aerospace

MDPI

Article

Flight-Data-Based High-Fidelity System Identification of DJI M600 Pro Hexacopter

Péter Bauer * and Mihály Nagy

Systems and Control Laboratory, Institute for Computer Science and Control (SZTAKI), HUN-REN, Kende utca 13-17, H-1111 Budapest, Hungary; nagy.mihaly@sztaki.hun-ren.hu
* Correspondence: bauer.peter@sztaki.hun-ren.hu

Abstract: Research and industrial application can require custom high-level controllers for industrial drones. Thus, this paper presents the high-fidelity dynamic and control model identification of the DJI M600 Pro hexacopter. This is a widely used multicopter in the research and industrial community due to its high payload capability and reliability. To support these communities, the focus of control model identification was on the exploration and implementation of DJI Onboard Software Development Kit (OSDK) functionalities, also including some unconventional special modes. Thus, the resulting model can be controlled with the same OSDK functionalities as the real drone, making control development and application time effective. First, the hardware and software structure of the additional DJI M600 onboard system are introduced. Then, the postulated dynamic and control system models are shown. Next, real flight test campaigns generating data for system identification are presented. Then, the mass and inertial properties are estimated for TB47S and TB48S battery sets and the custom Forerunner UAV payload. Dynamic system model identification includes the aerodynamic effects and considers hover, vertical, and horizontal forces together with static horizontal wind components and finally the rotational moments and dynamics. The control system components were identified following the structure of OSDK, including vertical, horizontal, and yaw loops. After identification, the model was validated and refined based on an unused flight test and software-in-the-loop simulation data. The simulation is provided by DJI and was also compared to real flight results. This comparison showed that the DJI simulation covers the dynamics of the real drone well, but it requires being connected to the drone and needs the controllers onboard to be implemented in advance, which limits applicability and increases development time. This was another motivation to introduce a standalone simulation in Matlab Simulink, which covers all the important modes of OSDK control and can be run solely in Matlab without any hardware support. The constructed model will be published for the benefit of the research and industrial community.

Keywords: system identification; DJI M600 Pro; high-fidelity simulation; real flight test

MSC: 93B30

Citation: Bauer, P.; Nagy, M. Flight-Data-Based High-Fidelity System Identification of DJI M600 Pro Hexacopter. *Aerospace* **2024**, *11*, 79. https://doi.org/10.3390/aerospace11010079

Academic Editor: Gokhan Inalhan

Received: 17 November 2023
Revised: 21 December 2023
Accepted: 1 January 2024
Published: 15 January 2024

1. Introduction

Multirotor platforms with different payloads are widely applied in research and industrial applications. The DJI M600 Pro is a preferred platform due to its high payload capacity (5–6 kg) and redundant systems [1–8]. Most of the applications only utilize the factory capabilities but for specific tasks, custom high-level control is required [5,9–13]. The tuning requires the identification of system dynamics to be able to simulate system responses before field testing ([9,10]).

As the DJI M600 is a widely applied platform (see the references above) and the Forerunner UAV (see Figure 1) was a specific application which required a custom velocity controller (see [12,13]), our research team (see authors of [11]) decided to create a high-fidelity identification model of a DJI M600. The goal was to cover the whole flight envelope

and also the factory control loops applied in DJI OSDK (Onboard Software Development Kit see [14]), providing the opportunity for simulation and controller development at any level (except for the lowest levels controlling: pitch and roll angular rates and thrust). Identification was based on real flight data, considering also the wind disturbances. The resulting Matlab Simulink simulation model is openly published (see the Supplementary Material) for the advantage of the research and industrial community.

Figure 1. DJI M600 Pro Forerunner UAV platform with extra payload systems.

System identification of multicopters is a widely researched topic [9,10,15–25]. From a model structure point of view, sources can be grouped as low-fidelity ([9,10,19,20,22–25]) and high-fidelity ([15–18,21]) methods, the latter considering details of aerodynamics and flight mechanics. Scheduled linear models are considered to be low fidelity, as their basic model is a linearized low-fidelity one. The applied methodologies for parameter estimation can be transfer function identification from input–output data [10,17,20], time domain parameter estimation for a fixed model structure (gray box identification) [9,18,19,21], frequency domain system identification [22,24,25] and even neural network-based system modeling [23]. The considered data sources can be wind tunnel and test bench [17–19,21] or real flight [9,10,22–25] data.

Closest to our goal was the article [9] identifying the dynamics of the DJI M600 hexacopter. Similarities include the consideration of low-level controllers and delays; however, there are significant differences, which are listed below.

- The article presents parameter-dependent linearized models instead of high-fidelity nonlinear ones.
- The article considers almost zero roll and pitch angles (near hover state), resulting in body and inertial forces close to each other and the neglect of roll and pitch rotational dynamics. This approximation limits model validity to low speeds (until 1.5–2 m/s).

On the contrary, our approach includes the following features:

- Nonlinear dynamical and aerodynamical model with rotor inertial effects, quadratic air drag, rotor hub forces and the effect of rotor inflow on ascend and descend dynamics.
- Coverage of the full flight envelope with maximum $\pm 25^\circ$ roll and pitch angles, 18 m/s forward and side speed, 3–5 m/s ascend and descend speed and 8 m/s wind resistance (see [26]).
- Modeling of the factory control loops from low to high-level, also including the identified special modes. Thus, the required control inputs of the simulation model are the same as those required to control the drone through DJI OSDK, providing an easy way to integrate and apply the simulation-tested controllers onboard.

The structure of the paper follows. Section 2 introduces the additional hardware and software structure of the DJI M600 Pro (further referenced as M600) applied for system

identification. Then, Section 3 introduces the postulated dynamic and control model structure of the DJI M600 targeted to be identified in the article. Section 4 introduces the flight test campaigns and the basic analysis of flight data, which has led to the discovery of two special modes. Section 5 discusses mass and inertia estimation, while Section 6 introduces the process of dynamic system model identification. After dynamic system identification, Section 7 discusses the identification of control system parameters targeting to cover the DJI OSDK modes. Finally, Section 8 shows the results of high-fidelity model validation and refinement, and Section 9 concludes the paper.

2. Hardware and Control Software Structure

The stock DJI M600 was equipped with a custom onboard system to transform it into a Forerunner UAV (see Figures 1–3). Figure 2 shows the connection of the onboard Nvidia Jetson Xavier NX computer to the autopilot of the DJI M600 through the OSDK UART port. This made it possible to control the drone with custom controllers sending OSDK commands through the UART port. Thus, the system was capable of flying special system identification maneuvers. Flight data logging for analysis and system identification was also completed through OSDK, reading and saving the specific packets at 50 Hz frequency. Flight test maneuvers were executed also through the stock ground control station UgCS ([27], see later).

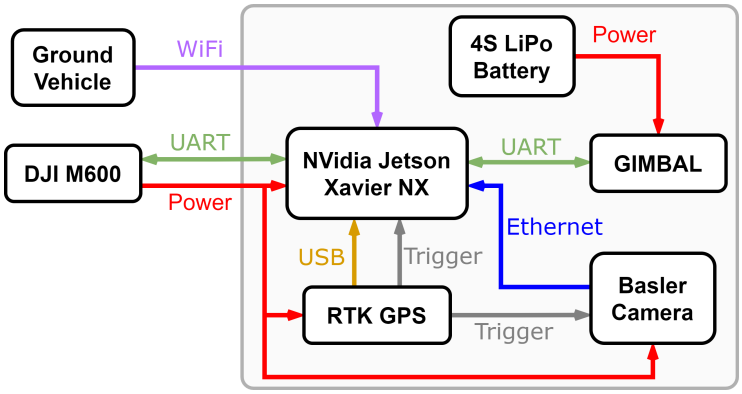

Figure 2. Block scheme of the drone onboard hardware in the Forerunner UAV setup.

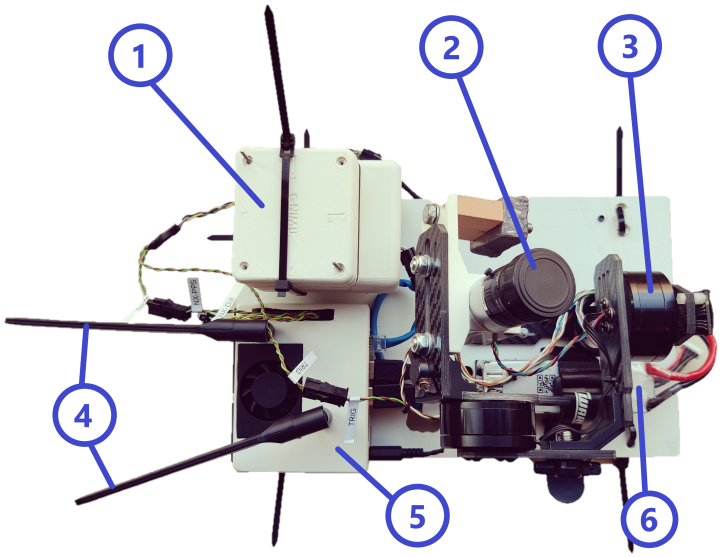

Figure 3. Onboard hardware system of the aerial vehicle. 1: RTK GNSS electronics, 2: Basler camera, 3: Gimbal, 4: WiFi antennas, 5: Nvidia Jetson Xavier NX, 6: 4S LiPo battery.

The possibilities of OSDK control are summarized in Table 1 based on [14].

Table 1. Possible OSDK control mode settings.

Logic Identifier	Mode Identifier	Explanation
HorizontalCoordinate	HORIZONTAL_GROUND	Set the x–y of ground frame as the horizontal frame (NEU)
	HORIZONTAL_BODY	Set the x–y of body frame as the horizontal frame (FRU)
HorizontalLogic	HORIZONTAL_ANGLE	Set the control mode to control pitch and roll angle of the vehicle
	HORIZONTAL_VELOCITY	Set the control mode to control horizontal vehicle velocities
	HORIZONTAL_POSITION	Set the control mode to control position offsets of pitch and roll directions
	HORIZONTAL_ANGULAR_RATE	Set the control mode to control rate of change of the vehicle's attitude
VerticalLogic	VERTICAL_VELOCITY	Set the control mode to control the vertical speed of UAV, upward is positive
	VERTICAL_POSITION	Set the control mode to control the height of UAV
	VERTICAL_THRUST	Set the control mode to directly control the thrust
YawLogic	YAW_ANGLE	Set the control mode to control yaw angle
	YAW_RATE	Set the control mode to control yaw angular velocity

The high-fidelity simulation model is designed to implement every possibility except for the HORIZONTAL_ANGULAR_RATE and VERTICAL_THRUST cases, as these are low-level controls rarely applied in high-level mission controllers. Moreover, the direct actuation of such low-level controls can be dangerous. Thus, only the other modes were tested in the flight test campaigns to correctly identify OSDK functionalities.

3. High-Fidelity Simulation Model Structure

The high-fidelity simulation model was targeted to describe the 6 degrees of freedom (DoF) nonlinear system dynamics of the hexacopter including engine dynamic and aerodynamic effects, and the controller structure was based on the OSDK functionalities (see Table 1). The main structure is shown in Figure 4, while the controller details are shown in Figure 5.

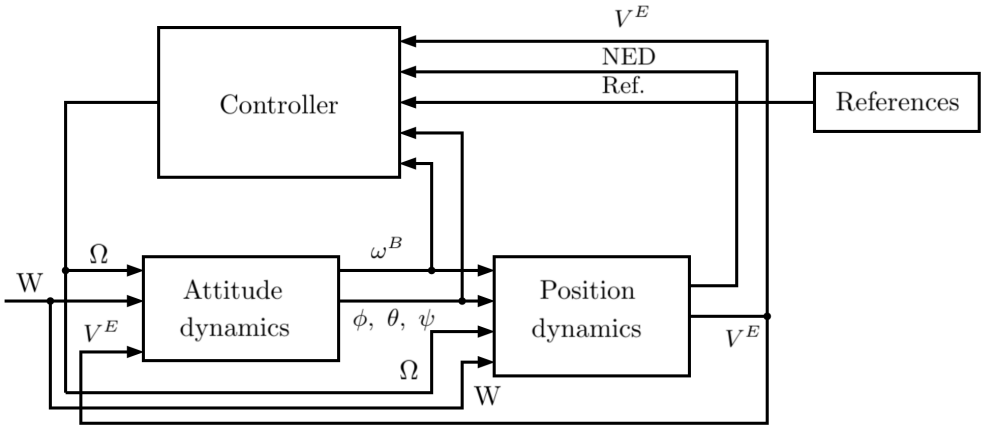

Figure 4. Block diagram of DJI M600 Pro 6DoF simulation model.

Figure 4 shows that the 6 DoF dynamics were separated into attitude (rotational) and position (translational) dynamics, the former including the yaw (Section 6.4) and roll/pitch (Section 6.3), the latter including the vertical (Section 6.2) and horizontal (Section 6.1) dynamics. The References block covers the generation of an OSDK control flag (see Table 1 for the different modes) and the related X, Y, Z and yaw references.

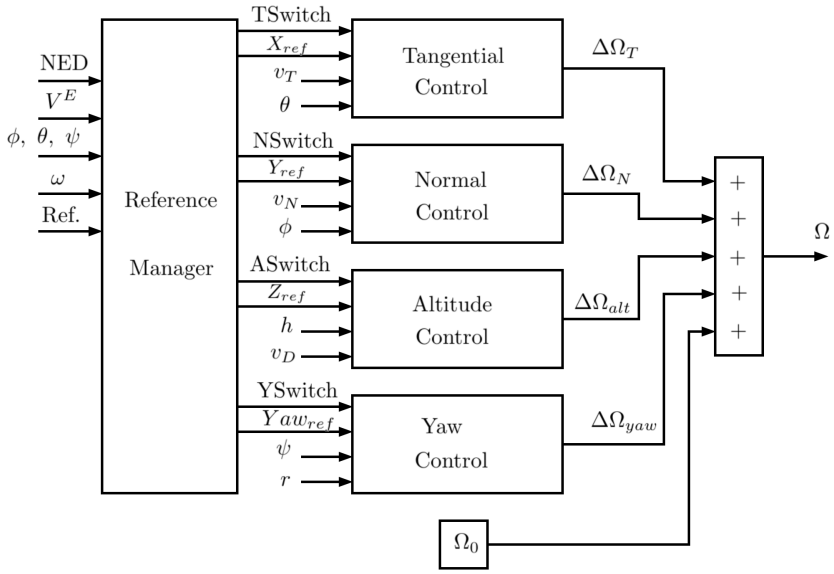

Figure 5. Block diagram of identified DJI M600 controller implementing OSDK modes.

The controller block is detailed in Figure 5 having the speeds of the six engines as output (Ω). In addition to the references, switching signals (TSwitch, NSwitch, ASwitch and YSwitch) were applied to the controllers determining if velocity or angle (TSwitch, NSwitch), altitude or velocity (ASwitch) and yaw angle or yaw rate (YSwitch) are tracked in the actual control (see Table 1 for OSDK control modes). The resulting engine speed is the sum of the controller incremental outputs $\Delta\Omega$ and the initial speed for hovering Ω_0. The overall engine speed is simply saturated because this did not cause any problems in the simulation, although it is well known that saturation can even cause instability [28]. The Reference Manager block obtains the references including OSDK flag and the states of the drone and generates the switching signals and the related X_{ref}, Y_{ref}, Z_{ref} and Yaw_{ref} references. Handling of the position kick mode (Section 8.3.1) and ground reference mode (Section 8.3.2) were also implemented inside this block.

3.1. Coordinate Systems and Transformations

Before detailing the model equations, the applied coordinate systems are considered as shown in Figures 6 and 7. They are the north–east–down (NED) and Body systems. The former is in the local tangential plane of the globe with axes pointing to the north, east and inside the globe (down). The latter has its *X*-axis pointing toward the front of the drone, *Y*-axis pointing to the right and *Z*-axis pointing downward (see Figure 7). Because of the short distances flown by a hexacopter drone, the NED system can be considered as the inertial system. The rotational transformation between the NED and body system was defined with the well-known Euler angles, having roll ϕ, pitch θ and yaw ψ motions (see, e.g., [29]).

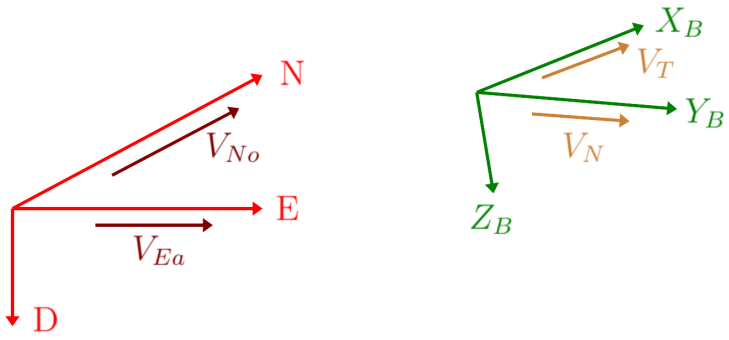

Figure 6. The applied coordinate systems.

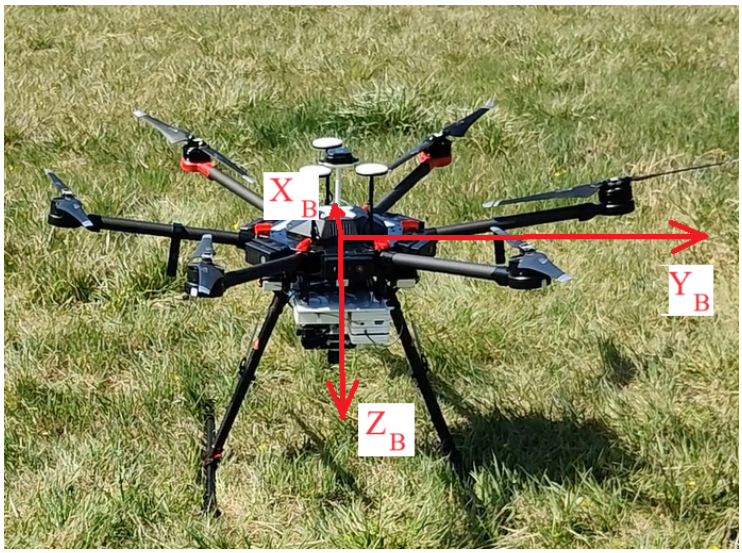

Figure 7. The body coordinate system on M600.

The horizontal velocities deduced from the DJI OSDK autopilot description (see [14]) were V_{No}, V_{Ea} north and east velocities defined in the NED system and V_T, V_N tangential and normal velocities defined parallel to the NE horizontal plane but aligned with body yaw orientation (body misalignment with the NE horizontal plane is visualized in Figure 6 with the velocity vectors being unaligned with the system). The transformation between them is defined in (1).

$$\begin{bmatrix} V_T \\ V_N \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} V_{No} \\ V_{Ea} \end{bmatrix} \tag{1}$$

In addition to the coordinate systems, the engine layout and geometry of the hexacopter is crucial in modeling its dynamics. It is presented in Figure 8 together with engine rotational directions (L is for left, R is for right, F is for front, B is for back and S is for side).

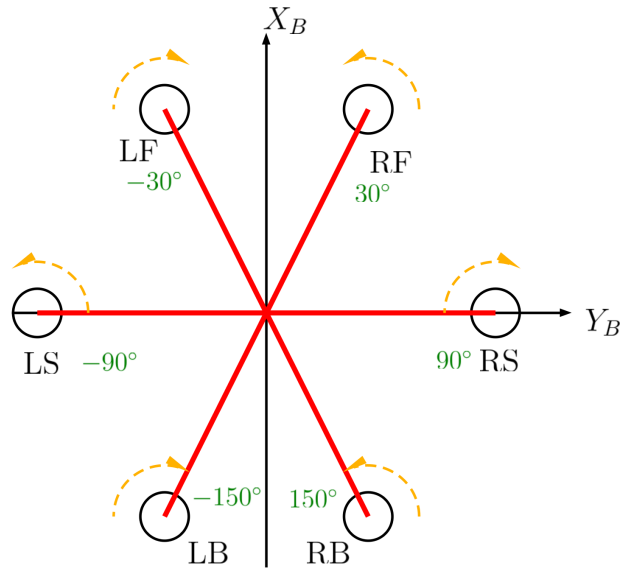

Figure 8. Engines, their angular positions and their rotational directions in body system of DJI M600.

3.2. Details of the 6DoF Dynamic Model

Figure 4 shows that the inputs to the attitude dynamics block are engine angular rates $\Omega = \begin{bmatrix} \Omega_{LB} & \Omega_{LF} & \Omega_{LS} & \Omega_{RB} & \Omega_{RF} & \Omega_{RS} \end{bmatrix}^T$, windspeed vector $W^E = \begin{bmatrix} W_N & W_E & W_D \end{bmatrix}^T$

and groundspeed $V^E = \begin{bmatrix} V_{No} & V_{Ea} & V_D \end{bmatrix}^T$, while its outputs are the body angular rate $\omega^B = \begin{bmatrix} p & q & r \end{bmatrix}^T$ and the Euler angles (ϕ, θ, ψ).

This block implements the hexacopter rotational dynamics assuming a diagonal inertia matrix $J = \langle J_x \quad J_y \quad J_z \rangle$ (which is not completely true, as shown in Table 4, but it is a good approximation), the dynamics of the Euler angles and the calculation of the torques acting on the hexacopter body. The dynamic equations can be obtained from, e.g., [29].

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 1/J_x & 0 & 0 \\ 0 & 1/J_y & 0 \\ 0 & 0 & 1/J_z \end{bmatrix} \left(\begin{bmatrix} (J_y - J_z)qr \\ (J_z - J_x)pr \\ (J_x - J_y)pq \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \right) \tag{2}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi tg\theta & c\phi tg\theta \\ 0 & c\phi & -c\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3}$$

Here, τ_x, τ_y, τ_z are the torques acting on the hexacopter body and c, s, tg are shorthands for cosine, sine and tangent.

The inputs to the position dynamics block are the body angular rate ω^B, Euler angles, engine angular rates Ω and windspeed vector W. Its outputs are the NED position and velocity V^E. This block implements the hexacopter translational dynamics again deduced from [29].

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + g^B + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \underbrace{\begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}}_{T_{EB}} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{5}$$

Here, m is the mass of the hexacopter, T_{EB} is the body to earth rotational matrix, $V^B = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = T_{BE} V^E$ is the inertial velocity vector in the body system ($T_{BE} = T_{EB}^T$), $g^B = T_{BE} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$ is the gravity vector in the body system, f_x, f_y, f_z are the body forces acting on the hexacopter and N, E, D are the NED positions.

