Bulletin of the JSME Vol.8, No.5, 2021
Mechanical Engineering Journal

Nonlinear model predictive control for hexacopter with failed rotors based on quaternions — simulations and hardware experiments —

Yusuke AOKI*, Yuta ASANO**, Akihiko HONDA**, Norizumi MOTOOKA**, Kenta HOSHINO* and Toshiyuki OHTSUKA*

* Department of Systems Science, Graduate School of Informatics, Kyoto University
Kyoto, Kyoto 606-8501, Japan
E-mail: hoshino@i.kyoto-u.ac.jp

** Advanced Technology R&D Center, Mitsubishi Electric Corporation
Amagasaki, Hyogo 661-8661, Japan

Received: 27 May 2021; Revised: 2 August 2021; Accepted: 8 September 2021

Abstract

This work applies real-time nonlinear model predictive control (NMPC) to fault-tolerant control problems of an unmanned aerial vehicle (UAV) with failed rotors. In the control problem, a hexacopter with up to three failed rotors out of the six available rotors is considered. The NMPC approach includes a quaternion-based nonlinear model of the hexacopter as well as constraints in the thrusts to consider the inherent nonlinearities of UAVs. The proposed method aims to achieve real-time optimization of the NMPC in the on-board computers without any linearization. We explore all possible scenarios in up to three rotor failures and demonstrate control designs in the NMPC for these scenarios. The simulation results indicate that by using the quaternion model, the position and attitude of a hexacopter can be controlled from a large inclined initial state with a non-zero angular velocity and falling velocity. Moreover, the results reveal that the quaternion model is superior to the Euler angle model in terms of the computation time. We also conduct hardware experiments using an actual hexacopter with a failed rotor to demonstrate the real-time NMPC optimization. The results of the simulations and hardware experiments demonstrate that the NMPC can deal with various operation conditions of a hexacopter in a unified manner, with only minor modifications in the performance index.

Keywords : Nonlinear control, Predictive control, Optimal control, Unmanned aerial vehicle, Fault-tolerant control, Quaternion

1. Introduction

The emergence of control methodologies for unmanned aerial vehicles (UAVs) has resulted in various applications, and advanced control methods for UAVs constitute ongoing research topics. To date, UAVs have been demonstrated to be useful in agriculture (Rasmussen et al., 2013), logistics (Fink et al., 2011), disaster relief (Greer et al., 2002), surveying (Murphy et al., 2008), and biological animal research (Selby et al., 2011), among other fields. With the increasing demand for UAVs, the need for further advanced control methods has expanded to enable UAVs to fly over people, particularly for applications such as safety requirements in urban environments. However, UAVs are at risk of accidents such as crashes or collisions. For example, an accident occurred in which a drone crashed into a crowd, injuring six people, in Japan (KYODO, 2017). Such possible situations necessitate the development of a control method for UAVs to counter the risk of failure.

One possible failure that may cause serious accidents in UAVs is rotor failure, which requires the development of fault-tolerant control. The flight of UAVs is inherently fragile owing to the loss of thrusts by rotors; if one rotor fails, the specified equilibrium may no longer be stabilized with the thrusts that are generated by the remaining rotors. Various fault-tolerant control methods have been proposed for UAVs, such as sliding mode control (Sharifi et al., 2010), gain-scheduled

Paper No.21-00204
J-STAGE Advance Publication date : 18 September, 2021
[DOI: 10.1299/mej.21-00204]

PID control (Milhim et al., 2010; Sadeghzadeh et al., 2012), PD control (Du et al., 2015), model predictive control (Yu et al., 2013; Aoki et al., 2018, 2019; Izadi et al., 2011; Kamel et al., 2015), LQ control (Mueller and D'Andrea, 2014), and adaptive control (Avram et al., 2017). Zhang et al. (2013) presented comparisons between such methods for fault-tolerant control. Four of these studies, namely Sharifi et al. (2010), Milhim et al. (2010), Sadeghzadeh et al. (2012), and Yu et al. (2013), considered a reduction in the thrusts in the rotors rather than a complete stop, and Du et al. (2015) considered a case in which only one rotor stopped. In addition, Avram et al. (2017) proposed nonlinear adaptive control as a fault-tolerant control method against a reduction of the thrusts of quadcopters, where the attitude and altitude of quadcopters are controlled. Although Avram et al. (2017) can take into account failures of several rotors, their method cannot be adapted to failures of complete stops of rotors. Therefore, it is necessary to develop fault-tolerant control dealing with complete stops of several rotors, which likely happen in actual accidents. Mueller and D'Andrea (2014) considered cases of quadcopters in which one or two rotors stopped for a linearized model. The linearized model was invalid if the state deviated significantly from the equilibrium. Thus, it is necessary to employ nonlinear UAV models in fault-tolerant control, because a large inclined attitude is likely to occur. Hexacopters may be more suitable for fault tolerance in actual applications than quadcopters from the viewpoint of the mechanical structure. They may be robust against rotor failures owing to their redundancies in rotor controls. Furthermore, as noted in Nascimento and Saska (2019), it is necessary to consider the number of rotors in fault-tolerant control. Therefore, it is desirable to develop a nonlinear control method for the fault-tolerant control of hexacopters with failed rotors. Kamel et al. (2015) considered tracking control for a nonlinear hexacopter model with one failed rotor and presented experimental results. However, it remains necessary to develop a fault-tolerant control method for a nonlinear hexacopter model with multiple rotor failures. Moreover, the control method should enable implementation in the on-board computers of hexacopters.

As implied by Kamel et al. (2015), Aoki et al. (2019, 2018), nonlinear model predictive control (NMPC) is a promising approach for the fault-tolerant control of hexacopters, owing to its flexibility and applicability to nonlinear systems. Ideally, the NMPC can achieve feedback control for nonlinear systems by solving receding-horizon optimal control problems in real time. Thus, if suitable receding-horizon control problems can be formulated, fault-tolerant control problems with failed rotors can be performed. In our previous studies (Aoki et al., 2018, 2019), we developed NMPC for fault-tolerant control using nonlinear models of hexacopters with failed rotors. These works achieved fault-tolerant control with two or three failed rotors out of the six available rotors. However, certain issues remain unsolved. The first is the singularity issue in the Euler angle representation for the attitude of a hexacopter (Aoki et al., 2018, 2019). The representation may fall into singularities, which is known as gimbal lock. This should be resolved in actual applications, because the large inclined attitude that is likely to occur with rotor failures may cause a singular attitude. The above issue can be resolved by employing quaternions in the attitude representation instead of Euler angles, as in many other studies (Fresk and Nikolakopoulos, 2013; Alaimo et al., 2013). Another issue is to validate the real-time implementation of NMPC on hardware, which is inevitable in actual industrial applications. Aoki et al. (2018, 2019) demonstrated the validity of fault-tolerant control only in simulations. Therefore, it is indispensable to verify the executability of real-time computation for NMPC in hardware experiments. Hexacopters exhibit redundancies in the rotor control. However, the real-time implementation of NMPC in hexacopters is more challenging than that of quadcopters. This is because a higher number of inputs causes an increase in the computational effort required for the NMPC. Therefore, an implementable NMPC method should be provided for the fault-tolerant control of hexacopters. The NMPC implementation should avoid the increase in computational effort of the NMPC to solve the receding-horizon control problems in real time.

This study provides an NMPC approach for fault-tolerant control using nonlinear hexacopter models with failed rotors, aimed at real-time implementation in on-board computers. The considered failures of the rotors are modeled as complete stops. We demonstrate the design of a performance index in receding-horizon optimal control problems for all possible scenarios for up to three failed rotors. This enables the implementation of the NMPC method for various failure cases in a unified manner. In terms of real-time implementation, this study employs quaternions in the attitude expressions of the hexacopter, which is an extension of previous works (Aoki et al., 2018, 2019). The quaternions do not have any singularities in the attitude expression. An additional motivation for introducing the quaternion attitude representation is to reduce the computational cost of the NMPC. Quaternions are known to be more computationally efficient than Euler angles in attitude representations (Fresk and Nikolakopoulos, 2013; Alaimo et al., 2013). This study shows that quaternions are also favorable in the NMPC implementation. NMPC for UAVs involves substantial computation of the attitude representation. Accordingly, the quaternion representation in NMPC is expected to be superior to the Euler angle representation in terms of the computational efficiency.

We present simulation and experimental results to validate the proposed approach. In the simulations, we considered seven failure cases: 1) one rotor stops, 2) two rotors in opposite positions stop, 3) two rotors separated by a functioning rotor stop, 4) two adjacent rotors stop, 5) three rotors in alternate positions stop, 6) two adjacent rotors and a separated rotor stop, and 7) three adjacent rotors stop. These cases consider all possible failures of up to three hexacopter rotors, and this study demonstrates a control design for the position and attitude control in such cases. We controlled the position and attitude from a large inclined initial state with a non-zero angular velocity and falling velocity, which causes singularity in the Euler angle model. Furthermore, the simulation results reveal that the quaternion model is superior to the Euler angle model in terms of the computation time. In addition to the simulations, we conducted hardware experiments using an actual hexacopter with a failed rotor to demonstrate real-time optimization for NMPC, which is another extension of previous works (Aoki et al., 2018, 2019). To demonstrate the effectiveness of the proposed control method, we compared a controller for a hexacopter with six available rotors and a controller for a hexacopter with a failed rotor in terms of the control performance when one rotor stops. Although we conducted experiments for only one failed rotor cases due to physical limitations in the experiments, the results showed that NMPC could achieve the real-time optimization for the actual hexacopter. The results of the simulations and hardware experiments demonstrate that NMPC can deal with various operation conditions of a hexacopter in a unified manner with only minor modifications in the performance index.

The remainder of this paper is organized as follows. In Section 2, we describe the state equation of a hexacopter and explain the cases of failed rotors. In Section 3, we formulate the NMPC of the hexacopter for each case of failed rotors. In Sections 4 and 5, we present the results of the simulations and hardware experiments, respectively. Finally, the paper is concluded in Section 6.

