

Fault-Tolerant Control of a Quadrotor Based on a Nonlinear Model

Kenta Hoshino and Toshiyuki Ohtsuka

Abstract This chapter addresses the fault-tolerant control (FTC) of a quadrotor, framed as a benchmark problem. When one of the quadrotor's four rotors fails, causing deviations in position and attitude, a fault-tolerant controller is activated to restore the quadrotor's original position and maintain stable flight. FTC for quadrotors is an important practical issue for ensuring dependability as these platforms find broader societal applications and roles. This is particularly challenging because a linear model is no longer valid for control design owing to the large attitude motions and limitations on available thrusts. We first present a nonlinear model and control objectives of a quadrotor and then define the benchmark problem and performance metrics. For future comparative studies, we also provide software tools for implementing nonlinear model predictive control (NMPC) as a baseline solution for the benchmark problem. The users can easily modify the software tools to implement and evaluate other control strategies. Finally, we present a design example of the NMPC and assess its efficacy through numerical simulations covering failure scenarios.

1 Introduction

Fault-tolerant control (FTC) for quadrotors, and multirotors more generally, is an important practical issue for ensuring dependability as these platforms find broader societal applications and roles. A fault-tolerant controller is activated when one or more of the rotors fail, causing deviations in position and attitude, to restore the multirotor's original position and to maintain stable flight. FTC of a multirotor is particularly challenging because a linear model is no longer valid for control design owing to the large attitude motions and limitations on available thrusts. Given the practical and societal demand, this field has been extensively studied for

Kenta Hoshino and Toshiyuki Ohtsuka
Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501,
Japan, e-mail: hoshino@i.kyoto-u.ac.jp, ohtsuka@i.kyoto-u.ac.jp

the last decade (see [25, 23] for reviews). The FTC has been addressed through a wide range of approaches, including gain-scheduled PID control [14], observer-based control [9], sliding mode control [22, 12], backstepping-based control [4], and model predictive control (MPC) [10, 3, 17]. Most of the studies in this field address the partial or complete loss of actuators' outputs; the situation is commonly referred to as the loss of effectiveness (LOE) [14, 9]. The gain-scheduled PID control, demonstrated in [14], involves the switching of several PID controllers for quadrotors experiencing LOE. One study [9] presents a nonlinear-observer-based FTC. Similarly, [12] combines a sliding-mode controller with a nonlinear observer to address the FTC under LOE conditions. MPC schemes [10, 3, 17] develop the FTC under LOE, leveraging their inherent flexibility to achieve diverse control objectives. Furthermore, [15] introduces a nonlinear FTC method handling failures in up to three rotors. Considering the current trend of the quadrotor FTC, it is necessary to develop a benchmark for comparing the various FTC methods.

In this chapter, we define an FTC benchmark problem for a quadrotor with one rotor failed. The failure of one rotor is a probable accident in practical use of a quadrotor, and its FTC is a baseline for FTC of general multirotors and general failure scenarios. As a benchmark problem, we consider both partial loss and complete loss of a thrust, and we focus on FTC after the rotor failure is detected by some fault diagnosis mechanism. In particular, we deal with a nonlinear model of the quadrotor and also consider the constraints in the thrusts. We first present the nonlinear model and control objectives of the FTC and then define the benchmark problem and performance metrics. We also provide software tools for implementing nonlinear model predictive control (NMPC) as a baseline solution for the benchmark problem. The users can easily modify the software tools to implement and evaluate other control strategies. Finally, we present a design example of the NMPC and assess its efficacy through numerical simulations covering failure scenarios.

2 Problem Description

2.1 Nonlinear Model of a Quadrotor

The configuration of a quadrotor is depicted in Fig. 1. The coordinate axes e_j^i and e_j^b ($j = 1, 2, 3$) denote the inertial frame and body-fixed frame, respectively. The state vector of the quadrotor is defined as

$$x = [\xi^\top \ v^\top \ q^\top \ \omega^\top]^\top \in \mathbb{R}^{13}, \quad (1)$$

where $\xi = [\xi_1 \ \xi_2 \ \xi_3]^\top \in \mathbb{R}^3$ and $v = \dot{\xi}$ denote the position and velocity of the quadrotor in the inertial frame, respectively, $q = [q_0 \ q_1 \ q_2 \ q_3]^\top \in \mathbb{R}^4$ denotes the unit quaternion representing the attitude of the quadrotor, and $\omega = [\omega_1 \ \omega_2 \ \omega_3]^\top \in \mathbb{R}^3$ denotes the angular velocity of the quadrotor in the body-fixed frame. The control input vector $u = [u_1 \ u_2 \ u_3 \ u_4]^\top \in \mathbb{R}^4$ consists of thrusts generated by the four rotors.

For details of the quaternion representation of the attitude of a rigid body, the readers are referred to [16, 6, 24, 7]. Unlike the Euler angle representation, the quaternion representation has no singularity and is more computationally efficient.