As on a multicopter, the torques are mostly caused by the rotor forces acting on the body first; the force models should be postulated and then the torque models. For a multicopter, the basic equilibrium state is hovering, and every control action is relative to this, increasing or decreasing the engine thrusts. This fact was considered in the control model structure (see Figure 5) where there is a base engine angular rate vector Ω_0 for hovering, and the controllers generate angular rate changes $\Delta\Omega$ according to it.

The f_z body vertical force model should include the effects of vertical air drag and engine thrust. The air drag can be represented with the second-order model $K_z V_{az} |V_{az}|$ also considering the sign dependence. Here, V_{az} is the vertical airspeed component (see (9)) and K_z is the unknown drag coefficient. The vertical engine forces were represented applying the usual thrust coefficient-based model of propellers (see, e.g., [16,18,30]) and considering that the propeller thrust can change with the motion of the propeller, especially in ascension and descension [31]. Thrust coefficient diagrams in the article show that the coefficient depends linearly on the advance ratio, increases in descension (backward motion of propeller in the air) and decreases in ascension (forward motion of propeller in the air). Thus, the hover thrust model for one motor can be well described as:

$$T_{z_i} = -\left(c_{T0} + \frac{K_c}{100}\mu_i\right)\rho A R^2 \Omega_i^2 \tag{6}$$

Here, ρ = 1.225 kg/m^3 is air density at sea level according to International Standard Atmosphere (ISA), A = 0.2231 m^2 is propeller area and R = 0.2665 m is propeller radius. Substituting the advance ratio $\mu_i = \frac{V_{az}}{R\Omega_i}$ results in (7). Note that here, the angular rate of one engine results from hover and control rates as: $\Omega_i = \Omega_{0_i} + \Delta\Omega_{T_i} + \Delta\Omega_{N_i} + \Delta\Omega_{alt_i} + \Delta\Omega_{yaw_i}$

$$T_{z_i} = -c_{T0}\rho A R^2 \Omega_i^2 - \frac{K_c}{100} V_{az} \rho A R \Omega_i \tag{7}$$

Finally, summing up all the engine and air drag effects, the vertical force model is the following:

$$\begin{aligned} f_z &= -c_{T0}\rho A R^2 \sum_i \Omega_i^2 - \frac{K_c}{100} V_{az} \rho A R \sum_i \Omega_i - K_z V_{az} |V_{az}| \\ i &\in \left[LB, LF, LS, RB, RF, RS\right] \end{aligned} \tag{8}$$

$$\begin{aligned} V_a^E &= V^E - W^E \\ V_a^B &= T_{BE} V_a^E = \begin{bmatrix} V_{ax} \\ V_{ay} \\ V_{az} \end{bmatrix} \end{aligned} \tag{9}$$

The horizontal body forces were modeled with hub force and air drag forces as:

$$\begin{aligned} f_x &= -\frac{K_{hx}}{100}\rho A R V_{ax} \sum_i \Omega_i - K_x V_{ax} |V_{ax}| \\ f_y &= -\frac{K_{hy}}{100}\rho A R V_{ay} \sum_i \Omega_i - K_y V_{ay} |V_{ay}| \end{aligned} \tag{10}$$

The applied hub force model $-\frac{K_{hx}}{100}\rho A R V_{ax} \sum_i \Omega_i$ was derived with the simplification of the blade element-momentum theory models from, e.g., [15,16]. This simplified model is explicitly referenced in [21] (page 78 Remark 4.1), but here, time-varying Ω_i engine speeds were considered. The time dependence of the variables is not denoted for the sake of simplicity. Note that all force models include the effect of the unknown wind disturbance W^E through the airspeed terms, so the wind should also be estimated.

After postulating the force models, the torque models were constructed. In the horizontal rotational dynamic model of the rotor inertial forces, the air drag and the momentums resulting from the thrust forces of the six propellers were considered.

Considering [32], Figure 8 and the thrust model of the engines (7), the following model equations were made. Note that the air drag from the vertical airspeed V_{az} was assumed to have zero torque effect on the body; only the air drag from rotational motion was considered.

$$\begin{aligned} \tau_x =& -qJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\ & c_{T0}\rho A R^2 \frac{l}{2}(\Omega_{LB}^2 + \Omega_{LF}^2 + 2\Omega_{LS}^2 - \Omega_{RB}^2 - \Omega_{RF}^2 - 2\Omega_{RS}^2) + \\ & \frac{K_c(V_{az})}{100} V_{az}\rho A R \frac{l}{2}(\Omega_{LB} + \Omega_{LF} + 2\Omega_{LS} - \Omega_{RB} - \Omega_{RF} - 2\Omega_{RS}) - K_p p|p| \\ \tau_y =& pJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\ & c_{T0}\rho A R^2 \frac{\sqrt{3}l}{2}(-\Omega_{LB}^2 + \Omega_{LF}^2 - \Omega_{RB}^2 + \Omega_{RF}^2) + \\ & \frac{K_c(V_{az})}{100} V_{az}\rho A R \frac{\sqrt{3}l}{2}(-\Omega_{LB} + \Omega_{LF} - \Omega_{RB} + \Omega_{RF}) - K_q q|q| \end{aligned} \tag{11}$$

Here, $l = 0.60285$ m is the physical length of the engine arms from the CG and $J_r = 7 \times 10^{-4}$ kg·m^2 is the inertia of the BLDC (brushless DC) engine rotor and propeller together obtained from [33].

For the vertical torque, the torque of propellers around the vertical (rotational) axes was derived and considered as a main effect together with the air drag from rotational speed: $-K_r r|r|$. In [31], the power coefficient of the motors is also almost linearly dependent on the advance ratio, leading to the following vertical torque model.

$$\begin{aligned} P_i &= c_P \rho A R^3 \Omega_i^3 = M_{z_i} \Omega_i \\ M_{z_i} &= c_P \rho A R^3 \Omega_i^2 \\ c_P &= c_{P0} + K_p \frac{V_{az}}{R\Omega_i} \\ M_{z_i} &= c_{P0} \rho A R^3 \Omega_i^2 + K_p \rho A V_{az} R^2 \Omega_i \end{aligned} \tag{12}$$

Here, P_i is the motor power, M_{z_i} is the motor vertical torque and c_P is the power coefficient. Considering the rotor rotational directions from Figure 8 and the air drag, the postulated yaw dynamic model was:

$$\begin{aligned} &\tau_z = \\ &\frac{c_{P0}}{100} \rho A R^3 (-\Omega_{LB}^2 - \Omega_{LF}^2 + \Omega_{LS}^2 + \Omega_{RB}^2 + \Omega_{RF}^2 - \Omega_{RS}^2) + \\ &K_p \rho A V_{az} R^2 (-\Omega_{LB} - \Omega_{LF} + \Omega_{LS} + \Omega_{RB} + \Omega_{RF} - \Omega_{RS}) - K_r r|r| \end{aligned} \tag{13}$$

Here, it is assumed that while hovering, the sum of engine speed terms is zero:

$$\begin{aligned} -\Omega_{LB0}^2 - \Omega_{LF0}^2 + \Omega_{LS0}^2 + \Omega_{RB0}^2 + \Omega_{RF0}^2 - \Omega_{RS0}^2 = 0 \\ -\Omega_{LB0} - \Omega_{LF0} + \Omega_{LS0} + \Omega_{RB0} + \Omega_{RF0} - \Omega_{RS0} = 0 \end{aligned} \tag{14}$$

3.3. Details of Control Loop Models

Table 1 shows that there are three control logics (horizontal, vertical and yaw) which should be realized in four control loops (tangential, normal, altitude and yaw) as the horizontal control consists of tangential and normal components, as shown in Figure 5. The controller models were postulated based on conventional control system design knowledge including PID control and anti-windup to have a simple control structure (as presented in [29] for UAVs).

As conventional position control is not present on DJI M600 (see Section 4.1.1), only tangential and normal velocity tracking control was identified considering the following signal flow: velocity reference with saturation → velocity error → pitch/roll reference with anti-windup (AW) → pitch/roll error → engine speeds (assuming engine dynamics included in the controller). The tangential controller scheme is shown in Figure 9. The normal velocity tracking is the same with V_N and ϕ signals instead of V_T and θ. The ΔV signal is required to make an internal gain switching inside the PI+AW block; this will be detailed in Section 8. The velocity error to the angle reference part was postulated as a PI (proportional–integral) controller as the integral term is required to compensate wind in case of zero velocity reference. AW was required because angle references have to be saturated at ±25° according to DJI specification [26]. Note that the $d(el)$ delay term was only inserted during model verification and refinement in Section 8.

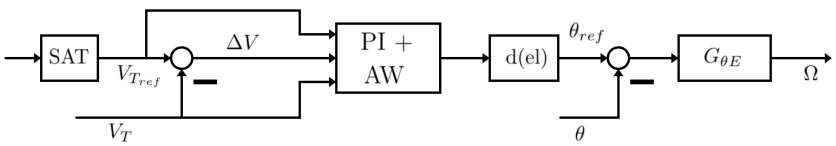

Figure 9. Scheme of tangential velocity controller.

The postulated altitude controller structure is presented in Figure 10. The assumed signal flow in the controller is: altitude reference (h_{ref}) → altitude error (with D down position) → saturated vertical speed reference (gain + saturation) → vertical speed error → engine speeds (assuming engine dynamics included in the controller). Note that the $d(el)$ delay term was only inserted during model verification and refinement in Section 8.

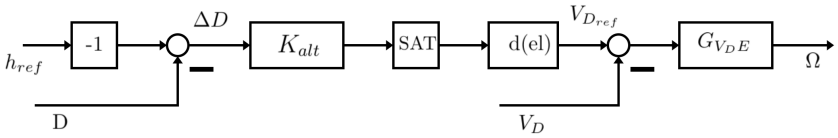

Figure 10. Scheme of altitude controller.

The assumed structure of the yaw angle controller was yaw angle error → yaw rate reference with saturation → yaw rate error → engine speeds (assuming engine dynamics included in the controller). The controller scheme is shown in Figure 11. Note that the $d(el)$ delay terms were considered both in identification and during model verification and refinement in Section 8.

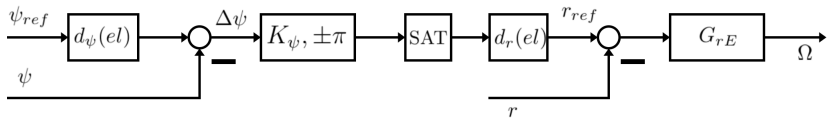

Figure 11. Scheme of yaw angle controller.

After postulating the model structure, flight data collection for system identification was completed.

4. Flight Data Collection and Basic Model Analysis

Three flight test campaigns were conducted for the collection of persistently excited relevant flight data, each time improving the test procedures based on the experiences of the previous tests.

On 5 November 2021, back and forth and ascend/descend maneuvers were flown in high wind conditions, as listed below. The back and forth flights were aligned with the wind direction to have larger and smaller airspeeds (groundspeed was the same) and make it easier to distinguish the wind effect from other dynamics. Stop and turn control mode was applied to have yaw excitation in the hovering stat and thus proper data for yaw dynamic and control identification. Ascend/descend were performed together with position hold to have proper excitation of the vertical dynamics and control. The trajectories were generated with UgCS software. In all of the cases, the wind direction was approximately estimated from direction measurement with a flag. From now on, this test day is referenced as $FT1$ (Flight Test 1).

1. $FT1/1$ Fly against wind and with wind between two waypoints about 100 m away from each other with groundspeed limited to 5 m/s.
2. $FT1/2$ Fly against wind and with wind between two waypoints about 100 m away from each other with groundspeed limited to 10 m/s.
3. $FT1/3$ Fly against wind and with wind between two waypoints about 180 m away from each other with groundspeed limited to 14 m/s.
4. $FT1/4$ Hold position and make about 60 m ascend and descend maneuvers limiting vertical speed to ± 3 m/s.

The 180 m long back and forth flight trajectory is shown in Figure 12.

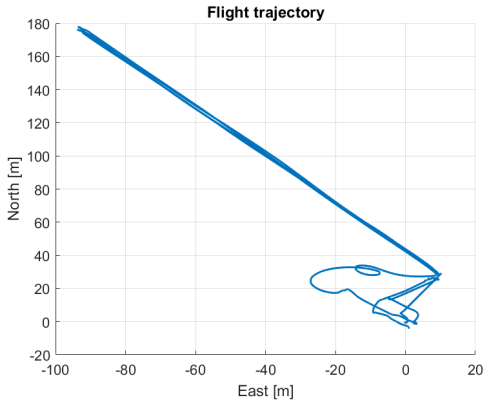

Figure 12. Straight back and forth flight trajectories.

Based on the experiences of the previous flight campaign, new flight tests were performed on 14 April 2022 applying the stop-and-turn mode of the autopilot to fly between given waypoints defined through the UgCS software as follows (called $FT2$). The change was to fly triangle patterns to have wind disturbance acting from multiple directions to excite the cross-dynamics of the hexacopter. Otherwise, again, the vertical dynamics (ascend/descend) and the yaw dynamics were excited to identify all dynamics and OSDK logic (see Table 1) possible.

1. $FT2/1$ Hold position and make about 60 m ascend and descend maneuvers limiting vertical speed to ± 3 m/s.
2. $FT2/2$ Fly a triangular pattern back and forth with 100 m side length and groundspeed limited to 5 m/s.
3. $FT2/3$ Fly a triangular pattern back and forth with 100 m side length and groundspeed limited to 10 m/s.
4. $FT2/4$ Fly a triangular pattern back and forth with 180 m side length and groundspeed limited to 14 m/s.

Both the 100 m and 180 m side triangle trajectories are shown in Figure 13.

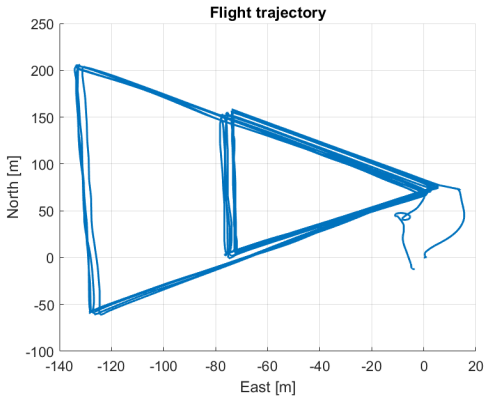

Figure 13. Triangle flight trajectories.

Finally, a third flight test campaign was flown on 12th August 2022 as $FT3$ to collect data from the basic OSDK operational modes listed in Table 1 (except for HORIZONTAL_ANGULAR_RATE and VERTICAL_THRUST cases as stated before). All of the maneuvers were tailored to utilize and excite one specific mode of the OSDK logic. In these flights, the onboard system commanded the drone through DJI OSDK (see [14]) with the proper mode switching flag and references. DJI provides a simulator for DJI M600 through DJI Assistant 2, and before flight testing, the designed special maneuvers were verified in this simulator. During this verification, it turned out that for a step position reference command, the drone flies away (instead of moving with the given distance), which is why

in the flight testing, 'position kick' commands were applied. For details of the specialties of position mode, see Section 4.1.1.

1. $F3/1$ Kick command in X and Y position (see the related special mode in Section 4.1.1).
2. $F3/2$ Normal V_N and tangential V_T velocity doublet tracking with $3,\ 5,\ 10$ m/s amplitudes.
3. $F3/3$ Vertical (down V_D) velocity doublet tracking with $3,\ 5$ m/s amplitudes.
4. $F3/4$ Yaw angle ψ doublet tracking with $45°, 90°$ amplitudes.
5. $F3/5$ Roll ϕ and pitch θ doublet tracking with $10°$ amplitude.

Normal and tangential velocity refer to the forward and side ground relative velocities in a body system (see Figure 6). Data from this last flight test campaign were utilized in the identification of special behaviors of the DJI controller, the verification of the DJI simulator of M600 in DJI Assistant 2 and the refinement and final validation of the M600 simulation model.

4.1. Discovered Special Cases

Pre-analysis of control actions in the DJI simulator before flight testing showed two special cases where unexpected behavior was experienced. Unexpected is considered from a control engineering point of view. These are the horizontal position and angle-tracking modes. These special behaviors were underlined by the flight test experiences, and this section compares the simulation and flight test results of these modes.

4.1.1. Horizontal Position Reference Tracking

A step position reference did not result in the change of position with the given distance as assumed (e.g., 10 m move to the north); rather, the drone had continuous motion with constant velocity until stopping the controller. Considering the fact that the step change of any other reference caused the tracking of that reference, this was surprising (see, e.g., Figure 16), but such behavior is also discussed in the DJI description [34]. So, in horizontal position mode, the DJI M600 did not track the given position signal; rather, it moved away (this is underlined by the position response to a position kick maneuver in Figure 14).

Consequently, position kick commands were sent to the system observing the position, velocity and angular responses both in DJI simulation and real flight, as presented in Figures 14 and 15. The figures show that for a 5 m (side) position kick, there was about 2–2.5 m displacement of the drone with a speed up and slow down maneuver.

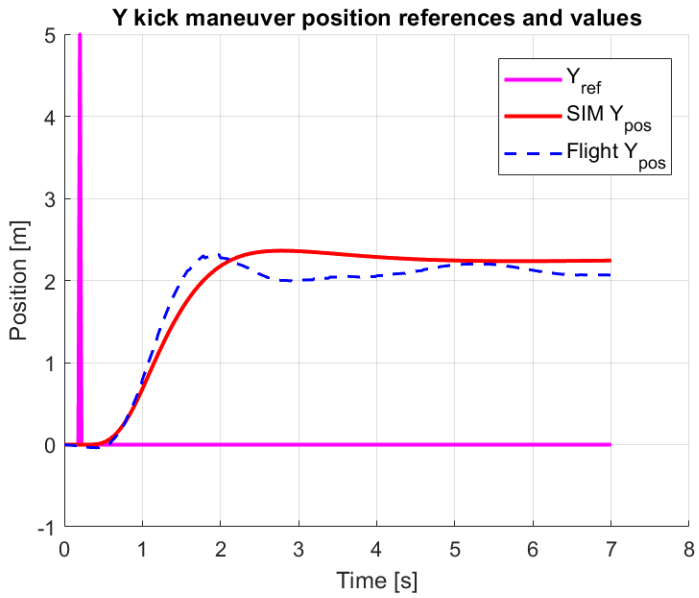

Figure 14. Effect of position kick command on position.

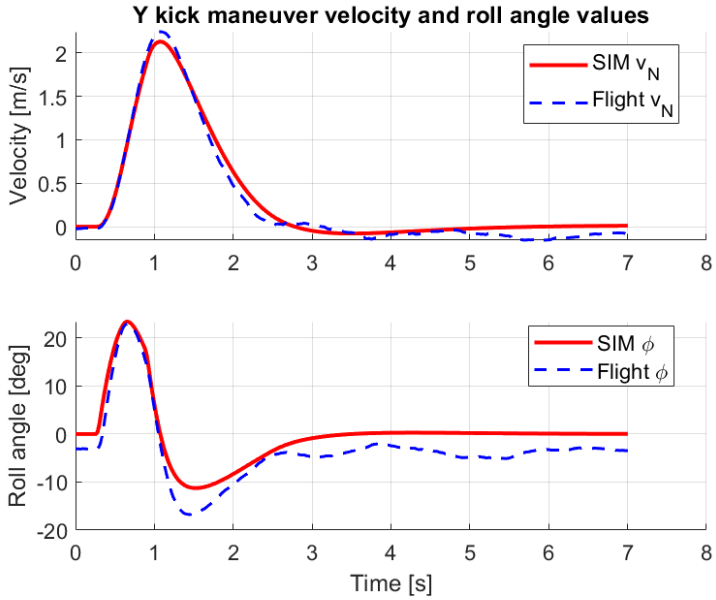

Figure 15. Effect of position kick command on velocity and angle.

4.1.2. Horizontal Angle Reference Tracking

Another specialty is the handling of angle references, as a step up and back to zero angle reference caused different behavior at the changes. The step to nonzero value caused good tracking of the nonzero signal, but upon jumping to the zero value, a braking mode was activated, which caused high overshoot of the angle because it started to track a zero velocity reference with high angle values stopping the vehicle as fast as possible. So, the zero angle reference is rather implemented as a zero velocity one, stopping the vehicle (see Figures 16 and 17 presenting both DJI simulation and real flight behavior).

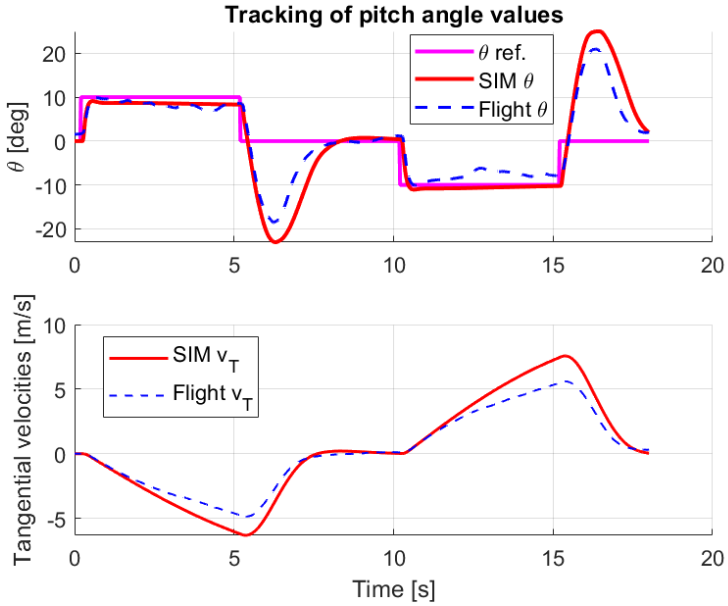

Figure 16. Pitch doublet command response.

