aerospace

MDPI

Article

Experimental Validation of a Stepwise Automatic Determination Method for TECS Parameters in ArduPilot Based on Steady-State Assessment

Ryoya Fukada, Kazuaki Hatanaka * and Mitsutomo Hirota

Graduate School of Engineering, Muroran Institute of Technology, Muroran 050-0071, Japan; 24042044@muroran-it.ac.jp (R.F.); hirota@muroran-it.ac.jp (M.H.)
* Correspondence: hatnac@muroran-it.ac.jp

Abstract

We propose a stepwise in-flight method for automatically determining flight-envelope-related parameters for the longitudinal control of small fixed-wing unmanned aerial vehicles (UAVs), including pitch-angle limits, maximum climb and sink rate limits, and the cruise (trim) throttle. The method performs steady-state evaluation using onboard state estimates and sequentially updates the parameter set of ArduPilot's energy-based longitudinal controller (Total Energy Control System, TECS). The algorithm was implemented in ArduPilot Plane v4.6.1 via Lua scripting, enabling real-time parameter determination and immediate application during flight. The proposed procedure was assessed in software-in-the-loop (SITL) simulations and further validated through flight experiments. The results demonstrated that the target parameters could be automatically identified during flight and implemented in real time. The proposed method is expected to reduce reliance on expert trial-and-error and contribute to improving portability across airframes and configuration changes.

Keywords: fixed-wing UAV; ArduPilot; Total Energy Control System; parameter tuning; on-board control

Academic Editor: Changzhu Wei

Received: 13 January 2026
Revised: 5 February 2026
Accepted: 14 February 2026
Published: 17 February 2026

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

In Japan, a new regulatory framework for Unmanned Aerial Vehicles (UAVs) has been implemented to enable Level-4 operations, defined as beyond-visual-line-of-sight flights over populated areas without visual observers [1]. This regulatory change has raised expectations for the use of UAVs in applications such as wide-area reconnaissance during disasters, long-range logistics, and pesticide spraying over large-scale farmland. When conducting missions that cover wide areas and extend over long durations, the choice of aircraft platform becomes a critical factor. At present, rotary-wing UAVs, typified by helicopters and multicopters, are widely deployed. Because they are capable of Vertical Take-Off and Landing (VTOL), rotary-wing UAVs do not require a runway for take-off and landing and offer excellent loitering capabilities, making them well suited for tasks requiring the focused monitoring of specific locations. Indeed, Colomina and Molina [2] reported various practical deployment cases that exploit these advantages, particularly for surveillance applications. However, as noted by Cai et al. [3], the flight principle of rotary-wing UAVs requires continuous high energy consumption to generate lift, which inherently limits flight endurance and range. Moreover, increasing the payload generally necessitates

scaling up the airframe, which increases power consumption and poses challenges from an operational cost perspective.

To address the above issues, this study considers fixed-wing UAVs that fly by utilizing lift generated by their main wings, as in conventional aircraft. Watts et al. [4] reported that, compared with rotary-wing UAVs, fixed-wing UAVs are more energy-efficient, can achieve higher speeds and longer-range flights, and can carry larger payloads. These characteristics are consistent with the mission requirements for UAV operations—including Level-4 flights—such as wide-area situational awareness over vast disaster regions (e.g., wildfires), long-distance transportation of supplies, agricultural applications over large-scale farmland, and infrastructure inspection. However, to fully leverage these advantages, the aircraft must be operated safely and conveniently even by users without specialized expertise, and an autonomous flight control system is therefore indispensable. Fixed-wing UAV operations also involve specific challenges, such as securing a runway for takeoff and landing and managing airspeed to avoid stall, although these issues are being mitigated [5]. Nevertheless, there remains a need to establish control technologies that enable stable flight performance of fixed-wing UAVs under diverse operating conditions, including for non-expert users [6].

Open-Source Software (OSS) autopilot systems have become widely used as a technological foundation for implementing autonomous flight control. A representative example is ArduPilot Plane v4.6.1 [7], which is also used in this study and has been under development since 2007 by Mackay et al. ArduPilot is built on a Linux-based architecture, offering good compatibility with embedded hardware and supporting operation on a wide range of flight controllers. Moreover, OSS autopilot systems such as ArduPilot provide broad functionality and versatility across diverse platforms—including fixed-wing, rotary-wing, and VTOL aircraft—and are therefore widely adopted in both academic research and commercial applications [8,9].