2. Modeling

2.1. State equation of hexacopter

Our hexacopter was based on the configuration depicted in Fig. 1. The rotors were numbered and the coordinate axes were set as shown in Fig. 1. The coordinate axes e^{i}_j and e^{b}_j $(j = 1, 2, 3)$ denote the inertial frame and body-fixed frame, respectively. The state vector and control input vector of the hexacopter are defined as follows:

$$x = [\xi^{\mathrm{T}}\ \dot{\xi}^{\mathrm{T}}\ \omega^{\mathrm{T}}\ q^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{13}, \quad u = [f_1\ \cdots\ f_6]^{\mathrm{T}} \in \mathbb{R}^6, \tag{1}$$

where $\xi = [x\ y\ z]^{\mathrm{T}} \in \mathbb{R}^3$ is the hexacopter position, $\omega = [\omega_1\ \omega_2\ \omega_3]^{\mathrm{T}} \in \mathbb{R}^3$ is the vector of the angular velocity in the body frame, $q = [q_0\ q_1\ q_2\ q_3]^{\mathrm{T}}$ is the vector of the unit quaternion describing the hexacopter attitude, and $f_1, \ldots, f_6$ are the six independent thrusts that are generated by the six rotors. In this study, the quaternion is employed in the attitude representation (see Appendix A for details). The values of the elements of q can be obtained from the unit vector representing the rotation axis direction $n = (n_1, n_2, n_3)^T$ and its rotation angle α, as illustrated in Fig. 1, using

$$q = \left[\cos\left(\frac{\alpha}{2}\right)\ n_1 \sin\left(\frac{\alpha}{2}\right)\ n_2 \sin\left(\frac{\alpha}{2}\right)\ n_3 \sin\left(\frac{\alpha}{2}\right)\right]^T. \tag{2}$$

The sum of the thrusts f_{all} and torque τ can be written as $f_{\mathrm{all}} = \Sigma_{i=1}^{6} f_i \in \mathbb{R}$ and $\tau = Tu - D\omega \in \mathbb{R}^3$, respectively, where the matrices $T \in \mathbb{R}^{3\times6}$ and $D \in \mathbb{R}^{3\times3}$ are defined as follows:

$$T = \begin{bmatrix} -\frac{1}{2}l & -l & -\frac{1}{2}l & \frac{1}{2}l & l & \frac{1}{2}l \\ -\frac{\sqrt{3}}{2}l & 0 & \frac{\sqrt{3}}{2}l & \frac{\sqrt{3}}{2}l & 0 & -\frac{\sqrt{3}}{2}l \\ k & -k & k & -k & k & -k \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \gamma \end{bmatrix}, \tag{3}$$

in which l is the distance from the hexacopter mass center to each rotor, k is a coefficient representing the relationship between the thrust and reaction torque, and γ is the coefficient of air resistance. In this study, we consider only the

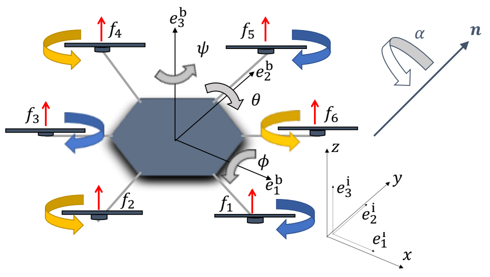

Fig. 1 Hexacopter configuration and coordinate axes

aerodynamic drag in the yaw axes and ignore those in the roll and pitch axes because the study of Mueller and D'Andrea (2014) indicates that the assumption employing the form of $D\omega$ is supported by experimental data.

From Newton's equation of motion, the translational motion can be described as

$$m\ddot{\xi} = QF^{\mathrm{b}} - mge_3^{\mathrm{i}}, \tag{4}$$

where m is the hexacopter mass, g is the gravitational acceleration, $Q \in \mathbb{R}^{3\times3}$ is defined as

$$Q = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}, \tag{5}$$

and F^b is described as $F^b = [0\ 0\ f_{\mathrm{all}}]^{\mathrm{T}}$. According to Euler's equation regarding rigid body dynamics and the kinematics equation of a quaternion, the rotational motion can be described as

$$\begin{cases} I\dot{\omega} + \omega \times I\omega = Tu - D\omega \\ \dot{q} = \frac{1}{2}\Omega q, \end{cases} \tag{6}$$

where $I \in \mathbb{R}^{3\times3}$ is the inertia matrix of the hexacopter, and the matrix $\Omega \in \mathbb{R}^{4\times4}$ is the skew-symmetric matrix given by

$$\Omega = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}. \tag{7}$$

Accordingly, from Eqs. (4) and (6), the hexacopter state equation can be expressed as

$$\dot{x} = \begin{bmatrix} \dot{\xi} \\ \frac{QF^b}{m} - ge_3^{\mathrm{i}} \\ -I^{-1}(\omega \times I\omega - Tu + D\omega) \\ \frac{1}{2}\Omega q \end{bmatrix} := f(x, u). \tag{8}$$

In this study, quaternions were used in the nonlinear model of the hexacopter. This is one of the extensions of this study from our previous works (Aoki et al., 2018, 2019), which employed Euler angles. Quaternions can prevent singularity issues that may occur in Euler angle representations. This study introduces quaternions not only because of singularity issues but also the computational advantages. As noted in several studies on the control of UAVs (Fresk and Nikolakopoulos, 2013; Alaimo et al., 2013), the quaternion representation is more computationally efficient than the Euler angle representation. In this study, the quaternions allowed for the real-time implementation of NMPC for the fault-tolerant control of hexacopters in the simulations and experiments.

2.2. Cases of failed rotors

The aim of this research was to control the position and attitude of a hexacopter with various cases of failed rotors in a unified manner using NMPC. It was assumed that we had the means to detect the loss of thrusts in the failed rotors and to determine which rotors failed. For the modeling, all rotors were assumed to be on the vertices of a regular hexagon, as indicated in Fig. 1. Therefore, we considered the following seven failure cases without loss of generality, as shown in Figs. 2 to 8. It should be noted that these cases included all possible failure cases for up to three rotors.

Failure of one rotor (case 1) In case 1 (Fig. 2), rotor no. 6 stops, and the component f_6 of the control input u in Eq. (8) is set to zero.

Failure of two rotors in opposite positions (case 2) In case 2 (Fig. 3), rotor nos. 3 and 6 stop, and the components f_3 and f_6 of the control input u in Eq. (8) are set to zero. As can be observed from the configuration of the rotors, this case is similar to that of a quadcopter, but differs in terms of the rotor rotation direction.

Failure of two rotors separated by a functioning rotor (case 3) In case 3 (Fig. 4), rotor nos. 2 and 6 stop, and the components f_2 and f_6 of the control input u in Eq. (8) are set to zero. Note that because only rotor no. 4 rotates in a different direction and the other three rotate in the same direction, the rotation of the yaw direction is very difficult to control.

Failure of two adjacent rotors (case 4) In case 4 (Fig. 5), rotor nos. 1 and 6 stop, and the components f_1 and f_6 of the control input u in Eq. (8) are set to zero. As can be observed from the configuration of the rotors, no stable static equilibrium exists, and finding a stable equilibrium with non-zero angular velocity is not a trivial task.

Failure of three rotors in alternate positions (case 5) In case 5 (Fig. 6), rotor nos. 2, 4, and 6 stop and the components f_2, f_4, and f_6 of the control input u in Eq. (8) are set to zero. Note that because all of the remaining rotors rotate in the same direction, the yaw direction inevitably rotates.

Failure of two adjacent rotors and a separated rotor (case 6) In case 6 (Fig. 7), rotor nos. 1, 3, and 6 stop and the components f_1, f_3, and f_6 of the control input u in Eq. (8) are set to zero. Similar to case 4, as can be observed from the configuration of the rotors, no stable static equilibrium exists and it is necessary to specify a stable equilibrium with non-zero angular velocity.

Failure of three adjacent rotors (case 7) In case 7 (Fig. 8), rotor nos. 1, 5, and 6 stop and the components f_1, f_5, and f_6 of the control input u in Eq. (8) are set to zero. As in cases 4 and 6, no static equilibrium exists and it is necessary to specify a stable equilibrium with non-zero angular velocity.

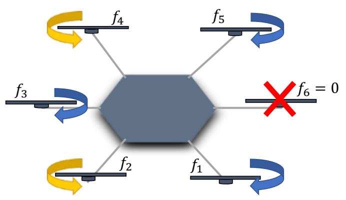

Fig. 2 One rotor stops (case 1)

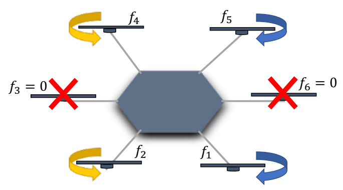

Fig. 3 Two rotors in opposite positions stop (case 2)

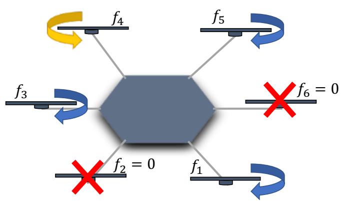

Fig. 4 Two rotors separated by a functioning rotor stop (case 3)

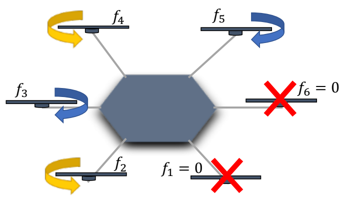

Fig. 5 Two adjacent rotors stop (case 4)

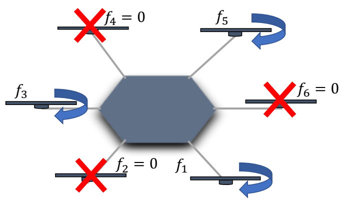

Fig. 6 Three rotors in alternate positions stop (case 5)

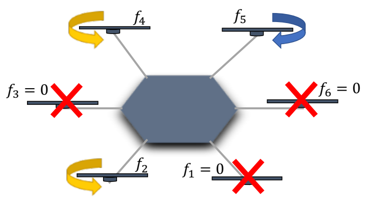

Fig. 7 Two adjacent rotors and a separated rotor stop (case 6)

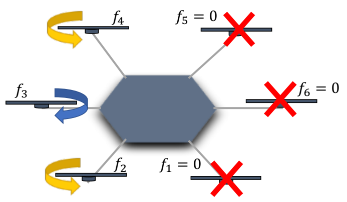

Fig. 8 Three adjacent rotors stop (case 7)

3. Control method

The control problem of a hexacopter with failed rotors can be considered as an optimal control problem. In this study, NMPC was used to perform feedback control by numerically solving an optimal control problem. For this purpose, we used a real-time numerical algorithm known as C/GMRES (Ohtsuka, 2004, 2011). In this section, we summarize the NMPC and formulate the hexacopter control problem.

3.1. Overview of NMPC

We consider the state equation (8) with an equality constraint $C(x(t), u(t)) = 0$, where C is a d-dimensional vector-valued function. An inequality constraint can be transformed into an equality constraint by introducing a dummy input, as demonstrated in Ohtsuka (2004) (see also Section 3.3). Thus, only equality constraints are considered in this study.

NMPC methods solve a finite-horizon optimal control problem at each instant and update the solution to the optimal control problem in succession so as to use the optimal control input as though it acts as feedback control input. In general, NMPC methods solve the following finite-horizon control problem to predict the optimal system behavior:

$$\begin{aligned} &\min_{\bar{u}(\cdot)} \; J \\ &\text{s.t.} \;\; \dot{\bar{x}}(\tau; t) = f(\bar{x}(\tau; t), \bar{u}(\tau; t)), \quad C(\bar{x}(\tau; t), \bar{u}(\tau; t)) = 0, \end{aligned} \tag{9}$$

where $\bar{x}(\tau; t)$ and $\bar{u}(\tau; t)$ denote the state and control input of (8) for the prediction of the finite horizon with $\tau \in [t, t + T]$ and $T \geq 0$, satisfying $\bar{x}(t; t) = x(t)$ with the value of the actual state $x(t)$ at t, and J is a performance index given by

$$J := \varphi(\bar{x}(t + T; t), p(t + T)) + \int_t^{t+T} L(\bar{x}(\tau; t), \bar{u}(\tau; t), p(\tau))d\tau \tag{10}$$

with a time-varying parameter $p(\tau)$. We use overline expressions for variables to denote predictions on the horizon. In this problem, various control problems can be formulated by selecting the appropriate functions φ and L for the performance index. Note that the predicted state $\bar{x}(\tau; t)$ and input $\bar{u}(\tau; t)$ for $\tau \in [t, t + T]$ are simply introduced to predict the evolution of the state and control input; these do not necessarily coincide with the actual state and control input.