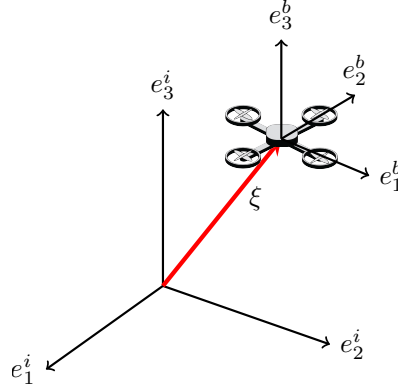


Fig. 1 Configuration of a quadrotor.

From the Newton–Euler equations of rigid-body motion, the state equation of the quadrotor is obtained as follows [8, 1, 24] (see [13] for a review of the modeling of UAVs):

$$\begin{cases} \dot{\xi} = v \\ \dot{v} = -g e_3^i + \frac{1}{m} e_3^b(q) s^\top u \\ \dot{q} = \frac{1}{2} \Omega(\omega) q \\ \dot{\omega} = -J^{-1} \omega^\times J \omega - J^{-1} D \omega + J^{-1} T u, \end{cases} \quad (2)$$

which is also represented in a vector form,

$$\dot{x} = f(x, u), \quad x = [x_1 \cdots x_{13}]^\top. \quad (3)$$

In state equation (2), g designates the gravitational acceleration, $e_3^i = [0 \ 0 \ 1]^\top$ denotes the unit vector of the vertical direction in the inertial frame, m represents the mass of the quadrotor, $J \in R^{3 \times 3}$ indicates the inertia matrix of the quadrotor, and $e_3^b(q)$, s , $\Omega(\omega)$, $\omega^\times$, D , and T are defined as follows:

$$e_3^b(q) = \begin{bmatrix} 2(q_0q_2 + q_1q_3) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}^\top \in \mathbb{R}^3, \quad (4)$$

$$s = [1 \ 1 \ 1 \ 1]^\top \in \mathbb{R}^4, \quad (5)$$

$$\omega^\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3 \times 3}, \quad (6)$$

$$\Omega = \begin{bmatrix} 0 & -\omega^\top \\ \omega & -\omega^\times \end{bmatrix} \in \mathbb{R}^{4 \times 4}, \quad (7)$$

$$D = \text{diag}[0, 0, d_3] \in \mathbb{R}^{3 \times 3}, \quad (8)$$

$$T = \begin{bmatrix} 0 & \ell & 0 & -\ell \\ -\ell & 0 & \ell & 0 \\ k & -k & k & -k \end{bmatrix} \in \mathbb{R}^{3 \times 4}, \quad (9)$$

where d_3 denotes the coefficient of aerodynamic drag in the yaw axis, ℓ designates the moment arm, that is, the distance between the center of mass of the quadrotor and each rotor, and k is a coefficient representing the relationship between the thrust and reaction torque in each rotor. The use of aerodynamic drag in the present form is supported by an experimental study [15].

In practice, the thrusts of the rotors are constrained within a certain range as

$$u_{\min} \leq u_i \leq u_{\max} \quad (i = 1, 2, 3, 4). \quad (10)$$

In most quadrotors, each rotor rotates in one fixed direction and cannot stop completely, which implies that $u_{\min} > 0$.

2.2 Control Objective

We consider FTC of the quadrotor to restore and maintain stable flight after some of the rotors lose their thrusts and the position and attitude of the quadrotor deviates from their reference values. In particular, we focus on reduction of one thrust u_1 by a constant thrust reduction factor c_1 ($0 \leq c_1 \leq 1$) in this chapter. More general cases of failures in the rotors and flight conditions could be considered with suitable modifications of the problem formulation. We assume that the loss of the thrust is identified by some algorithms for fault detection and fault isolation in a certain interval of time after the failure, and the FTC is then activated [25]. We regard the time and state at the activation of the FTC as the initial time $t = 0$ and initial state $x(0) = x_0$ of the FTC problem.

The reduction of thrust u_1 can be represented with vector s and matrix T in state equation (2) modified to s_{fail} and T_{fail} defined as

$$s_{\text{fail}} = [c_1 \ 1 \ 1 \ 1]^\top \in \mathbb{R}^4, \quad (11)$$

$$T_{\text{fail}} = \begin{bmatrix} 0 & \ell & 0 & -\ell \\ -c_1 \ell & 0 & \ell & 0 \\ c_1 k & -k & k & -k \end{bmatrix} \in \mathbb{R}^{3 \times 4}. \quad (12)$$

The terms s_{fail} and T_{fail} coincide with s and T , respectively, when $c_1 = 1$, which implies no reduction in thrust. Additionally, as seen from (2), the thrust produced by a rotor specified by u_1 is completely lost when $c_1 = 0$. The constraints in (10) are the same for all thrusts $u_1, \dots, u_4$ before and after the failure, and therefore, u_1 is regarded as a thrust command rather than the actual thrust after the failure.

