}} \in \mathbb{R}^{3\times 3}$ gathers the first 3 rows of all the other columns except the i^{th} column of M in (6). E.g.:
$$\widehat{M_{!2}} = \begin{bmatrix} K_T & \cancel{K_T} & K_T & K_T \\ 0 & \cancel{-LK_T} & 0 & LK_T \\ -LK_T & \cancel{0} & LK_T & 0 \end{bmatrix}. \tag{13}$$
- The thrust, T, roll, τ_ϕ, and pitch, τ_θ, torques are gathered into:
$$\hat{\mathsf{u}} = [T\ \tau_\phi\ \tau_\theta]^\top. \tag{14}$$
- The four speed references from (7) are represented by:
$$\boldsymbol{\Omega}_r = [\Omega_{1,r} \dots \Omega_{4,r}]^\top, \tag{15}$$
while the three rotor speed references except the i^{th} stuck rotor are gathered into $\boldsymbol{\Omega}_{r,!i} \in \mathbb{R}^3$. E.g.:
$$\boldsymbol{\Omega}_{r,!2} = [\Omega_{1,r}\ \Omega_{3,r}\ \Omega_{4,r}]^\top. \tag{16}$$

3. FAULT TOLERANT CONTROL DESIGN

Fig. 1 presents the hierarchical control scheme to counteract the influences of the fault induced by a single stuck rotor. At high level, the position controller tracks the reference 3D trajectory $\xi_r = [x_r\ y_r\ z_r]^\top$ by providing the references of thrust, T_r, and roll, ϕ_r, pitch, θ_r, angles. We apply the feedback linearization controller introduced in our previous work (Nguyen et al., 2018) which provides the references (T_r, ϕ_r, θ_r) constrained as follows:

$$T_r \leqslant T_{\max} \text{ and } |\phi_r|, |\theta_r| \leqslant \epsilon_{\max}, \tag{17}$$

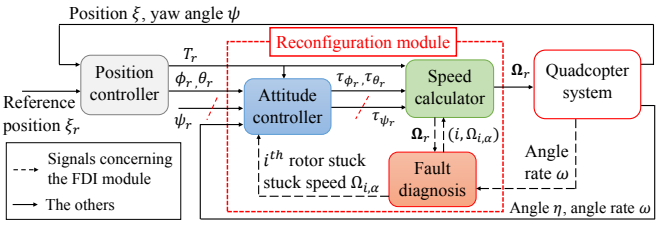


Fig. 1. Hierarchical FTC scheme for a quadcopter system.

where $T_{\max}$, the maximum thrust and $\epsilon_{\max}$, the maximum angle are parameters to be desired. The reader is referred to (Nguyen et al., 2018) for more details on this feedback linearization controller while we concentrate only on the reconfiguration module design hereinafter.

3.1 FTC module

This section presents the FTC (Fault Tolerant Control) module which consists of the attitude controller and the speed calculator as shown in Fig. 1. The FTC law alternates between the healthy and under fault modes. Each of these correspond to a particular form of the NMPC attitude controller and the speed calculator.
At first, the discretized rotation dynamics are required for the prediction part of the NMPC. A typical example is to apply Euler's discretization method with sampling time Δt to the continuous model given in (3):

$$\bar{\eta}(s+1) = \bar{\eta}(s) + \Delta t W\left(\bar{\eta}(s)\right)\bar{\Omega}(s), \tag{18a}$$

$$\bar{\omega}(s+1) = J^{-1}\left(-\bar{\omega}(s) \times \left(J\bar{\Omega}(s)\right) + \bar{\tau}(s)\right), \tag{18b}$$

with $\bar{\eta}(s)$, the predicted angle and $\bar{\omega}(s)$, the predicted angle rate at time step s. $\bar{\Omega}$, J and $W(\cdot)$ are given in (3). The discrete dynamics (18) will be used in both healthy and faulty modes of the FTC module detailed hereinafter.