The optimal control $\bar{u}_{\text{opt}}(\tau; t)$ $(t \leq \tau \leq t + T)$ that minimizes J is determined as a time function within the horizon. Accordingly, the actual control input $u(t)$ at time t is given by the initial value of $\bar{u}_{\text{opt}}(\tau; t)$. That is, $u(t) = \bar{u}_{\text{opt}}(t; t)$ is applied to the actual system (8). The control $\bar{u}_{\text{opt}}(t; t)$ acts as a state feedback control. This feedback control method is referred to as NMPC or nonlinear receding-horizon control.

3.2. C/GMRES method

We used the C/GMRES method (Ohtsuka, 2004) to solve the NMPC problems in real time. We outline the C/GMRES algorithm based on Ohtsuka (2004) with certain modifications and additional expositions.

To describe the algorithm, we introduce the Hamiltonian

$$H(x, \lambda, u, \mu, p) := L(x, u, p) + \lambda^{\mathrm{T}} f(x, u) + \mu^{\mathrm{T}} C(x, u), \tag{11}$$

where $\lambda \in \mathbb{R}^{13}$ is the adjoint variable and $\mu \in \mathbb{R}^d$ is the Lagrange multiplier. Subsequently, we consider the time discretization of problem (9) on the prediction horizon with the time step $\Delta\tau = T/N$, where N is the number of grids on the horizon and T is the length of the horizon in (10). Using this setting, we obtain the discretized approximation of the Euler–Lagrange equation of (9) by means of the Hamiltonian (11):

$$\bar{x}_{i+1} = \bar{x}_i + f(\bar{x}_i, \bar{u}_i)\Delta\tau, \quad \bar{x}_0 = x(t), \tag{12a}$$

$$\bar{\lambda}_i = \bar{\lambda}_{i+1} - \frac{\partial H}{\partial x}(\bar{x}_i, \bar{\lambda}_{i+1}, \bar{u}_i, \bar{\mu}_i, p_i)^{\mathrm{T}}\Delta\tau, \tag{12b}$$

$$\bar{\lambda}_N = \frac{\partial \varphi}{\partial x}(\bar{x}_N, p_N)^{\mathrm{T}}, \tag{12c}$$

$$\frac{\partial H}{\partial u}(\bar{x}_i, \bar{\lambda}_{i+1}, \bar{u}_i, \bar{\mu}_i, p_i) = 0, \tag{12d}$$

$$C(\bar{x}_i, \bar{u}_i) = 0, \tag{12e}$$

where $\bar{x}_i$, $\bar{\lambda}_i$ $\bar{u}_i$, $\bar{\mu}_i$, and p_i correspond to the discretized approximations of $\bar{x}(t + i\Delta\tau; t)$, $\bar{\lambda}(t + i\Delta\tau)$, $\bar{u}(t + i\Delta\tau)$, $\bar{\mu}(t + i\Delta\tau)$, and $p(t + i\Delta\tau)$, respectively, for $i = 0, \ldots, N$. We may use notations such as $\bar{x}_i(t)$ instead of $\bar{x}_i$ to specify the time instant t, which is the left endpoint of the prediction horizon. Thereafter, once a sequence of the control input $\{\bar{u}_i\}_{i=0}^{N-1}$ and a sequence of the Lagrange multiplier $\{\bar{\mu}_i\}_{i=0}^{N-1}$ have been provided, we can obtain a sequence of the state $\{\bar{x}_i\}_{i=0}^{N}$ using Eq. (12a) as a function of $\{\bar{u}_i\}_{i=0}^{N-1}$ and $\{\bar{\mu}_i\}_{i=0}^{N-1}$. Moreover, once the sequence of the state has been provided in addition to the control input, a sequence of $\{\bar{\lambda}_i\}_{i=0}^{N}$ is given by Eqs. (12b) and (12c). If these sequences $\{\bar{x}_i\}_{i=0}^{N}$, $\{\bar{u}_i\}_{i=0}^{N-1}$, $\{\bar{\lambda}_i\}_{i=0}^{N}$, and $\{\bar{\mu}_i\}_{i=0}^{N-1}$ satisfy conditions (12d) and (12e), such a sequence of $\bar{u}_i$ is an approximate solution to the receding-horizon control of (9). To obtain the sequences $\{\bar{u}_i\}_{i=0}^{N-1}$ and $\{\bar{\mu}_i\}_{i=0}^{N-1}$ as an approximate optimal solution to the receding-horizon control problem (9), we solve the equation

$$F(U(t), x(t), t) := \begin{bmatrix} \frac{\partial H}{\partial u}(\bar{x}_0(t), \bar{\lambda}_1(t), \bar{u}_0(t), \bar{\mu}_0(t), p_0(t))^{\mathrm{T}} \\ C(\bar{x}_0(t), \bar{u}_0(t)) \\ \vdots \\ \frac{\partial H}{\partial u}(\bar{x}_{N-1}(t), \bar{\lambda}_N(t), \bar{u}_{N-1}(t), \bar{\mu}_{N-1}(t), p_{N-1}(t))^{\mathrm{T}} \\ C(\bar{x}_{N-1}(t), \bar{u}_{N-1}(t)) \end{bmatrix} = 0 \tag{13}$$

with respect to $U(t) = [\bar{u}_0(t)^{\mathrm{T}}, \bar{\mu}_0(t)^{\mathrm{T}}, \dots, \bar{u}_{N-1}(t)^{\mathrm{T}}, \bar{\mu}_{N-1}(t)^{\mathrm{T}}]^{\mathrm{T}}$. Note that $F(U(t), x(t), t)$ also denotes the dependence on the value of the actual state $x(t)$. If it is possible to solve Eq. (13) in real time, we can obtain the control input according to $u(t) = \bar{u}_0(t)$; however, this is challenging. Ohtsuka (2004) proposed the introduction of the differential equation

$$\dot{F}(U(t), x(t), t) = -\zeta F(U(t), x(t), t), \tag{14}$$

where $\zeta > 0$ and $\dot{F}(U(t), x(t), t)$ is the time derivative of $F(U(t), x(t), t)$. Note that $F(U(t), x(t), t) \equiv 0$ if $F(U(0), x(0), 0) = 0$ and the value $F(U(t), x(t), t)$ tends towards zero in (14), even when $F(U(0), x(0), 0) \neq 0$. According to Eq. (14) and the chain rule, we obtain

$$\frac{\partial F}{\partial U}\dot{U}(t) = -\zeta F(U(t), x(t), t) - \frac{\partial F}{\partial x}\dot{x}(t) - \frac{\partial F}{\partial t}, \tag{15}$$

where $\dot{x}(t)$ is given by (8) with the values $x(t)$ and $u(t)$. This equation is linear with respect to $\dot{U}(t)$. It is assumed that the regularity of the Jacobian matrix $\frac{\partial F}{\partial U}$, $\dot{U}(t)$ can be obtained by numerical methods such as the GMRES method (Saad and Schultz, 1986) with low computational effort, even when the size of the matrix $\frac{\partial F}{\partial U}$ is large. Therefore, integrating $\dot{U}(t)$ yields the trajectory of $U(t)$ such that $F(U(t), x(t), t)$ satisfies Eq. (14). In the implementation, if the above procedure provides $U(t)$ within each sampling period for the control, $U(t)$ determines the control input $u(t)$ provided to the actual system at each t as $u(t) = \bar{u}_0(t)$.

In the implementation of C/GMRES, it is generally required that $F(U(0), x(0), 0) = 0$ holds to guarantee $F(U(t), x(t), t) \approx 0$ for each $t \geq 0$ with (14). However, this is often difficult. Therefore, we allow the terminal time T in (10) to be time varying; for example,

$$T(t) := T_f(1 - e^{-\alpha t}) \tag{16}$$

with $T_f, \alpha > 0$. When using such a function $T(t)$, the time step $\Delta\tau = T(0)/N$ becomes zero at $t = 0$. Therefore, according to (13), we only need to solve

$$\frac{\partial H}{\partial u}(\bar{x}_0(0), \bar{\lambda}_1(0), \bar{\mu}_0(0), p_0(0)) = 0, \ C(\bar{x}_0(0), \bar{u}_0(0)) = 0 \tag{17}$$

with respect to $\bar{u}_0(0)$ and $\bar{\mu}_0(0)$ when $t = 0$. As $\bar{x}_0(0)$ and $\bar{\lambda}_1(0)$ are given by $\bar{x}_0(0) = x(0)$ and $\bar{\lambda}_1(0) = \frac{\partial \varphi}{\partial x}(x(0), p(0))$, it is often possible to solve the equations in (17) numerically or analytically. Once $\bar{u}_0(0)$ and $\bar{\mu}_0(0)$ have been obtained, we can begin the algorithm guaranteeing $F(U(0), x(0), 0) = 0$.

The C/GMRES algorithm is summarized as follows:

Algorithm 1 C/GMRES

```
1: Set the sampling period for the control $\Delta t$. Set $t = 0$, measure the value of $x(0)$, and set $\bar{x}_0(0) = x(0)$ and $\bar{\lambda}_1(0) = \frac{\partial \varphi}{\partial x}(\bar{x}_0(0), p_0(0))$. Solve (17) with respect to $\bar{u}_0(0)$ and $\bar{\mu}_0(0)$ analytically or numerically.
2: Determine the control input for system (8) at $t$ as $u(t) = \bar{u}_0(t)$.
3: Calculate the sequences $\{\bar{x}_i(t)\}_{i=0}^{N}$ and $\{\bar{\lambda}_i(t)\}_{i=0}^{N}$ with (12a), (12b), and (12c).
4: Solve Eq. (15) with respect to $\dot{U}(t)$ using the GMRES method with the number of iterations $k_{\max}$, and integrate $\dot{U}(t)$ as $U(t + \Delta t) = U(t) + \dot{U}(t)\Delta t$.
5: Measure $x(t + \Delta t)$ and set $t := t + \Delta t$. Return to step 2.
```

Note that the value of Δt is not necessarily identical to the value of $\Delta\tau$. The highlight of C/GMRES is the tracing of the evolution of the control sequence vector $U(t)$ in t with Eq. (15). This is in contrast to solving Eq. (13) at each sampling time. C/GMRES enables the control input $u(t) = \bar{u}_{\mathrm{opt}}(t)$ to be obtained for each sampling period. We employ this method for the fault-tolerant control of hexacopters. This method solves the receding-horizon control problem within each sampling time, in which we use the nonlinear model (8) without any linearization. The automatic code generation system for C/GMRES, namely AutoGenU, has been published for symbolic computation languages such as Mathematica and Maple (Ohtsuka, 2015). We used AutoGenU for Maple (Cybernet Systems Co., 2016) in the simulations and experiments.

Moreover, the introduction of the quaternion model helps to avoid an increase in computational efforts in the computation of NMPC. Euler angles, quaternions, and other methods can be used for the attitude representation of UAVs. When using the Euler angle representation, the attitude matrix corresponding to Q in (5) is given by the trigonometric functions of the roll, pitch, and yaw angles (see Appendix B.1). As noted in Fresk and Nikolakopoulos (2013), Alaimo et al. (2013), the computational efforts for calculating the trigonometric functions tend to be large. However, the quaternion representations yield the attitude matrix Q of (5) with the multiplications of the quaternion vector elements, which requires less computational effort than the Euler angle representation (Fresk and Nikolakopoulos, 2013; Alaimo et al., 2013). In C/GMRES, the state equation of (8) and its derivatives are required in various operations, as can be observed in (12b), (13), and (15). The quaternion is effective in terms of computational efforts for achieving real-time calculation of the NMPC in this study.