However, ArduPilot employs a conventional cascaded feedback PID control architecture. Li et al. [10] noted that this architecture requires re-tuning when aircraft dynamics change. This requirement suggests inherent limitations in achievable control accuracy under uncertainties—such as variations in aircraft weight, shifts in the center of gravity, and environmental disturbances like wind. To address this issue, the same authors [10] proposed an adaptive module that operates in parallel with the existing PID controller, presenting an approach to improve robustness against model uncertainty and disturbances. In addition, the ArduPilot development team provides the AUTOTUNE function [11] as a practical measure to mitigate this challenge. AUTOTUNE intentionally excites the vehicle in flight and automatically computes the roll- and pitch-axis PID gains from the measured response, and Matt et al. [12] demonstrated that it is highly effective for ensuring baseline stability. While these approaches are grounded in advanced control theory and can be powerful for constructing systems robust to environmental changes, they often require substantial modifications to the existing architecture and entail complex implementation.

Next, the longitudinal control law of ArduPilot, which is the focus of this study, employs the Total Energy Control System (TECS), which coordinates throttle and pitch-angle control based on the allocation of kinetic and potential energy [13]. This approach is considered useful from the viewpoints of flight efficiency and stability [14]. However, TECS parameters—such as cruise throttle, the upper limits of climb and sink rates, and pitch-angle limits—are airframe-dependent. As described in the official ArduPilot documentation [15], these parameters have traditionally relied heavily on empirical manual adjustment through flight experiments with human input. Although automation of TECS parameter adjustment has been discussed in the official ArduPilot issue tracker, it has not been updated since 2022 [16].

Based on this situation, this study aims to automatically determine TECS parameters and proposes a stepwise method to determine appropriate TECS parameters for a small fixed-wing UAV using ArduPilot. The proposed method leverages ArduPilot's Lua scripting functionality ("Lua Scripts", hereafter Lua) that enables user customization of control logic [17]. This functionality allows new behaviors to be added to the autonomous flight control system without modifying the core flight program written in C++. In particular, Lua enables flexible implementation of various processes, including acquisition of the UAV state, modification of control parameters, log output, and communication with a Ground Control Station (GCS). Moreover, because the TECS parameters to be determined are static, it is necessary to appropriately perform steady-state evaluation of the state variables in order to conduct state identification of the UAV.

In this study, baseline performance of the attitude controller for the target small fixed-wing UAV was ensured in advance by combining ArduPilot with Software-in-the-Loop (SITL) simulation [18]. Accordingly, the proposed method assumes that the UAV is capable of a certain level of autonomous flight. The TECS parameters are identified through multiple flight phases in accordance with the procedure described in the TECS for Speed and Height Tuning Guide (TECS-TG) [15], and an automated sequence that includes these phases was designed in Lua.

The proposed method is not a simple automation of the manual parameter-setting procedure described in the conventional TECS-TG. In the manual procedure, an operator selects an appropriate flight segment and determines the TECS parameters under the implicit assumption that the segment is sufficiently in a steady state. In contrast, the proposed method quantitatively assesses the steady-state condition of the aircraft state variables using the mean absolute error (MAE) and integrates Phase transitions with parameter updates. In addition, the procedure explicitly specifies retries after unsuccessful attempts and safety actions. Consequently, the proposed method is implemented as a highly reproducible parameter-determination process that does not depend on the operator's experience.

To clarify the effectiveness of the proposed method, this paper places particular emphasis on the practical applicability of the automatic sequence designed in Lua. Section 2 describes the list of TECS parameters that are automatically determined in each phase, together with the processing flow implemented in Lua. Section 3 describes the execution procedure of the designed automatic sequence, presents the SITL simulation results, and validates the robustness of the proposed method. Section 4 reports the flight-experiment results obtained with a model aircraft to examine the validity of the proposed method, and discusses its effectiveness. Finally, Section 5 concludes this study.

2. TECS Parameters and Lua Script Design

2.1. Target TECS Parameters for Automatic Determination

As described in Section 1, the Total Energy Control System (TECS) is a method that regulates the vehicle's total energy rate (i.e., the sum of the rate of change in kinetic and potential energy) using the throttle, while controlling the energy distribution rate (i.e., the exchange between these two forms of energies) using the elevator. For this control logic to function properly, the airframe-specific performance limits (envelope) of the controlled vehicle and the reference point at which the net energy balance becomes zero must be identified accurately. These quantities strongly depend on airframe-specific aerodynamic characteristics, thrust, and weight. The official ArduPilot documentation also states that appropriate settings of such airframe-dependent parameters—as addressed for automatic determination in this study—are required to fully exploit TECS performance [15]. Therefore, the automation targets in this study were selected as the most fundamental and important set of parameters that define the physical flight envelope of a given airframe.