The control objective of FTC is to regulate the state x from a given initial state $x(0) = x_0$ to a given reference state x_{ref} after the failure, that is, for the state equation (2) with s and T replaced by s_{fail} and T_{fail} , respectively. In case of hovering, the reference position ξ_{ref} can be assumed to be zero without loss of generality, and the reference quaternion q_{ref} is given by $q_{\text{ref}} = [\pm 1 \ 0 \ 0 \ 0]^\top$ to maintain the upright attitude. Note that q_{ref} represents the same upright attitude regardless of the sign of q_0 , and, since $\sum_{i=0}^3 q_i^2 = 1$ holds, it suffices to regulate $q_1 = q_2 = q_3 = 0$ to achieve the upright attitude. Reference velocity v_{ref} and reference angular velocity ω_{ref} are also zero to maintain the reference position and attitude. Therefore, the reference state x_{ref} is given by

$$x_{\text{ref}} = [0 \ 0 \ 0 \ 0 \ 0 \ \pm 1 \ 0 \ 0 \ 0 \ 0 \ 0]^\top. \quad (13)$$

3 Benchmark Design

We employ the Parrot Mambo minidrone¹ as the platform for the benchmark problem. Table 1 summarizes its physical parameters taken from a Simulink Supprot Package². We assume there is neither disturbance nor modelling error in the system, all state variables are measurable at sampling period of 1 ms without noises, and the thrust reduction factor c_1 is known. We also assume the control inputs $u_1, \dots, u_4$ are applied without delay at every sampling time.

Benchmark Problem For a given thrust reduction factor $c_1 \in [0, 1]$, design a feedback controller for the quadrotor to regulate its state from a given initial state $x(0) = x_0$ to the reference state x_{ref} . Evaluate the controller with the following performance metrics for each component of the position, ξ_1 , ξ_2 , and ξ_3 , respectively.

- Steady state error
- Overshoot
- Settling time
- Undershoot

¹ <https://www.parrot.com/us/support/documentation/mambo-range>

² <https://www.mathworks.com/hardware-support/parrot-minidrones.html>

Table 1 Physical parameters of the quadrotor.

Parameter	Symbol	Value	Unit
Mass	m	0.063	kg
Gravitational acceleration	g	9.81	m/s ²
Moment arm	ℓ	0.0624	m
Reaction coefficient	k	0.0731	m
Moment of inertia	J_1	5.83×10^{-5}	kg·m ²
Moment of inertia	J_2	7.17×10^{-5}	kg·m ²
Moment of inertia	J_3	1.00×10^{-4}	kg·m ²
Drag coefficient	d_3	1.00×10^{-3}	N·m·s
Maximum thrust	$u_{\max}$	0.3266	N
Minimum thrust	$u_{\min}$	0.0065	N

When the initial value $\xi_i(0)$ is negative, the performance metrics are defined as follows. The steady state value $\bar{\xi}_i$ of a coordinate ξ_i is defined by its mean value over a certain time interval before the end of the simulation. For peak value $\xi_i^{\text{peak}} = \max_t \xi_i(t)$ of a coordinate ξ_i , the overshoot is defined by $100(\xi_i^{\text{peak}} - \bar{\xi}_i)/(\bar{\xi}_i - \xi_i(0))$ [%]. The settling time is the minimum time such that $\xi_i(t)$ enters and remains in the range of $\pm 2\%$ from $\bar{\xi}_i$. For the minimum value $\xi_i^{\min} = \min_t \xi_i(t)$ of a coordinate ξ_i , the undershoot is defined by $100(\xi_i(0) - \xi_i^{\min})/(\bar{\xi}_i - \xi_i(0))$ [%]. In particular, the undershoot of ξ_3 indicates the altitude loss at the beginning of the control due to the rotor failure.

Note that the state of the quadrotor may not converge to a constant value due to the unbalanced thrusts and reaction torques generated by the normal rotors. In particular, the quadrotor can keep rotating around the e_3^b axis, inducing rotations around other axes. Therefore, we focus on the performance metrics in the position. Moreover, in case the position also keeps oscillating and does not converge to a constant value, the settling threshold of $\pm 2\%$ might be relaxed to, for example, $\pm 5\%$ or the magnitude of steady-state oscillation might be used as an alternative performance metric.

The initial state is given by

$$x_0 = [-1 \ -1 \ -1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^T \quad (14)$$

for the normal operation and FTC. In the case of FTC, a more critical situation can be evaluated, for example, with an initial state,

$$x_0 = \left[-1 \ -1 \ -1 \ 0 \ 0 \ 0 \ \cos \frac{\theta_0}{2} \ 0 \ \sin \frac{\theta_0}{2} \ 0 \ 0 \ \omega_{20} \ 0 \right]^T, \quad (15)$$

where θ_0 and ω_{20} denote the initial values of the pitch angle and its angular velocity, respectively.