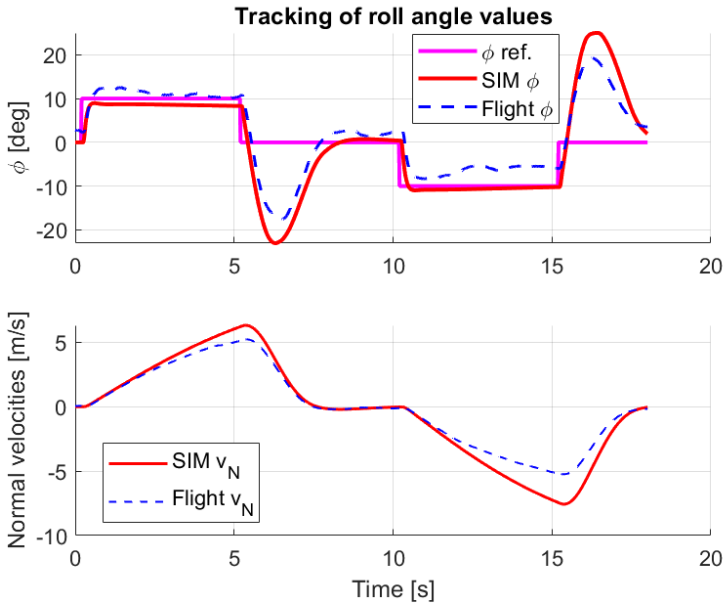

Figure 17. Roll doublet command response.

4.2. Verification of DJI Simulator

The *FT*3 flight test maneuvers, constructed to analyze the behavior of factory OSDK control loops, were also utilized to compare flight data and DJI Simulator (referred to as software-in-the-loop (SIL) later) performance and thus verify the factory simulator of DJI M600. Details of this verification are presented in [35]. The conclusion was that the DJI simulator is appropriate for OSDK-based control development and testing. However, there are certain drawbacks of this simulation, like the need to connect the drone to the simulation computer and the need to implement the control methods onboard the drone before evaluation. The high-fidelity Matlab Simulink simulator targeted in this work makes development and testing much easier, removing the needs to connect the physical drone and to implement the control code on the onboard computer before the first test runs. This speeds up control development and application, as only the simulation-tested and -verified controllers should be implemented and built for DJI simulator and flight testing.

The first step of high-fidelity system identification was the estimation of mass and inertial data, which is presented in the next section.

5. Estimation of Mass and Inertial Data

It is relatively easy to measure the mass of a drone, but the inertial data estimation is more complicated. One possible method is to measure it swinging the drone as a pendulum [19]; another possibility is to measure the size and mass of the parts and create a 3D model with an appropriate software computing the inertias by the software itself. As equipment for pendulum tests was not available and it was relatively easy to disassemble the drone, the latter method was chosen.

The complete disassembly of the drone was not necessary; only the mass of the main parts was measured, and their dimensions were approximated. The Onshape free online 3D modeling software [36] can determine inertia from a given 3D model, so that was used for the calculation. The approximate 3D model of the stock M600 without payload is visible in Figure 18 in flight configuration with retracted landing gears.

Figure 18 shows that the drone was decomposed into six main types of parts. The description of these parts with the measured weights is summarized in Table 2. Measurement of the folding arms and motors required the removal and disassembly of one arm. These data were then used to approximate the weight of the drone body. Measuring the battery and landing gear elements was considerably easier, since they are originally removable parts.

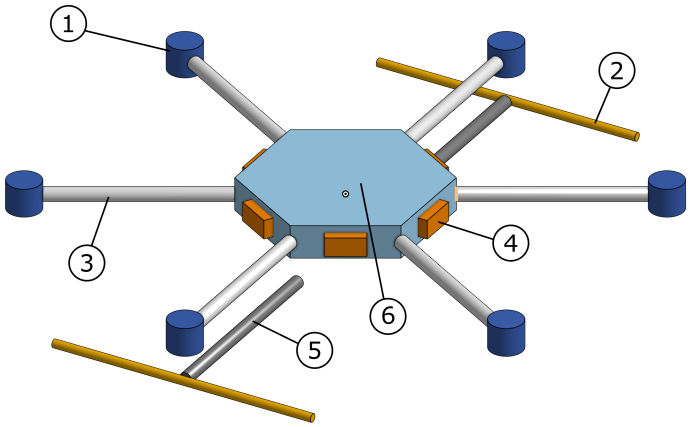

Figure 18. Stock DJI M600 3D model for inertia calculation.

Table 2. Mass of DJI M600 components for inertia calculations.

ID	Name	Mass (g)	Element Count
1	Motor	382	6
2	Landing gear	105	2
3	Folding arm	46	6
4	Battery	590/675	6
5	Landing gear arm	61	2
6	Body	2624	1

The drone can be equipped with two types of batteries. The two options are TB47S and TB48S; the latter has larger capacity. Hence, in Table 2, both weights are stated (the larger weight belongs to TB48S). Therefore, two masses and two inertia matrices can be calculated even for the stock M600.

Figures 19 and 20 show the 3D model of the M600 with the forerunner payload attached (for payload details, see Figures 2 and 3). The payload components are a bit more detailed than the M600 itself, but some lightweight elements, like the carbon-fiber parts of the gimbal, were omitted from the model, as the two electric motors dominate the gimbal weight and inertia. There are blank spaces visible in both 3D models, which are of course not present in real life. The mass of these elements was neglected or incorporated into the other elements. But since inertia calculations depend on the position of the individual elements, each part has to be placed in the exact position where they actually are.

The payload elements and their data are listed in Table 3. The basic elements of the stock drone are not repeated, but they were included in the inertia calculations of the payload version as well.

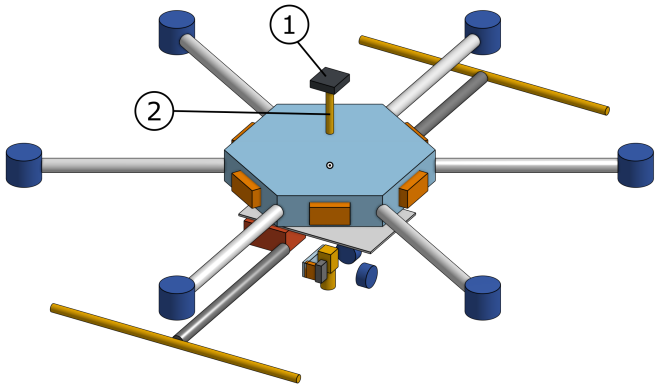

Figure 19. Payload DJI M600 3D model for inertia calculation: top view with additional GPS antenna.

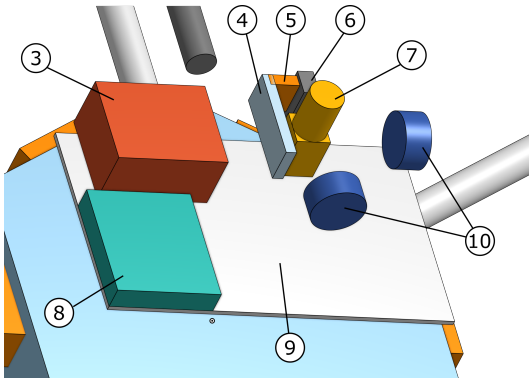

Figure 20. Payload DJI M600 3D model for inertia calculation: bottom view with onboard computer, GPS module, gimbal and camera.

Table 3. Mass of payload components for inertia calculations.

ID	Name	Mass (g)	Element Count
1	GNIMU antenna	112	1
2	GNIMU antenna stand	53	1
3	GNIMU	191	1
4	Gimbal part 1	13.3	1
5	Gimbal part 2	24	1
6	Gimbal part 3 (counter weight)	137	1
7	Basler camera	155.8	1
8	Nvidia Jetson Xavier NX	216	1
9	Base plate	325	1
10	Gimbal motors	199	2

Finally, the results of the mass and inertia calculations carried out by Onshape in all configurations are presented in Table 4. Note that off-diagonal values appeared in the inertia matrix because of the payload, but they are negligible compared to the diagonal values. This underlines that the diagonal inertia matrix was a valid assumption in Section 3.

Table 4. Results of mass and inertia calculations.

Configuration	Mass (g)	Inertia Matrix (g·cm^2)
Stock M600 with TB47S batteries	9530	$\begin{bmatrix} 6.894 \times 10^6 & 0 & 0 \\ 0 & 5.971 \times 10^6 & 0 \\ 0 & 0 & 1.275 \times 10^7 \end{bmatrix}$
Stock M600 with TB48S batteries	10,040	$\begin{bmatrix} 6.949 \times 10^6 & 0 & 0 \\ 0 & 6.026 \times 10^6 & 0 \\ 0 & 0 & 1.286 \times 10^7 \end{bmatrix}$
Payload M600 with TB47S batteries	11,155	$\begin{bmatrix} 7.398 \times 10^6 & -1375.236 & -8604.432 \\ -1375.236 & 6.524 \times 10^6 & 1225.41 \\ -8604.432 & 1225.41 & 1.287 \times 10^7 \end{bmatrix}$
Payload M600 with TB48S batteries	11,665	$\begin{bmatrix} 7.456 \times 10^6 & -1376.109 & -8849.862 \\ -1376.109 & 6.582 \times 10^6 & 1216.184 \\ -8849.862 & 1216.184 & 1.298 \times 10^7 \end{bmatrix}$

6. Dynamic System Model Identification

Before starting the identification, the maximum motor speed (equal to propeller speed in case of direct drive BLDC motors) was estimated based on the figures in [33] (page 9) and real flight data. It was set to $\Omega_{max} = 456.45$ rad/s.

First the force and then the torque model parameters were identified. As the vertical force depends on the airspeed, and thus the wind disturbance first, the horizontal force model parameters were identified including the north and east horizontal wind components and assuming $W_D = 0$ as the deterministic wind disturbance is dominantly horizontal. It was also assumed that the strength and direction of horizontal wind disturbance was constant during each flight test. Considering the 15–20 min maximum flight time, this can be approximately true.

6.1. Horizontal Force Model Identification

In case of the *FT*1 flight campaign, there was high wind but no crosswind excitation of the drone (flight against or with wind), while in case of *FT*2, there was crosswind excitation but low wind. Considering that in the postulated force model (10) both terms depend on the airspeed in low wind, it is hard to distinguish them, while in high wind, the air drag becomes dominant (considering $6\rho AR = 0.437$, $\Omega_{max}/100 = 4.56$ and so $6\rho AR\Omega_{max}/100 = 1.992$). Thus, in the high wind condition of *FT*1, the forward air drag parameter can be determined well, and so the cross-value should be derived based on engineering assumptions, while with *FT*2 data, the hub force coefficients can be determined if the air drag values are known and fixed. A repeated *FT*2 flight in high wind could be a good solution, but it is hard to organize all the logistics of a test flight, and it is not guaranteed that an attempt in case of high wind forecast will be successful (too high wind cancels all the flight tests). That is why the collected data were utilized in the best way to obtain realistic parameters.

In the *FT*1 flight test campaign, back-and-forth flights were made, meaning that the horizontal wind components can be aligned with the body system to make parameter identification easier. Flight directions were $\psi_{fwd} = -38^\circ$ and $\psi_{bckwd} = 142^\circ$ and considering positive tangential and normal wind velocities W_T, W_N for the former (can be obtained from NE wind analogously to (1)) results in $-W_T, -W_N$ velocities for the latter doing a 180° rotation. For the positive W_T, W_N values and considering V_T, V_N, V_D body aligned drone speeds, the horizontal airspeed components result were:

$$\begin{aligned} V_{ax} =& V_T\cos(\theta) - V_D\sin(\theta) - W_T\cos(\theta) \\ V_{ay} =& V_T\sin(\phi)\sin(\theta) + V_N\cos(\phi) + \\ & V_D\sin(\phi)\cos(\theta) - W_T\sin(\phi)\sin(\theta) - W_N\cos(\phi) \end{aligned} \tag{15}$$

From the translational dynamics (4), the horizontal force Equation (10) can be related to the measured body acceleration (a_x, a_y) of the vehicle as:

$$\begin{aligned} m\cdot a_x &= -\frac{K_{hx}}{100}\rho ARV_{ax}\sum_i \Omega_i - K_x V_{ax}|V_{ax}| \\ m\cdot a_y &= -\frac{K_{hy}}{100}\rho ARV_{ay}\sum_i \Omega_i - K_y V_{ay}|V_{ay}| \end{aligned} \tag{16}$$

From (16), the unknowns are $K_{hx}, K_{hy}, K_x, K_y, W_T$, and W_N, while the other variables are measured by the on-board system of the M600. For the estimation of parameters, two data sets were selected (*FT*1/1 and *FT*1/2), each set including hover, climb, against (FAW = forward against wind) and with (FWW = forward with wind) wind (both 5 m/s and 10 m/s ground speed) maneuvers for persistent excitation of the horizontal forces. The limits of air drag coefficients K_x, K_y were determined from the maximum wind tol-

erance 8 m/s and maximum bank angle 25° of the M600 considering maximum wind hovering flight (see technical specifications [26]):

$$m \cdot g \sin(25/180\pi) = K_{max} 8^2 \;\rightarrow\; K_{max} = 0.65$$

Note that in FAW flight, the air velocity can be higher and so the K_x limit is lower: that is why this is a realistic upper limit. The limits for K_{hx}, K_{hy} were assumed to be 3 and the tangential and normal wind limits were assumed to be 10 m/s (a bit above the maximum 8 m/s to allow free motion of the parameters for convergence). All of the parameter minimums were 0 as the parameters cannot change signs (invalid model). As (16) is nonlinear in the winds, the Matlab nonlinear least-squares solver *lsqcurvefit* was applied considering flight data Sets *FT*1/1 and *FT*1/2. Having all six parameters free, the results are as summarized in Table 5.

Table 5. Results of horizontal forces identification from *FT*1 flight campaign.

Parameter	K_{hx}	K_{hy}	K_x	K_y	W_T	W_N	$\tilde{K}_y$
Set *FT*1/1	0.7194	0.3096	0.0563	0.031	4.4635	6.609	0.0488
Set *FT*1/2	0.7028	0.289	0.0539	0	4.6476	10	0.0467
Average	0.7111	0.2993	0.0551	0.0155	4.555	8.3	0.04775

Table 5 shows that the longitudinal parameters K_{hx}, K_x, W_T are consistent between the two sets, while the lateral parameters K_{hy}, K_y, W_N are rather inconsistent, including a saturation of W_N at the maximum unrealistic 10 m/s upper limit. This underlines the statement that the cross-direction was not persistently excited.

However, as in the *FT*2 campaign, the air drag was not dominant, an approximation of K_y was obtained considering the geometric parameters of the M600. As the air drag is linearly proportional to the front area of the vehicle, which from the Y direction is $\sqrt{3}/2$ times smaller than from the X (see again Figure 8) $\tilde{K}_y \approx 0.866 K_x$ was applied (see Table 5).

After determining K_x and K_y from *FT*1, they were fixed and *FT*2 flight data (without air drag dominance due to the low wind) were utilized to obtain the hub force coefficients K_{hx}, K_{hy}. To cover a large range of maneuvers and thus provide persistent excitation, one complete 5 m/s speed triangle was combined with one 10 m/s triangle flown in the reverse direction (one CW and one CCW), forming four different data sets. The model structure was the same as postulated in Equations (15) and (16) completed with (17), showing the transformation from NE to TN wind components as from the triangle flights. Only the NE components can be determined.

$$\begin{aligned} W_T &= W_N \cos(\psi) + W_E \sin(\psi) \\ W_N &= -W_N \sin(\psi) + W_E \cos(\psi) \end{aligned} \tag{17}$$

Finally, making K_x, K_y fix from the previous model gave satisfactory results with only a few outliers as Table 6 shows. Outliers were defined as values more than 30% larger or smaller than the others. They are caused by turbulence and windgust disturbances in the real flight data.

Table 6. Results of horizontal forces identification from the second flight campaign (*FT*2), first run. Outliers are denoted with bold face.

Parameter	K_{hx}	K_{hy}	W_N	W_E
Set *FT*2/1	1.1045	0.8265	**2.4815**	–1.376
Set *FT*2/2	1.0527	0.9276	1.745	–1.1056
Set *FT*2/3	1.036	0.7077	1.613	**–0.67**
Set *FT*2/4	1.132	**0.66**	1.7222	–1.2163
Average	1.0813	0.8206	1.6934	–1.2326

In the next step, the resulting average K_{hx}, K_{hy} values were fixed, and the other parameters were fit to check model's validity (K_x, K_y should result the same as the fixed values from $FT1$). The results (again with outliers) are shown in Table 7.

Table 7. Results of horizontal forces identification from the second flight campaign ($FT2$), second run. Outliers are denoted with boldface.

Parameter	K_x	K_y	W_N	W_E
Set $FT2/1$	0.1028	0.0968	**2.57**	–1.41
Set $FT2/2$	0.0928	**0.2181**	1.747	–1.12
Set $FT2/3$	0.0987	**0.1582**	1.581	**–0.57**
Set $FT2/4$	0.1072	0.0934	1.667	–1.28
Average	0.1	0.0951	1.665	–1.27

The wind values are close to the previous results (also the outliers), while K_x and K_y are far from the previous ranges. As there is no other data set to be applied, the average results from $FT1$ (Table 5) and $FT2$ (Table 7) were finally considered as $K_x = 0.0775$, $K_y = 0.0714$. Fixing them and ensuring a fit for the hub force and wind disturbances gave the final model values in Table 8.

Table 8. Final results of horizontal forces identification from the second flight campaign ($FT2$). Outliers are denoted with boldface.

Parameter	K_{hx}	K_{hy}	W_N	W_E
Set $FT2/1$	0.9613	0.7881	**2.41**	–1.325
Set $FT2/2$	0.908	0.8951	1.668	–1.1098
Set $FT2/3$	0.8966	0.6713	1.574	**–0.623**
Set $FT2/4$	0.9911	**0.6138**	1.6592	–1.2107
Average	0.9392	0.7848	1.6337	–1.2112

Again, the outliers are at the same places. The average K_{hx}, K_{hy} values are smaller than in the first round (see Table 6) but larger than from the previous flight campaign, giving a model balanced between the two cases. The absolute value of average wind result was 2.03 m/s, while its direction was $-36.55°$ (southeast). The Windguru weather and wind forecast (see [37]) for 14 April 2022 is presented in Figure 21 (on-site wind measurements were not calculated due to the lack of equipment). The system identification flights were between 11 and 11:30, and the predicted wind for that time range is 2–3 m/s southeast. The system identification well covers the southeast (a bit more south) direction and is close to the 2–3 m/s with a 2 m/s estimate. So, the estimated model values can be accepted as realistic.

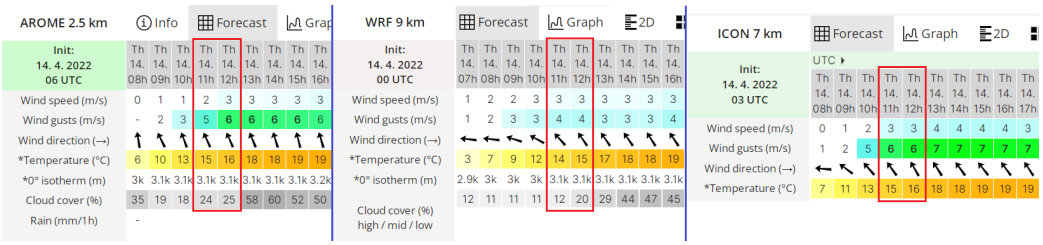

Figure 21. Weather and wind forecast for 14 April 2022 ($FT2$) (printscreen from [37]).

Fixing the hub force and air drag model parameters at the new values makes it possible to re-identify the wind disturbances in the two flights in $FT1$. The results are shown in Table 9. The table shows that there is only one outlier; the other results are close, so the averages can be easily calculated. The results show that the wind disturbances were lower than previously estimated, which is realistic, as the previous estimates were too close to

the maximum (8 m/s) wind tolerance of the M600, while there was no problem even with 14 m/s flight speeds. Unfortunately, there is no saved Windguru forecast for the day of that test.

Table 9. Re-identified wind disturbances in the first flight campaign ($FT1$). Outliers are denoted with boldface.

Set	Subset	W_N	W_E
1–2	1	–4.6152	–0.4295
	2	–4.599	–0.556
	3	–4.332	**–0.1331**
	4	–4.5963	–0.5295
Average	–	–4.5356	–0.505
3	1	–5.2753	–1.262
	2	–5.5744	–0.8514
Average	–	–5.4248	–1.0576

Evaluation of the final 3D simulation model showed that the maximum 18 m/s flight speed of M600 (in still air, see [26]) could not be achieved due to the too high air drag coefficients. So, fine tuning was performed, decreasing the K_x air drag in forward flight until reaching 18 m/s. Finally, $K_x = 0.072$ was a proper value and so K_y was decreased with the same ratio to $K_y = 0.0663$. Fixing K_x, K_y and re-identifying the other parameters for the second flight test campaign ($FT2$) gave the values listed in Table 10. Here, the spread of the values is lower than in the previous case (Table 8), showing that the refinement moved the parameters into the proper direction. Testing again for the reaching of 18 m/s showed that K_{hx} and K_{hy} should be slightly decreased to $K_{hx} = 0.94$ and $K_{hy} = 0.7641$ (the same ratio as for K_{hx}). The wind estimates were not updated, as they were close to the previous ones.

Table 10. New fit of parameters in the second flight campaign ($FT2$) after air drag model refinement. Outliers are denoted with boldface.

Parameter	K_{hx}	K_{hy}	W_N	W_E
Set 1	0.9963	0.7954	**2.4296**	–1.3371
Set 2	0.9433	0.901	1.6879	–1.0998
Set 3	0.9308	0.6786	1.5838	**–0.6343**
Set 4	1.0256	**0.6753**	1.6592	–1.2117
Average	0.974	0.7917	1.649	–1.2162

Figures 22–25 show the final model outputs compared to the flight measured horizontal forces for $FT1$ (hover, climb, FAW 5 m/s, FWW 5 m/s, FAW 10 m/s, FWW 10 m/s) and $FT2$ (5 m/s and 10 m/s triangle flights). Note that system identification was completed considering the specific sections where clear vertical or horizontal movements arise without vertical rotation. These sections were cut out from the continuous data. That is why there are gaps between the data sections in the figures: because the non-specific parts are not compared and thus not plotted.

The figures show that the measured accelerations and thus the dynamic forces are very noisy, and there can be large differences between the measured and the model forces. This is the reason why performance measures (such as (23)) were not applied, as they gave unacceptably large results for any model. However, the model well follows the range and dynamic changes of the measurements, so in light of the high uncertainty in the parameter identification, the model performance is acceptable.