3.3. Performance index and constraints for hexacopter with failed rotors

We describe the design of the performance index of the NMPC for the fault-tolerant control of hexacopters. Our aim is to develop a control method such that hexacopters get converged to a reference position after the complete stops of rotors happen, namely a kind of regulation problem under the failure of rotors. This study presents the design for the rotor failures described in Subsection 2.2 in a unified manner; we describe a performance index for all the cases of rotor failures described in Subsection 2.2. This study shows that the fault-tolerant control can be achieved for all the failure cases only by choosing the design parameters in the performance index. Furthermore, the constraints on the thrusts, which are significant in actual hexacopters, are considered.

In the nonlinear model of the hexacopter, which is expressed by Eq. (8), the thrust generated in each rotor is constrained as follows:

$$u_{\min} \le u_i \le u_{\max}. \tag{18}$$

The dummy inputs u_i' $(i = 1, \dots, 6)$ are subsequently introduced to transform this inequality into the following equality:

$$C_i(u_i, u_i') := \left(u_i - \frac{u_{\min} + u_{\max}}{2}\right)^2 + u_i'^2 - \left(\frac{u_{\max} - u_{\min}}{2}\right)^2 = 0, \tag{19}$$

where $u_{\min}$ and $u_{\max}$ are the minimum and maximum thrusts, respectively, that each rotor can generate. When u_i and u_i' satisfy the equality constraint (19), we obtain

$$\left(u_i - \frac{u_{\min} + u_{\max}}{2}\right)^2 \le \left(\frac{u_{\max} - u_{\min}}{2}\right)^2. \tag{20}$$

In the above inequality, the term $(u_{\max}+u_{\min})/2$ corresponds to the middle point of the interval $[u_{\min}, u_{\max}]$, and $|u_{\max} - u_{\min}|/2$ is the distance from the middle point to the boundaries of the interval $u_{\min}$ and $u_{\max}$. Thus, according to (20), u_i satisfies the inequality condition (18) when u_i and u_i' satisfy condition (19). The performance index is expressed as

$$\begin{aligned} J = &\sum_{i=1}^{9} s_i\Big(x_i(t+T) - x_{\mathrm{ref}i}(t+T)\Big)^2 + \sum_{i=0}^{3} s_{i+10}\Big(q_{ei}(t+T) - q_{\mathrm{zero}i}\Big)^2 \\ &+ \int_t^{t+T}\left[\sum_{i=1}^{9} s_i\Big(x_i(\tau) - x_{\mathrm{ref}i}(\tau)\Big)^2 + \sum_{i=0}^{3} s_{i+10}\Big(q_{ei}(\tau) - q_{\mathrm{zero}i}\Big)^2 + \sum_{i=1}^{6} r_i\Big(u_i(\tau) - u_{\mathrm{ref}i}\Big)^2 - \sum_{i=1}^{6} r_i' u_i'(\tau)\right] d\tau, \end{aligned} \tag{21}$$

where $u_i = f_i$ $(i = 1, \dots, 6)$ are the control inputs; $u_{\mathrm{ref}i}$ $(i = 1, \dots, 6)$ are the reference values of the inputs; $x_{\mathrm{ref}i}$ $(i = 1, \dots, 9)$ are the reference values of the positions, velocities, and angular velocities; q_{ref} is a reference vector of the quaternion; s_i $(i = 1, \dots, 13)$ are the weighting values of the state; r_i and r_i' $(i = 1, \dots, 6)$ are the weighting values of the inputs; $q_{ei} = (q_{\mathrm{ref}}^* q)_i$ $(i = 0, \dots, 3)$ are the quaternion errors; and $q_{\mathrm{zero}} = [1\ 0\ 0\ 0]^{\mathrm{T}}$ is the reference vector for the quaternion errors. In relation to the previous section, these reference vectors yield the parameter $p(t)$ in (10). The reference values of the states are determined to make the position of the hexacopter converged to a constant reference position. To this end, the values of $x_{\mathrm{ref}i}(t)$ for $i = 1, \dots, 8$, corresponding to the positions, velocities, and angular velocities except for the yaw angular velocity, are given as constant values. Only the value of $x_{\mathrm{ref}9}(t)$, corresponding to the yaw angular velocity, can be time varying because this angular velocity is used to produce a dynamic equilibrium under the rotor failures, as discussed below. A similar consideration determines the reference values of the inputs. They are given so that the hexacopter can stay at the reference position determined by $x_{\mathrm{ref}}(t)$; the total thrust by $u_{\mathrm{ref}}(t)$ should balance the mass of the hexacopter. The reference values of the inputs for each case are listed in Table 1. The weighting matrix of the state $S_f = \mathrm{diag}[s_1 \dots s_{13}] \in \mathbb{R}^{13\times 13}$ is presented in Table 2, and the weighting values of the inputs are displayed in Table 3. We selected these weights according to trial and error. However, this study reveals that the form of the performance index (21) is effective for the fault-tolerant control of hexacopters. Only the values of the weights and references in this performance index need to be changed to achieve fault-tolerant control in various situations. The terminal weights were set to be equal to the stage weights. Apart from cases 4, 6, and 7, the reference values of the state were set to $x_{\mathrm{ref}i} = 0$ $(i = 1, \dots, 9)$ and $q_{\mathrm{ref}} = [1\ 0\ 0\ 0]^{\mathrm{T}}$. In cases 4, 6, and 7, the reference states were set to $x_{\mathrm{ref}i} = 0$ $(i = 1, \dots, 8)$ and $q_{\mathrm{ref}} = [1\ 0\ 0\ 0]^{\mathrm{T}}$, but $x_{\mathrm{ref}9}$ was different from that in the other cases. See below for details of $x_{\mathrm{ref}9}$ in cases 4, 6, and 7. The linear penalty for the dummy inputs in the performance index was introduced to avoid singularities regarding the dummy inputs u'_i in (19). Without this term, singularity of the Jacobian $\frac{\partial F}{\partial U}$ in (15) may occur when $u'_i(t) = 0$ holds for a certain t. That is, the equality constraint (19) accompanies the indefiniteness of the sign of the dummy input $u'_i(t)$. This causes the issue whereby once $u'_i(t') = 0$ holds at a certain time instant t', the sign of $u'_i(t')$ cannot be determined when $u'_i(t')$ must be a non-zero value after t' when the final term in (21) is not assumed. The introduction of the final term in (21) enables such singularity issues to be avoided.

In case 1, in which one rotor stopped, the stabilization of the attitude was the top priority for maintaining the altitude. We set the reference values of the inputs to support the hexacopter by the thrusts of four rotors (u_1, u_2, u_4, and u_5) and reduced the thrust of the remaining rotor, u_3.

In case 2, in which two rotors in opposite positions stopped, the stabilization of the attitude was the top priority for maintaining the altitude, as in case 1. We set the same reference values of the inputs to support the hexacopter by the thrusts of four rotors (u_1, u_2, u_4, and u_5). As the rotation direction of the diagonal rotors differed, it was impossible to control the yaw angle independently of the roll and pitch angles. Therefore, the weight of the yaw angle was set to zero.

In case 3, in which two rotors separated by a functioning rotor stopped, the yaw angle was difficult to control because three out of the four rotors rotated in the same direction. Therefore, the weight of the yaw angle was set to zero.

In case 4, in which two adjacent rotors stopped, as no stable static equilibrium state existed, we attempted to maintain the horizontal attitude and altitude by rotation in the yaw direction. Therefore, the reference value $x_{\mathrm{ref}9}(t)$ for the component of the yaw angular velocity is expressed as follows:

$$x_{\mathrm{ref}9}(t) = \frac{\overline{\omega}_{3\mathrm{ref}}}{1 + \exp(-(t-2))}. \tag{22}$$

This reference value is a sigmoid function that nearly reaches $\overline{\omega}_{3\mathrm{ref}}$ for a sufficiently large t. We selected this form of (22) by trial and error. However, the parameter $\overline{\omega}_{3\mathrm{ref}}$ was determined by specifying a dynamic equilibrium condition. We describe the dynamic equilibrium of this control problem in Appendix B. We also present a procedure for determining the value of $\overline{\omega}_{3\mathrm{ref}}$. In the simulations, we used the value of $\overline{\omega}_{3\mathrm{ref}} = -11.2$ [rad/s] that was obtained by following the procedure in Appendix B. As the attitude of the hexacopter was inclined when certain rotors stopped, we first needed to control the roll and pitch angles. Thereafter, we controlled the yaw angular velocity towards $\overline{\omega}_{3\mathrm{ref}}$. This time-varying reference value was also effective for cases 6 and 7.

In case 5, in which three rotors in alternate positions stopped, the yaw angle was impossible to control because all of the remaining rotors rotated in the same direction. Therefore, the weight of the yaw angle was set to zero.

In case 6, in which two adjacent rotors and a separated rotor stopped, as no stable static equilibrium state existed, we attempted to maintain the horizontal attitude and altitude by rotation in the yaw direction. Therefore, the reference value $x_{\mathrm{ref}9}(t)$ in Eq. (22) was provided for the component of the yaw angular velocity, as in case 4.

Finally, we considered case 7, in which three adjacent rotors stopped. In this case, because the thrust existed only on one side of the hexacopter and there was no static equilibrium state, we attempted to maintain the horizontal attitude and altitude by rotation in the yaw direction. Therefore, the reference value $x_{\mathrm{ref}9}(t)$ in Eq. (22) was provided for the component of the yaw angular velocity, as in cases 4 and 6.

4. Simulations

4.1. Simulation conditions

Cases 1 to 7 were simulated using a PC (CPU: Intel(R) Core(TM) i5-5350U 1.80 GHz; RAM: 8.00 GB; OS: macOS Mojave). We implemented the C/GMRES method for fault-tolerant control using AutoGenU (Cybernet Systems Co., 2016), as noted in Section 3.2. AutoGenU generated codes that were written in the C language, and we conducted the following simulations by compiling them with the GNU gcc compiler. The codes implemented a predictor–corrector

Table 1 Reference values of inputs

	u_{ref1}	u_{ref2}	u_{ref3}	u_{ref4}	u_{ref5}	u_{ref6}
Case 1	$\frac{mg}{4}$	$\frac{mg}{4}$	0	$\frac{mg}{4}$	$\frac{mg}{4}$	–
Case 2	$\frac{mg}{4}$	$\frac{mg}{4}$	–	$\frac{mg}{4}$	$\frac{mg}{4}$	–
Case 3	$\frac{2mg}{5}$	–	$\frac{mg}{5}$	$\frac{mg}{5}$	$\frac{mg}{5}$	–
Case 4	–	$\frac{mg}{4}$	$\frac{mg}{4}$	$\frac{mg}{4}$	$\frac{mg}{4}$	–
Case 5	$\frac{mg}{3}$	–	$\frac{mg}{3}$	–	$\frac{mg}{3}$	–
Case 6	–	$\frac{mg}{3}$	–	$\frac{mg}{3}$	$\frac{mg}{3}$	–
Case 7	–	$\frac{mg}{3}$	$\frac{mg}{3}$	$\frac{mg}{3}$	–	–

Table 2 Weighting matrix of state

	S_f
Case 1	diag[1 1 10 0.1 0.1 0.1 0.1 0.1 0 0 25 25 0]
Case 2	diag[1 1 10 0.1 0.1 0.1 0.1 0.1 0 0 25 25 0]
Case 3	diag[1 1 30 0.1 0.1 0.1 0.1 0.1 0 0 50 50 0]
Case 4	diag[1 1 1 0.1 0.1 0.1 0.1 0.1 1 0 30 30 0]
Case 5	diag[1 1 10 0.1 0.1 0.1 0.1 0.1 0 0 25 25 0]
Case 6	diag[1 1 1 0.1 0.1 0.1 0.1 0.1 1 0 30 30 0]
Case 7	diag[1 1 1 0.1 0.1 0.1 0.1 0.1 1 0 30 30 0]

Table 3 Weighting values of inputs

	r_1	r_2	r_3	r_4	r_5	r_6	r'_1	r'_2	r'_3	r'_4	r'_5	r'_6
Case 1	1	1	1	1	1	0	0.01	0.01	0.01	0.01	0.01	0
Case 2	1	1	0	1	1	0	0.01	0.01	0	0.01	0.01	0
Case 3	1	0	1	1	1	0	0.01	0	0.01	0.01	0.01	0
Case 4	0	1	1	1	1	0	0	0.01	0.01	0.01	0.01	0
Case 5	1	0	1	0	1	0	0.01	0	0.01	0	0.01	0
Case 6	0	1	0	1	1	0	0	0.01	0	0.01	0.01	0
Case 7	0	1	1	1	0	0	0	0.01	0.01	0.01	0	0

method using the Adams–Bashforth predictor method and the Adams–Moulton method for numerical calculations of the state equation (8). In the codes, the C/GMRES method described in Section 3.2 solved the receding-horizon control problem at each step of the predictor–corrector method to simulate the control problems.