In the proposed method, eight TECS parameters were selected as targets for automatic determination, because they define the aircraft flight envelope and the reference condition for the energy balance. To represent the operational bounds of kinetic energy under the assumption of level flight, AIRSPEED_MIN, the minimum airspeed required to avoid stall, and AIRSPEED_MAX, the maximum airspeed to prevent overspeed and structural overload during steep descents, were selected as targets. To characterize climb capability under thrust-limited conditions and the associated energy allocation, TECS_PITCH_MAX, the maximum pitch angle sustainable at maximum throttle, and TECS_CLMB_MAX, the maximum rate of climb, were selected as targets. For descent performance, TECS_PITCH_MIN, the minimum pitch angle, TECS_SINK_MIN, the minimum sink rate at minimum throttle, and TECS_SINK_MAX, the maximum sink rate under neutral throttle, were selected as targets. For TECS_PITCH_MIN and TECS_SINK_MAX, direct measurement can be difficult in actual flight and an additional safety margin is required; therefore, these parameters were not set to simple measured values but were derived dependently from other parameters based on flight-dynamics relationships described later. In addition, TRIM_THROTTLE, which TECS uses as the reference corresponding to zero net energy balance (steady, level flight), was selected as a target, resulting in a total of eight parameters. The parameter names follow the naming convention of the official ArduPilot parameter list [7]. Table 1 summarizes the target parameters and defines the symbols used hereafter.

Table 1. TECS Parameters Targeted for Automatic Determination.

Parameter	Unit	Symbol	Meaning
AIRSPEED_MIN	m/s	V_{min}	Minimum airspeed
AIRSPEED_MAX	m/s	V_{max}	Maximum airspeed
TECS_PITCH_MAX	deg.	θ_{max}	Maximum pitch angle
TECS_CLMB_MAX	m/s	$\dot{h}_{max}$	Maximum rate of climb
TECS_PITCH_MIN	deg.	θ_{min}	Minimum pitch angle
TECS_SINK_MAX	m/s	$\dot{h}_{sink}^{max}$	Maximum sink rate
TECS_SINK_MIN	m/s	$\dot{h}_{sink}^{min}$	Minimum sink rate
TRIM_THROTTLE	%	T_{trim}	Cruise throttle setting

2.2. Design Requirements for Lua Script-Based Automatic Parameter Determination

We propose a TECS-parameter automatic determination method that is implemented primarily as a single script using the Lua Scripts feature provided by ArduPilot Plane v4.6. In addition, two minor source-code modifications were introduced. First, a small modification was made to the processing for generating climb flight in the climb phase (Phase 4) by reusing the TAKEOFF logic (takeoff.cpp [19]). Second, a small modification was also made to the altitude-demand update process throughout the flight (mode.cpp [19]). The purpose of the first modification is described in detail in Section 2.4.3. The second modification resolves an issue in which the altitude demand is not updated in real time during AUTO mode when the Lua script intervenes. These modifications do not alter the energy-management algorithm in the TECS-TG or the PID-gain structure itself; they were limited to interface adjustments to ensure consistent operation with mission items and parameters that can be overwritten from Lua Scripts.

We organized the Lua script as a stepwise determination flow in which the eight TECS parameters described in Section 2.1 were determined by dividing the flight into multiple phases (Phase 1–Phase 6). Specifically, we defined a deceleration phase (Phase 1) to determine V_{min}, an acceleration phase (Phase 2) to determine V_{max}, a transition phase (Phase 3) to enter the climb phase, a climb phase (Phase 4) to determine θ_{max}, the maximum rate of climb $\dot{h}_{max}$, θ_{min}, and the maximum sink rate $\dot{h}_{sink}^{max}$ simultaneously, a descent phase

(Phase 5) to determine the minimum sink rate $\dot{h}_{sink}^{min}$, and a steady level-flight phase (Phase 6) to determine T_{trim}. Figure 1 shows the correspondence between these flight phases and the TECS parameters described in Section 2.1 By explicitly separating the flight into phases as in Figure 1, we implemented the manual procedure based on the TECS-TG in the Lua script as independent processing blocks for each flight condition, which enabled sequential execution during flight.

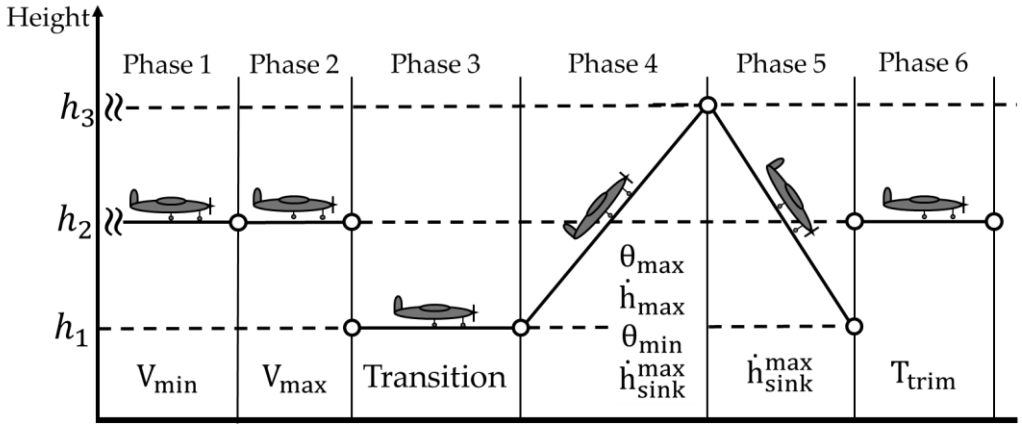


Figure 1. Multiple flight phases in the automatic sequence for determining TECS parameters.