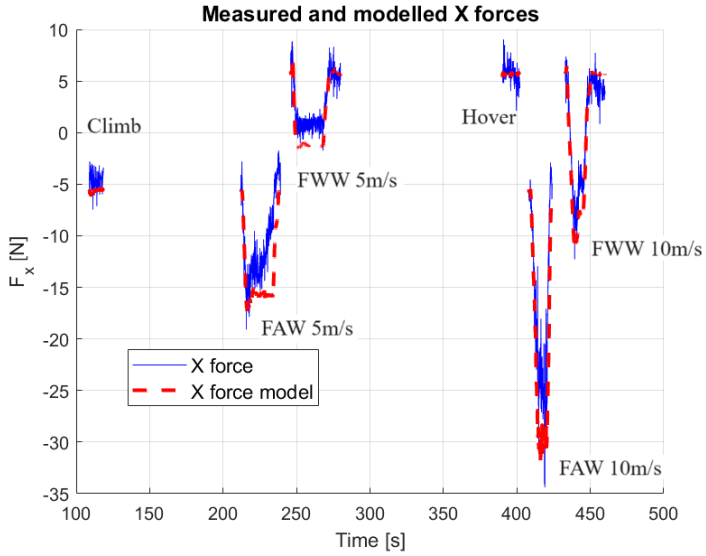

Figure 22. Longitudinal (X) forces in hover, climb, FAW and FWW flights and the outputs of the fitted model in the first flight campaign (*FT*1).

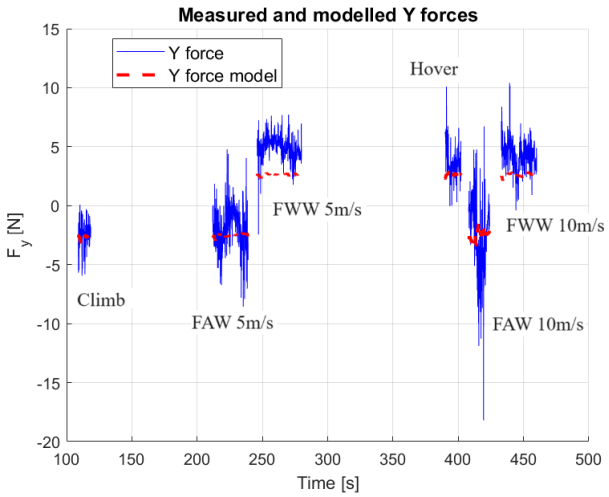

Figure 23. Lateral (Y) forces in hover, climb, FAW and FWW flights and the outputs of the fitted model in the first flight campaign (*FT*1).

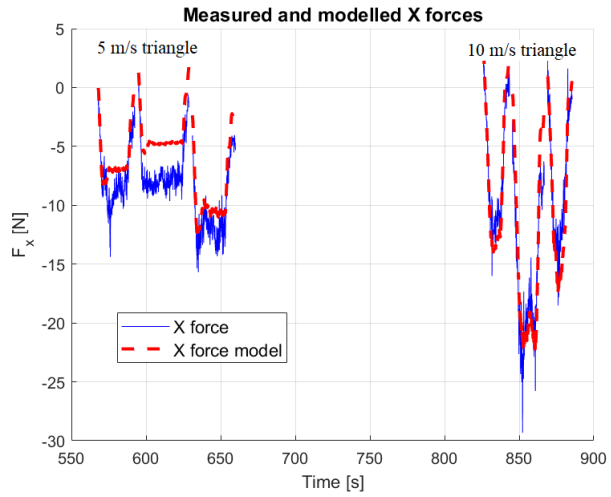

Figure 24. Longitudinal (X) forces in 5 m/s and 10 m/s triangle flights in the second flight campaign (*FT*2).

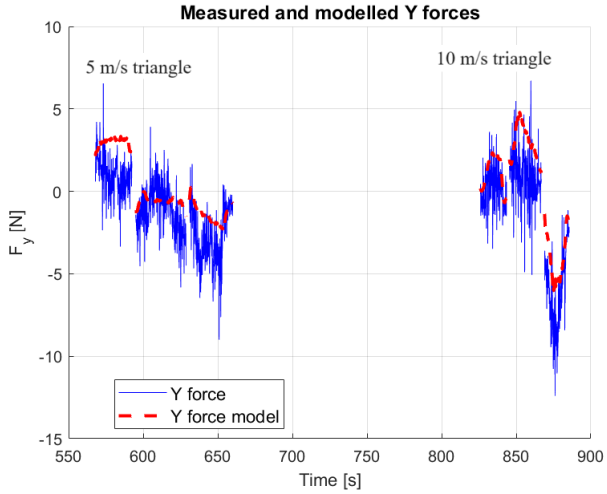

Figure 25. Lateral (Y) forces in 5 m/s and 10 m/s triangle flights in the second flight campaign (*FT*2).

6.2. Vertical Force Model Identification

Considering the vertical force model (8) first, the hover thrust coefficient c_{T0} and the hover engine speed Ω_0 were identified considering flight data sections with $V_a \approx 0$. In hover, the vertical gravitational force should be compensated by the engines:

$$mg\cos\theta\cos\phi = c_{T0}\rho AR^2\sum_i \Omega_i^2 \tag{18}$$

In the identification, $\sum_i \Omega_i^2$ is substituted with $6\overline{\Omega^2}$, which is the average squared engine angular rate.

$$\overline{\Omega^2} = \frac{\Omega_{LB}^2 + \Omega_{LF}^2 + \Omega_{LS}^2 + \Omega_{RB}^2 + \Omega_{RF}^2 + \Omega_{RS}^2}{6} \tag{19}$$

Hovering flight test sections were considered from *FT*1 and *FT*2 flights with TB48S batteries and so $m = 10.04$ kg mass and from *FT*2 flight with TB47S batteries and so $m = 9.53$ kg. c_{T0} was calculated as the mean of the ratio

$$c_{T0} = \frac{mg\cos\theta\cos\phi}{6\rho AR^2\overline{\Omega^2}}$$

for every flight section. Also, $\Omega_0 = \sqrt{\overline{\Omega^2}}$ was calculated as the average (of the averages) hover engine speed. The results are summarized in Tables 11 and 12 for $m = 9.53$ kg and $m = 10.04$ kg, respectively.

Table 11. Hover thrust coefficients and engine speeds for $m = 9.53$ kg.

FLY Nr.	***FT*2/1**	***FT*2/2**	***FT*2/3**	***FT*2/4**
c_{T0} [-]	0.0106	0.0102	0.0105	0.0101
Ω_0 [rad/s]	275.8	280.93	277.08	288.7

Table 12. Hover thrust coefficients and engine speeds for $m = 10.04$ kg.

FLY Nr.	***FT*1/1**	***FT*1/2**	***FT*1/3**	***FT*1/4**	***FT*2/1**	***FT*2/2**	***FT*2/3**	***FT*2/4**	***FT*2/5**
c_{T0} [-]	0.0107	0.0106	0.0107	0.0106	0.0101	0.0103	0.0104	0.0107	0.0106
Ω_0 [rad/s]	280.59	282.54	280.25	282.74	288.7	286.17	284.66	280.57	282.57

The tables show that the results are similar for the two different masses giving only slightly lower hover engine speed for the lower mass. As the final goal in the Forerunner

project was flight with payload finally, the data for the larger mass (Table 12) were averaged to obtain the setpoint:

$$c_{T0} = 0.0105 \;\; \Omega_0 = 283.032 \text{ rad/s}$$

Figures 26 and 27 show the engine speed and the vertical forces in the $FT1/3$ hover case. Figure 26 shows that the engine speeds are not all equal, as there is a need for roll and pitch to compensate for wind disturbance (position hold). However, their values are almost constant. Figure 27 shows the body vertical force from gravity and from the hover engine model, giving values very close to each other (errors between 0.02 and 0.1 N).

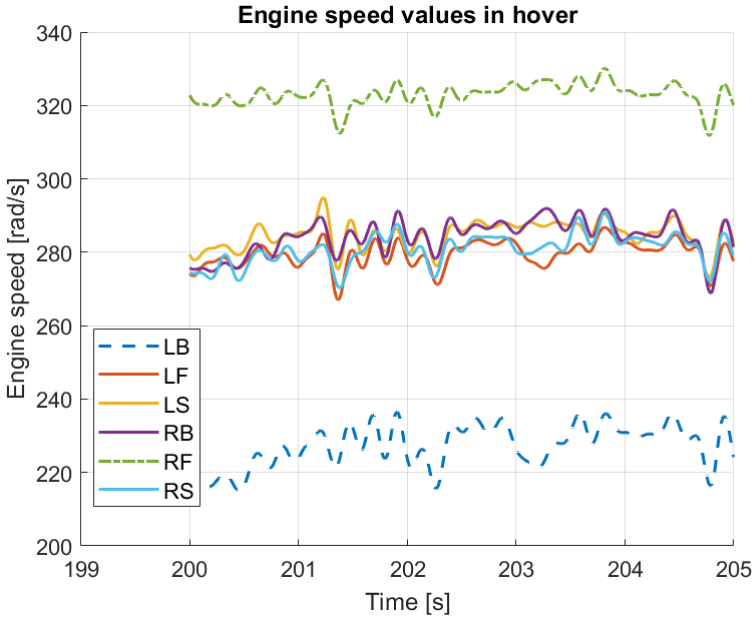

Figure 26. Engine speeds in $FT1/3$ hover case.

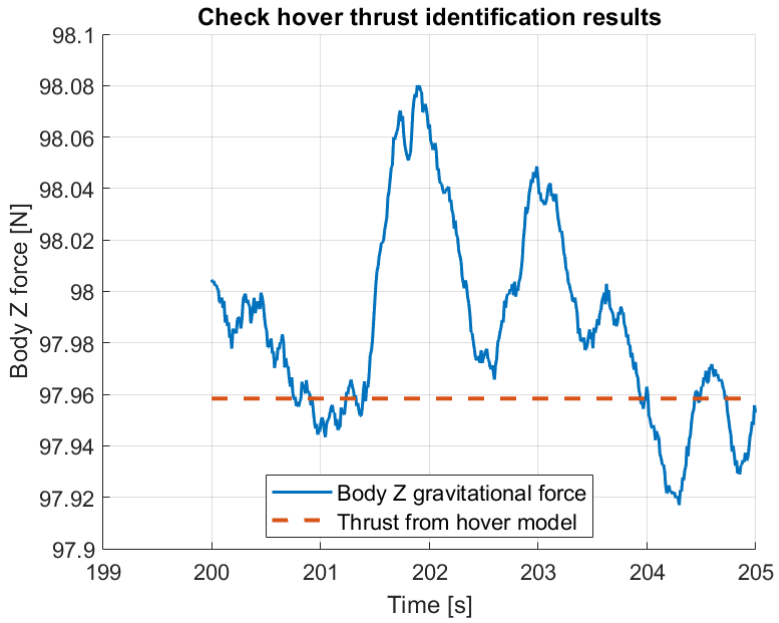

Figure 27. Gravitational and thrust force in hover ($FT1/3$ case).

After identifying the hover thrust coefficient and hover engine speed, the full thrust model was targeted considering $ma_z = f_z$, (8) and that c_{T0} is known.

Trying to fit the model (K_c, K_z) to ascend/descend/FAW flight data showed that a single vertical airspeed-dependent gain $K_c(V_{az})$ is sufficient to cover it; K_z was not required.

This was because (according to [31]) in ascending flight, the rotor thrust coefficient decreases while in descending flight, it increases, and this is a similar effect to the air drag (down in ascension and up in descension). As both effects are airspeed dependent, they cannot be distinguished. So, the force model in (8) was modified by deleting the air drag term and including an airspeed-dependent $K_c(V_{az})$ gain.

$$f_z = -c_{T0} 6\rho A R^2 \overline{\Omega^2} - \frac{K_c(V_{az})}{100} V_{az} 6\rho A R \overline{\Omega} \tag{20}$$

Again, the sums of engine angular rates were replaced by the average values for parameter identification, resulting in:

$$\overline{\Omega} = \frac{\Omega_{LB} + \Omega_{LF} + \Omega_{LS} + \Omega_{RB} + \Omega_{RF} + \Omega_{RS}}{6}$$

In the first flight campaign ($FT1$), the persistently excited data sections (regarding V_{az}) were ascend/descend, FAW 10 m/s and FAW 14 m/s. As the V_{az} airspeed values depend on the wind disturbances, their estimates from Table 9 were applied in the calculation. Separate K_c constant coefficients were determined through linear least-squares fit for the different V_{az} ranges. The estimated coefficients, together with the dominant V_{az} values from the first flight campaign, are shown in Table 13. Estimates from the second flight campaign are presented in Table 14 calculated with wind values from Table 10. The second flight campaign includes ascend and descend test maneuvers with TB47S batteries (smaller mass) and forward flight maneuvers along the triangles with TB48S batteries (larger mass). For this model, some of the K_c results are invalid, giving negative values, so only the positive results were considered in further calculation. The invalid values are at the smallest V_{az} part, so there was not enough excitation of the model.

Table 13. Resulting vertical force coefficients and related V_{az} values from the first flight campaign ($FT1$).

Maneuver	K_c	V_{az} [m/s]
Ascend	1.7419	–3.3
FAW 10 m/s	2.5236	–4
FAW 10 m/s	1.7482	–3.8
FAW 14 m/s	3.4947	–6
FAW 14 m/s	4.5452	–7
Ascend	2.6345	–3.5
Descend	9.4385	2.8
Descend	7.2495	2.7
Ascend	2.1952	–3.4

Table 14. Resulting vertical force coefficients and related V_{az} values from the second flight campaign ($FT2$).

Maneuver	K_c	V_{az} [m/s]
Ascend	3.5995	–3.15
Ascend	4.3	–3.1
Descend	2.895	3
Ascend	3.5535	–3.1
Descend	6.1913	2.9
Along 10 m/s	–2.77	–1.4
Diagonal 10 m/s	–2.8	–2.3
Cross 10 m/s	–0.28	–2
Along 14 m/s	4.22	–6
Cross 14 m/s	2.79	–4

The tabular data are visualized in Figure 28. The figure shows that most of the maneuvers were performed with negative V_{az} (drone motion relative to air), meaning ascension or forward motion with high pitch angles (high velocity). There were two outliers in the data in the first flight campaign: the 7.2495 and 9.4385 values, which were removed from data union (see the figure). The figure shows that there are many more data points for negative V_{az} values than for positive. What is more, only the positive V_{az} value is about 3 m/s (descend). Thus, the characteristic for positive velocity is uncertain. Considering the range of thrust coefficients (1.5–4.5) in the negative V_{az} range suggests that the two Flight 1 points in the positive range are outliers. This is underlined by the finally selected symmetric fit, which has much lower values. First, an asymmetric second-order curve was fit to the data:

$$K_c(V_{az}) = 0.0715V_{az}^2 + 0.3091V_{az} + 3.0277 \tag{21}$$

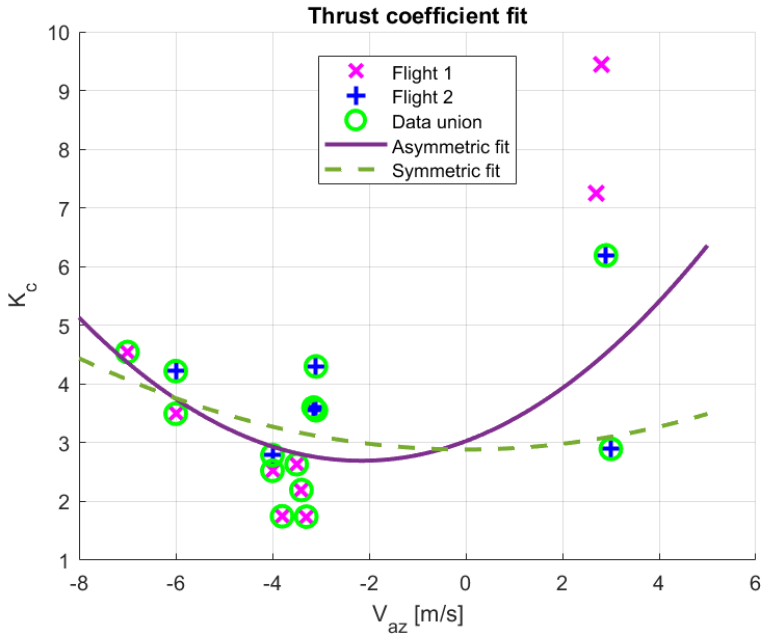

Figure 28. Thrust coefficient fit to all flight data.

However, later testing the vertical control of the model (after identification of the vertical controller, see Section 8.1) showed that with this asymmetric model, the output does not fit the flight measured data. This is illustrated by Figure 29, showing the differences (slower settling) in V_D vertical speed tracking with the asymmetric model (compare it to Figure 52). That is why a symmetric model was fit considering only the negative V_{az} range, as there were more measurement points:

$$K_c(V_{az}) = 0.0243V_{az}^2 + 2.8851 \tag{22}$$

Figures 30–32 show the flight measured $\Delta f_z = ma_z + c_{T0}6\rho AR^2\overline{\Omega^2}$ forces and the outputs of the $-\frac{K_c(V_{az})}{100}V_{az}6\rho AR\overline{\Omega}$ model for 3 m/s ascend and descend maneuvers. Figure 30 for a 3 m/s ascend shows that the symmetric model output is closer to the measured values than the asymmetric. Figure 31 for a 3 m/s descend maneuver shows that neither of the models cover the measured data well.

However, for another 3 m/s descend in the same flight test campaign, Figure 32 shows that the models cover the data, the symmetric model having a smaller average error. So for the case in Figure 31, possibly some windgust disturbance occurred, which is not considered in the model.

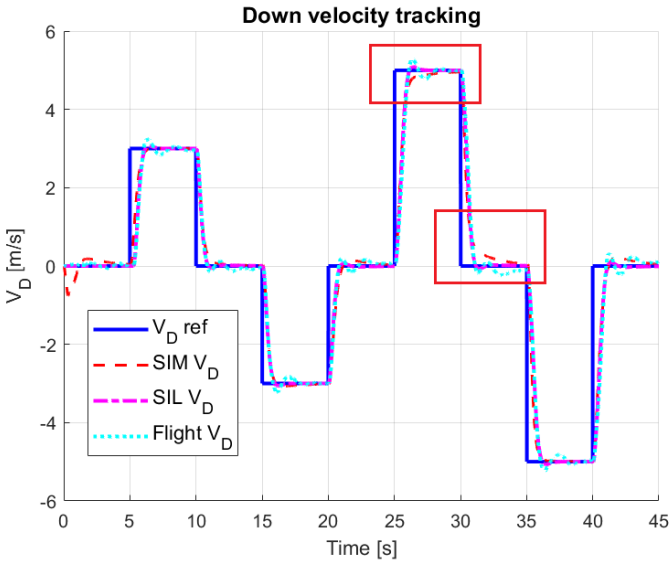

Figure 29. Slower settling of vertical velocity tracking with asymmetric thrust coefficient model.

Figure 30. Vertical force model outputs in ascend mode.

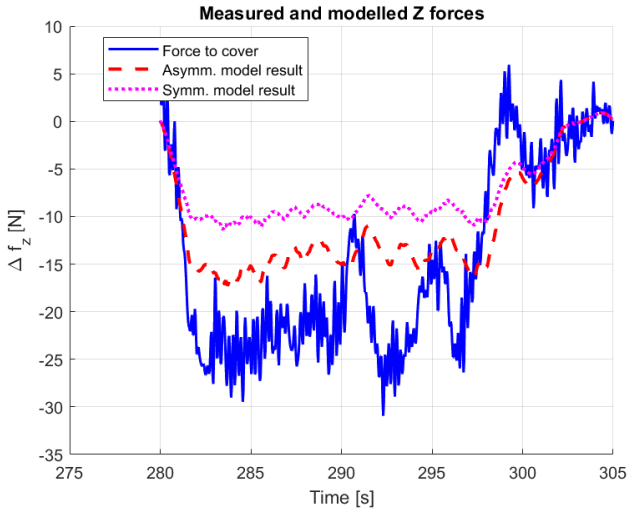

Figure 31. Vertical force model outputs in descend mode.

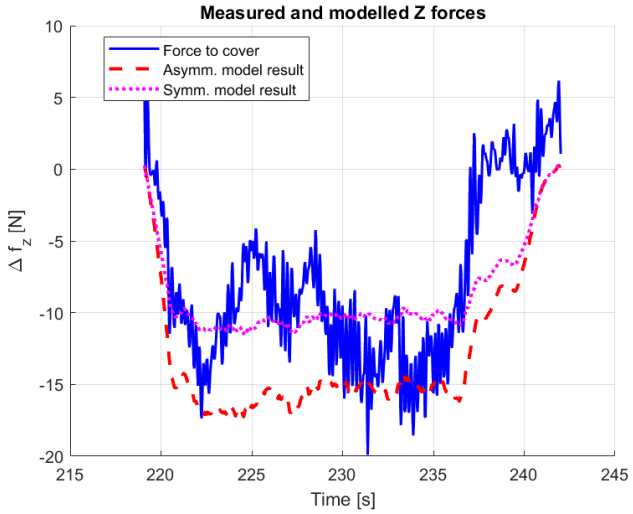

Figure 32. Vertical force model outputs in another descend mode.

As a quality measure, the time-averaged absolute percentage model error was applied (23). The measures for the three figures are summarized in Table 15.

$$e = \sum_i \left(\frac{|y_{meas_i} - y_{model_i}|}{|y_{meas_i}|} 100 \right) \tag{23}$$

Table 15. Quality measures for asymmetric and symmetric vertical force models.

Figure	Figure 30	Figure 31	Figure 32
Asymmetric	833.5	1216	2169
Symmetric	675.8	1453	1623

The table shows that except for the disturbed case when both models fail, the symmetric model gave better results. These results underline that the symmetric model is better, so it is applied in the final M600 simulation.