Nominal functioning: Under nominal operation, the healthy attitude controller tracks the three angle references $\eta_r = [\phi_r\ \theta_r\ \psi_r]^\top$ (with (ϕ_r, θ_r) obtained from the position controller as in (17)) by providing three torque references $\tau_r = [\tau_{\phi_r}\ \tau_{\theta_r}\ \tau_{\psi_r}]^\top$. Using the MPC strategy, the torque references $\tau_r(k)$ at time step k (i.e., time instant $k\Delta t$) is obtained as the solution of the following optimization problem over the prediction horizon N_p:

$$\tau_r(k) = \arg\min_{\bar{\tau}} \sum_{s=0}^{N_p-1} \Big(\left(\bar{\eta}(s) - \eta_r(k)\right)^\top Q_\eta \left(\bar{\eta}(s) - \eta_r(k)\right) + \left(\bar{\tau}(s+1) - \bar{\tau}(s)\right)^\top Q_{\Delta\tau} \left(\bar{\tau}(s+1) - \bar{\tau}(s)\right) \Big), \tag{19}$$

$$\text{subject to } \begin{cases} \text{dynamics (18)}, \\ \bar{\tau}(s) \in \mathcal{S}(T_r(k)),\ \forall s \in \{0 \dots N_p - 1\}, \\ \bar{\eta}(0) = \eta(k),\ \bar{\Omega}(0) = \Omega(k), \end{cases}$$

with the references, $\eta_r(k)$ and $T_r(k)$, obtained from the high level controller. $\eta(k)$ and $\Omega(k)$ are the actual values of the angles and angle rates at time step k. The set $\mathcal{S}(T_r(k)) \subset \mathbb{R}^3$ is given by:

$$\mathcal{S}(T_r(k)) = \left\{ \tau_r \in \mathbb{R}^3 \middle| 0 \leqslant M^{-1} \begin{bmatrix} T_r(k) \\ \tau_r \end{bmatrix} \leqslant \omega_{\max}^2 \right\}, \tag{20}$$

with M in (6) and $\omega_{\max}$ the maximum value of the rotor speed in (7). The set $\mathcal{S}(T_r(k))$ from (20) is constructed based on (11) and it gathers all the feasible values of $\tau_r \in \mathbb{R}^3$ such that the resulted reference rotor speeds $\boldsymbol{\Omega}_r$ as in (15) respect the rotor speed saturation.

Next, at time step k, the speed calculator block provides the reference rotor speeds $\boldsymbol{\omega}_r(k)$ given by:

$$\boldsymbol{\Omega}_r^2(k) = M^{-1}\mathsf{u}_r(k), \tag{21}$$

where $\mathsf{u}_r(k) = [T_r(k)\ \tau_r^\top(k)]^\top$ with $T_r(k)$ from (17) and $\tau_r(k)$ from (19). As $\tau_r(k) \in \mathcal{S}(T_r(k))$ from (20), the reference rotor speeds $\boldsymbol{\Omega}_r(k)$ calculated by (21) stay under the maximum rotor speed $\Omega_{\max}$ which ultimately lead to:

$$\boldsymbol{\Omega}(k) = \boldsymbol{\Omega}_r(k),\ T(k) = T_r(k),\ \tau(k) = \tau_r(k), \tag{22}$$

with $\boldsymbol{\Omega}(k)$ from (7), $T(k)$ and $\tau(k)$ from (11). This will be used for designing the FDI module later.

Under fault functioning (i^{th} rotor is stuck): Once the fault is detected and isolated, the attitude controller reconfigures to no longer control the yaw angle (recall also Fig. 1). At time step k, it provides only the references of the roll, $\tau_{\phi_r}(k)$, and pitch, $\tau_{\theta_r}(k)$, torques to track the angle references, $\phi_r(k)$ and $\theta_r(k)$ sent from the position controller at the high level (17). Then, the speed calculator block provides the three reference speeds $\boldsymbol{\Omega}_{r,!i}(k)$ (as defined in (16)) for the three remaining healthy rotors (except from the i^{th} stuck rotor) as follows:

$$\boldsymbol{\Omega}_{r,!i}^2(k) = \widehat{M}_{!i}^{-1}\hat{\mathsf{u}}_r(k) - \widehat{M}_{!i}^{-1}\widehat{M}_i\Omega_{i,\alpha}^2, \tag{23}$$

with $\hat{\mathsf{u}}_r(k) = [T_r(k)\ \tau_{\phi_r}(k)\ \tau_{\theta_r}(k)]^\top$, $\widehat{M}_i \in \mathbb{R}^3$ from (12), $\widehat{M}_{!i} \in \mathbb{R}^{3\times 3}$ from (13) and $\Omega_{i,\alpha}$ the stuck speed of the faulty i^{th} rotor as in (9). Then, by using the actual stuck speed as reference for the faulty i^{th} rotor, the four rotor speed references are given by:

$$\boldsymbol{\Omega}_r^2(k) = \boldsymbol{I}_{4,:i}\Omega_{i,\alpha}^2 + \boldsymbol{I}_{4,!i}\boldsymbol{\Omega}_{r,!i}^2(k), \tag{24}$$