As described in Section 1, we implemented the designed script as a periodically executed task running in parallel with the ArduPilot core. At each cycle, the script acquired the UAV state and mission progress, evaluated the phase-transition conditions, and performed automatic determination of TECS parameters. In particular, we used the API functions provided by the Lua Scripts feature [17] to obtain the current airspeed, longitudinal acceleration, rate of climb, altitude, and throttle percentage. When the temporal variations in these state estimates became sufficiently small and the flight was judged to be in a steady state, we determined the TECS parameters associated with the current phase. We used the mean absolute error (MAE) as the primary metric for steady-state evaluation; if the flight was judged not to have reached a steady state, we discarded the data collected in that phase and repeated the same flight condition. This design aimed to make the parameter determination less sensitive to environmental disturbances such as wind and to transient manual inputs, and the specific steady-state metrics and threshold settings based on MAE are described later in Section 2.3. We also designed the script to operate in close coordination with mission items in the flight plan, including loitering based on Way Point (WP) commands and climb flight using the TAKEOFF command; in particular, for Phase 1 and Phase 2, which determine V_{min} and V_{max}, and for Phase 6, which determines T_{trim}, we predefined the mission indices corresponding to the straight and turning segments in the flight plan.

We designed the script execution trigger as an explicit switching mechanism using an auxiliary switch on the RC transmitter, referred to as a "Proportional controller", as shown in Figure 2. The designed Lua script started at the instant when the operator changed the switch position, and we configured it to stop the script at any other position. When the sequence is stopped, the current Phase and the number of attempts were saved to enable resumption. This design was intended to allow the automatic determination of TECS parameters to be continuously progressed across multiple flights even when all phases cannot be completed within a single flight. Note that the safety design, such as the return conditions to a holding pattern and altitude constraints, is described in Section 2.5.

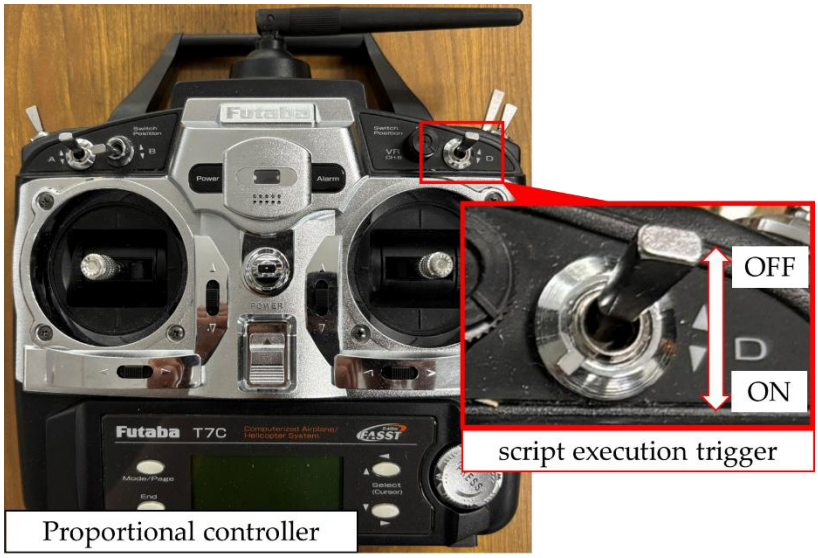


Figure 2. Script execution trigger using the RC transmitter auxiliary switch.

As summarized above, we designed the Lua script in this study to comprise the following elements:

1. Minor source-code modifications to support coordination with the mission plan and the TAKEOFF command;
2. Flight-phase partitioning and a stepwise determination flow based on the TECS-TG;
3. Real-time acquisition of UAV state estimates and their steady-state evaluation;
4. An explicit execution trigger using an auxiliary RC-transmitter switch, together with functions that allow interruption and resumption.

In the following subsections, for clarity, we first describe the details of the steady-state evaluation method and then present the TECS-parameter determination method for each phase.

2.3. State Variables and Evaluation Metrics for Steady-State Evaluation

TECS is typically designed and analyzed under the assumption of quasi-steady flight near a trim condition [13–15]. As described in the Introduction, the TECS tuning guide (TECS-TG) [15] recommends setting quantities such as the maximum rate of climb, the minimum sink rate, and a safe stall-limit airspeed using FBWA mode (a semi-autonomous mode that maintains level flight) and Loiter mode (an autonomous loitering mode). This recommendation is based on the premise that these values can be measured while reproducing constant-speed, steady climb, and steady glide conditions. Such conventional approaches can therefore be interpreted as relying on an implicit assumption that the target parameters are obtained from time intervals in which the state variables can be regarded as constant under a specific flight condition.