The physical parameters of the hexacopter are listed in Table 4. These parameters were measured using Firefly from AscTec (Ascending Technologies, 2019). The coefficient of air resistance γ was arbitrarily determined with reference to Mueller and D'Andrea (2014). Note that in cases 4 to 7, the input constraints were relaxed for the simulation studies: $u_{\mathrm{max}} = 12$ [N]. Although this maximum thrust u_{max} could not be generated by Firefly, we could still investigate the NMPC performance for a generic hexacopter. The computation parameters are listed in Table 5. Refer to Section 3.2 for details of these parameters.

4.2. Simulation results

In all cases, the optimal control problem in the finite horizon of $T_f = 1.0$ [s] was solved within 0.20 ms for the sampling period $\Delta t = 1$ [ms], whereas the computation time when using the Euler angle model was approxi-

Table 4 Physical parameters

Parameter	Meaning	Value
m	Mass	1.608 [kg]
g	Gravitational acceleration	9.81 [m/s^2]
l	Length of arms	0.216 [m]
k	Coefficient of rotors	1.60×10^{-2} [m]
I_{xx}	Moment of inertia around roll angle	0.0366 [kg · m^2]
I_{yy}	Moment of inertia around pitch angle	0.0327 [kg · m^2]
I_{zz}	Moment of inertia around yaw angle	0.0187 [kg · m^2]
I_{xy}	Product of inertia	0 [kg · m^2]
I_{yz}	Product of inertia	0.0046 [kg · m^2]
I_{zx}	Product of inertia	0.0065 [kg · m^2]
u_{min}	Minimum thrust	0.1046 [N]
u_{max}	Maximum thrust	6 [N]
γ	Coefficient of air resistance	1×10^{-2} [N · m · s/rad]

Table 5 Computation parameters

Parameter	Meaning	Value
Δt	Sampling period	1 [ms]
T_f	Final horizon length	1.0 [s]
α	Parameter for variable horizon	0.5
ζ	Parameter for stabilization of continuation method	1000
k_{max}	Number of iterations in GMRES	5
N	Number of grids on horizon	50

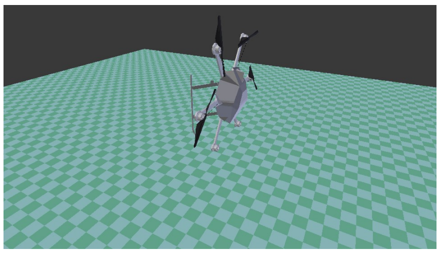

Fig. 9 Initial conditions (the CAD model of Firefly was obtained from Furrer et al. (2016))

mately 0.25 ms under the same conditions. That is, real-time optimization was achieved, and the quaternion model was superior to the Euler angle model in terms of the computation time. The C/GMRES method determines the vector $U(t) = [\bar{u}_0(t), \bar{\mu}_0(t), \ldots, \bar{u}_{N-1}(t), \bar{\mu}_{N-1}(t)]$, where $\bar{u}_i(t)$ and $\bar{\mu}_i(t)$ are the control inputs and Lagrange multipliers in the prediction horizon, respectively. In our setting, the dimension of the vector $U(t)$ was 900 because the dimension of the control input $\bar{u}_i(t)$ for each i was 12, including the dummy inputs, the dimension of the Lagrange multiplier $\bar{\mu}_i(t)$ for each i was six (see the description of the equality constraints (19)), and the number of grids N was 50. Our approach provided the control inputs in real time under nonlinear settings, although the computation in the NMPC involved a large dimension of $U(t)$. Furthermore, as demonstrated below, the quaternion model of the hexacopter was superior to the Euler angle model in terms of the control performance, because the position and attitude could be controlled from an initial state with a non-zero angular velocity and falling velocity, which caused singularity in the Euler angle model. The initial conditions were $\xi = [0\ 0\ 0]^{\mathrm{T}}$, $\dot{\xi} = [1\ 0\ -2]^{\mathrm{T}}$, $\omega = [0\ 2\ 0]^{\mathrm{T}}$, and $q = [0.697\ 0\ 0.717\ 0]^{\mathrm{T}}$ (see Fig. 9), except for case 7. These initial conditions were determined by considering the delay in the fault detection, and caused singularity in the Euler angles because this pitch angle was approximately $\pi/2$. Owing to these initial conditions, the hexacopter was likely to exhibit large transient behavior. Thus, we assumed that the hexacopter altitude was sufficiently high to not crash into the ground. We ignored the offset of the altitude and the initial altitude was set to zero.

The simulation results are depicted in Figs. 10 to 23 (Animations of simulation results are provided at http://www.ids.sys.i.kyoto-u.ac.jp/images/AokiUAV/aokiSim.mp4). As it is difficult to determine the attitude of a hexacopter from the time histories of the quaternion, we present the time histories of the Euler angles converted from the quaternion. In all cases except for case 7, the position and attitude were successfully controlled from a large inclined initial state with a non-zero angular velocity and falling velocity.

First, we considered case 1, in which one rotor stopped. As illustrated in Figs. 10 and 11, we could maintain the position, and the roll and pitch angles following rotor failure. Interestingly, the four rotor nos. 1, 2, 4, and 5 mainly supported the hexacopter, but sometimes rotor no. 3 was used.

Second, we considered case 2, in which two rotors in opposite positions stopped. As indicated in Figs. 12 and 13, The closed-loop responses were almost the same as those in case 1, except for the yaw angle. This is because $u_{\mathrm{ref}3}$ was zero and u_3 was sufficiently small in case 1.

Third, we considered case 3, in which two rotors separated by a functioning rotor stopped. Figures 14 and 15 show that the position $\xi = [x\ y\ z]$ converged to $\xi = [0\ 0\ 0]$. In this case, it was difficult to control the yaw angle owing to the arrangement of the functioning rotors. In the control design for this case, we set the weight for the yaw angle in (21) to zero. Therefore, we specifically allowed the hexacopter to rotate around the yaw axis. Accordingly, the values of the roll and pitch angles exhibited slight deviations from zero, as indicated in Fig. 14, because of the rotation around the yaw axis. As illustrated in Figs. 14 and 15, the rotation around the yaw axis increased with time because three out of four rotors rotated in the same direction and the torque around the yaw axis was large. After a certain period, the torque in the yaw direction was balanced with the air resistance and the yaw angular velocity ω_3 converged to a constant.

Fourth, we considered case 4, in which two adjacent rotors stopped. Recall that we introduced a dynamic equilibrium and time-varying reference (22) for the yaw angular velocity. Figures 16 and 17 indicate that the hexacopter control converged to $\xi = [0\ 0\ 0]$ with a slight residual error in z, despite the rotor failures. Moreover, the value of the yaw angle decreased. This is because we used the dynamic equilibrium, in which the reference of the yaw angular velocity converged to $\bar{\omega}_{3\mathrm{ref}} = -11.2$ (see Eq. (22)). Moreover, the attitude z became approximately $z = -20$ in the transient behavior. The hexacopter required this transient behavior to recover its position, and roll and pitch angles from the large inclined initial attitude (see Fig. 9). Furthermore, as illustrated in these figures, the roll angle ϕ converged not to 0 rad, but to 2π rad, which also corresponded to the desired attitude. In contrast, the realization of similar behaviors with the Euler angle model is not trivial. An advantage of quaternions is their ability to identify α rad and $\alpha \pm 2n\pi$ rad $(n = 1, 2, \ldots)$ naturally.

Fifth, we considered case 5, in which three rotors in alternate positions stopped. In this case, the functioning rotors were located in alternate positions on the vertices of the hexagon, and all of the functioning rotors rotated in the same direction. Therefore, it was not easy to control the yaw angle. In the control design, we set the weight for the yaw angle to zero. As illustrated in Figs. 18 and 19, the position of the hexacopter converged to zero with a slight residual error in z. Note that the rotation around the yaw axis increased with time because all three rotors rotated in the same direction, and the force around the yaw axis was large. After a certain period, the torque in the yaw direction was balanced with the air resistance and the yaw angular velocity ω_3 converged to a constant, as in case 3.

Sixth, we considered case 6, in which two adjacent rotors and a separated rotor stopped. As in case 4, a dynamic equilibrium was considered. As indicated in Figs. 20 and 21, the hexacopter was controlled to recover its position. Furthermore, because of the large inclined initial attitude, the altitude exhibited large transient behavior. Note that the roll angle ϕ converged not to 0 rad, but to 2π rad, as in case 4.

Finally, we considered case 7, in which three adjacent rotors stopped. Unlike the other cases, the initial conditions were $\xi = [0\ 0\ 0]^{\mathrm{T}}$, $\dot{\xi} = [0\ 0\ 0]^{\mathrm{T}}$, $\omega = [0\ 0\ 0]^{\mathrm{T}}$, and $q = [1\ 0\ 0\ 0]^{\mathrm{T}}$. Although the hexacopter in this case could be recovered from the same initial conditions as in the other cases, we selected the initial conditions to demonstrate an aggressive maneuver that could be realized by NMPC. As illustrated in Figs. 22 and 23, the position and attitude were controlled from a horizontal attitude and static state by providing the reference for the yaw angular velocity $x_{\mathrm{ref}9}(t)$. Although the simulation was started from the horizontal attitude and static state, the hexacopter was largely inclined and fell at a high velocity immediately after the simulation started. Therefore, it was difficult for the attitude to recover from this initial state, as in the other cases. Furthermore, the roll angle ϕ converged not to 0 rad, but to -2π rad.