with $\boldsymbol{I}_{4,:i} \in \mathbb{R}^4$ the i^{th} column and $\boldsymbol{I}_{4,!i} \in \mathbb{R}^{4\times 3}$ the matrix gathering the three columns beside the i^{th} one of the identity matrix $\boldsymbol{I}_4 \in \mathbb{R}^{4\times 4}$.
In order to avoid the saturation effects on $\boldsymbol{\Omega}_{r,!i}(k)$ from (23), the torque references, $\tau_{\phi_r}(k)$ and $\tau_{\theta_r}(k)$, are required to be within the feasible set $\mathcal{S}_i(T_r(k)) \subset \mathbb{R}^2$:

$$\mathcal{S}_i(T_r(k)) = \left\{ \begin{bmatrix} \tau_{\phi_r} \\ \tau_{\theta_r} \end{bmatrix} \in \mathbb{R}^2 \middle| \right. \tag{25}$$
$$\left. 0 \leqslant \widehat{M}_{!i}^{-1} \begin{bmatrix} T_r(k) \\ \tau_{\phi_r} \\ \tau_{\theta_r} \end{bmatrix} - \widehat{M}_{!i}^{-1}\widehat{M}_i\Omega_{i,\alpha}^2 \leqslant \Omega_{\max}^2 \right\}.$$

Proposition 1. Assuming the i^{th} rotor under fault, let us constrain $[\tau_{\phi_r}(k)\ \tau_{\theta_r}(k)]^\top \in \mathcal{S}_i(T_r(k))$ as in (25). The rotors track their speed references $\boldsymbol{\Omega}_r(k)$ from (24). Then, the following hold:

i) the actual values of the four rotor speeds $\Omega(k)$, of thrust, $T(k)$, and of roll, $\tau_\phi(k)$, pitch, $\tau_\theta(k)$, torques equal their references.
ii) the uncontrolled yaw torque $\tau_\psi(k)$ as in (18) is calculated by the following formulation:

$$\tau_\psi(k) = 4M_{4,i}\Omega_{i,\alpha}^2 + M_{4:}\boldsymbol{I}_{4,!i}\widehat{M}_{!i}^{-1}\hat{\mathsf{u}}_r(k), \tag{26}$$

with $M_{4,i}$ the i^{th} element of $M_{4:}$, the 4^{th} row of M, $\widehat{M}_{!i}$ from (13), $\hat{\mathsf{u}}_r$ from (23) and $\Omega_{i,\alpha}$ the stuck speed of the faulty i^{th} rotor.

Proof. Since $[\tau_{\phi_r}(k)\ \tau_{\theta_r}(k)]^\top \in \mathcal{S}_i(T_r(k))$ from (25), the speed references $\boldsymbol{\Omega}_r(k)$ from (24) do not exceed their maximum value $\Omega_{\max}$. Thus, the actual rotor speeds under fault $\boldsymbol{\Omega}_{i,\alpha}(k)$ equal to their references $\boldsymbol{\Omega}_r(k)$ (i.e. the i^{th}

stuck rotor receives the actual stuck speed as its reference). Then, the actual thrust and torques $\mathsf{u}(k)$ in (11) are:

$$\mathsf{u}(k) = M\boldsymbol{\Omega}_{i,\alpha}^2(k), \tag{27}$$

which validates point i) and also provides $\tau_\psi(k) = M_{4:}\boldsymbol{\Omega}^2(k)$. Combining this with (23)-(24) leads to (26) (by also using $M_{4:}\left(\boldsymbol{I}_{4,:i} - \boldsymbol{I}_{4,!i}\widehat{M}_{!i}^{-1}\widehat{M}_i\right) = M_{4,i}$). □

Finally, the NMPC optimization problem at time step k of the attitude controller under fault of the i^{th} stuck rotor is formulated as follows:

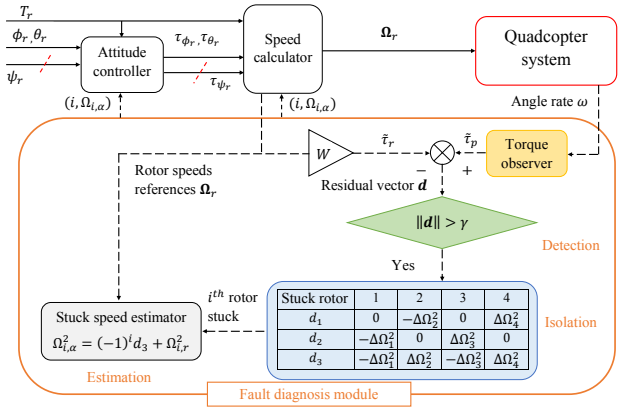
$$\begin{aligned}\widehat{\mathcal{T}}_r(k) = \arg\min_{\widehat{\bar{\mathcal{T}}}} \sum_{s=0}^{N_f-1} &\left((\widehat{\bar{\eta}}(s) - \hat{\eta}_r(k))^\top Q_{\hat{\eta}}(\widehat{\bar{\eta}}(s) - \hat{\eta}_r(k))\right.\\ &\left. + \left(\widehat{\bar{\mathcal{T}}}(s+1) - \widehat{\bar{\mathcal{T}}}(s)\right)^\top Q_{\Delta\hat{\mathcal{T}}}\left(\widehat{\bar{\mathcal{T}}}(s+1) - \widehat{\bar{\mathcal{T}}}(s)\right)\right),\end{aligned} \tag{28}$$

$$\text{s.t.} \begin{cases} \text{dynamics (18)}, \\ \bar{\mathcal{T}}_\psi(s) = 4M_{4,i}\Omega_{i,\alpha}^2 + M_{4:}\boldsymbol{I}_{4,!i}\widehat{M}_{!i}^{-1}\widehat{\bar{\mathsf{u}}}(s), \\ \widehat{\bar{\mathcal{T}}}(s) \in \mathcal{S}_i(T_r(k)),\ \forall s \in \{0 \dots N_f - 1\}, \\ \bar{\eta}(0) = \eta(k),\ \bar{\omega}(0) = \omega(k). \end{cases}$$

with $\hat{\eta}_r(k) = [\phi_r(k)\ \theta_r(k)]^\top$ the roll and pitch angle references, $T_r(k)$ the thrust reference, all obtained from the high level controller at time step k. Note that, by constraining the whole predicted torques $\widehat{\bar{\mathcal{T}}}(s)$ along the prediction horizon N_f to stay within the set $\mathcal{S}_i(T_r(k))$, Proposition 1 is validated. Thus, the prediction model of yaw torque $\bar{\mathcal{T}}_\psi(s)$ can be taken from (26) and the resulted actual roll, pitch torques equal their references $\widehat{\mathcal{T}}_r(k)$ obtained from (28). Ultimately, the actual roll, pitch angles $\hat{\eta}(k)$ track their references $\hat{\eta}_r(k)$ and leads to the stability of the whole hierarchical control scheme.

3.2 FDI module design

This section introduces the design of a fault diagnosis module (as shown in Fig. 2) which can detect the stuck fault, identify the i^{th} stuck rotor and estimate its stuck speed $\Omega_{i,\alpha}$.



Stuck rotor	1	2	3	4
d_1	0	$-\Delta\Omega_2^2$	0	$\Delta\Omega_4^2$
d_2	$-\Delta\Omega_1^2$	0	$\Delta\Omega_3^2$	0
d_3	$-\Delta\Omega_1^2$	$\Delta\Omega_2^2$	$-\Delta\Omega_3^2$	$\Delta\Omega_4^2$

Fig. 2. Fault diagnosis module.

Torque normalization:
We define the normalized torques as follows

$$\tilde{\mathcal{T}} = \begin{cases} W\boldsymbol{\Omega}^2, & \text{under nominal condition,} \\ W\boldsymbol{\Omega}_{i,\alpha}^2, & \text{if the } i^{th} \text{ rotor stuck,} \end{cases} \tag{29}$$

with $\boldsymbol{\Omega}$, $\boldsymbol{\Omega}_{i,\alpha}$ from (11) and the matrix W given by:

$$W = \begin{bmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 \end{bmatrix}. \tag{30}$$

Next, by a similar way, we also construct the normalized desired torques $\tilde{\mathcal{T}}_d$ as follows:

$$\tilde{\mathcal{T}}_d = W\boldsymbol{\Omega}_r^2, \tag{31}$$

with $\boldsymbol{\Omega}_r$ the rotor speeds references obtained by (21) under nominal functioning and by (24) under fault of the i^{th} stuck rotor.