Therefore, in the Lua script that we designed to realize the proposed method, we incorporated the above concept as explicit logic. Our basic policy was to automatically detect, onboard, a time interval that can be regarded as "steady-state" in each phase and then automatically determine the TECS parameters based on that interval. In this subsection, we define, in a general form, the state variables used for this purpose and the evaluation metric based on the mean absolute error (MAE). We adopted MAE because, compared with variance-based measures or squared-error metrics, it is less sensitive to outliers and provides an intuitive measure of the magnitude of deviation within a time window. In particular, in the fields of process control and time-series analysis, MAE is widely used as one of the indices for control performance evaluation [20].

First, we define the state vector $\mathbf{x}(t)$, which summarizes the state variables used for steady-state evaluation—airspeed V, longitudinal acceleration $\dot{V}$, rate of climb $\dot{h}$, and altitude h—as expressed in Equation (1).

$$\mathbf{x}(t) = \begin{bmatrix} V(t) & \dot{V}(t) & \dot{h}(t) & h(t) \end{bmatrix}^T \tag{1}$$

* Here, the superscript $[\,]^T$ denotes the transpose of a vector/matrix.

We select the components to be evaluated from Equation (1) for each phase according to the physical meaning of the TECS parameter to be determined. Similarly, we define the target values of the required state variables in each phase as a target-value vector, as shown in Equation (2).

$$\mathbf{x^{ref}} = \begin{bmatrix} V^{ref} & \dot{V}^{ref} & \dot{h}^{ref} & h^{ref} \end{bmatrix}^T \tag{2}$$

Although pitch angle θ and throttle T are not included in the steady-state evaluation variables, we also monitor them. Let V_{cruise} denote an arbitrarily specified cruise airspeed; examples of reference conditions used for each flight condition are given as follows. In accordance with TECS-TG [15], the meanings of the subscripts of throttle T are the same as those in Equation (2).

- Steady level-flight condition: $V^{ref} = V_{cruise}$, $\dot{V}^{ref} = 0$, $\dot{h}^{ref} = 0$;
- Steady gliding-flight condition: $V^{ref} = V_{cruise}$, $T^{ref} = T_{min}$, $\theta(t) < 0$.

As described above, within the designed Lua script, we perform steady-state evaluation using only the required components in Equation (1). In addition, in this study, we treat throttle T as a rate scaled within ArduPilot.

Next, to judge steady-state flight onboard, we use an evaluation metric based on a finite-length time window. Let the sampling interval be Δt and the evaluation window length be T_{obs}; then the number of samples per window, N, is given by Equation (3).

$$N = \frac{T_{obs}}{\Delta t} \tag{3}$$

Let the window start time be t_0; the state variables at discrete time k are expressed by Equation (4).

$$\mathbf{x}[k] = \mathbf{x}(t_0 + k\Delta t) \tag{4}$$

In addition, we define the error between the state variables at discrete time k and their target values as in Equation (5).

$$\mathbf{e}[k] = \mathbf{x}[k] - \mathbf{x}^{ref} \tag{5}$$

Accordingly, denoting the i-th component of **e**[k] in Equation (5) as $e_i[k]$ and using N given by Equation (3), we define the MAE of each component as in Equation (6).

$$J_i = \frac{1}{N}\sum_{k=0}^{N-1} |e_i[k]| \tag{6}$$

We then define the allowable MAE threshold for each component as in Equation (7).

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_V & \varepsilon_{\dot{V}} & \varepsilon_{\dot{h}} & \varepsilon_h \end{bmatrix}^T \tag{7}$$

Finally, we regard the flight segment as steady-state when the condition in Equation (8) is satisfied.

$$J_i \leq \varepsilon_i \tag{8}$$

In addition, ε_i is independent of the Phase and is a fixed value assigned to each state variable, computed from prior experimental data obtained in a steady level-flight segment.

With the steady-state evaluation logic described above, we judged whether the data obtained in each phase were sufficiently steady. The specific combinations of observed variables and the assignment of reference values in each phase are described in Section 3 and subsequent sections.

2.4. TECS Parameter Automatic Determination Method in Each Phase

In this subsection, we describe the Lua script–based automatic sequence designed according to the requirements in Section 2.2 and the steady-state metrics in Section 2.3. When the script trigger switch is turned on, the sequence starts and automatically determines TECS parameters while transitioning from Phase 1 to Phase 6.