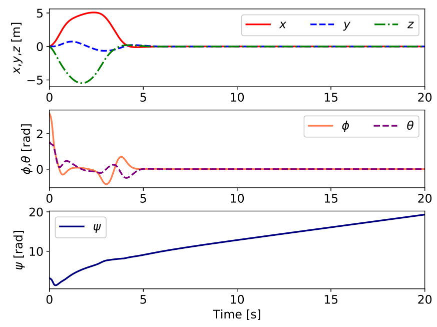

Fig. 10 Time histories of state variables in case 1

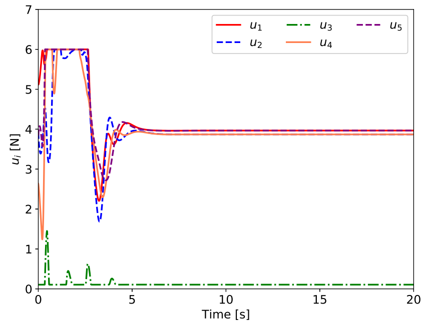

Fig. 11 Time histories of input variables in case 1

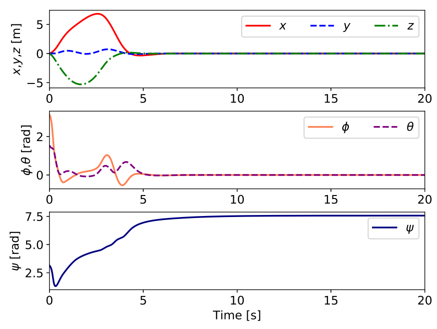

Fig. 12 Time histories of state variables in case 2

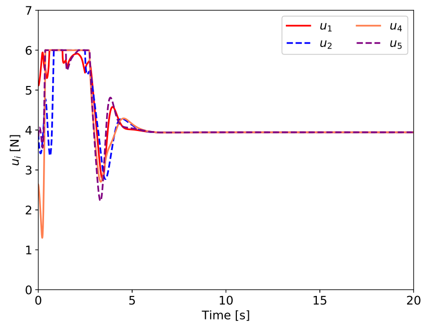

Fig. 13 Time histories of input variables in case 2

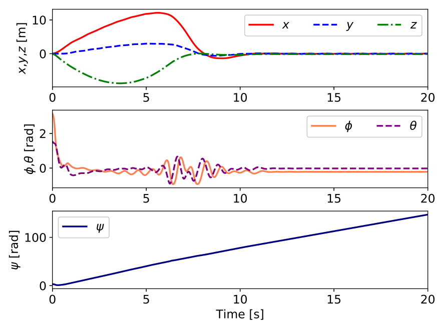

Fig. 14 Time histories of state variables in case 3

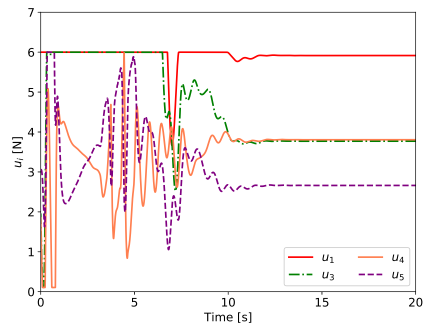

Fig. 15 Time histories of input variables in case 3

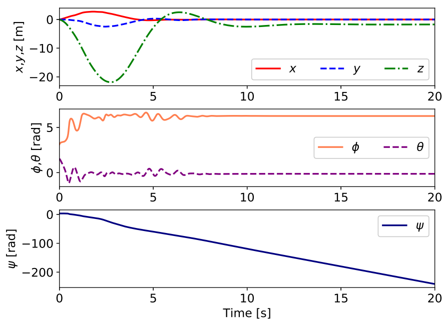

Fig. 16 Time histories of state variables in case 4

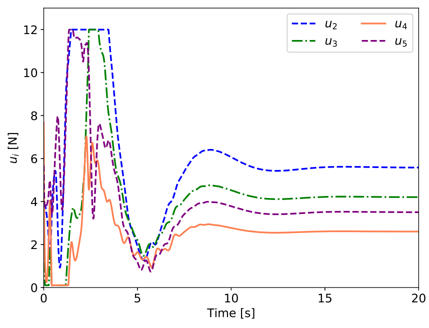

Fig. 17 Time histories of input variables in case 4

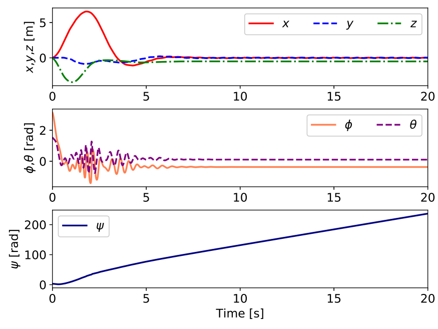

Fig. 18 Time histories of state variables in case 5

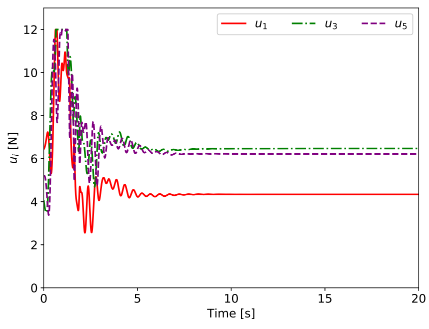

Fig. 19 Time histories of input variables in case 5

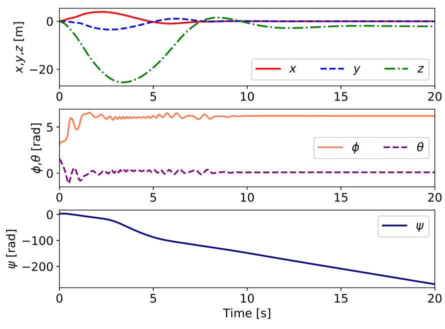

Fig. 20 Time histories of state variables in case 6

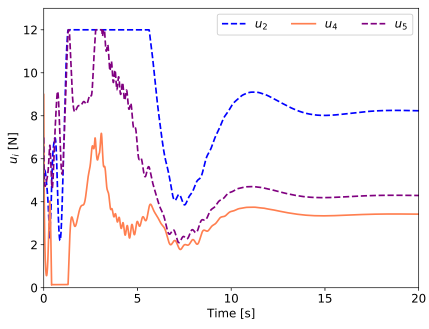

Fig. 21 Time histories of input variables in case 6

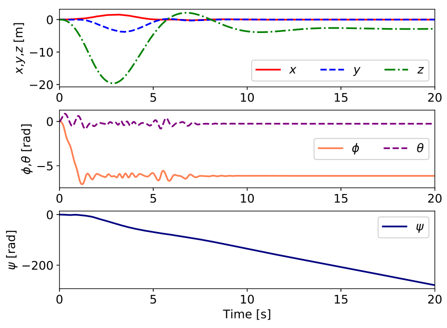

Fig. 22 Time histories of state variables in case 7

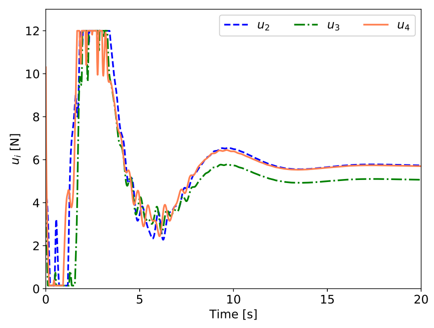

Fig. 23 Time histories of input variables in case 7

5. Hardware experiments

5.1. Experimental conditions

We describe the experimental results when using an actual hexacopter to validate the effectiveness of the proposed method in practical control environments. We used AscTec Firefly (Ascending Technologies, 2019) in the experiments. Figure 24 presents a simplified block diagram of the system. The motion capture system used six cameras, OptiTrack Flex 13 (NaturalPoint Inc., 2020), and four reflective markers to track the position of the hexacopter. The marker position data were sent to a PC via a USB at 120 Hz. The position of the hexacopter was calculated by tracking software (Motive: Tracker 1.10.3 Final). The hexacopter position data were sent to an on-board PC via an XBee at 20 Hz. On the on-board PC, the velocity of the hexacopter was calculated from the difference between two positions measured at adjacent sampling time instants. The on-board processor obtained the Euler angles and angular velocity from the inertial measurement unit, and sent these to the on-board PC at 700 Hz. The Euler angles were converted into a quaternion, and the control inputs were calculated using these state variables (position, velocity, angular velocity, and quaternion) on the on-board PC. Note that a quaternion can always be determined by Euler angles, even in the case of gimbal lock, and therefore, this controller never falls into singularities. The control inputs were converted into PWM signals and then sent to the motor controllers via the on-board processor. The details of the experimental equipment are listed in Table 6. Note that the rotor failures were simulated by deliberately degrading the input value. Owing to the limitation in the actual hexacopter, the value was set to $u_{\rm min}$, as indicated in Table 7, but not zero.

We conducted the experiments for the two cases, namely a normal operation and a fault tolerant operation, abbreviated to as NO and FTO in the below, which were established in consideration of the experimental environment. We demonstrate the effectiveness of the proposed controller for a hexacopter with failed rotors in the comparison between the NO and FTO cases. In case of NO, during hovering flight with six rotors available, one rotor suddenly stopped, and the controller for the six rotors was still used even after the failure. In case of FTO, during hovering flight with six rotors available, one rotor suddenly stopped, and 0.3 s after the rotor stopped, we switched from a controller for a hexacopter with six available rotors to another controller for a failed rotor. In this study, we investigated only the one failed rotor cases due to physical limitations in the experiments, though such experimental results sufficed to show the executability of real-time computation for NMPC in hardware experiments. Experiments with multiple failed rotors will be conducted in future works.

5.2. Experimental results

In the hardware experiments, the NMPC for the quaternion model was successfully implemented in an actual hexacopter in real time. The constraints are given by Eq. (19), and the performance index is determined using Eq. (21). The reference values of the inputs are listed in Table 7, the weighting matrix of a state is shown in Table 8, and the weighting values of the inputs are listed in Table 9. The sampling period was set to $\Delta t = 1.5$ [ms], and the values of the physical parameters and other computation parameters were the same as those of the simulation (see Tables 4 and 5). The sampling period was shorter than that of other studies, such as Kamel et al. (2015), in which the controller was running at 200 Hz; that is, the sampling period was 5 [ms]. The reference values of the state were set to $x_{{\rm ref}i} = 0$ $(i = 1, \ldots, 9)$ and $q_{\rm ref} = [1\ 0\ 0\ 0]^{\rm T}$. In the experiments, we found that the controller was sensitive to noise when the control inputs approached $u_{\rm min}$ or $u_{\rm max}$. Moreover, we found that the sensor for angular velocity did not work when the angular velocity was excessively high. Therefore, we determined the reference values of the inputs for a failed rotor so that the control

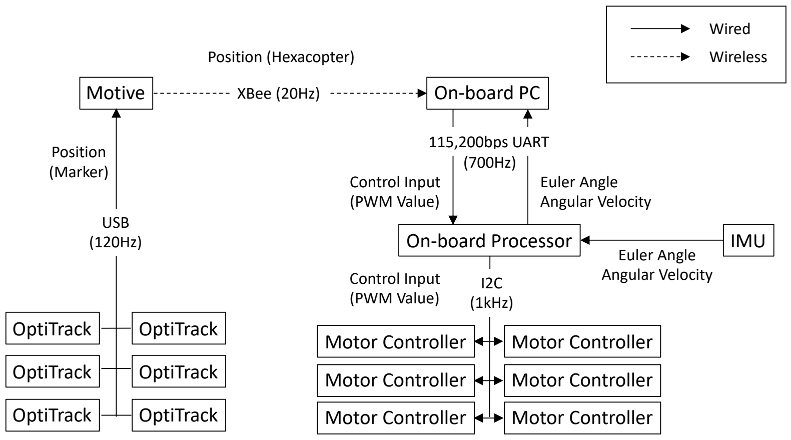

Fig. 24 Experimental equipment diagram

Table 6 Equipment specifications

Hexacopter	AscTec Firefly
Motion capture	OptiTrack Flex 13
Tracking software	Motive: Tracker 1.10.3 Final
On-board PC	Mastermind 3a (CPU: Intel(R) Core(TM) i7-3517UE CPU @ 1.70 GHz, RAM: 4.00 GB, OS: Ubuntu 14.04.5 LTS)
Battery	AP03 (5000 mAh, 11.1 V, 3S1P)