Torque observer (yellow block in Fig. 2):
The observer estimate the value of the actual normalized torque $\tilde{\mathcal{T}}$ as in (29) from the available angle rate ω based on the Euler discretization of the rotation dynamics (18):

$$\mathcal{T}_p(k) = J\left(\frac{\omega(k) - \omega(k-1)}{\Delta_{\text{FDI}}}\right) + \omega(k) \times (J\omega(k)), \tag{32}$$

with $\mathcal{T}_p(k)$ the estimated torque at time step k, $\omega(k)$ the angle rate, Δ_{FDI} the sampling time of the FDI module (can be chosen smaller than the sampling time of the attitude controller Δt as in (18) to enhance the accuracy, but being limited by the feedback rate of ω). Then, the normalized estimated torque $\tilde{\mathcal{T}}_p$ is calculated as follows:

$$\tilde{\mathcal{T}}_p(k) = \text{diag}\left\{\frac{1}{LK_T}, \frac{1}{LK_T}, \frac{1}{b}\right\}\mathcal{T}_p(k), \tag{33}$$

with L, K_T, b the physical parameters from (5).

Residual vector and its functioning:
Recalling the FDI module scheme in Figure 2, the residual vector $\boldsymbol{d} = [d_1\ d_2\ d_3]^\top$ is simply taken as follows:

$$\boldsymbol{d} = \tilde{\mathcal{T}}_p - \tilde{\mathcal{T}}_d, \tag{34}$$

with $\tilde{\mathcal{T}}_p$, the estimated normalized torque as in (33) and $\tilde{\mathcal{T}}_d$, the normalized desired torque as in (31), both taken at the same time instants.

Proposition 2. (Fault detection). Let us consider the residual vector $\boldsymbol{d}$ calculated as in (34). The followings hold:

1) If the attitude controller given in Section 3.1 is functioning in the appropriate mode, i.e., nominal mode (19) under nominal case and under-fault mode (28) corresponding to the right scenario (i^{th} rotor being stuck at $\Omega_{i,\alpha}$), then the expectation of the norm of the residual vector $\|\boldsymbol{d}\|$ is zero:
$$\mathbf{E}[\|\boldsymbol{d}\|] = 0, \tag{35}$$
with $\mathbf{E}[x]$ is the expectation of a variable x, sometimes called the mean value of x.
2) If the real rotor speeds are not tracking their references, then, the norm of the residual vector $\|\boldsymbol{d}\|$ varies around a non-zero value. More precisely, we consider two following scenarios:
   2a) If the attitude controller is functioning in the nominal mode (19) and the i^{th} rotor is stuck at the speed of $\Omega_{i,\alpha}$, then we have that:
   $$\mathbf{E}[\|\boldsymbol{d}\|] = \sqrt{2}|\Omega_{i,\alpha}^2 - \Omega_{i,r}^2|, \tag{36}$$
   with $\Omega_{i,r}$ the speed reference of the i^{th} rotor.
   2b) If the attitude controller is functioning in under-fault mode (28) corresponding to the right i^{th}

stuck rotor but with the wrong stuck speed Ω_{i,α_w} instead of the actual stuck speed $\Omega_{i,\alpha}$, then:

$$\mathbf{E}[\|\boldsymbol{d}\|] = \sqrt{2}|\Omega_{i,\alpha}^2 - \Omega_{i,\alpha_w}^2|. \tag{37}$$

Proof. We assume that the torques estimator as in (32)–(33) works properly and hence, the estimated normalized torque $\widetilde{\tau}_p$ from (33) varies around its actual value $\widetilde{\tau}$ from (29), i.e. $\mathbf{E}[\widetilde{\tau}_p] = \widetilde{\tau}$. Then, we have that:

$$\mathbf{E}[\boldsymbol{d}] = \widetilde{\tau} - \widetilde{\tau}_d = \begin{cases} W(\boldsymbol{\Omega}^2 - \boldsymbol{\Omega}_r^2), & \text{under nominal case,} \\ W(\boldsymbol{\Omega}_{i,\alpha}^2 - \boldsymbol{\Omega}_r^2), & \text{with } i^{th} \text{ stuck rotor,} \end{cases} \tag{38}$$

with W from (30), $\boldsymbol{\Omega}_r$, ($\boldsymbol{\Omega}$ and $\boldsymbol{\Omega}_{i,\alpha}$) the references and the actual rotor speeds under nominal and faulty cases. For both scenarios under point 1), the actual rotor speeds equal their references which leads to $\mathbf{E}[\boldsymbol{d}] = \mathbf{0}$ as in (35). Next, regarding both scenarios considered at point 2), their common problem is that the i^{th} faulty rotor ($i \in \{1, \dots, 4\}$) becomes stuck and does not follow its reference. Hence, the residual vector $\boldsymbol{d}$ varies around a non-zero value:

$$\mathbf{E}[\boldsymbol{d}] = W_i \Delta\Omega_i^2, \tag{39}$$

in which, W_i is the i^{th} column of the matrix W from (30). The term $\Delta\Omega_i^2$ describes the speed tracking mismatch of the i^{th} rotor and is defined as follows:

$$\Delta\Omega_i^2 = \begin{cases} \Omega_{i,\alpha}^2 - \Omega_{i,r}^2, & \text{under point 2a,} \\ \Omega_{i,\alpha}^2 - \Omega_{i,\alpha_w}^2, & \text{under point 2b,} \end{cases} \tag{40}$$

with $\Omega_{i,\alpha}$, $\Omega_{i,r}$ and Ω_{i,α_w} given as in (36)–(37). Due to the construction of W from (30), $\|W_i\| = \sqrt{2}, \forall i \in \{1, \dots, 4\}$ which further provides $\mathbf{E}[\|\boldsymbol{d}\|] = \|W_i\||\Delta\Omega_i^2| = \sqrt{2}|\Delta\Omega_i^2|$ and hence, both (36)–(37) are validated. □

From Proposition 2, the two faulty scenarios 2a (36) and 2b (37) can be distinguished from the point 1 (35) and be detected by checking (c.f. Figure 2):

$$\|\boldsymbol{d}\| > \gamma, \tag{41}$$

with $\gamma \in \mathbb{R}_+$ the threshold designed as follows:

$$\gamma = \sqrt{2}\beta\Omega_{\max}^2, \tag{42}$$

with $\beta \in (0, 1)$ the tuning parameter describing the ratio of the acceptable tracking error (e.g. $|\Omega_{i,\alpha}^2 - \Omega_{i,r}^2|$ as in (36)) to the maximum squared speed $\Omega_{\max}^2$ as in (7).

Proposition 3. (Fault isolation and estimation). Considering two scenarios 2a (36) and 2b (37), after detecting the faults, the i^{th} faulty rotor can be isolated (i.e., re-checked for scenario 2b) by using the look-up Table 1 in which the residual vector $\boldsymbol{d} = [d_1\ d_2\ d_3]$ is from (34) and the notation $\Delta\Omega_i^2$ is defined in (40). Next, after obtaining the

Table 1. Stuck rotor identification.

Stuck rotor	1	2	3	4
d_1	0	$-\Delta\Omega_2^2$	0	$\Delta\Omega_4^2$
d_2	$-\Delta\Omega_1^2$	0	$\Delta\Omega_3^2$	0
d_3	$-\Delta\Omega_1^2$	$\Delta\Omega_2^2$	$-\Delta\Omega_3^2$	$\Delta\Omega_4^2$

index i, the actual stuck speed $\Omega_{i,\alpha}$ is estimated by:

$$\Omega_{i,\alpha}^2 = \begin{cases} (-1)^i d_3 + \Omega_{i,r}^2, & \text{under point 2a,} \\ (-1)^i d_3 + \Omega_{i,\alpha_w}^2, & \text{under point 2b,} \end{cases} \tag{43}$$

of Proposition 2, with $\Omega_{i,\alpha}$ the actual stuck speed of the i^{th} rotor and $\Omega_{i,r}$, Ω_{i,α_w} two references under two cases 2a (36) and 2b (37) of Proposition 2.

Proof. The look-up Table 1 is constructed by using the relation $\mathbf{E}[\boldsymbol{d}] = W_i \Delta\Omega_i^2$ as in (39) with W_i the i^{th} column of W from (30). Then, by generalizing the last row of Table 1, we arrive to:

$$\mathbf{E}[d_3] = (-1)^i \Delta\Omega_i^2, \tag{44}$$

with $\Delta\Omega_i^2$ as in (40), which further leads to the use as in (43), completing the proof. □

Note that, due to realistic noises and implementation mismatches, the isolation algorithm can be relaxed by only checking the signs of the elements (d_1, d_2, d_3) instead of following exactly the indications given in Table 1.

4. SIMULATION RESULTS

In this section, we consider the simulation model (1)-(7) of a quadcopter platform characterized by:

- $m = 0.028$, $J_x = J_y = 1.4 \times 10^{-5}$, $J_z = 2.2 \times 10^{-5}$;
- $L = 0.065$, $K_T = 3.16 \times 10^{-10}$, $b = 7.94 \times 10^{-12}$;
- each rotor has maximum speed $\omega_{\max} = 22000$.