6.3. Horizontal Torque Model Identification

The postulated model is presented in (11). Combining it with the rotational dynamics (2) results in (24).

$$\begin{aligned}
J_x\dot{p} =& qr(J_y - J_z) - qJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\
& c_{T0}\rho AR^2\frac{l}{2}(\Omega_{LB}^2 + \Omega_{LF}^2 + 2\Omega_{LS}^2 - \Omega_{RB}^2 - \Omega_{RF}^2 - 2\Omega_{RS}^2) + \\
& \frac{K_c(V_{az})}{100}V_{az}\rho AR\frac{l}{2}(\Omega_{LB} + \Omega_{LF} + 2\Omega_{LS} - \Omega_{RB} - \Omega_{RF} - 2\Omega_{RS}) - K_p p|p| \\
J_y\dot{q} =& pr(J_z - J_x) + pJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\
& c_{T0}\rho AR^2\frac{\sqrt{3}l}{2}(-\Omega_{LB}^2 + \Omega_{LF}^2 - \Omega_{RB}^2 + \Omega_{RF}^2) + \\
& \frac{K_c(V_{az})}{100}V_{az}\rho AR\frac{\sqrt{3}l}{2}(-\Omega_{LB} + \Omega_{LF} - \Omega_{RB} + \Omega_{RF}) - K_q q|q|
\end{aligned} \tag{24}$$

In (24), the first four terms on the right-hand side are known from angular rate measurement, inertia and vertical force identification. On the left-hand side, smoothed numerical differentiation from [38] was applied to calculate $\dot{p}, \dot{q}$ so those are also known. Theoretically, only the air drag terms (K_p, K_q) should be identified. Subtracting the known terms of the right-hand side gives residual terms (ΔL for roll and ΔM for pitch) which

should be covered with the air drag. The residuals were far from zero and the air drag component (plotted with $K_p = 10, K_q = 10$) had a completely different characteristic, so it was impossible to cover the residuals with any constant K_p, K_q as Figures 34–36 show. So, additional model terms were required. Several combinations of $V_{ax}, V_{ay}, \Omega_i, \phi, \theta, \|V_a\|$, and $\|W\|$ were tested to find something characteristically similar to the residuals. Finally, the wind disturbance-based models below were postulated to cover the residuals, as they were very similar to the residual characteristic (see Figures 34–36).

$$\begin{aligned} &K_{Wp}\|W\|\frac{l}{200}(\Omega_{LB} + \Omega_{LF} + 2\Omega_{LS} - \Omega_{RB} - \Omega_{RF} - 2\Omega_{RS}) \\ &K_{Wq}\|W\|\frac{\sqrt{3}l}{200}(-\Omega_{LB} + \Omega_{LF} - \Omega_{RB} + \Omega_{RF}) \end{aligned} \tag{25}$$

here, $\|W\| = \sqrt{W_N^2 + W_E^2}$ is the Euclidean norm of the wind disturbance and K_{Wp}, K_{Wq} are gains to be determined. From engineering intuition, it would be better to apply the airspeed $\|V_a\| = \sqrt{V_{ax}^2 + V_{ay}^2}$ instead of the wind speed $\|W\|$; however, it gave unacceptable results. To successfully determine the gains, persistently excited flight sections were required, which were inside a short time range to have a valid constant wind disturbance assumption and thus apply the estimated constant wind values. Hover, 5 m/s, 10 m/s, 14 m/s, $+180^\circ$ yaw and -180° yaw maneuvers were considered from the flights, forming different combined data sets from them to give a wide range of excitation. Parameters estimated from the $FT1$ flight campaign are shown in Table 16; those from $FT2$ are in Table 17 together with the average values. Note that multiple take-offs were considered from the first flight campaign.

The average data showed that the gain is wind strength dependent, so polynomials were fitted to the $\|W\| - K_W$ curves. Raw data and fitted polynomials are shown in Figure 33.

Table 16. Pitch and roll moment parameters from first flight campaign ($FT1$).

Take-Off	Set	K_{Wq}	K_{Wp}	$\|W\|$
1	Set $FT1/1$	–2.3902	–2.4444	4.5636
1	Set $FT1/2$	–2.3927	–2.4537	4.5636
AVERAGE	-	–2.3914	–2.449	4.5636
2	Set $FT1/1$	–1.9803	–2.007	5.527
2	Set $FT1/2$	–2.0158	–2.0136	5.527
AVERAGE	-	–1.9981	–2.0103	5.527

Table 17. Pitch and roll moment parameters from second flight campaign ($FT2$).

Set	K_{Wq}	K_{Wp}	$\|W\|$
Set $FT2/1$	–5.2789	–5.3329	2
Set $FT2/2$	–5.216	–5.1729	2
Set $FT2/3$	–5.2445	–5.2666	2
Set $FT2/4$	–5.2464	–5.2241	2
Set $FT2/5$	–5.208	–5.25	2
Set $FT2/6$	–5.1866	–5.1786	2
Set $FT2/7$	–5.1939	–5.086	2
AVERAGE	–5.2249	–5.2159	2

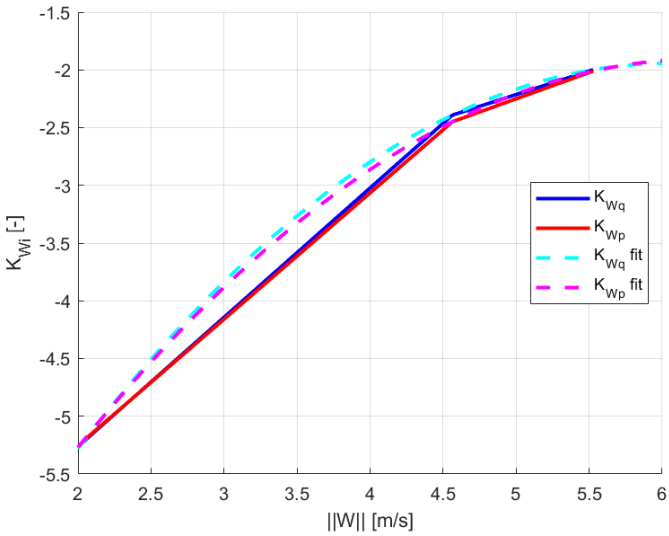

Figure 33. $\|W\| - K_W$ curves and fitted polynomials.

Figure 33 shows that a negative second-order function can possibly well cover the data. This is underlined by the resulted best fit (measured by mean squared error) expressions ($W = \|W\|$):

$$K_{Wq} = -0.2W^2 + 2.46W - 9.39$$

$$K_{Wp} = -0.183W^2 + 2.3W - 9.136$$

Some illustrative results with the averaged parameters are plotted in Figures 34–36, showing acceptable coverage of the residuals by the fitted model.

However, during simulation testing, this model form gave instabilities, so finally, the fitted additional term (with K_W) was removed, but air drag effects $-10p|p|$ and $-10q|q|$ were added, resulting in the final roll and pitch dynamic models (26). The 10 coefficient of air drag was heuristically tuned based on the damping behavior of the model for roll and pitch excitation.

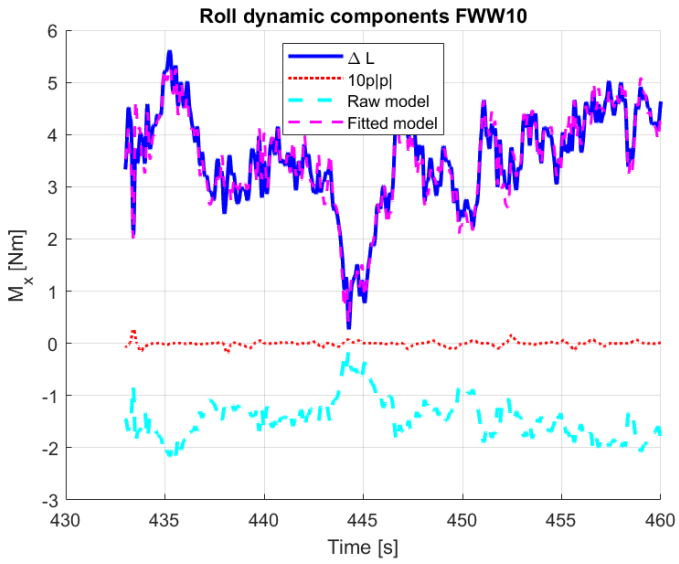

Figure 34. Roll model with averaged parameters (FWW 10 m/s flight section).

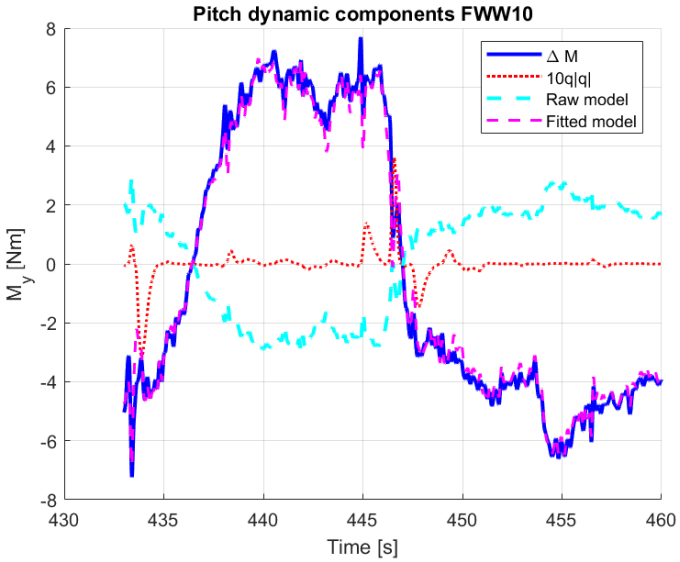

Figure 35. Pitch model with averaged parameters (FWW 10 m/s flight section).

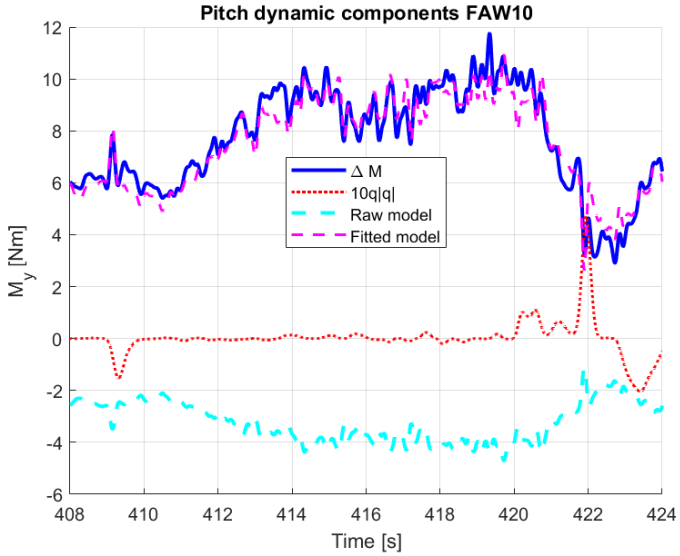

Figure 36. Roll model with averaged parameters (FAW 10 m/s flight section).

$$
\begin{aligned}
\tau_x = &- qJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\
&c_{T0}\rho AR^2\frac{l}{2}(\Omega_{LB}^2 + \Omega_{LF}^2 + 2\Omega_{LS}^2 - \Omega_{RB}^2 - \Omega_{RF}^2 - 2\Omega_{RS}^2) + \\
&\frac{K_c(V_{az})}{100}V_{az}\rho AR\frac{l}{2}(\Omega_{LB} + \Omega_{LF} + 2\Omega_{LS} - \Omega_{RB} - \Omega_{RF} - 2\Omega_{RS}) - 10p|p| \\
\tau_y = &pJ_r(\Omega_{LB} + \Omega_{LF} - \Omega_{LS} - \Omega_{RB} - \Omega_{RF} + \Omega_{RS}) + \\
&c_{T0}\rho AR^2\frac{\sqrt{3}l}{2}(-\Omega_{LB}^2 + \Omega_{LF}^2 - \Omega_{RB}^2 + \Omega_{RF}^2) + \\
&\frac{K_c(V_{az})}{100}V_{az}\rho AR\frac{\sqrt{3}l}{2}(-\Omega_{LB} + \Omega_{LF} - \Omega_{RB} + \Omega_{RF}) - 10q|q|
\end{aligned}
\tag{26}
$$

Identification of the roll and pitch controllers was flawless for this model, which was similar to the final roll and pitch behavior of the DJI M600 simulation model. Thus, model mismatch from the neglection of this effect (questionable from an engineering point of view) was accepted.

6.4. Yaw Torque Model Identification

The yaw torque model is postulated in (13) together with the assumption that the hover speed of the motors is the same, so the hover parts cancel out (14).

However, the voltages of the motors differ from each other as shown, e.g., in Figure 37. As engine speed is proportional to voltage (see, e.g., [33]), this means that the hover speeds were not the same, so the mean hover speed effects were subtracted from the model before identification. This was because only the engine angular rate changes, causing the yaw rotation, should be considered to have a model relative to the hover state (see the control structure in Figure 5). The considered averaged hover values are shown below:

$$\begin{aligned} T_{c_{P0}} &= \overline{\rho A R^3(-\Omega_{LB0}^2 - \Omega_{LF0}^2 + \Omega_{LS0}^2 + \Omega_{RB0}^2 + \Omega_{RF0}^2 - \Omega_{RS0}^2)/100} \\ T_{K_p} &= \overline{\rho A V_{az} R^2(-\Omega_{LB0} - \Omega_{LF0} + \Omega_{LS0} + \Omega_{RB0} + \Omega_{RF0} - \Omega_{RS0})} \end{aligned} \tag{27}$$

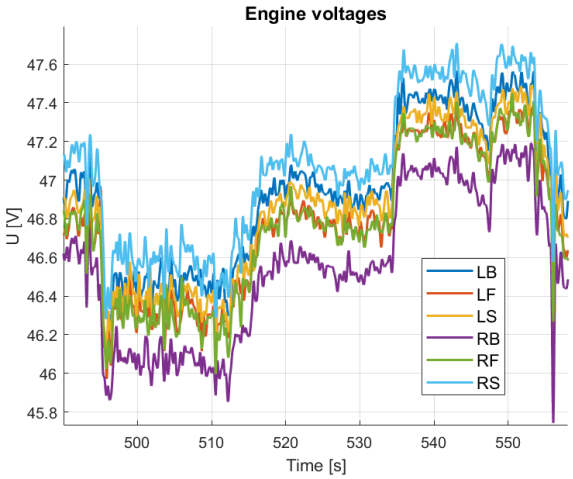

Figure 37. Engine voltages during flight.

Subtracting the hover terms from the τ_z torque (13) and considering the rotational dynamics (2) resulted in the complete yaw dynamics (28).

$$\begin{aligned} &J_z\dot{r} = pq(J_x - J_y) + \\ &c_{P0}\left[\frac{\rho A R^3}{100}(-\Omega_{LB}^2 - \Omega_{LF}^2 + \Omega_{LS}^2 + \Omega_{RB}^2 + \Omega_{RF}^2 - \Omega_{RS}^2) - T_{c_{P0}}\right] + \\ &K_p\left[\rho A V_{az} R^2(-\Omega_{LB} - \Omega_{LF} + \Omega_{LS} + \Omega_{RB} + \Omega_{RF} - \Omega_{RS}) - T_{K_p}\right] - K_r r|r| \end{aligned} \tag{28}$$

In the above model, the unknowns were c_{P0}, K_p and K_r; however, hover and $\pm 180^\circ$ yaw rotation maneuvers were applied to identify the model parameters and the M600 held position during these maneuvers, so the V_{az} velocity was small and was not properly excited. On the contrary, during FAW or FWW maneuvers, the vertical rotation was zero. So finally, the K_p term was removed from the model together with $pq(J_x - J_y)$, while the measured term was negligibly small (see Figures 38–40) due to the almost zero roll and pitch rates in hover or vertical rotation.

Parameter identification was completed with the least squares (LS) method selecting hover sections followed by positive and negative yaw rotations, so three flight sections/data sets were considered. The results are summarized in Table 18, including both $FT1$ and $FT2$ results. The table shows that the parameters are close to each other in one flight, and the c_{P0} values are globally close to each other. The K_r values are more uncertain, showing that the dominant term is c_{P0}. Finally, the averaged values were applied as they gave satisfactory results shown in Figures 38–40. In the figures, $J_z dr$ is the left side with inertial

torque, 'Inertial' means the $pq(J_x - J_y)$ term and 'Model' is the output of the postulated model with average identified parameters. The figures show the negligible value of the $pq(J_x - J_y)$ inertial term and the good fit of the model both for hover and yaw rotations. Again, the performance measures (23) were too large to make a proper model selection, while the qualitative fit can be seen in the figures.

Table 18. Yaw moment parameters.

FT	Set	K_r	c_{P0}
1	Set 1	0.3035	0.2906
	Set 2	0.3099	0.2961
	Set 3	0.3092	0.2927
2	Set 1	0.4064	0.2733
	Set 2	0.356	0.2552
	Set 3	0.4403	0.2693
AVERAGE		0.3542	0.2795

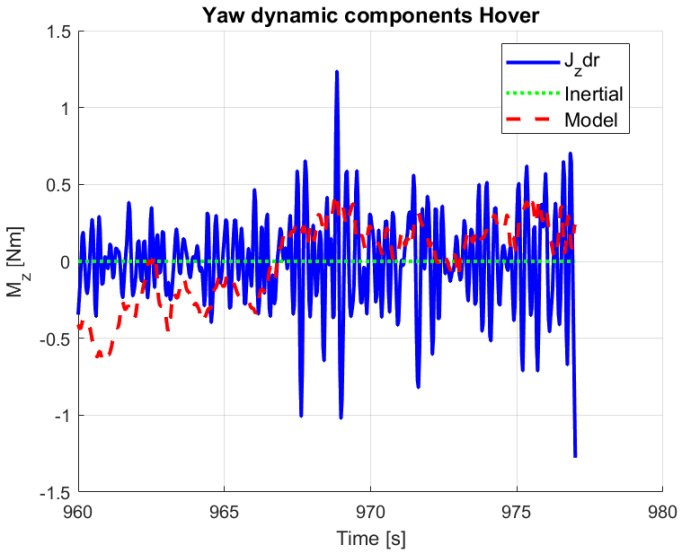

Figure 38. Yaw torque model in hover.

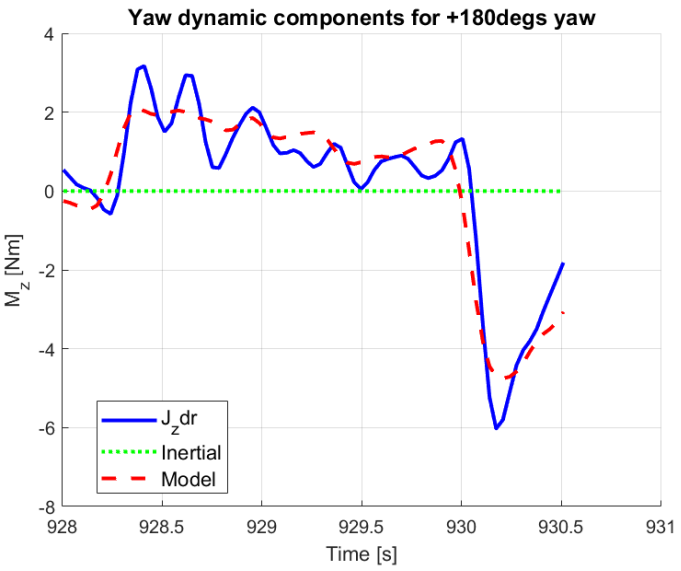

Figure 39. Yaw torque model in +180 degs yaw turn.

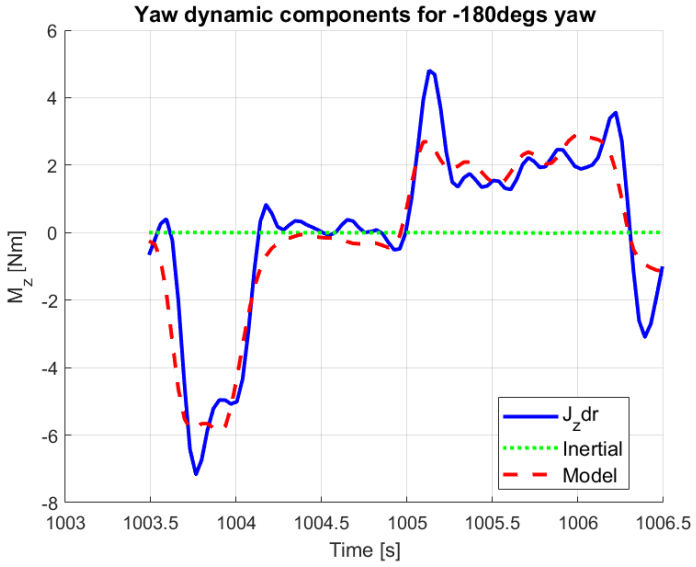

Figure 40. Yaw torque model in –180 degs yaw turn.

7. Control System Identification

After identifying the dynamic and aerodynamic parts of the system model, the factory DJI control loops were identified, considering the possibilities of OSDK control summarized in Table 1.

As vertical and yaw control can be identified in the position hold model and thus are relatively independent from the horizontal motion of the drone, they were identified first. The horizontal control of the drone was only identified after. In most of the cases, transfer function identification was completed on noisy measured data, so the goal was only to cover the characteristic dynamics without considering turbulence and noise effects in the models.

7.1. Vertical Control Identification

The altitude controller structure is postulated and discussed in Section 3.3 in Figure 10. It was straightforward to generate the vertical speed reference from the altitude error with a proper gain and saturation. Upon the step change of the altitude reference, there was a large step change in altitude error, so the maximum saturation of V_D (this case 3 m/s) was immediately activated. Thus, the altitude error to V_D reference gain could only be identified in the sections where the altitude approached the reference, and so the scaled error decreased below the saturation limit. These sections were cut out from the flight data, and an LS optimal gain fit was performed, resulting in $K_{alt} = -0.3769$ for $FT1$ and $K_{alt} = -0.3865$ for $FT2$. The method is illustrated in Figure 41. The difference between the two gains is about 2.5%, so the average value can be applied:

$$K_{alt} = -0.3817$$

The engine speeds can be generated directly from the V_D tracking error, assuming that the engine dynamics is included in the controller. For the identification of the controller, two ascend–descend sections were selected from $FT1$ and two ascend sections were selected from $FT2$. Note that while there was hovering between ascend and descend in the first campaign, there was an immediate transition to descend in the second, as shown in Figure 42, so in this case, only the ascend dynamics were considered for identification.

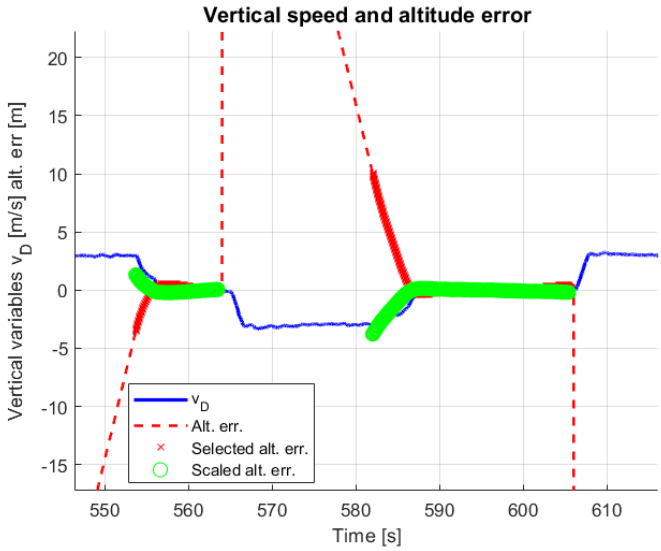

Figure 41. Conversion of altitude error to V_D reference on selected sections (highlighted with x and o symbols).