4 Accessing the Benchmark

Jupyter notebooks for simulating the FTC of the quadrotor are provided at a GitHub repository³. This repository is based on the automatic code generation tool for NMPC, AutoGenU for Jupyter⁴ [11], and contains the following notebooks for the present benchmark problem.

- QuadrotorFTC.ipynb
- RunSimQuadrotorFTC.ipynb
- PlotSimQuadrotorFTC.ipynb

QuadrotorFTC.ipynb is a notebook for automatic generation of C++ code and its Python interface for NMPC of the quadrotor as well as the closed-loop simulation by the generated C++ code. A design example and simulation results of NMPC will be given in the next section as the baseline. RunSimQuadrotorFTC.ipynb is a notebook for the closed-loop simulation by the Python interface generated by QuadrotorFTC.ipynb, and PlotSimQuadrotorFTC.ipynb is a notebook for plotting a simulation result by QuadrotorFTC.ipynb or RunSimQuadrotorFTC.ipynb. Although RunSimQuadrotorFTC.ipynb uses the Python interface generated by QuadrotorFTC.ipynb for NMPC, it can also be used for the closed-loop simulation of any other user-defined controllers in Python. PlotSimQuadrotorFTC.ipynb can also be used for plotting the closed-loop responses of the user-defined controllers.

First, a user needs to set up a JupyterLab or Jupyter Notebook⁵ together with a C++ compiler, a Python environment, and some other tools according to the README.md in the repository⁶. Then, the user can clone the repository to a local working directory QuadrotorFTC_Benchmark with the following command.

```
git clone https://github.com/ohtsukalab/QuadrotorFTC_Benchmark
--recursive
```

The working directory QuadrotorFTC_Benchmark contains the above three notebooks together with various files and directories for automatic code generation.

Next, the user opens QuadrotorFTC.ipynb by JupyterLab and runs it to generate C++ code and a Python interface for NMPC. The generated C++ source files are saved in the QuadrotorFTC_Benchmark/generated/QuadrotorFTC directory. QuadrotorFTC.ipynb also builds the executable file from the generated C++ source files and performs the closed-loop simulation. The log files of the simulation results are saved in QuadrotorFTC_Benchmark/generated/QuadrotorFTC/log directory. Then, the user can also run the closed-loop simulation with RunSimQuadrotorFTC.ipynb. An advantage of using RunSimQuadrotorFTC.ipynb is that the user can change some parameters in the model and NMPC in Python. That is, once a Python interface is generated by QuadrotorFTC.ipynb, the

³ https://github.com/ohtsukalab/QuadrotorFTC_Benchmark

⁴ <https://ohtsukalab.github.io/autogenu-jupyter>

⁵ <https://jupyter.org/>

⁶ https://github.com/ohtsukalab/QuadrotorFTC_Benchmark#readme

user can modify some of the parameters in NMPC without regenerating the C++ source files and Python interface. As the simulation model of the quadrotor is redefined in `RunSimQuadrotorFTC.ipynb`, the user can also perform the closed-loop simulation with model–plant mismatch.

Finally, the user can plot the simulation result in the log files by running `PlotSimQuadrotorFTC.ipynb`. `PlotSimQuadrotorFTC.ipynb` plots time histories of some variables and evaluates the performance metrics in the previous section. It also plots the trajectory of the quadrotor in the three-dimensional space and generates its animation.

For Windows users and Mac users, pre-built Python interfaces are available in the zip files at the release page⁷ of the repository. If a user downloads and extracts `QuadrotorFTC_Benchmark_Win.zip` (for Windows) or `QuadrotorFTC_Benchmark_Mac.zip` (for Mac), the user can directly run `RunSimQuadrotorFTC.ipynb` for simulation of FTC by NMPC and `PlotSimQuadrotorFTC.ipynb` for plotting the simulation result. The user does not need to run `QuadrotorFTC.ipynb` for generating C++ code and Python interfaces for NMPC.

5 Discussion for Future Comparison

5.1 Nonlinear Model Predictive Control

In this section, we employ NMPC as the baseline for FTC of the quadrotor, which can deal with the nonlinear model with input constraints. We define the performance index for NMPC by a simple quadratic function as follows:

$$J = \varphi(x(t+T)) + \int_t^{t+T} L(x(\tau), u(\tau)) d\tau, \quad (16)$$

$$\varphi(x) = \frac{1}{2} (x - x_{\text{ref}})^\top S_f (x - x_{\text{ref}}), \quad (17)$$

$$L(x, u) = \frac{1}{2} (x - x_{\text{ref}})^\top S (x - x_{\text{ref}}) + \frac{1}{2} (u - u_{\text{ref}})^\top R (u - u_{\text{ref}}), \quad (18)$$