The quadcopter tracks a feasible smooth trajectory which respects the maximum rotor speed $\omega_{\max}$ and other initial/final conditions. The yaw angle reference ψ_r is fixed at zero. The fault scenario is as follows:
- *From* 0 *to* 2.5 *seconds:* nominal functioning;
- *From* 2.5 *to* 3 *seconds:* the 4^{th} rotor is stuck at its previous rotating speed (as shown in Fig.3 with $\omega_{4,\alpha_1} = \alpha_1\Omega_{\max}$ with $\alpha_1 = 0.66$);
- *From* 3 *to* 4 *seconds:* the 4^{th} rotor is stuck at another speed of $\omega_{4,\alpha_2} = \alpha_2\Omega_{\max}$ with $\alpha_2 = 0.68$.

The design parameters concerning the FTC controller are gathered in Table 2. The position and attitude controllers (cf. Fig.1) run at the frequencies of 10 Hz and 100 Hz, respectively. The NMPC optimization problems are solved using Pyomo (Hart et al., 2011) in Python 3.0.

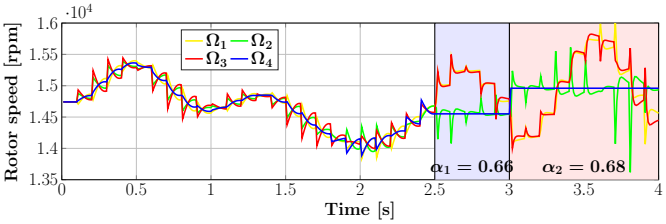


Fig. 3. Rotor speeds values under stuck fault simulation.

We firstly provide the results of the four rotor speeds in Figure 3. After the nominal operation from 0 to 2.5 seconds, the 4^{th} rotor is stuck at the previous speed represented by $\alpha_1 = 0.66$ from 2.5 to 3 seconds and at $\alpha_2 = 0.68$ from 3 to 4 seconds which is illustrated by the blue line remaining constant at two different values.

The fault diagnosis module detailed in Section 3.2 succeeds in detecting the stuck faults within two sampling time periods (0.02 seconds) as given in Figure 4 by using the threshold γ from (41) with $\beta = 10^{-2}$, i.e., providing an acceptable speed tracking mismatch of 10% of the maximum rotor speed $\Omega_{\max} = 22000$ rpm. Note that, a smaller value of γ can be employed but may result in excessive sensitivity (i.e., false alarms). As a part of the fault diagnosis module, the result of the torque observer (32) is given in Figure 5. We provide only the estimated value of the roll torque τ_{ϕ_p} as in (32) plotted as a solid

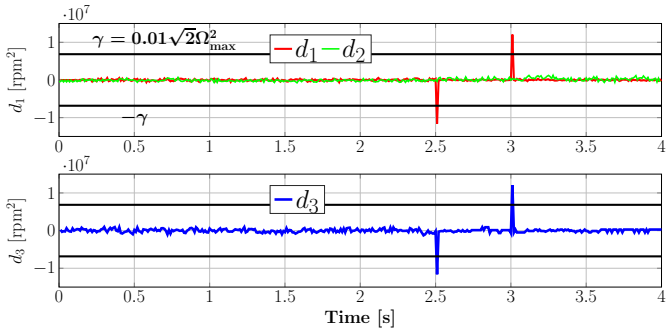


Fig. 4. Values of residual vector $\boldsymbol{d} = [d_1\ d_2\ d_3]^\top$.

green line since the estimated results are noisy due to the usage of the backward Euler method in (32) (can be seen from the enlarging circle) which significantly reduce the clarity of the figure.

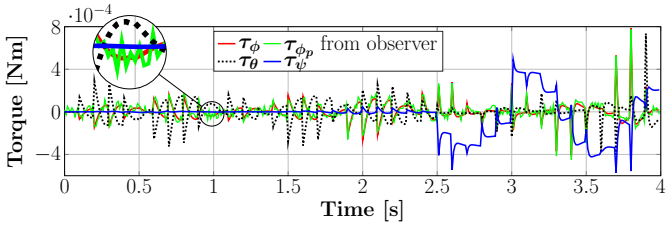


Fig. 5. Torques results.