Table 7 Reference values of inputs (hardware experiments)

	u_{ref1}	u_{ref2}	u_{ref3}	u_{ref4}	u_{ref5}	u_{ref6}
Six rotors available	$\frac{mg}{6}$	$\frac{mg}{6}$	$\frac{mg}{6}$	$\frac{mg}{6}$	$\frac{mg}{6}$	$\frac{mg}{6}$
One failed rotor	$\frac{15mg}{56}$	$\frac{11mg}{56}$	$\frac{mg}{14}$	$\frac{11mg}{56}$	$\frac{15mg}{56}$	–

Table 8 Weighting matrix of state (hardware experiments)

	S_f
Six rotors available	diag[1 1 1 1 1 1 1 1 1 0 1 1 10]
One failed rotor	diag[1 1 10 0.1 0.1 0.1 0.1 0.1 0 0 25 25 0]

Table 9 Weighting values of inputs (hardware experiments)

	r_1	r_2	r_3	r_4	r_5	r_6	r'_1	r'_2	r'_3	r'_4	r'_5	r'_6
Six rotors available	1	1	1	1	1	1	0.01	0.01	0.01	0.01	0.01	0.01
One failed rotor	1	1	1	1	1	0	0.01	0.01	0.01	0.01	0.01	0

inputs did not approach the bounds of the constraints and the yaw angular velocity did not become too high. There is a possibility to find such appropriate reference values using NMPC by introducing more strict conditions on the control input and additional constraints on the angular velocities. Further investigation will be included in future works.

The experimental environment is depicted in Fig. 25 and the experimental results are illustrated in Figs. 26 to 29 (Videos of experimental results are provided at http://www.ids.sys.i.kyoto-u.ac.jp/images/AokiUAV/aokiExp.mp4). First, we considered the NO case, in which although a rotor stopped at approximately $t = 5$ [s], we continued to use the controller for the nominal operation to learn how it responded to the rotor failure. As indicated in Figs. 26 and 27, the experiment failed immediately after a rotor stopped, because the hexacopter fell into the safety net surrounding the laboratory (see Fig. 25). This result demonstrates that the controller for a hexacopter with six available rotors cannot control a hexacopter with a failed rotor adequately; thus, the controller should be switched to deal with the failure.

Second, we considered the FTO case, in which 0.3 s after a rotor stopped at approximately $t = 5$ [s], the controller was switched to that for a hexacopter with a failed rotor. As indicated in Figs. 28 and 29, even after the rotor stopped, by switching the controller, the hexacopter could maintain its altitude with yaw rotation and small circular motion near the target point. This circular motion was caused by the yaw torque when the rotor stopped. This result demonstrates that the proposed controller works successfully in an actual hexacopter. Moreover, comparing the results in NO and FTO cases, we conclude that the controller for the hexacopter with a failed rotor is essential even when only one rotor stops in the case of a limited range of maneuvers.

6. Conclusions

This study has presented the NMPC approach for the fault-tolerant control problem with a nonlinear model of a

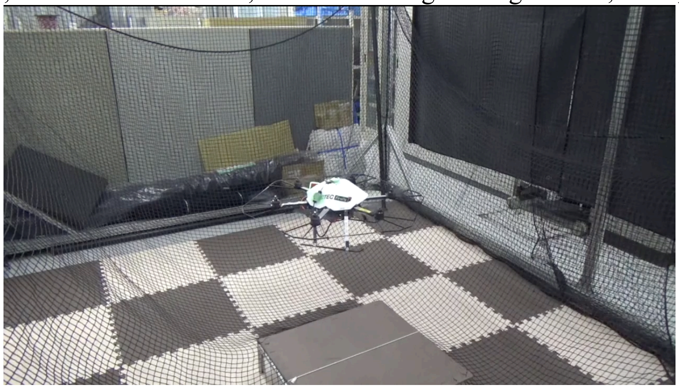
Fig. 25 Experimental environment

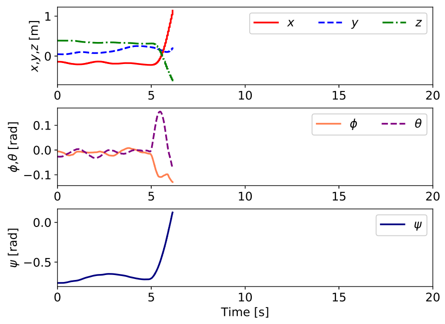

Fig. 26 Time histories of state variables in normal operation (NO case)

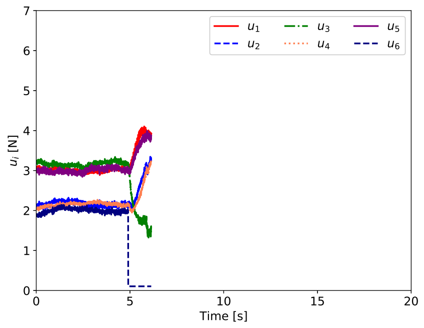

Fig. 27 Time histories of input variables in normal operation (NO case)

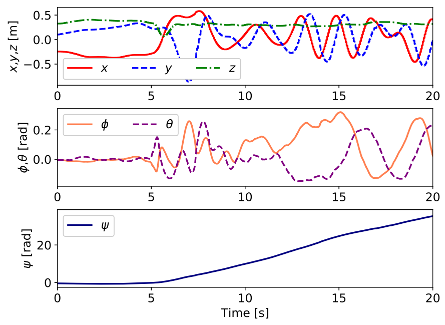

Fig. 28 Time histories of state variables in fault tolerant operation (FTO case)

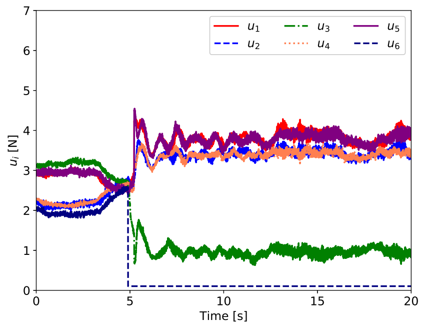

Fig. 29 Time histories of input variables in fault tolerant operation (FTO case)

hexacopter with failed rotors. The NMPC approach was developed to control the position and attitude of the hexacopter for rotor failures of complete stops. This study provided the performance index of NMPC for the fault-tolerant control and showed that only tuning design parameters in the performance index is required for various failure cases. As a result, the attitude of the hexacopter could be stabilized from a large inclined initial state with a non-zero angular velocity and falling velocity in six cases, and from a horizontal attitude and static state in one case. In terms of the computation time, the quaternion model was also demonstrated to be superior to the Euler angle model. In particular, in cases 4 (in which two adjacent rotors stopped), 6 (in which two adjacent rotors and a separated rotor stopped), and 7 (in which three adjacent rotors stopped), the desired attitude was achieved with a roll angle of $\pm 2\pi$ rad, which is not easily possible with the Euler angle model. Moreover, the hardware experiments indicated that our proposed controller could achieve real-time optimization for NMPC on an on-board PC, and it could effectively control a hexacopter with a failed rotor. From the simulation and hardware experiment results, we can conclude that NMPC for the quaternion model is a promising approach for controlling a hexacopter with failed rotors in a unified manner with the appropriate definition of a performance index.

Future work will include the control of a hexacopter with four or five failed rotors from a large inclined initial state with a non-zero angular velocity and falling velocity. The integration of NMPC and fault detection is another direction of future research.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number 15H02257.

Appendix A. Quaternions

This appendix briefly introduces quaternions (see Choset et al. (2005, Appendix E) for details of quaternions.). Our hexacopter model is based on quaternions for attitude, instead of Euler angles. Quaternions are generally described as

$$q = q_0 + q_1 i + q_2 j + q_3 k, \tag{A.1}$$

where q_l $(l = 0, \ldots, 3)$ are scalars and i, j, k are the basis vectors corresponding to the e^{i}_1, e^{i}_2, and e^{i}_3 axes in Fig. 1, respectively. These basis vectors have the following multiplication rules:

$$i^2 = j^2 = k^2 = -1,\ ij = k,\ jk = i,\ ki = j. \tag{A.2}$$

The conjugate of the quaternion is expressed as

$$q^* = q_0 - q_1 i - q_2 j - q_3 k, \tag{A.3}$$

and the norm of the quaternion $||q||$ is defined as

$$||q|| = \sqrt{q^* q}. \tag{A.4}$$

Moreover, a unit quaternion is a quaternion that has a norm equal to one, and any unit quaternion can be described as

$$q = \cos\left(\frac{\alpha}{2}\right) + n \sin\left(\frac{\alpha}{2}\right), \tag{A.5}$$

where n is a unit vector that represents the rotation axis direction and α is a scalar that represents the rotation angle (Fig. 1). This can also be written as follows using vector notations:

$$q = \left[\cos\left(\frac{\alpha}{2}\right)\ n_1 \sin\left(\frac{\alpha}{2}\right)\ n_2 \sin\left(\frac{\alpha}{2}\right)\ n_3 \sin\left(\frac{\alpha}{2}\right)\right]^{\mathrm{T}}, \tag{A.6}$$

where $n = (n_1\ n_2\ n_3)^{\mathrm{T}}$. The unit quaternion can be used to describe the attitude of the hexacopter. The deviation from a reference quaternion q_{ref} is expressed as follows:

$$q_{\mathrm{e}} = q^*_{\mathrm{ref}} q, \tag{A.7}$$

where q_{e} is known as the quaternion error. Furthermore, the quaternion can be converted into ZYX Euler angles as follows:

$$\phi = \arctan\left(\frac{2(q_0 q_1 + q_2 q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right), \tag{A.8}$$

$$\theta = \arcsin\left(-2(q_1 q_3 - q_0 q_2)\right), \tag{A.9}$$

$$\psi = \arctan\left(\frac{2(q_1 q_2 + q_0 q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right), \tag{A.10}$$

where ϕ, θ, and ψ are the roll, pitch, and yaw angles, respectively.

Appendix B. Derivations of reference value $\overline{\omega}_{3\mathrm{ref}}$

This appendix presents the derivation of the reference value $\overline{\omega}_{3\mathrm{ref}}$ in Eq. (22) that is used to control the hexacopter with failed rotors, as outlined in Section 3.3. In Section 3.3, the value $\overline{\omega}_{3\mathrm{ref}}$ must be determined so that a hexacopter can maintain its altitude and attitude. This means that we need to investigate the equilibrium of the partial state of the hexacopter to determine $\overline{\omega}_{3\mathrm{ref}}$. We refer to this equilibrium of the partial state as a dynamic equilibrium. In the following, we determine $\overline{\omega}_{3\mathrm{ref}}$ by deriving the dynamic equilibrium conditions.