where S_f , S , and R are weighting matrices, x_{ref} is the reference state defined by (13), and $u_{\text{ref}} \in \mathbb{R}^4$ denotes the reference input, respectively. The weighting matrices S_f and S for the state are typically selected to be positive semi-definite, and the weighting matrix R for input is selected to be positive definite. Regarding the input reference u_{ref} , we impose a balance condition

$$c_1 u_{\text{ref}1} + \sum_{i=2}^4 u_{\text{ref}i} = mg \quad (19)$$

⁷ https://github.com/ohtsukalab/QuadrotorFTC_Benchmark/releases

so that the sum of the reference thrusts balances the gravitational force. Although reference thrusts satisfying (19) are not unique, we adopt the same value for all thrusts as

$$u_{\text{ref}i} = \frac{mg}{c_1 + 3} \quad (i = 1, \dots, 4). \quad (20)$$

At each sampling time t of NMPC, we solve an optimal control problem (OCP) to find an optimal control input $u_{\text{opt}}(\tau; x(t))$ ($t \leq \tau \leq t + T$) minimizing the performance index J in (16) subject to the state equation (3) and the input constraint (10). Note that the optimal control $u_{\text{opt}}(\tau; x(t))$ depends on the current state $x(t)$ of the actual system because the initial condition for the OCP to be solved at time t is given by $x(t)$. If we can solve the OCP sufficiently fast within the sampling period, then we give the actual input to the system by the initial value of the optimal control over the horizon $[t, t + T]$ as

$$u(t) = u_{\text{opt}}(t; x(t)). \quad (21)$$

Therefore, NMPC is a type of state feedback even if the OCP is solved numerically.

Since it is crucial for implementation of NMPC to solve an OCP in real time within the sampling period, real-time optimization methods tailored for NMPC have been actively studied for more than two decades [19, 18] (see also [2, 21, 20] for various topics in MPC). Our software tools employ the C/GMRES method [18], which is one of the real-time optimization methods for NMPC.

The horizon length T for NMPC is given by a time-dependent function $T = T_f(1 - e^{-\alpha t})$ with $T_f = 0.4$ [s] and $\alpha = 1$, and the parameter for stabilization of the continuation method is given by $\zeta = 1000$ in C/GMRES. The number of discretization steps over the horizon is given by $N = 100$. The inequality constraints on the input u_i are transformed to equality constraints by introducing dummy inputs u_{di} ($i = 1, 2, 3, 4$) as

$$C_i(u_i, u_{di}) = (u_i - \bar{u}_i)^2 + u_{di}^2 - u_{\text{mmi}}^2 = 0, \quad (22)$$

where

$$\bar{u}_i = \frac{u_{\text{max}} + u_{\text{min}}}{2}, \quad u_{\text{mmi}} = \frac{u_{\text{max}} - u_{\text{min}}}{2}. \quad (23)$$

A penalty term $-w_d \sum_{i=1}^4 u_{di}$ is also added to the stage cost $L(x, u)$ in (18) to avoid a singularity when the dummy input becomes zero. The weights in the cost function (16) are selected as

$$S_f = S = \text{diag}(5, 5, 50, 1, 1, 1, 0, 1, 1, 1, 0.1, 0.1, 0.1), \quad R = I, \quad w_d = 100 \quad (24)$$

Since the sign of $x_{\text{ref}7} = q_{\text{ref}0}$ is not unique, and it suffices to regulate $q_1 = q_2 = q_3 = 0$, we set the weights for q_0 to be zero.

5.2 Simulation Results

Code generation for NMPC and closed-loop simulations were performed on a PC (OS: Windows 11 Pro, CPU: Intel Core i7-1165G7 2.80 GHz, RAM 32 GB). The average computation time per update of NMPC was 0.14 ms for all cases. Therefore, NMPC can be implemented for FTC of the quadrotor in real time for the sampling period of 1 ms.

Figures 2–4 show time histories of the position $(x, y, z) = (\xi_1, \xi_2, \xi_3)$, Euler angles (ϕ, θ, ψ) (roll, pitch, and yaw angles), and control inputs in NMPC for normal operation ($c_1 = 1$), partial failure ($c_1 = 0.5$), and complete failure ($c_1 = 0$), respectively, with the same initial state x_0 in (14). Here, we use symbol x to denote the coordinate ξ_1 for better readability although it is same as the state vector. Note also that the actual thrusts are given by $(c_1 u_1, u_2, u_3, u_4)$. Tables 2–4 summarize the performance metrics for each case. In all the cases, NMPC successfully regulates the position of the quadrotor near the reference position without violating the input constraint. In Fig. 4, the two inputs u_2 and u_4 saturate at the maximum value, and the altitude drops more than 2.5 m before recovering, which demonstrates the critical situation of the failure. In Figs. 3 and 4 for the cases of failure ($c_1 = 0.5, 1$), the attitude angles keep oscillating, and, in particular, the yaw angle ψ keeps rotating due to the unbalance in the thrusts. Although those oscillations in the attitude also induce small oscillations in the position, the position settles within $\pm 2\%$ of the steady state.