Using the information of the faults provided by the fault diagnosis module, the proposed FTC scheme ensures the trajectory tracking capability of the quadcopter system as shown by the convergence of the 3D positions over time in Figure 6. Finally, the computing time of the NMPC attitude controllers is plotted in Figure 7. It can be observed that under nominal cases, on average, the NMPC controller (19) (plotted in red line) requires 49.5 milliseconds per step (given by blue line) to compute while the NMPC scheme under fault as in (28) (plotted in green line) yields the average computing time of 60 milliseconds. This is due to the use of the prediction model for the yaw torque given by (26) and the longer prediction horizon $N_f = 8$ (w.r.t. $N_p = 5$ under nominal functioning as given in Table 2). We also notice that the computation times are higher than the sampling time (i.e., 0.01 seconds) which would make the control strategy, in its present form, unsuitable for implementation. However, we are confident that further work (e.g., to refine the algorithm's implementation) and/or use advanced solving techniques (Mayne et al., 2000) will mitigate the issue.

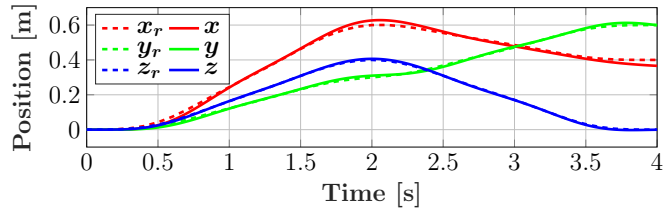


Fig. 6. Position tracking results.

Table 2. Parameters of the attitude controller (cf. Fig.1).

Under nominal functioning	Parameters	Q_η	$Q_{\Delta\mathcal{T}}$	N_p
	Values	$\boldsymbol{I}_3$	$0.1\boldsymbol{I}_3$	5
Under fault of a unique stuck rotor	Parameters	$Q_{\hat{\eta}}$	$Q_{\Delta\widehat{\mathcal{T}}}$	N_f
	Values	$\boldsymbol{I}_2$	$0.1\boldsymbol{I}_2$	8

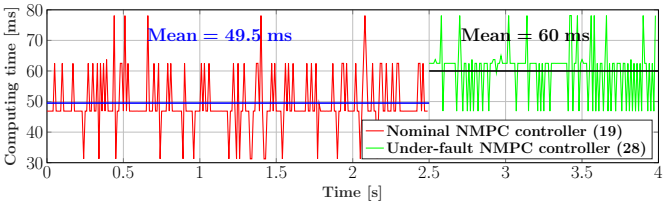


Fig. 7. Computing time of the NMPC attitude controllers.

5. CONCLUSION

This paper presented the designs of a trajectory tracking FTC (Fault Tolerant Control) controller for a quadcopter system under rotor saturation constraints and fault of a unique stuck rotor. The reconfiguration module employs an NMPC scheme to ensure the constraints on the four rotor speeds and functions in two different modes, i.e., nominal mode and 'under-fault' mode. The FDI (Fault Detection and Isolation) module estimates the differences between the references of torques and their actual values in order to provide the speed tracking errors of the rotors and then, to detect the stuck fault. The proposed methods were validated through extensive simulations and show promise for experimental tests. Future works will analyze the impacts of measurement errors and of the internal dynamics of the rotors.

REFERENCES

Chen, W. and Jiang, J. (2005). Fault-tolerant control against stuck actuator faults. *IEE Proceedings-Control Theory and Applications*, 152(2), 138–146.

Freddi, A., Lanzon, A., and Longhi, S. (2011). A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC Proceedings Volumes*, 44(1), 5413–5418.

Hart, W.E., Watson, J.P., and Woodruff, D.L. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3), 219–260.

Hasan, A. and Johansen, T.A. (2018). Model-based actuator fault diagnosis in multirotor uavs. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1017–1024. IEEE.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Nascimento, T.P. and Saska, M. (2019). Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*.

Nguyen, N.T., Prodan, I., and Lefèvre, L. (2018). Flat trajectory design and tracking with saturation guarantees: a nano-drone application. *International Journal of Control*, 1–14, Article in Press.

Nguyen, N.T., Prodan, I., Stoican, F., and Lefèvre, L. (2017). Reliable nonlinear control for quadcopter trajectory tracking through differential flatness. *IFAC-PapersOnLine*, 50(1), 6971–6976.

Shao, X., Hu, Q., Shi, Y., and Jiang, B. (2018). Fault-tolerant prescribed performance attitude tracking control for spacecraft under input saturation. *IEEE Transactions on Control Systems Technology*.

Stoican, F., Petzke, F., Prodan, I., and Streif, S. (2018). Hierarchical control with guaranteed fault diagnosability. *IFAC-PapersOnLine*, 51(24), 1105–1110.