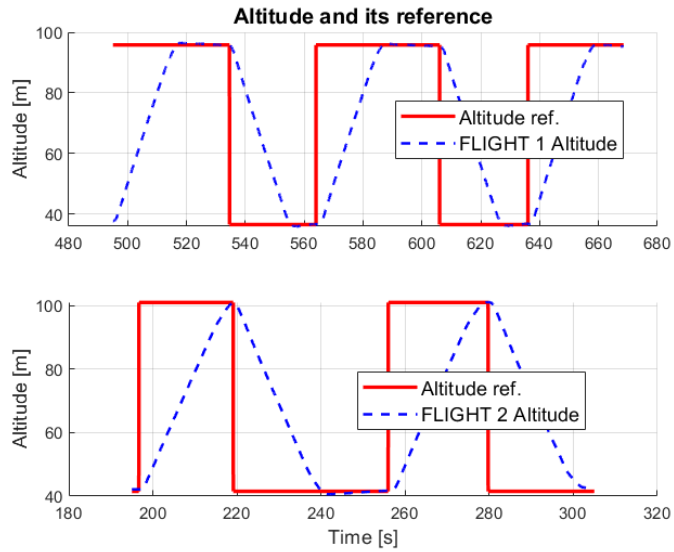

Figure 42. Flight altitude and the generated reference signal in flight campaigns $FT1$ and $FT2$.

Continuous time controller transfer function identification was completed using the Matlab *tfest* function with the vertical speed error as input and the average engine speed (between the six engines) relative to the hover average value as output. The denominator order was selected by trial and error, and a second-order denominator gave acceptable results for all flights (1, 2 for the first campaign and 3, 4 for the second):

$$G_{V_D E_1} = \frac{-23.27s - 8.273}{s^2 + 1.023s + 0}, \ 55.44\%$$

$$G_{V_D E_2} = \frac{-16.53s - 10.76}{s^2 + 0.9894s + 0}, \ 64.2\%$$

$$G_{V_D E_3} = \frac{-21.99s - 13.16}{s^2 + 1.395s + 0}, \ 55.05\%$$

$$G_{V_D E_4} = \frac{-7.688s + 0.5073}{s^2 + 1.523s + 0}, \ 7.9\%$$

The parameters were close to each other, and integral controllers resulted as expected; however, the numerator of $G_{V_DE_4}$ was an outlier (more than 30% difference from the other parameters). This was underlined by the fit measures provided by Matlab *compare* function. The measure of $G_{V_DE_4}$ was much lower, while the others were close to each other. $G_{V_DE_2}$ gave the best measure, but the first numerator coefficients are closer to each other in cases of $G_{V_DE_1}, G_{V_DE_3}$ obtained, respectively, from $FT1$ and $FT2$. That is why, finally, an average function was calculated from $G_{V_DE_1}, G_{V_DE_2}, G_{V_DE_3}$ to balance the disturbance effects between the different flights.

$$G_{V_DE} = \frac{-20.6s - 10.731}{s^2 + 1.1358s} = \frac{-9.448(1.9197s + 1)}{0.88s^2 + s} \tag{29}$$

A detailed evaluation of the result (test on flight data) led to the decrease of the time constant from $T = 0.88s$ to $T = 0.5s$. The fit quality values of the final controllers are presented in Table 19 giving balanced performance for $FT1$ but worse and unacceptable values for $FT2$. The worst value is about 0%, but Figure 43 shows that the trend of the function is acceptable. The best fit is presented in Figure 44.

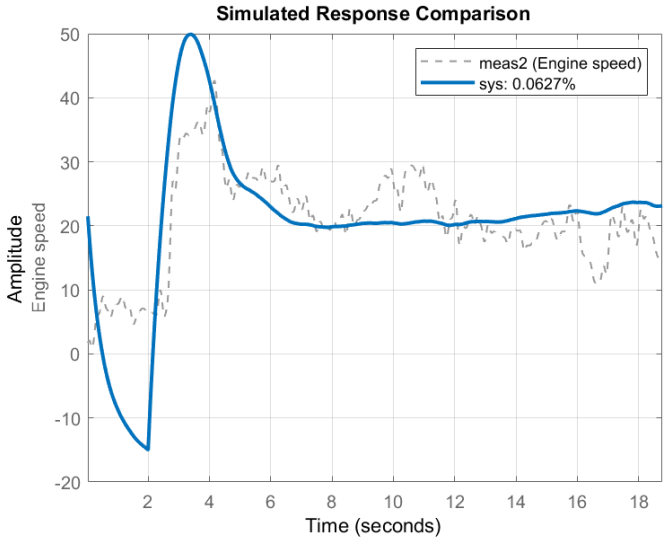

Figure 43. Worst altitude control function fit.

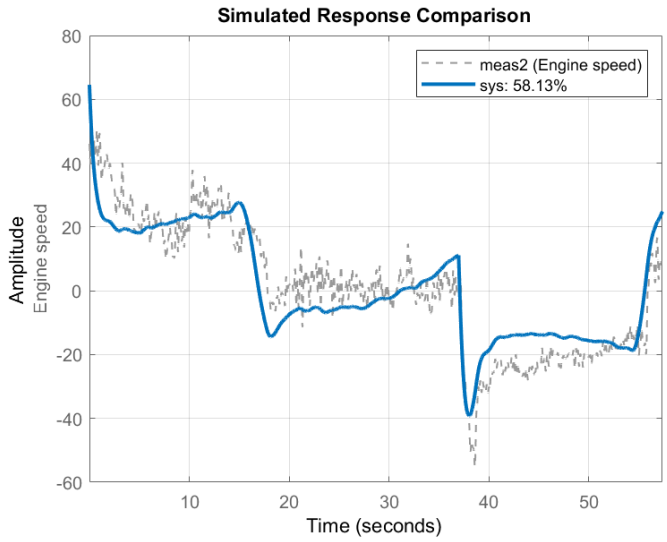

Figure 44. Best altitude control function fit.

Table 19. Vertical controller fit to flight data.

FLY	Set	Fit Quality
*FT*1	1	58.08%
*FT*1	2	58.13%
*FT*2	3	43.33%
*FT*2	4	0.0627%

7.2. Yaw Control Identification

The yaw control model is postulated in Section 3.3 in Figure 11. For this identification, $\pm 180^\circ$ rotations from the first flight campaign (*FT*1) and the largest rotations from the second (*FT*2) were selected. As the latter used a triangular flight pattern, there were no 180° rotations. As $\pm 180^\circ$ rotation means opposite orientation, the roll and pitch values required to maintain hover against the wind swap with each other, which also means a change in the engine speeds. As on the *FT*1 flight test, there was high wind speed; this effect was significant. So in this case, the yaw control engine speed components were calculated by subtracting a spatially changing engine speed characteristic whose value depend on the northeast angular positions of the engines. That is why first, the NE angular positions of the engines (body positions are shown in Figure 8) and their speeds were registered before (at -38° M600 yaw angle) and after (at 142° M600 yaw angle) the yaw rotations and plotted in Figures 45 and 46. The figures show that the average values give a close to sinusoidal change, so a sinusoidal curve was fitted as $\Omega_0(\varphi) = 275 - 43\sin(\varphi + 70^\circ)$ where φ is the engine actual angular position in the NE system. The engine speeds for yaw identification were calculated relative to this curve. In case of the second flight campaign *FT*2, the wind was very low, so no such compensation was required.

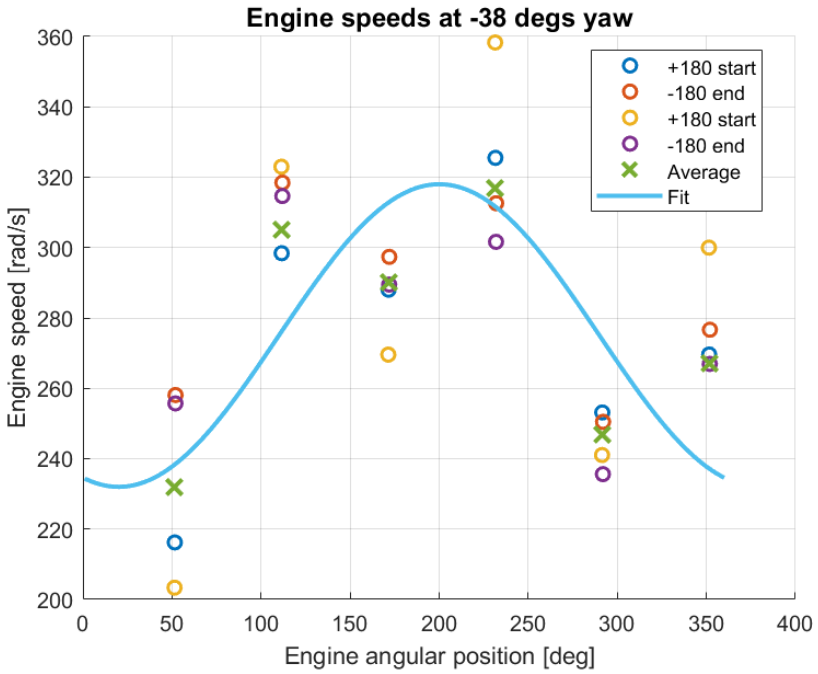

Figure 45. Engine speed effect of engine angular position at -38° yaw.

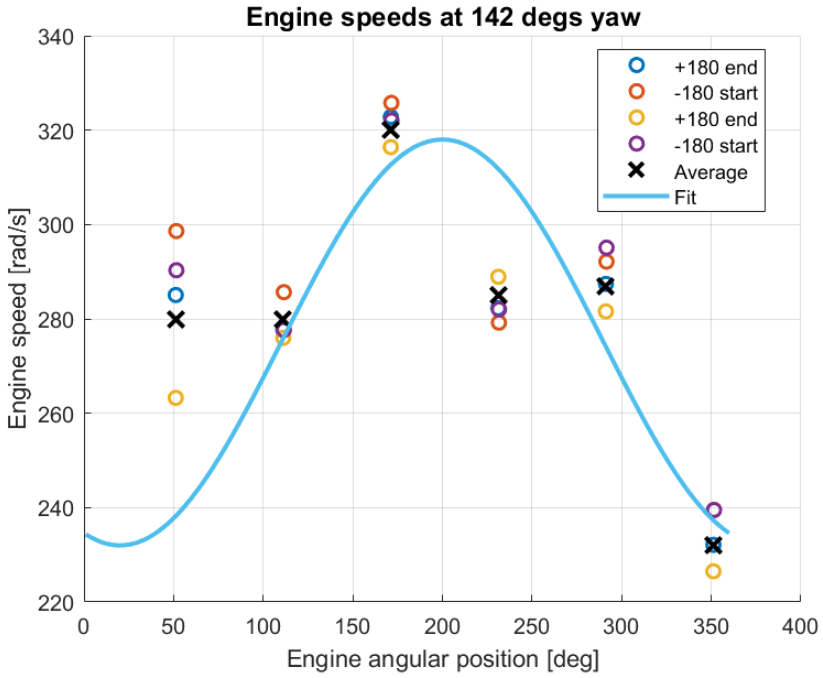

Figure 46. Engine speed effect of engine angular position at 142° yaw.

In the next step, the yaw angle step references were estimated (in $FT1$ and $FT2$, UgCS-generated trajectories were used, so the onboard DJI references were not known) considering the flight yaw angle changes and the start of yaw rate changes. A constant gain was applied to generate the yaw rate reference from the yaw angle error considering the yaw rate saturations, which are 0.72 rad/s and -1.605 rad/s for $FT1$ and $+1.92$ rad/s and -1.745 rad/s for the $FT2$ flight campaign (set by the DJI system independently from the user). $K_\psi = 2$ was the first result but finally, tuning with yaw control simulation $K_\psi = 1.5$ gave satisfactory results as Figure 47 shows. For unlimited applicability, the yaw angle error $\psi_{ref} - \psi$ should be checked for unrealistic large values upon changes from 180° to $-180°$ or vice versa. This is denoted by the $\pm\pi$ in the K_ψ gain block in Figure 11.

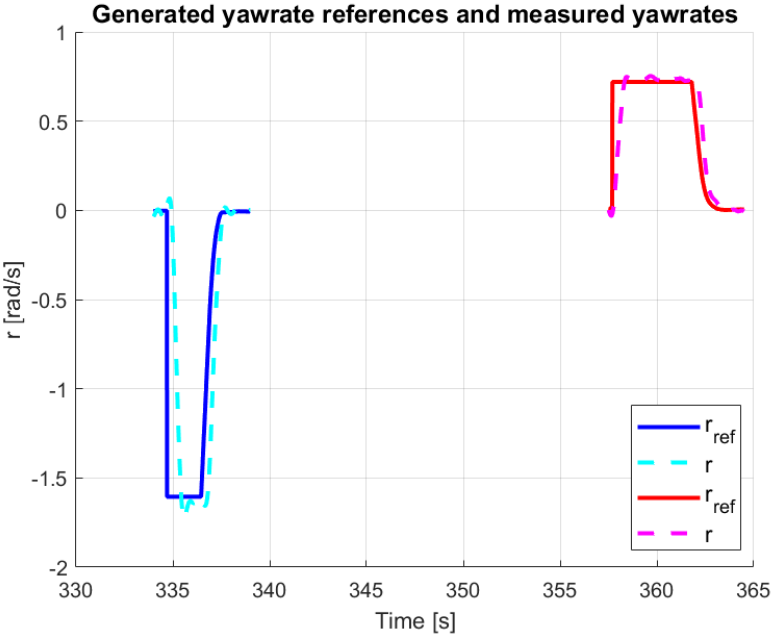

Figure 47. Yaw rate reference from yaw angle error and flight yaw rate.

After generating the yaw rate reference from the angle error, the yaw rate error to engine speed (relative to the hover speeds) transfer functions were identified. Considering the engine rotational directions in Figure 8, their reaction torques are opposite to the rotation so RF, RB, LS engines give positive torques and RS, LF, LB give negative torques. That is why the controller dynamics were identified separately for these two groups ($G_{rE}+$ and $G_{rE}-$), again applying the Matlab *tfest* function with second-order transfer function denominators. However, first the delay between the yaw rate error and engine speed output in the measured data was corrected, being about 0.26*s* in all flights. Later (see Section 8), detailed delay analysis was applied, resulting in a separate yaw angle and yaw rate reference delays, as shown in Figure 11.

The transfer function identification results are shown in Table 20 with averaged functions for the two $+180^\circ$ and -180° rotations. Four sets from each flight campaign were considered.

Table 20. Yaw controller transfer functions.

FLY	Set	$G_{rE}+$	$G_{rE}-$
*FT*1	1–3	$G_{rE1}+ = \frac{76.4(s+0.43)}{s(0.046s+1)}$	$G_{rE1}- = \frac{-193.8544(s+0.1243)}{s(0.6932s+1)}$
*FT*1	2–4	$G_{rE2}+ = \frac{85.9512}{s(0.0964s+1)}$	$G_{rE2}- = \frac{-77.4487(s+0.5384)}{s(0.0134s+1)}$
*FT*2	5–7	$G_{rE3}+ = \frac{68.43(s-0.1377)}{s(0.11s+1)}$	$G_{rE3}- = \frac{-76.8125(s+0.4649)}{s(0.0653s+1)}$
*FT*2	6–8	$G_{rE4}+ = \frac{X}{s(0.0428s+1)}$	$G_{rE4}- = \frac{-67.3(s+0.14)}{s(0.226s+1)}$

Regarding the $G_{rE}+$ results, the numerator gains were close to each other except for the fourth case where the numerator was negative and thus invalid. The gains for the $G_{rE}-$ case were also similar to the $G_{rE}+$ case, too. However, the numerator time constants were very different for $G_{rE}+$, while for $G_{rE}-$, the 1 and 4 and the 2 and 3 numerator time constants were similar. Regarding the denominator time constants, there were slower (above 0.1 s) and faster (below 0.1 s) values. Finally, after evaluating and tuning on all of the data sets to obtain the best overall results, the following form was fixed:

$$G_{rE} = \frac{\pm 97.2(s+0.3)}{s(0.0767s+1)} \tag{30}$$

The fit qualities are shown in Table 21. The best result (Set 5/$G_{rE}-$ with 70.45% fit quality) and worst result (Set 1/$G_{rE}-$ with 24.13% fit quality) are plotted in Figure 48 and Figure 49, respectively. Considering the amplitudes, the worst result is also acceptable, only it is delayed at the beginning. Overall, the fit quality is above 40–50% in most of the cases.

Table 21. Data fit of final yaw controller transfer function.

FLY	Set	$G_{rE}+$	$G_{rE}-$
*FT*1	1	25.16%	24.13%
*FT*1	2	45.37%	29.25%
*FT*1	3	42.01%	33.01%
*FT*1	4	69.72%	51.36%
*FT*2	5	42.63%	70.45%
*FT*2	6	52.07%	64.4%
*FT*2	7	43.78%	62.7%
*FT*2	8	48.59%	50.13%

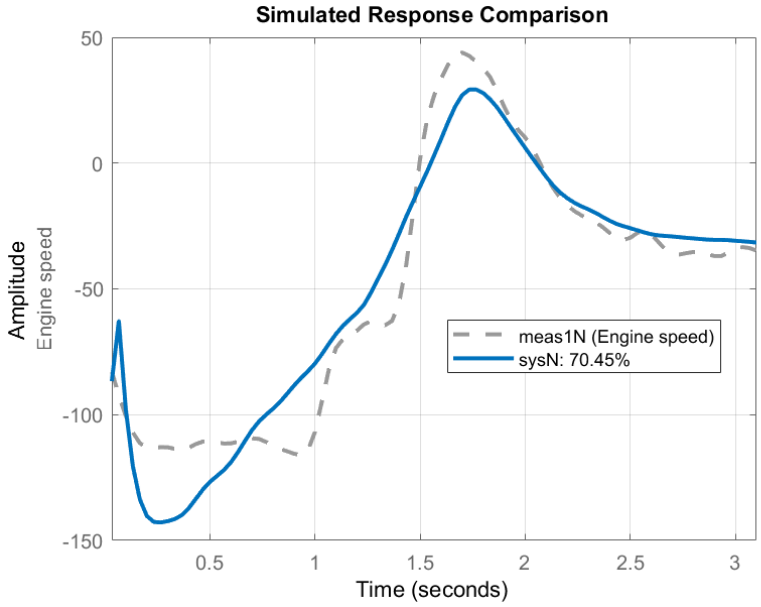

Figure 48. Yaw angle control model output and flight data in the best case.

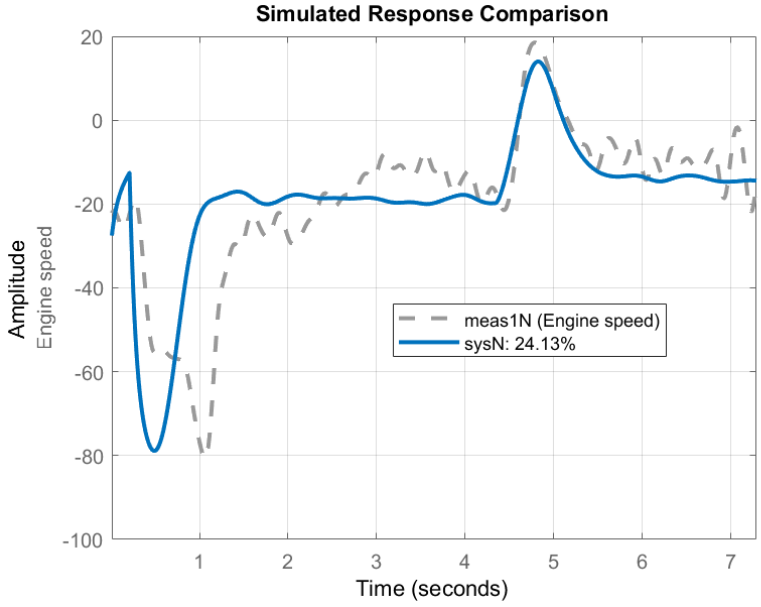

Figure 49. Yaw angle control model output and flight data in the worst case.

7.3. Horizontal Velocity Control Identification

The horizontal control structure is postulated in Section 3.3 in Figure 9. After saturating the velocity reference, pitch or roll angle references (depending on tangential or normal control) should be generated from the velocity error. However, as the wind disturbances must be compensated with nonzero pitch and roll angles, a PI controller was required for this task to be able to hover with zero velocity (and so nonzero angles). Integral control requires AW ($\theta_{max} = \phi_{max} = 25°$) to quickly react with the angles for any changes. However, this introduced a switching nonlinearity into the system, making closed-form transfer function identification impossible. That is why the PI gains were tuned by trial-and-error. There was an attempt to identify the pitch (roll) angle error to engine speed control dynamics subtracting hovering engine speeds, but the results were unsatisfactory when tested in simulation. So, finally, two-step tuning was performed after building a longitudinal simulation including altitude and pitch dynamics.

The postulated angle-tracking controller transfer function was similar to G_{alt} and G_{ψ} without the integral effect (as the engine speeds generate torque and thus angular acceleration, the system has integral property):

$$G_{\theta} = \frac{g(T_d s + 1)}{Ts + 1} \tag{31}$$

Note that for pitch control, only the front and back four engines were applied with negative controller dynamics for the back ones. In roll control, the right engine reactions were considered negative. First, the g gain (pure P controller) was tuned to have a proper initial response $g = 100$, which was satisfactory. Then, the time constant T and derivative effect T_d were tuned to have short settling time (pitch (roll) control is the innermost loop, so it should be fast) without oscillations, $T = 0.01s$ was satisfactory together with $T_d = 0.1s$, so the final control transfer function (both for pitch and roll) is:

$$G_{\theta(\phi)} = \frac{100(0.1s + 1)}{0.01s + 1} \tag{32}$$

After identifying the pitch (roll) error to engine speed control dynamics, the gains of the PI AW controller generating the reference angles from the velocity error were determined applying the longitudinal control model again by trial-and-error to fit flight data with model outputs. Finally, $P = 0.07$ and $I = 0.03$ constant gains were determined with negative signs in case of tangential control (there, a negative pitch down causes positive forward velocity).

After finishing the whole system identification, SIL simulation (provided by DJI) and flight data-based validation and refinement of the model were performed.