Finally, Figure 5 and Table 5 present the time histories and performance metrics, respectively, for complete failure ($c_1 = 0$) with the initial state (15) of a non-zero initial pitch angle $\theta_0 = \pi/4$ [rad] and a non-zero initial pitch rate $\omega_{20} = 1$ [rad/s]. This is a more critical situation than that in the previous cases starting from hovering. Figure 5 and Table 5 exhibit a significant altitude loss at the beginning of the control and a larger overshoot in x (ξ_1). However, NMPC still regulates the position of the quadrotor successfully near the reference state.

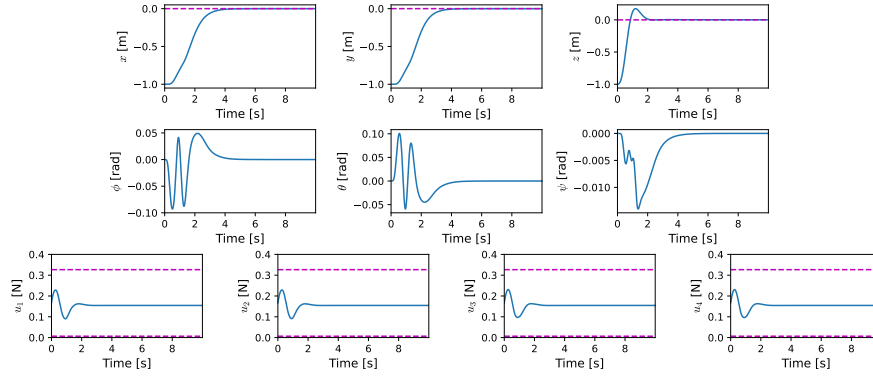


Fig. 2 Time histories of nonlinear model predictive control (NMPC) for normal operation, $c_1 = 1$.

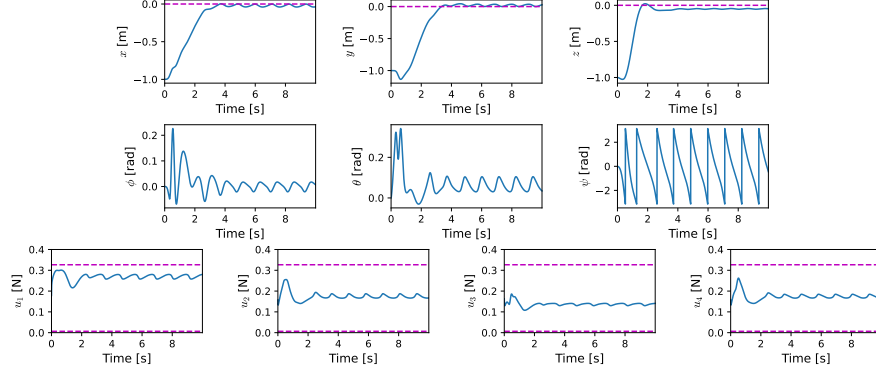


Fig. 3 Time histories of NMPC for partial failure, $c_1 = 0.5$.

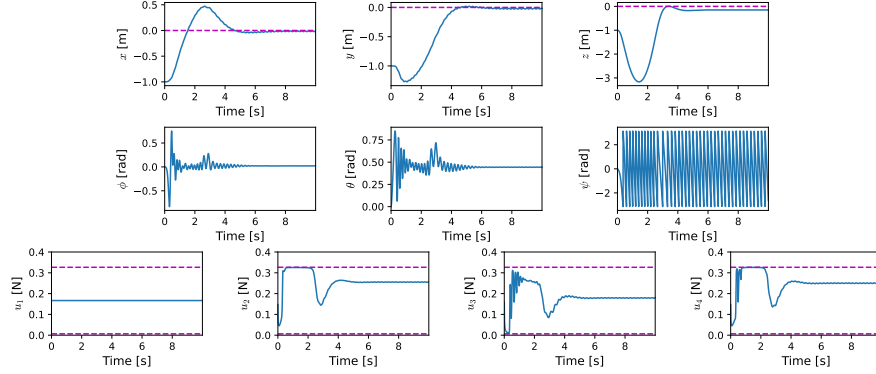


Fig. 4 Time histories of NMPC for complete failure, $c_1 = 0$.