Phase transitions are controlled by waypoint (WP) operations in a GCS-predefined flight plan, executed from within the Lua script. If the steady-state condition is not achieved within the prescribed time window or altitude margin, the UAV returns to a safe holding pattern, repeats the current phase, or forcibly transitions to the next phase. These behaviors keep the UAV within a safe flight envelope under gusts or timeouts. The safe-state definition, holding return maneuver, and retry conditions are described as safety design in Section 2.5.

In the following subsubsections, we describe the role of each phase, the steady-state–based determination method, and its correspondence with the TECS-TG.

2.4.1. Phase 1: Determination of the Minimum Airspeed

In Phase 1, we determine the minimum airspeed V_{min}, which specifies the lowest value that does not lead to stall [15]. As the variables for steady-state evaluation, we select airspeed $V(t)$, longitudinal acceleration $\dot{V}(t)$, and rate of climb $\dot{h}(t)$. Next, letting the number of deceleration steps be N_{dec}, we define the deceleration airspeed table $V_{dec}^{N_{dec}}$ designed in advance as shown in Equation (9).

$$V_{dec}^{N_{dec}} = \left\{ V_{dec}^{(1)}, V_{dec}^{(2)}, \ldots, V_{dec}^{(l)} \right\} \left(N_{dec} = 1, 2, \ldots, l \right) \tag{9}$$

Reference airspeed V^{ref} was decreased stepwise in accordance with Equation (9). When the steady-state condition in Equation (8) was satisfied, the state was judged to be steady and N_{dec} was updated to $N_{dec} + 1$. Here, the time at which the evaluation started was defined as t_0, and the forced termination time was defined as t_f $(t_0 + \Delta t_{out})$, where $\Delta t_{out} > \Delta t$ was set to provide a sufficient margin for performing the steady-state evaluation described in Section 2.3.

Finally, for any deceleration step N_{dec} for which a steady state was not judged within the measurement interval $t \in [t_0, t_f]$, $V_{dec}^{N_{dec}-1}$ was determined as the minimum airspeed V_{min}. The same procedure was applied when N_{dec} reached the maximum deceleration step l. The processing flow of Phase 1 is shown in Figure 3. In each processing step, ArduPilot controlled the aircraft to maintain level flight while tracking the cruise airspeed specified by the deceleration airspeed table.

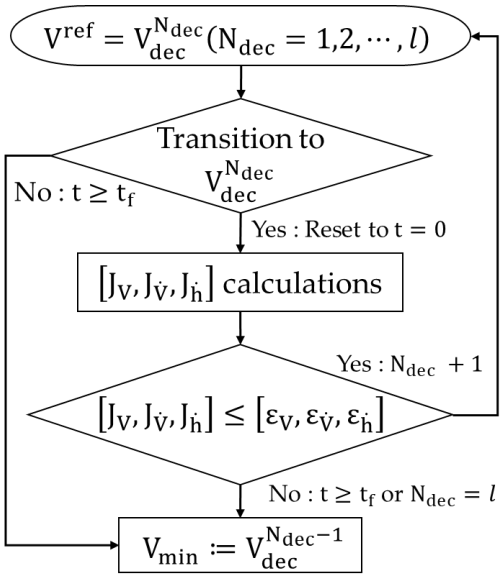


Figure 3. Phase 1 Processing Flowchart.

2.4.2. Phase 2: Determination of the Maximum Airspeed

In Phase 2, we determine the maximum airspeed V_{max} to preserve the airframe structure and suppress overspeed associated with steep descents [15]. In this phase as well, the state variables selected for steady-state evaluation are the same as those in Phase 1, namely airspeed V, longitudinal acceleration $\dot{V}(t)$, and rate of climb $\dot{h}$. Let the number of acceleration trials be N_{acc}; then, the predesigned acceleration airspeed table $V_{acc}^{N_{acc}}$ is defined as shown in Equation (10).

$$V_{acc}^{N_{acc}} = \left\{ V_{acc}^{(1)}, V_{acc}^{(2)}, \ldots, V_{acc}^{(m)} \right\} (N_{acc} = 1, 2, \ldots, m) \quad (10)$$

In contrast to Phase 1, the reference airspeed V^{ref} is increased stepwise in accordance with Equation (10). When Equation (8) is satisfied, the flight condition is judged to be steady, and N_{acc} is updated to $N_{acc} + 1$. The subsequent procedure follows the same logic as Phase 1; therefore, the detailed description is omitted here. The processing flow of Phase 2 is shown in Figure 4. In each processing step, ArduPilot is controlled to maintain level flight while tracking the cruise airspeed specified by the acceleration airspeed table.