8. SIL and Flight Data-Based Verification, Refinement and Special Modes

After identifying the system dynamics and the control loops, the Matlab Simulink 3D simulation model was constructed considering the full 6DoF rigid body motion of the drone (see Figure 4 in Section 3). After model construction, model validation and refinement are required, which was completed running the simulation model with the same OSDK reference inputs as in SIL and real flight and comparing the results. The flight test part was based completely on the *FT*3 flight campaign, whose data were not applied in the identification process, so this provided the validation of the model. Of course, the model was initialized for the initial flight state. The next sections show the identified model outputs and discuss the refinements, presenting the finally applied model and its outputs compared to SIL and flight data. First, the inner and then the outer control loops were refined.

8.1. Vertical Control

First, the tracking of the vertical velocity reference of the model (SIM in the figures) was compared to SIL and flight test results applying the controller in (29). The results are shown in Figures 50 and 51. The figures show that the model was too slow with large overshoot, and there was a delay in both SIL and flight tracking, which was not present in the simulation model. Thus, the gain of the controller was increased, and the numerator time constant decreased as (33) shows. Additionally, a $0.07s$ delay was applied at the $V_{D_{ref}}$ reference, as shown in Figure 10. The improved tracking results are shown in Figures 52 and 53. The figures show that the tracking became very similar to SIL and flight results, though not all of the delay was modeled at the beginning of the reference. However, at the other transients, the delay difference is smaller.

$$G_{V_D E} = \frac{-18(2s + 1)}{0.1s^2 + s} \tag{33}$$

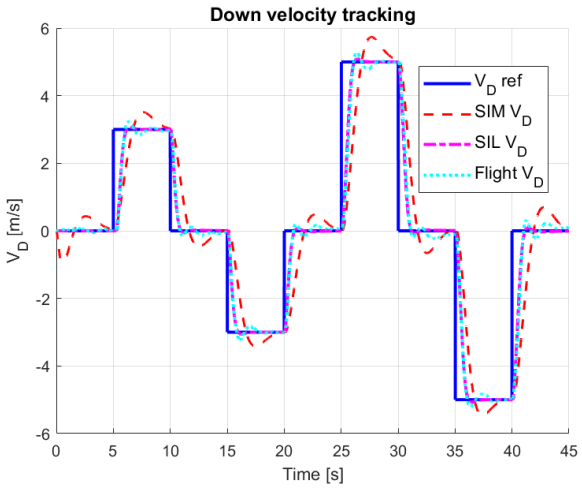

Figure 50. Comparison of vertical velocity tracking results.

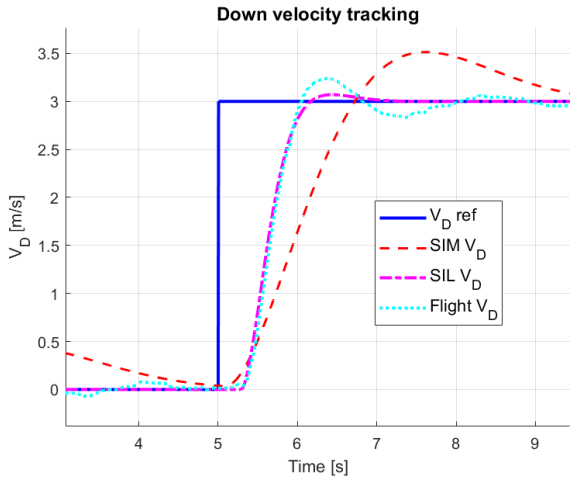

Figure 51. Zoomed start of vertical velocity tracking.

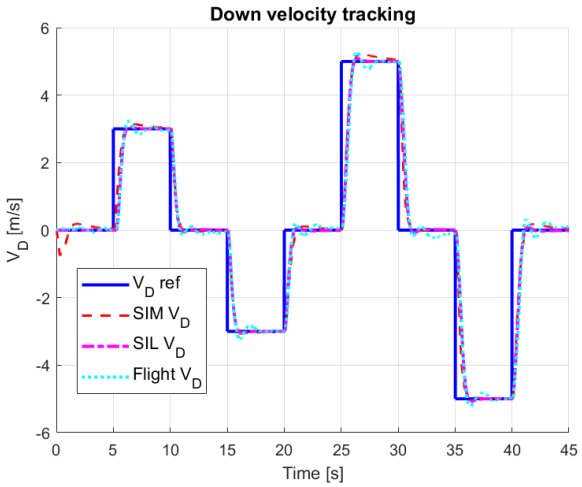

Figure 52. Comparison of vertical velocity tracking results with improved controller.

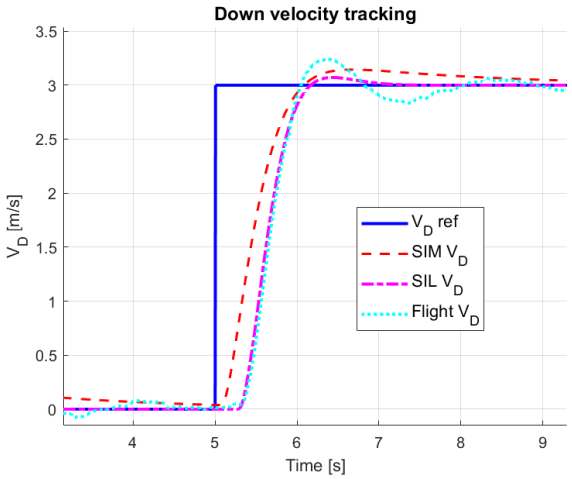

Figure 53. Zoomed start of vertical velocity tracking with improved controller.

After fine tuning the vertical velocity tracking control, the altitude tracking was tested again by comparison to SIL and flight results (see Figures 54 and 55). The figures show that both altitude and vertical velocity dynamics are very similar to SIL and flight, so there is no need to change the K_{alt} gain and the method of vertical velocity reference generation or the controller.

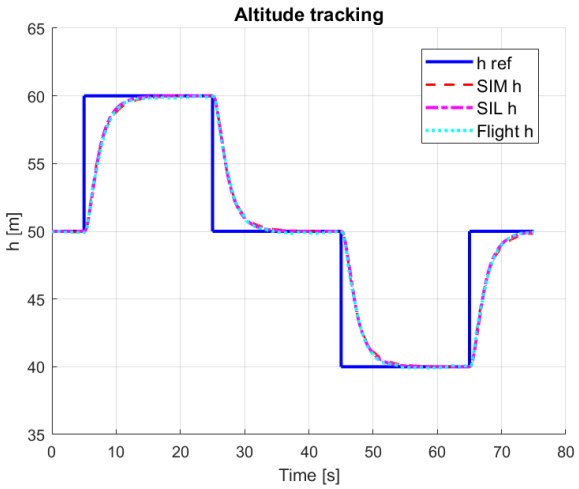

Figure 54. Comparison of altitude tracking results.

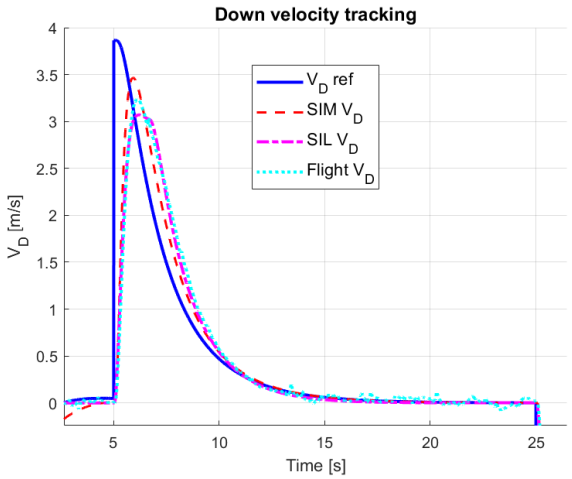

Figure 55. Vertical velocity tracking in altitude tracking control.

8.2. Yaw Control

In case of yaw control, first, the yaw rate tracking was evaluated and fine tuned. Unfortunately, the yaw rate was only saved onboard the drone, and this log file was damaged during the SIL test, so only the comparison to flight test results was possible.

Figure 56 shows the yaw rate tracking with controller from (30) and zero reference delay. The figure shows that the model was slow and did not converge fast enough in case of larger yaw rate reference. Thus, the controller time constant was decreased, and the gain increased, resulting in (34). Also, 0.02 s delay was added to the yaw rate reference based on detailed evaluation of the data. The improved yaw rate tracking is shown in Figure 57 with fast increase and settling.

$$G_{rE} = \frac{200(s+0.3)}{0.06s^2 + s} \tag{34}$$

After tuning the yaw rate inner loop, the yaw angle tracking control was evaluated first, with a $K_\psi = 1.5$ yaw angle error to yaw rate gain. Figure 58 shows that the tracking was slow and did not have any overshoot. So finally, $K_\psi = 2.5$ was applied, reproducing the fast dynamics but without overshoot (note that the first identified parameter was $K_\psi = 2$ closer to the final value; see Section 7.2). With larger K_ψ, part of the overshoots could be reproduced but with too fast dynamics. Thus, finally, the dynamics similar to SIL and flight was preserved, sacrificing the overshoot. The tracking results of the final control are shown in Figure 59 together with the yaw rate dynamics in Figure 60. The latter shows a faster increase in yaw rate and the lack of overshoot.

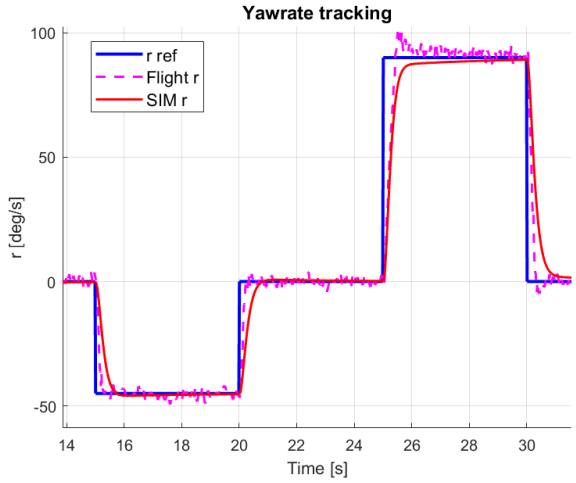

Figure 56. Yaw rate tracking.

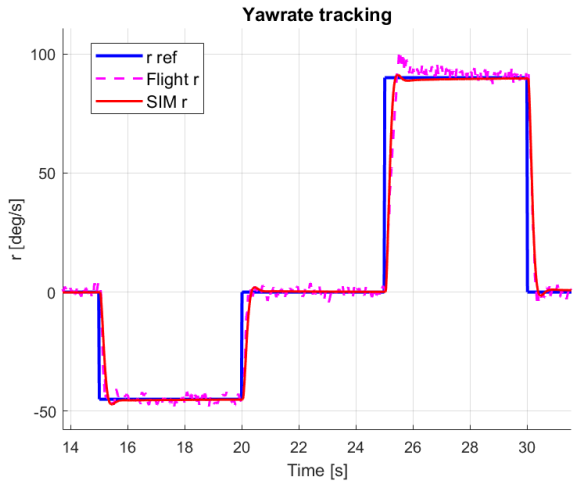

Figure 57. Yaw rate tracking with improved controller.

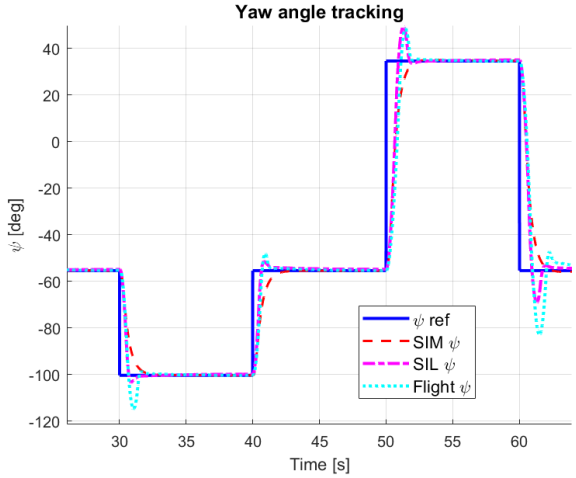

Figure 58. Yaw angle tracking.

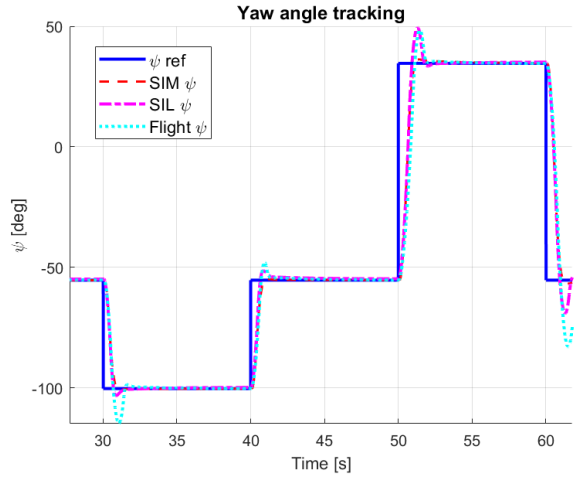

Figure 59. Yaw angle tracking with improved controller.

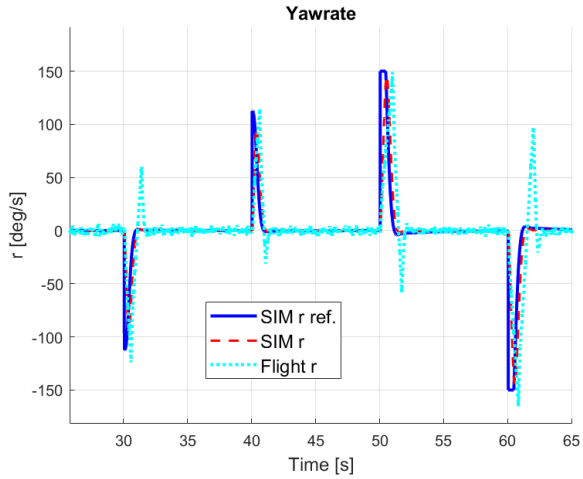

Figure 60. Yaw rate in yaw angle tracking with improved controller.

8.3. Horizontal Control

First, the roll and pitch angle tracking control was verified, and the handling of the special mode discussed in Section 4.1.2 was solved. The final tracking results are presented in Figures 61 and 62. The normal tracking of the angle references worked well with the identified model (see (32)), giving even slightly better results than the SIL simulation (compared to the flight response). Only 0.07 s delay was added to the angle references.

The special mode includes switching from angle reference tracking to zero velocity reference tracking, and also the increase in velocity tracking gain was observed, meaning that a braking mode is activated in this case. The figures show that even this braking mode behavior is well described by the model having outputs close to the SIL results.

Zero velocity tracking is switched in the model when the OSDK control is switched to angle tracking mode and both roll and pitch references are zero. The activation of the braking mode is more complex, following the rules below:

- If $V_{ref} = 0$ and $|V| > 1$ (consequently $|\Delta V| > 1$) and the controller is in normal velocity tracking mode: switch braking mode meaning upscale of the P gain $P' = 1.8P$ and zeroing out the integral state of the PI controller.
- If $|\Delta V| < 1$ and controller is in braking mode: switch back to normal tracking mode by applying the P gain again (integral state remains unchanged)

So, the braking mode is activated when the commanded velocity is zero, the velocity tracking error is large and even the velocity itself is large. Note that this is why the ΔV velocity tracking error is an input of the velocity tracking PI controllers (see Figure 9).

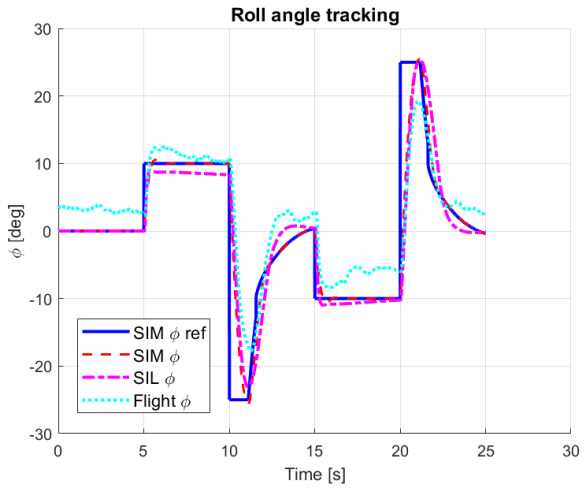

Figure 61. Roll angle tracking.

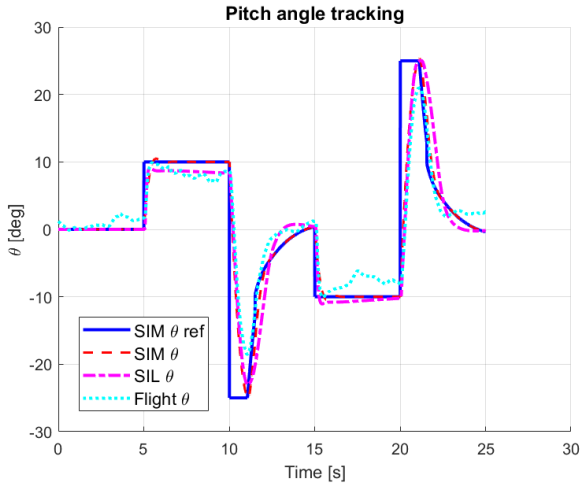

Figure 62. Pitch angle tracking.

After validating the roll and pitch angle control and its special mode, tangential and normal velocity tracking control were evaluated. The results are shown in Figures 63–66. Figures 63 and 64 show that the velocity tracking is pretty good with the original identified $P = 0.07$ and $I = 0.03$ parameters; even the overshoots are reproduced well. The only

difference is the earlier settling of the model to nonzero values and a later settling to zero values, but both are acceptable.

Figures 65 and 66 show that the roll and pitch angles are also reproduced well by the model being closer to the flight data than to SIL results. Note that the flight data are shifted due to the roll and pitch compensation of wind disturbances. Shifting its theoretically zero sections back to zero, the nonzero sections approach the model output.

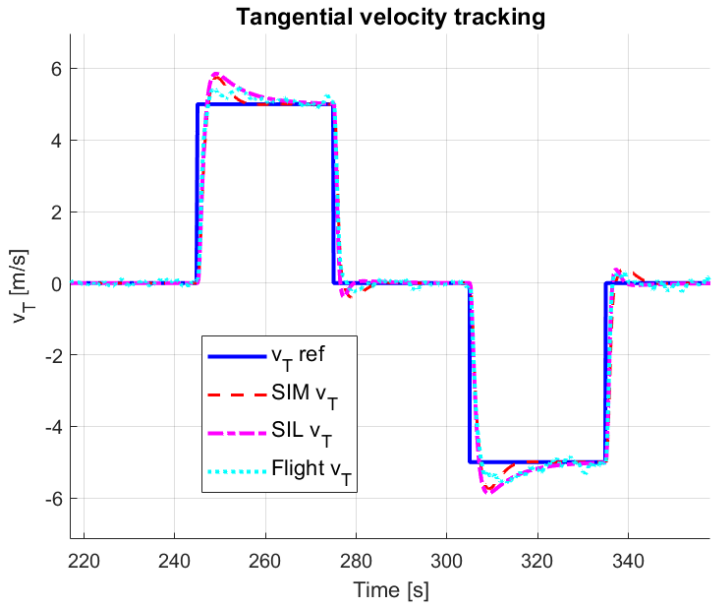

Figure 63. Tangential velocity tracking.

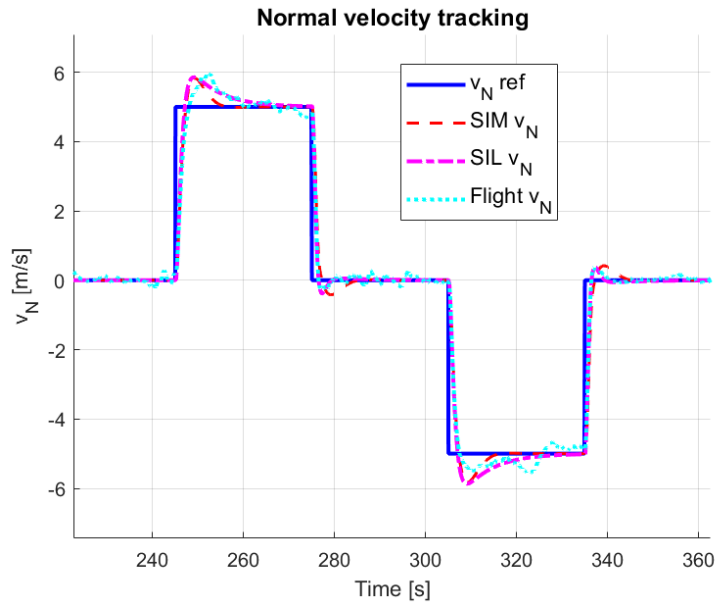

Figure 64. Normal velocity tracking.

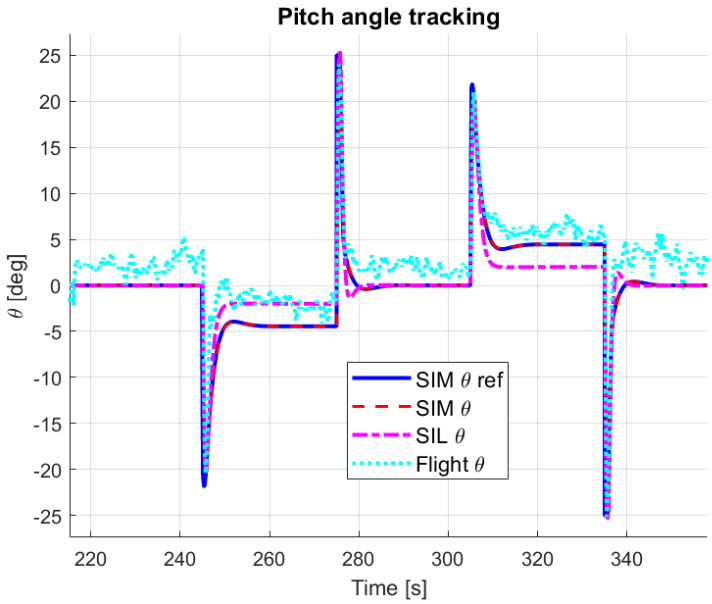

Figure 65. Pitch angle tracking during tangential velocity tracking.