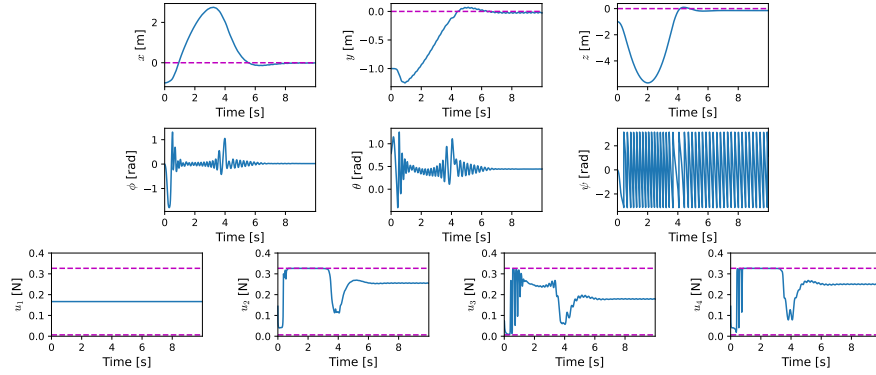


Fig. 5 Time histories of NMPC for complete failure, $c_1 = 0$, with $\theta_0 = \pi/4$ [rad] and $\omega_{20} = 1$ [rad/s].

Table 2 Performance metrics of NMPC for normal operation ($c_1 = 1$).

Performance metric	Unit	x	y	z
Steady state error	m	0.00	0.00	0.00
Overshoot	%	0.00	0.00	17.29
Settling time	s	3.59	3.45	1.92
Undershoot	%	0.00	0.00	0.00

Table 3 Performance metrics of NMPC for partial failure ($c_1 = 0.5$).

Performance metric	Unit	x	y	z
Steady state error	m	-0.02	0.02	-0.05
Overshoot	%	2.59	2.53	7.42
Settling time	s	9.42	4.74	3.54
Undershoot	%	0.00	13.49	2.72

Table 4 Performance metrics of NMPC for complete failure ($c_1 = 0$).

Performance metric	Unit	x	y	z
Steady state error	m	-0.02	-0.02	-0.15
Overshoot	%	49.43	4.12	18.22
Settling time	s	6.35	6.16	5.08
Undershoot	%	0.00	27.82	254.74

Table 5 Performance metrics of NMPC for complete failure ($c_1 = 0$) with $\theta_0 = \pi/4$ [rad] and $\omega_{20} = 1$ [rad/s].

Performance metric	Unit	x	y	z
Steady state error	m	-0.01	-0.02	-0.15
Overshoot	%	280.09	9.58	2950
Settling time	s	7.94	6.56	6.22
Undershoot	%	0.00	25.70	551.01

5.3 Future Directions

Although NMPC successfully recovers the position of the quadrotor with one rotor failed, there are still some issues to be addressed. First, real-time optimization for NMPC is computationally demanding and may not be implemented in embedded flight controllers for low-cost quadrotors. Therefore, control methods with low computation costs are preferable as long as they can cope with the nonlinear dynamics and input constraints in quadrotors. In addition to extending traditional nonlinear control methods, it would also be worth exploring appropriate use of machine learning techniques [5]. Second, NMPC in the present setting requires accurate measurements of all the state variables and information about the thrust reduction factor, which are not necessarily available in practice. Integrating state estimation and fault diagnosis and developing feedback control methods based only on available measurements are possible directions of research. Finally, the above described NMPC has a steady-state error in the position of the quadrotor. Although the steady-state error may be accept-

able in the critical situation of a rotor failure, a systematic and simple technique to compensate the steady-state error can be useful in some tasks for a quadrotor.

Acknowledgements This work was partly supported by JSPS KAKENHI Grant Number 22H01510.

Appendix

This appendix summarizes some properties of the quaternion and Euler angles (see [6, Appendix E] and [7] for details). The coordinate axes of the body-fixed frame are represented in terms of the quaternion $q = [q_0 \ q_1 \ q_2 \ q_3]^\top$ as follows:

$$\begin{bmatrix} e_1^b & e_2^b & e_3^b \end{bmatrix} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(-q_0q_3 + q_1q_2) & 2(q_0q_2 + q_1q_3) \\ 2(q_0q_3 + q_1q_2) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(-q_0q_1 + q_2q_3) \\ 2(-q_0q_2 + q_1q_3) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad (25)$$

A quaternion $q = [q_0 \ q_1 \ q_2 \ q_3]^\top$ can be converted to ZYX Euler angles consisting of roll angle ϕ , pitch angle θ , and yaw angle ψ as follows:

$$\phi = \arctan \left(\frac{2(q_0q_1 + q_2q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2} \right), \quad (26)$$

$$\theta = \arcsin (-2(q_1q_3 - q_0q_2)), \quad (27)$$

$$\psi = \arctan \left(\frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2} \right). \quad (28)$$