Appendix B.1. Equation of motion of hexacopter based on Euler angles

To derive the reference value $\overline{\omega}_{3\mathrm{ref}}$ in Eq. (22), we first formulate a nonlinear dynamic model of a hexacopter based on the ZYX Euler angles ϕ, θ, and ψ. Our hexacopter is based on the configuration depicted in Fig. 1. To derive the equation of motion, we refer to Nonami et al. (2010, Chapter 8). The equations of motion of the hexacopter can be described as

$$m\ddot{\xi} = RF^{\mathrm{b}} - mge_3^{\mathrm{i}}, \tag{B.11}$$

$$I\dot{\omega} + \omega \times I\omega = \tau + \tau_r, \tag{B.12}$$

where $\tau = Tu$ is the torque, $\tau_r = [0\ 0\ -\gamma\dot{\psi}]^{\mathrm{T}}$ is the torque of the air resistance, ω is the angular velocity, and $R \in \mathbb{R}^{3\times 3}$ is a rotation matrix that is defined as

$$R = R_\psi R_\theta R_\phi = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi, \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}. \tag{B.13}$$

The parameters m, g, I, T, and γ are the same as those in Section 2.1. Note that $s(\cdot)$ and $c(\cdot)$ are abbreviations for $\sin(\cdot)$ and $\cos(\cdot)$, respectively. The angular velocity ω is represented in terms of the Euler angles:

$$\omega = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \in \mathbb{R}^3. \tag{B.14}$$

Appendix B.2. Dynamic equilibrium conditions

We derive the dynamic equilibrium conditions by using Eqs. (B.11) and (B.12). In the dynamic equilibrium conditions, the altitude z, roll angle ϕ, pitch angle θ, yaw angular velocity $\dot{\psi}$, and control inputs u_i $(i = 1, \ldots, 6)$ are constant. If ϕ, θ, and $\dot{\psi}$ have constant values of ϕ_{de}, θ_{de}, and $\dot{\psi}_{\mathrm{de}}$, respectively, the corresponding angular velocity ω_{de} is expressed by Eq. (B.14), as follows:

$$\omega_{\mathrm{de}} = \dot{\psi}_{\mathrm{de}} \begin{bmatrix} -\sin\theta_{\mathrm{de}} \\ \sin\phi_{\mathrm{de}}\cos\theta_{\mathrm{de}} \\ \cos\phi_{\mathrm{de}}\cos\theta_{\mathrm{de}} \end{bmatrix} = \text{const.} \tag{B.15}$$

Moreover, by using Eqs. (B.12) and (B.15), we can obtain the dynamic equilibrium conditions in the attitude dynamics, as follows:

$$\tau_{\mathrm{de}} = \omega_{\mathrm{de}} \times I\omega_{\mathrm{de}} - \tau_{r\mathrm{de}}, \tag{B.16}$$

where $\tau_{\mathrm{de}} = Tu_{\mathrm{de}}$ for a constant input vector u_{de} and $\tau_{r\mathrm{de}} = [0\ 0\ -\gamma\dot{\psi}_{\mathrm{de}}]^{\mathrm{T}}$ is the torque of the air resistance. Furthermore, if the altitude z is constant, Eqs. (B.11) and (B.13) yield the following condition:

$$\left(\sum_{i=1}^{6} u_{\mathrm{de}i}\right)\cos\phi_{\mathrm{de}}\cos\theta_{\mathrm{de}} = mg, \tag{B.17}$$

where $u_{\mathrm{de}i}$ $(i = 1, \ldots, 6)$ represent the elements of u_{de}.

Appendix B.3. Solution for dynamic equilibrium conditions

To derive the reference value $\overline{\omega}_{3\mathrm{ref}}$, we determine ϕ_{de}, θ_{de}, $\dot{\psi}_{\mathrm{de}}$, and $u_{\mathrm{de}i}$ $(i = 1, \ldots, 6)$ satisfying the dynamic equilibrium conditions given by the four equations in Eqs. (B.16) and (B.17). As we have $N_r + 3$ variables for N_r functioning rotors, we introduce the performance index J_{de}, as follows:

$$J_{\mathrm{de}} = \max_i |u_{\mathrm{de}i} - u_{\mathrm{mean}}|, \quad u_{\mathrm{mean}} = \frac{u_{\mathrm{min}} + u_{\mathrm{max}}}{2}, \tag{B.18}$$

to determine the variables uniquely, where u_{min} and u_{max} are the minimum and maximum thrusts, respectively, as indicated in Section 3.3. That is, we determine the $N_r + 3$ variables ϕ_{de}, θ_{de}, $\dot{\psi}_{\mathrm{de}}$, and $u_{\mathrm{de}i}$ by minimizing the performance index I_{de} subject to the four equations given by Eqs. (B.16) and (B.17). By solving this problem, we obtain ϕ_{de}, θ_{de}, $\dot{\psi}_{\mathrm{de}}$, and $u_{\mathrm{de}i}$. Subsequently, by using ϕ_{de}, θ_{de}, $\dot{\psi}_{\mathrm{de}}$, and Eq. (B.15), we can obtain ω_{de} and use $\omega_{\mathrm{de}3}$ as the reference $\overline{\omega}_{3\mathrm{ref}}$.

References

Alaimo, A., Artale, V., Milazzo, C., Ricciardello, A., Trefiletti, L. Mathematical modeling and control of a hexacopter. Proceedings of 2013 International Conference on Unmanned Aircraft Systems (ICUAS) (2013). pp. 1043–1050.

Aoki, Y., Asano, Y., Honda, A., Motooka, N., Ohtsuka, T. Nonlinear model predictive control of position and attitude in a hexacopter with three failed rotors. Proceedings of 6th IFAC Conference on Nonlinear Model Predictive Control (NMPC) (2018). pp. 262–267.

Aoki, Y., Asano, Y., Honda, A., Motooka, N., Ohtsuka, T. Nonlinear model predictive control of position and attitude in a hexacopter with two failed rotors. Transactions of the Institute of Systems, Control, and Information Engineers, Vol. 32, No. 4, (2019), pp. 168–176. (in Japanese).

Ascending Technologies. http://wiki.asctec.de/display/AR/AscTec+Firefly (2019). (accessed 2019-12-18).

Avram, R. C., Zhang, X., Muse, J. Nonlinear adaptive fault-tolerant quadrotor altitude and attitude tracking with multiple actuator faults. IEEE Transactions on Control Systems Technology, Vol. 26, No. 2, (2017), pp. 701–707.

Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., Thrun, S., Arkin, R. C. Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT press (2005).

Cybernet Systems Co. AutoGenU. https://www.maplesoft.com/applications/view.aspx?SID=153555 (2016). (accessed on 2020-10-30).

Du, G.-X., Quan, Q., Cai, K.-Y. Controllability analysis and degraded control for a class of hexacopters subject to rotor failures. Journal of Intelligent & Robotic Systems, Vol. 78, No. 1, (2015), pp. 143–157.

Fink, J., Michael, N., Kim, S., Kumar, V. Planning and control for cooperative manipulation and transportation with aerial robots. The International Journal of Robotics Research, Vol. 30, No. 3, (2011), pp. 324–334.

Fresk, E., Nikolakopoulos, G. Full quaternion based attitude control for a quadrotor. Proceedings of 2013 European Control Conference (ECC) (2013). pp. 3864–3869.

Furrer, F., Burri, M., Achtelik, M., Siegwart, R. Robot Operating System (ROS): The Complete Reference (Volume 1), chapter RotorS—A Modular Gazebo MAV Simulator Framework. Springer International Publishing (2016). pp. 595–625. (The CAD model is taken from https://github.com/ethz-asl/rotors_simulator/blob/master/rotors_description/meshes/firefly.dae.) (accessed 2020-11-14).

Greer, D., McKerrow, P., Abrantes, J. Robots in urban search and rescue operations. Proceeding of 2002 Australasian Conference on Robotics and Automation (2002). pp. 27–29.

Izadi, H. A., Zhang, Y., Gordon, B. W. Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation. Proceedings of 18th IFAC World Congress, Vol. 18 (2011). pp. 6343–6348.

Kamel, M., Alexis, K., Achtelik, M., Siegwart, R. Fast nonlinear model predictive control for multicopter attitude tracking on SO (3). Proceedings of 2015 IEEE Conference on Control Applications (CCA) (2015). pp. 1160–1166.

KYODO. 6 people hurt as drone crashes into crowd at central Japan park. https://english.kyodonews.net/news/2017/11/91f561b0a49e-6-people-hurt-as-drone-crashes-into-crowd-at-central-japan-park.html?phrase=gifu&words=Gifu (2017). (accessed 2020-11-1).

Milhim, A., Zhang, Y., Rabbath, C.-A. Gain scheduling based PID controller for fault tolerant control of quad-rotor UAV. Proceedings of 2010 AIAA Infotech@Aerospace (2010). p. 3530.

Mueller, M. W., D'Andrea, R. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA) (2014). pp. 45–52.

Murphy, R. R., Steimle, E., Griffin, C., Cullins, C., Hall, M., Pratt, K. Cooperative use of unmanned sea surface and micro aerial vehicles at Hurricane Wilma. Journal of Field Robotics, Vol. 25, No. 3, (2008), pp. 164–180.

Nascimento, T. P., Saska, M. Position and attitude control of multi-rotor aerial vehicles: A survey. Annual Reviews in Control, Vol. 48, (2019), pp. 129–146.

NaturalPoint Inc. (2020). https://www.optitrack.com/cameras/flex-13/ (accessed 2020-11-14).

Nonami, K., Kendoul, F., Suzuki, S., Wang, W., Nakazawa, D. Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles. Springer Science & Business Media (2010).

Ohtsuka, T. A continuation/GMRES method for fast computation of nonlinear receding horizon control. Automatica, Vol. 40, No. 4, (2004), pp. 563–574.

Ohtsuka, T. Introduction to Nonlinear Optimal Control. Corona Publishing (2011). (in Japanese).

Ohtsuka, T. A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control. Proceedings of 2015 European Control Conference (ECC) (2015). pp. 73–86.

Rasmussen, J., Nielsen, J., Garcia-Ruiz, F., Christensen, S., Streibig, J. C. Potential uses of small unmanned aircraft systems (UAS) in weed research. Weed Research, Vol. 53, No. 4, (2013), pp. 242–248.

Saad, Y., Schultz, M. H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, Vol. 7, No. 3, (1986), pp. 856–869.

Sadeghzadeh, I., Mehta, A., Chamseddine, A., Zhang, Y. Active fault tolerant control of a quadrotor UAV based on gain scheduled PID control. Proceedings of 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (2012). pp. 1–4.

Selby, W., Corke, P., Rus, D. Autonomous aerial navigation and tracking of marine animals. Proceedings of 2011 Australasian Conference on Robotics and Automation (2011). pp. 1–7.

Sharifi, F., Mirzaei, M., Gordon, B. W., Zhang, Y. Fault tolerant control of a quadrotor UAV using sliding mode control. Proceedings of 2010 Conference on Control and Fault-Tolerant Systems (SysTol) (2010). pp. 239–244.

Yu, B., Zhang, Y., Minchala, I., Qu, Y. Fault-tolerant control with linear quadratic and model predictive control techniques against actuator faults in a quadrotor UAV. Proceedings of 2013 Conference on Control and Fault-Tolerant Systems (SysTol) (2013). pp. 661–666.

Zhang, Y., Chamseddine, A., Rabbath, C. A., Gordon, B. W., Su, C.-Y., Rakheja, S., Fulford, C., Apkarian, J., Gosselin, P. Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed. Journal of the Franklin Institute, Vol. 350, No. 9, (2013), pp. 2396–2422.