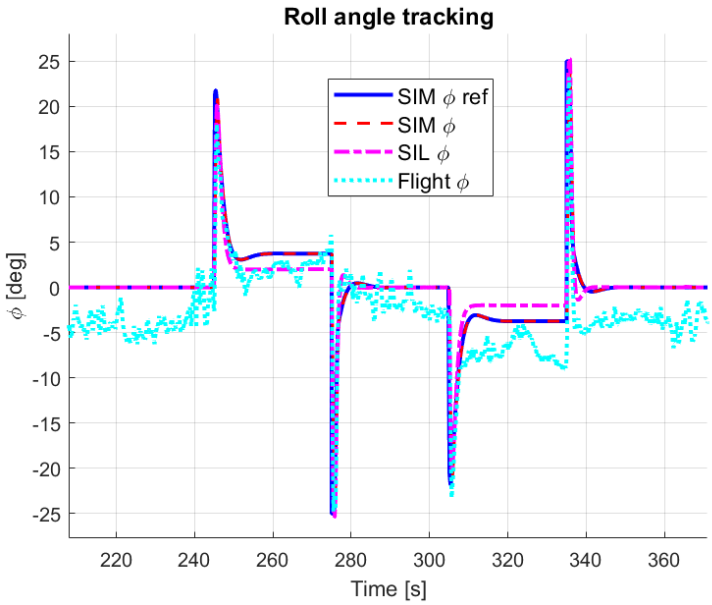

Figure 66. Roll angle tracking during normal velocity tracking.

8.3.1. Position Kick Control Identification

The position kick control is a special mode as discussed in Section 4.1.1. Flight and SIL testing showed that for a position kick reference, the drone accelerates to some velocity (as tracking a half-doublet speed reference) and then stops; for a step position reference, the drone keeps a constant velocity. The questions were the length and value of the half-doublet velocity reference and the value of the constant velocity reference.

Analysis of different position kick tracking maneuvers showed that the value of the half-doublet velocity reference is the value of the position kick signal. Its length was identified through a series of inverse model calculations applying SIL simulation data, as it is not affected by wind disturbance, and its validity for this maneuver was verified in Section 4.1.1 as Figure 15 shows. First, the transfer functions from pitch and roll references ($8 - 9^\circ$ to avoid any saturation in the system) to pitch and roll angles were identified (with Matlab *tfest* function) giving two similar functions with 97.02% and 97.15% fit quality:

$$G_{\theta_{ref}\theta} = \frac{104.8}{s^2 + 13.87s + 121.1}, \quad G_{\phi_{ref}\phi} = \frac{103.4}{s^2 + 14.6s + 119.3}$$

The two functions and their fit quality were very close to each other, so their average was considered as

$$G = \frac{104.1}{s^2 + 14.235s + 120.2}$$

As the inner references of the DJI controller could not be logged, the inverse of this transfer function was applied to estimate the pitch and roll references in position kick flight from the pitch and roll angle outputs. As the transfer function was stable, its inverse could be calculated giving an improper function, but its execution in discrete time was feasible.

After estimating the pitch and roll references, they were smoothed, and the inverse of the PI controller (generating angle references from velocity error) was applied to estimate the velocity references generated by the position kick control. The length of the half-doublet velocity reference was estimated based on this signal and resulted as $0.65s$. Plots of the main steps are shown in Figures 67 and 68.

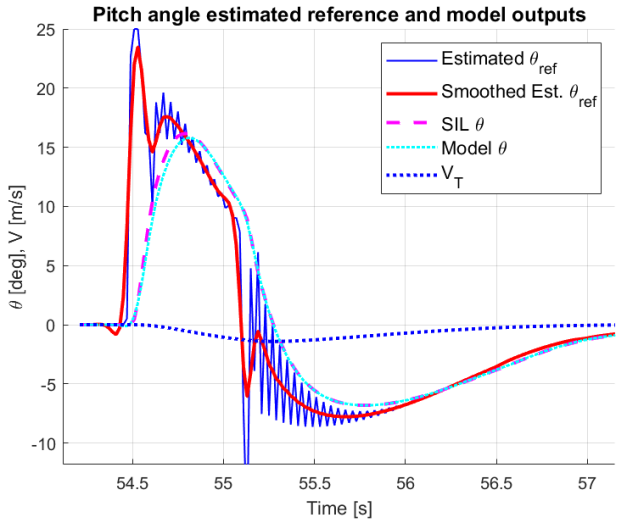

Figure 67. Pitch reference identification from tangential position kick maneuver.

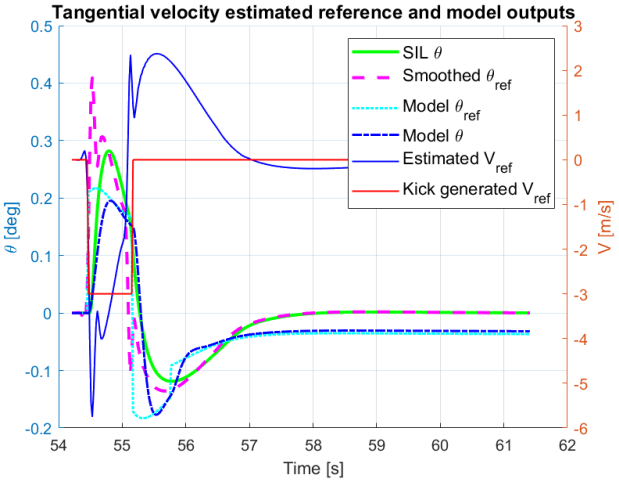

Figure 68. Model behavior in tangential position kick maneuver.

Figure 67 shows the raw and smoothed estimated pitch reference and the transfer function output driven by the smoothed reference compared to the SIL simulation output. The model pitch output is very close to the SIL one. Figure 68 shows the estimated velocity

reference and the generated one based on the width of the estimated signal covering only the first half-doublet wave. The second half is not required to generate a behavior similar to the real system. The pitch references and dynamics are also plotted comparing the smoothed estimated reference to the one generated from the velocity reference and the SIL and model outputs. Comparison of the outputs shows that qualitatively they are similar, but the values are different from each other.

Refinement and final validation of this model was performed on the full 3D simulation of M600, resulting in 0.75*s* width of the velocity reference, giving better results in velocity and position responses. Figure 69 shows the position transients after a tangential kick command. The figure shows that the maximum value of the position move is about the same as in SIL, but the model moves back after reaching the maximum distance. This is caused by a transient of the velocity shown in Figure 70 and could not be removed from the model. The figure also shows that the maximum velocities are about the same as in SIL. The width of the velocity reference was increased to achieve this state. Finally, Figure 71 shows the pitch angle reference and transients having similar peak values and dynamics. As this position kick mode is not the most practical and important one, model refinement was finished at this stage.

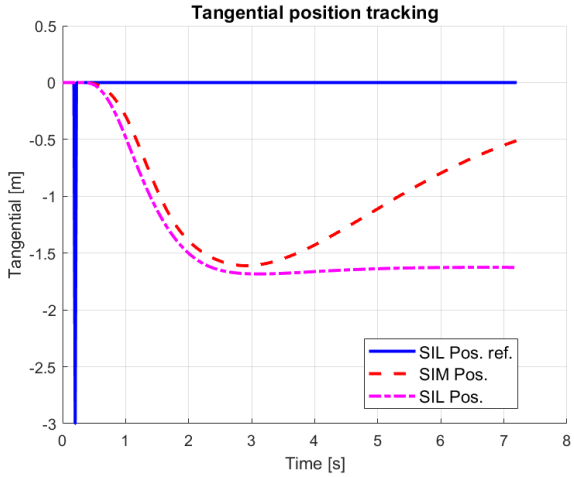

Figure 69. Position transients in tangential position kick maneuver.

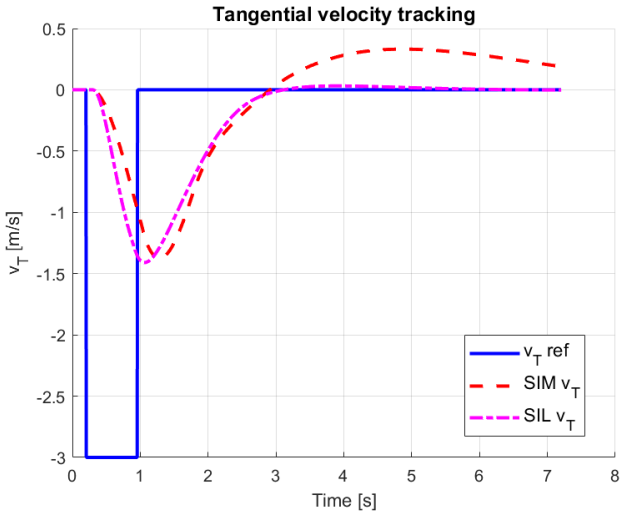

Figure 70. Velocity transients in tangential position kick maneuver.

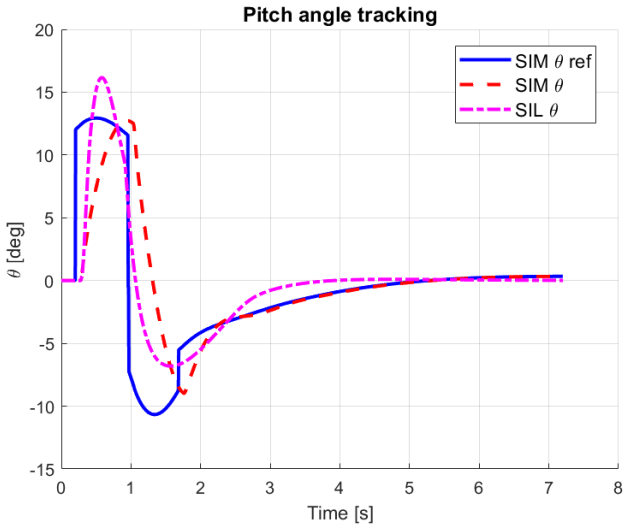

Figure 71. Pitch angle tracking in tangential position kick maneuver.

Regarding the application of step position reference (P_{ref}) instead of the position kick command SIL and flight data analysis showed that for smaller position references, half of them were set as reference velocity, but for larger references, the velocity was limited to 3 m/s, so finally the reference velocity model was:

$$V_{ref} = sign\left(P_{ref}\right) \min\left(\frac{|P_{ref}|}{2}, 3\right)$$

both in tangential and normal direction.

8.3.2. Control in Northeast Directions

In Table 1, the HORIZONTAL_GROUND option means that the horizontal position, velocity or angle references should be implemented along north and east earth coordinate system axes. Although the system identification was performed with the HORIZONTAL_BODY option, this mode was also implemented in the model to extend applicability. For position and velocity references, simply the inverse of the transformation in (1) was applied. The pitch and roll angle references are now understood around the east (α_E) and north (α_N) axes, so the transformation matrix from a rotated body to the NED system should be constructed (based on ψ, α_E, α_N) and the body referenced Euler angles (which drive the controllers in the model) obtained from it. The rotation steps are as follows. Rotation from body to NED:

$$T_\psi = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{35}$$

Rotation around the north axis (ground 'roll' rotation):

$$T_N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha_N) & -sin(\alpha_N) \\ 0 & sin(\alpha_N) & cos(\alpha_N) \end{bmatrix} \tag{36}$$

Rotation around the east axis (ground 'pitch' rotation):

$$T_E = \begin{bmatrix} cos(\alpha_E) & 0 & sin(\alpha_E) \\ 0 & 1 & 0 \\ -sin(\alpha_E) & 0 & cos(\alpha_E) \end{bmatrix} \tag{37}$$

The final rotation from the transformed body to NED system results as:

$$T_{EB} = T_E T_N T_\psi \tag{38}$$

The body referenced Euler angles can be obtained from this and applied in the tangential-normal control setup. After identification, verification and refinement of the dynamics and control loops the conclusion can be drawn.

9. Conclusions

This paper presents the high-fidelity simulation model identification of a DJI M600 Pro hexacopter, including not only system but also controller dynamics, focusing on the OSDK (Onboard Software Development Kit) control modes and presenting even the special cases. Such a model is required in research and development projects aiming to design custom mission controllers for this drone, which is widely used by universities and research institutes.

Real flight and software-in-the-loop (SIL) test data were applied to identify the model also validating the SIL simulation provided by DJI. First, the hardware and control software structure of the drone was introduced. Then, the planned structure of the high-fidelity simulation model was presented. After introducing the three flight test campaigns and the discovered special modes first, the mass and inertial properties were measured and calculated with both battery setups (TB47S and TB48S) and a custom Forerunner UAV payload. Then, the system dynamics (including dynamic and aerodynamic effects) was identified, including horizontal forces and wind disturbances, hover and vertical forces and finally, horizontal and yaw torque dynamics based on real flight data.

After identification of the dynamics, the control loops following the structure of OSDK control possibilities were identified, including also engine dynamics. Vertical, yaw and horizontal control were considered. Before publishing the model, unused flight and SIL data-based verification and refinement was completed for vertical, horizontal and yaw control considering all OSDK control modes except for the horizontal angular rate and vertical thrust, which are the innermost control levels not required for mission planning and control and being dangerous when directly excited by a custom controller. Finally, position kick control (special implementation of horizontal position control by DJI) and the consideration of horizontal references in a northeast (Earth) coordinate system instead of the multicopter body were presented. The refinement and validation was successful; the model covers well all OSDK control modes—even the special ones.

The paper ends with an overview of the resulting high-fidelity simulation and control model implemented in Matlab Simulink as a Supplementary Materials.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/aerospace11010079/s1.

Author Contributions: Experiment design, data processing, model identification P.B.; code building, data logging, experiment execution M.N.; model implementation P.B. and M.N. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002). Project no. TKP2021-NVA-01 has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme. The DJI M600 Pro research platform was provided by the "Developing innovative automotive testing and analysis competencies in the West Hungary region based on the infrastructure of the Zalaegerszeg Automotive Test Track" GINOP-2.3.4-15-2020-00009 project.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to confidentiality reasons.

Acknowledgments: The authors gratefully acknowledge the help of Sandor Csurgai BSc student in collecting and processing simulation and in-flight system responses during his summer internship in 2022.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. UNM. The University of New Mexico, Center for Advancement of Spatial Informatics Research & Education: DJI M600 Equipment. 2023. Available online: https://aspire.unm.edu/facilities/equipment-computing/dji-m600.html (accessed on 21 December 2023).
2. SeaBee. SeaBee Research Infrastructure: DJI M600 Pro. 2023. Available online: https://seabee.no/dji-m600-pro/ (accessed on 21 December 2023).
3. ONERA. ONERA: DJI M600 Application. 2023. Available online: https://www.onera.fr/fr/actualites/imagerie-laser-3d-haute-resolution-embarquee-sur-drones (accessed on 21 December 2023).
4. Sapkota, R. Using UAS Imagery and Computer Vision to Support Site-Specific Weed Control in Corn. Master's Thesis, North Dakota State University of Agriculture and Applied Science, Fargo, ND, USA, 2021.
5. Zhang, J.; Chadha, R.G.; Velivela, V.; Singh, S. P-CAL: Pre-computed Alternative Lanes for Aggressive Aerial Collision Avoidance. In Proceedings of the 12th International Conference on Field and Service Robotics (FSR '19), Tokyo, Japan, 29–31 August 2019.
6. Zhang, F.; Song, J. Real-Time Calibration of Gyro-Magnetometer Misalignment. *IEEE Robot. Autom. Lett.* **2018**, *3*, 849–856. [CrossRef]
7. Bates, J.S.; Montzka, C.; Schmidt, M.; Jonard, F. Estimating Canopy Density Parameters Time-Series for Winter Wheat Using UAS Mounted LiDAR. *Remote Sens.* **2021**, *13*, 710. [CrossRef]
8. Nguyen, T.M.; Yuan, S.; Cao, M.; Lyu, Y.; Nguyen, T.H.; Xie, L. NTU VIRAL: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *Int. J. Robot. Res.* **2021**, *41*, 270–280. [CrossRef]
9. Recalde, L.F.; Guevara, B.S.; Carvajal, C.P.; Andaluz, V.H.; Varela-Aldas, J.; Gandolfo, D.C. System Identification and Nonlinear Model Predictive Control with Collision Avoidance Applied in Hexacopters UAVs. *Sensors* **2022**, *22*, 4712. [CrossRef] [PubMed]
10. Sa, I.; Kamel, M.; Khanna, R.; Popovic, M.; Nieto, J.I.; Siegwart, R.Y. *Dynamic System Identification, and Control for a Cost Effective Open-Source VTOL MAV*; Technical Report; Autonomous Systems Lab., Department of Mechanical and Process Engineering: Zurich, Switzerland, 2017.
11. Bauer, P.; Hiba, A.; Nagy, M.; Simonyi, E.; Kuna, G.I.; Kisari, A.; Drotar, I.; Zarandy, A. Encounter Risk Evaluation with a Forerunner UAV. *Remote Sens.* **2023**, *15*, 1512. [CrossRef]
12. Hiba, A.; Bauer, P.; Nagy, M.; Simonyi, E.; Kisari, A.; Kuna, G.I.; Drotar, I. Software-in-the-loop simulation of the forerunner UAV system. *IFAC-PapersOnLine* **2022**, *55*, 139–144. [CrossRef]
13. Bauer, P.; Nagy, M.; Kuna, G.I.; Kisari, A.; Simonyi, E.; Hiba, A.; Drotar, I. Stability focused evaluation and tuning ofspecial ground vehicle tracking algorithms. In Proceedings of the 22nd World Congress of the International Federation of Automatic Control (IFAC WC 2023), Yokohama, Japan, 9–14 July 2023.
14. DJI. DJI::OSDK::Control Class Reference. 2023. Available online: https://developer.dji.com/onboard-api-reference/classDJI_1_1OSDK_1_1Control.html (accessed on 21 December 2023).
15. Kroo, I.M.; Shantz, M.; Kunz, P.; Fay, G.; Cheng, S.J.; Fábián, T. *The Mesicopter: A Miniature Rotorcraft Concept Phase II Interim Reporbt*; Technical Report; Stanford University: Stanford, CA, USA, 2001.
16. Bouabdallah, S. Design and Control of Quadrotors with Application to Autonomous Flying. Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2007. [CrossRef]
17. Fogelberg, J. Navigation and Autonomous Control of a Hexacopter in Indoor Environments. Master's Thesis, Lund University, Lund, Sweden, 2013.
18. Kaya, D.; Kutay, A. Modeling, Simulation, and System Identification of a Quadrotor Helicopter. In Proceedings of the 7th Ankara International Aerospace Conference, Ankara, Turkey, 11–13 September 2013.
19. Capello, E.; Park, H.; Tavora, B.; Guglieri, G.; Romano, M. Modeling and experimental parameter identification of a multicopter via a compound pendulum test rig. In Proceedings of the 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Cancún, México, 23–25 November 2015; pp. 308–317. [CrossRef]
20. Reséndiz, V.M.A.; Rivas-Araiza, E.A. System Identification of a Quad-rotor in X Configuration from Experimental Data. *Res. Comput. Sci.* **2016**, *118*, 77–86. [CrossRef]
21. Bangura, M. Aerodynamics and Control of Quadrotors. Ph.D. Thesis, Australian National University, Canberra, Australia, 2017.
22. Ivler, C.M.; Rowe, E.S.; Martin, J.; Lopez, M.J.S.; Tischler, M.B. System Identification Guidance For Multirotor Aircraft: Dynamic Scaling and Test Techniques. In Proceedings of the Vertical Flight Society's 75th AnnualForum & Technology Display, Philadelphia, PA, USA, 13–16 May 2019.
23. Rosales, C.; Soria, C.M.; Rossomando, F.G. Identification and adaptive PID Control of a hexacopter UAV based on neural networks. *Int. J. Adapt. Control Signal Process.* **2019**, *33*, 74–91. [CrossRef]
24. Yuksek, B.; Saldiran, E.; Cetin, A.; Yeniceri, R.; Inalhan, G. System Identification and Model-Based Flight Control System Design for an Agile Maneuvering Quadrotor Platform. In Proceedings of the AIAA Scitech 2020 Forum AIAA, Orlando, FL, USA, 6–10 January 2020. [CrossRef]
25. Niemiec, R.; Ivler, C.; Gandhi, F.; Sanders, F. Multirotor electric aerial vehicle model identification with flight data with corrections to physics-based models. *CEAS Aeronaut. J.* **2022**, *13*, 575–596. [CrossRef]
26. DJI. DJI M600 Pro Hexacopter. 2017. Available online: https://www.dji.com/hu/matrice600-pro (accessed on 21 December 2023).

27. Engineering, S. SPH Engineering: UgCS Professional Drone Mission Planning Software. 2023. Available online: https://www.sphengineering.com/flight-planning/ugcs (accessed on 21 December 2023).
28. Ducard, G.; Hua, M.D. WCA: A New Efficient Nonlinear Adaptive Control Allocation for Planar Hexacopters. *IEEE Access* **2023**, *11*, 37714–37748. [CrossRef]
29. Beard, R.W.; McLain, T.W. *Small Unmanned Aircraft*; Princeton University Press: Princeton, NJ, USA, 2012.
30. Bouabdallah, S.; Siegwart, R. Full control of a quadrotor. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 153–158. [CrossRef]
31. Damaren, C.J.; Kolaei, A.; Barcelos, D.; Bramesfeld, G. Experimental Analysis of a Small-Scale Rotor at Various Inflow Angles. *Int. J. Aerosp. Eng.* **2018**, *2018*, 2560370. [CrossRef]
32. Nagy, M. Development and Simulation Testing of a Forerunner UAV System. Master's Thesis, Budapest University of Technology and Economics, Budapest, Hungary, 2021.
33. DJI. E2000 Pro Tuned Propulsion System Manual. 2016. Available online: https://dl.djicdn.com/downloads/e2000/20161220/E2000+Pro+User+Manual+V1.4.pdf (accessed on 21 December 2023).
34. DJI. DJI OSDK Flight Control Sample. 2023. Available online: https://developer.dji.com/onboard-sdk/documentation/sample-doc/flight-control.html (accessed on 21 December 2023).
35. Bauer, P.; Nagy, M.; Csurgai, S. Simulation Model Verification and Special ControlModes of DJI M600 Pro Multicopter. In Proceedings of the Submitted to the 12th International Conference on Mechatronics and Control Engineering (ICMCE 2024), Budapest, Hungary, 25–27 January 2024.
36. Onshape Free CAD Platform. Available online: https://www.onshape.com/en/products/free (accessed on 11 January 2024).
37. Hornik, V. Windguru Website. Available online: https://www.windguru.cz (accessed on 11 January 2024).
38. Morelli, E. System Identification Programs for AirCraft (SIDPAC). In Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit, Monterey, CA, USA, 5–8 August 2002. [CrossRef]