Conversely, ZYX Euler angles ϕ , θ , and ψ are converted to a quaternion q as follows:

$$q = \begin{bmatrix} \cos(\frac{\psi}{2}) \cos(\frac{\theta}{2}) \cos(\frac{\phi}{2}) + \sin(\frac{\psi}{2}) \sin(\frac{\theta}{2}) \sin(\frac{\phi}{2}) \\ \cos(\frac{\psi}{2}) \cos(\frac{\theta}{2}) \sin(\frac{\phi}{2}) - \cos(\frac{\psi}{2}) \sin(\frac{\theta}{2}) \sin(\frac{\phi}{2}) \\ \cos(\frac{\psi}{2}) \sin(\frac{\theta}{2}) \cos(\frac{\phi}{2}) + \sin(\frac{\psi}{2}) \cos(\frac{\theta}{2}) \sin(\frac{\phi}{2}) \\ \sin(\frac{\psi}{2}) \cos(\frac{\theta}{2}) \cos(\frac{\phi}{2}) - \cos(\frac{\psi}{2}) \sin(\frac{\theta}{2}) \sin(\frac{\phi}{2}) \end{bmatrix}. \quad (29)$$

References

1. Alaimo, A., Artale, V., Milazzo, C., Ricciardello, A., Trefiletti, L.: Mathematical modeling and control of a hexacopter. In: 2013 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1043–1050 (2013)
2. Allgöwer, F., Findeisen, R., Nagy, Z.K.: Nonlinear model predictive control: From theory to application. *Journal of the Chinese Institute of Chemical Engineers* **35**(3), 299–316 (2004)
3. Aoki, Y., Asano, Y., Honda, A., Motooka, N., Hoshino, K., Ohtsuka, T.: Nonlinear model predictive control for hexacopter with failed rotors based on quaternions—simulations and hardware experiments—. *Mechanical Engineering Journal* **8**(5), 21–00204 (2021)

4. 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* **26**(2), 701–707 (2017)
5. Baskaya, E., Bronz, M., Delahaye, D.: Fault detection & diagnosis for small uavs via machine learning. In: 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), pp. 1–6 (2017)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W.: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT press (2005)
7. Fresk, E., Nikolakopoulos, G.: Full quaternion based attitude control for a quadrotor. In: 2013 European control conference (ECC), pp. 3864–3869 (2013)
8. Hamel, T., Mahony, R., Lozano, R., Ostrowski, J.: Dynamic modelling and configuration stabilization for an X4-flyer. *IFAC Proceedings Volumes* **35**(1), 217–222 (2002)
9. Hao, W., Xian, B.: Nonlinear adaptive fault-tolerant control for a quadrotor UAV based on immersion and invariance methodology. *Nonlinear Dynamics* **90**, 2813–2826 (2017)
10. Izadi, H.A., Zhang, Y., Gordon, B.W.: Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation. *IFAC Proceedings Volumes* **44**(1), 6343–6348 (2011)
11. Katayama, S., Ohtsuka, T.: Automatic code generation tool for nonlinear model predictive control with Jupyter. *IFAC Proceedings Volumes* **53**(2), 7033–7040 (2020)
12. Li, T., Zhang, Y., Gordon, B.W.: Passive and active nonlinear fault-tolerant control of a quadrotor unmanned aerial vehicle based on the sliding mode control technique. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* **227**(1), 12–23 (2013)
13. Mahony, R., Kumar, V., Corke, P.: Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine* **19**(3), 20–32 (2012)
14. Milhim, A., Zhang, Y., Rabbath, C.A.: Gain scheduling based PID controller for fault tolerant control of quad-rotor UAV. In: AIAA Infotech Aerospace 2010, p. 3530. AIAA (2010)
15. Mueller, M.W., D’Andrea, R.: Stability and control of a quadcopter despite the complete loss of one, two, or three propellers. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 45–52 (2014)
16. Murray, R.M., Li, Z., Sastry, S.S.: *A Mathematical Introduction to Robotic Manipulation*. CRC press (1994)
17. Nan, F., Sun, S., Foehn, P., Scaramuzza, D.: Nonlinear MPC for quadrotor fault-tolerant control. *IEEE Robotics and Automation Letters* **7**(2), 5047–5054 (2022)
18. Ohtsuka, T.: A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica* **40**(4), 563–574 (2004)
19. Ohtsuka, T., Fujii, H.A.: Real-time optimization algorithm for nonlinear receding-horizon control. *Automatica* **33**(6), 1147–1154 (1997)
20. Raković, S.V., Leving, W.S. (eds.): *Handbook of Model Predictive Control*. Birkhäuser (2019)
21. Rawlings, J.B., Mayne, D.Q., Diehl, M.M.: *Model Predictive Control: Theory, Computation, and Design*, second edn. Nob Hill Publishing (2020)
22. Sharifi, F., Mirzaei, M., Gordon, B.W., Zhang, Y.: Fault tolerant control of a quadrotor uav using sliding mode control. In: 2010 Conference on Control and Fault-Tolerant Systems (SysTol), pp. 239–244 (2010)
23. Shraim, H., Awada, A., Youness, R.: A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control. *IEEE Aerospace and Electronic Systems Magazine* **33**(7), 14–33 (2018)
24. Tayebi, A., McGilvray, S.: Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology* **14**(3), 562–571 (2006)
25. 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* **350**(9), 2396–2422 (2013)