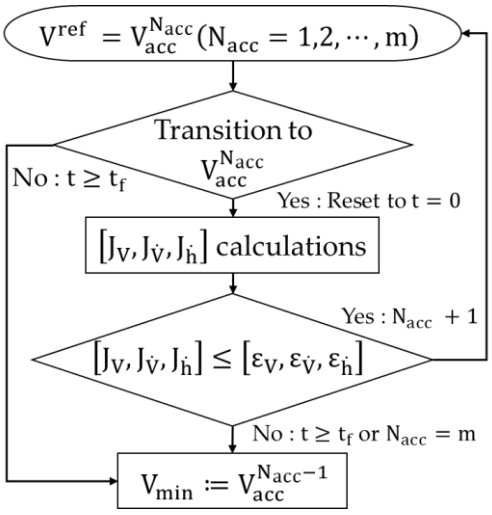


Figure 4. Phase 2 Processing Flowchart.

2.4.3. Pre-Climb Transition and Determination of the Maximum Rate of Climb, Pitch-Angle Limits, and the Maximum Sink Rate

Phase 3 is an acceleration phase performed before the climb-performance evaluation in Phase 4. Its purpose is to suppress stall immediately after the transition to climbing flight. Assuming that the target UAV maintains level flight, the script ensures that the airspeed just before rotation reaches the target rotation speed. Here, letting the target rotation speed with respect to the current airspeed $V(t)$ be V_R, the termination condition of Phase 3 is satisfied when the relationship in Equation (11) holds.

$$V(t) \geq V_R \tag{11}$$

Phase 4 determines the following parameters simultaneously based on the climb performance under the application of the maximum throttle, as described in [15].

- Maximum rate of climb: $\dot{h}_{max}$;
- Maximum pitch angle: θ_{max};
- Minimum pitch angle: θ_{min};
- Maximum sink rate: $\dot{h}_{sink}^{max}$.

In the TECS-TG [15], $\dot{h}_{max}$ and θ_{max} are defined as follows.

- $\dot{h}_{max}$: the maximum rate of climb that an aircraft can achieve while flying at the target airspeed with the throttle at its maximum setting;
- θ_{max}: the pitch angle at which an aircraft can climb while maintaining the target airspeed with the throttle at its maximum setting.

From these definitions, it is evident that $\dot{h}_{max}$ and θ_{max} are in a trade-off relationship. Related difficulties in deriving the maximum rate of climb have been addressed using optimization-based approaches [21]. However, because ArduPilot is intended to be model-free, it is difficult to perform optimization calculations derived from the equations of motion. Therefore, we designed logic that allows the aircraft to continue climbing while maintaining the target airspeed and searches for θ_{max}.

First, we defined θ_{R0} as the initial climb pitch angle as well as the maximum pitch angle during climb. After the aircraft completed rotation, we designed the target climb pitch angle to be updated dynamically according to a simple scheduling law proportional to airspeed. Here, the scheduling law designed using the target airspeed V^{ref} and the climb pitch-angle command θ_{cmd} is defined as Equation (12).

$$\theta_{cmd}(t) = \theta_{R0} \cdot \frac{V(t)}{V^{ref}} (\theta_{min,climb} \leq \theta_{cmd}(t) \leq \theta_{R0}) \tag{12}$$

Note that Equation (12) corresponds to a safety mechanism in ArduPilot's existing TAKE-OFF function, which acts when the aircraft has not reached the prescribed airspeed before rotation. In this study, we configured this mechanism to operate continuously. In addition, $\theta_{min,climb}$ is the lower bound of the climb pitch angle, which was set to avoid an excessive reduction in stall margin. As indicated by Equation (12), when the airspeed $V(t)$ is smaller than the target airspeed V^{ref}, the pitch-angle command θ_{cmd} is reduced below θ_{R0}; this prevents an excessive angle of attack under thrust-limited conditions and assists airspeed recovery during climb.

Next, in this phase, we select airspeed $V(t)$ and longitudinal acceleration $\dot{V}(t)$ as the variables for steady-state evaluation. During the climb, the pitch angle is updated dynamically in accordance with Equation (12); when Equation (8) is satisfied, the flight condition is judged to be steady, and the current rate of climb $\dot{h}(t)$ is determined as the maximum rate of climb $\dot{h}_{max}$, while the pitch angle $\theta(t)$ is determined as the maximum pitch

angle θ_{max}. Note that the method for combining $\dot{h}_{max}$ and θ_{max} was not the maximization of an objective function. A target airspeed was set as a constraint condition, and a boundary search was adopted to determine the allowable θ_{max} based on a feasibility judgement of whether the target airspeed could be maintained. Physically, maintaining the airspeed becomes difficult in proportion to an increase in the pitch angle; therefore, the feasibility judgement tends to become monotonically stricter. Accordingly, this search differs in nature from an optimization problem that converges to a local optimum. On the other hand, the feasibility judgement (steady-state judgement) may fluctuate due to transient responses and measurement errors. For this reason, as described in Section 2.3, we reduced the influence of misjudgements by the judgement based on the steady-state evaluation time window T_{obs} and by retries.

Furthermore, as described above, we also determined the minimum pitch angle θ_{min} and the maximum sink rate $\dot{h}_{sink}^{max}$. First, we derived θ_{min} in a simple manner from the geometric relationship with the maximum pitch angle θ_{max} determined above, as expressed in Equation (13).

$$\theta_{min} := \left(\theta_{max} - \Delta\theta_{margin}\right) \tag{13}$$

Here, $\Delta\theta_{margin}$ is a constant pitch-angle safety margin; in this study, we set it to approximately 5 deg. When tuning the pitch axis using the AUTOTUNE function [11], positive and negative pitch-rate commands are applied abruptly. If the absolute values of θ_{max} and θ_{min} are identical, the aircraft is expected to gradually lose altitude under the effect of gravity. Therefore, we adopted a practical determination method for θ_{min} that does not strictly follow the TECS-TG definition [15] represented by Equation (13). In contrast, for $\dot{h}_{sink}^{max}$, we derived it from the relationship between the sink rate and the flight-path angle γ. In general flight-performance analysis [22], the desired $\dot{h}_{sink}^{max}$ and γ are expressed as Equations (14) and (15).

$$\dot{h}_{sink}^{max} = V_{max} \cdot \sin\gamma \tag{14}$$

$$\gamma = \theta - \alpha \tag{15}$$

Note that V_{max} in Equation (14) was the value determined in Phase 2, and α in Equation (15) denotes the angle of attack. In addition, because we did not command a descent at the maximum airspeed in the flight experiments, α was expected to be on the order of a few degrees; therefore, we assumed α = 0 deg. [23]. Consequently, the maximum sink rate $\dot{h}_{sink}^{max}$ derived in this study is expressed as Equation (16).

$$\dot{h}_{sink}^{max} := V_{max} \cdot \sin\theta_{min} \tag{16}$$

Finally, the processing flow of Phases 3 and 4 is shown in Figure 5. In Phase 3, ArduPilot is controlled to maintain level flight while tracking the specified cruise airspeed. In Phase 4, variations in roll angle are assumed to be small, and pitch-angle control is assumed to provide sufficient tracking performance, such that the climb maneuver is executed as intended.

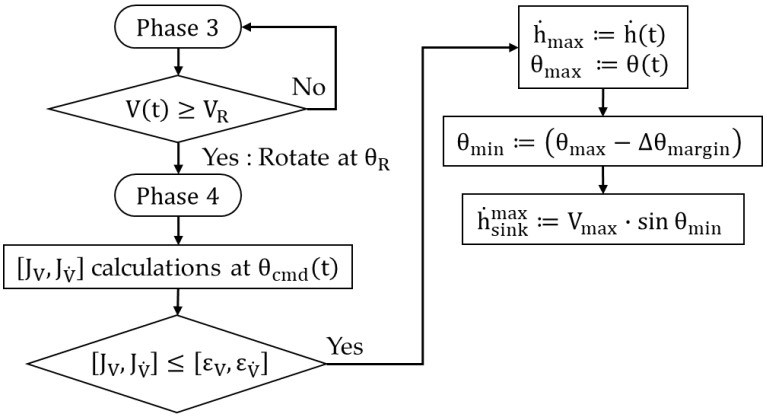


Figure 5. Phase 3&4 Processing Flowchart.

2.4.4. Phase 5: Determination of the Minimum Sink Rate

In Phase 5, we determine the minimum sink rate $\dot{h}_{sink}^{min}$ from the gliding performance under the application of the minimum throttle command. First, starting from the altitude increased in Phase 4, we transition to a glide while fixing the commanded throttle to its minimum and detect a steady gliding condition in which the airspeed and the sink rate are sufficiently steady. At this time, to ignore altitude error and prioritize airspeed control, we change TECS's speed-weight parameter TECS_SPDWEIGHT (ranges from 0 to 2) to a "speed-prioritized" value different from that used during cruise. This parameter controls the relative weighting between altitude and airspeed: when it is 1, TECS controls altitude and airspeed with equal weighting, whereas when it is 2, only the airspeed error is considered in the feedback control. Letting the Phase 5 start time be t_{glide}, we give the switching rule of TECS_SPDWEIGHT (TSWT) in Equation (17).

$$\mathrm{TSWT}(t) = \begin{cases} 1 & (t < t_{glide}) \\ 2 & (t \geq t_{glide}) \end{cases} \tag{17}$$

Equation (15) allows the pitch angle to be controlled even during a glide with the throttle rate fixed at its minimum, such that the airspeed is maintained constant and only the sink rate can be evaluated.

Following the above procedure, we selected the airspeed $V(t)$ as the variable for steady-state evaluation. When Equation (8) was satisfied, we determined the current sink rate $\dot{h}_{sink}(t)$ as the minimum sink rate $\dot{h}_{sink}^{min}$. The processing flow of Phase 5 is shown in Figure 6.

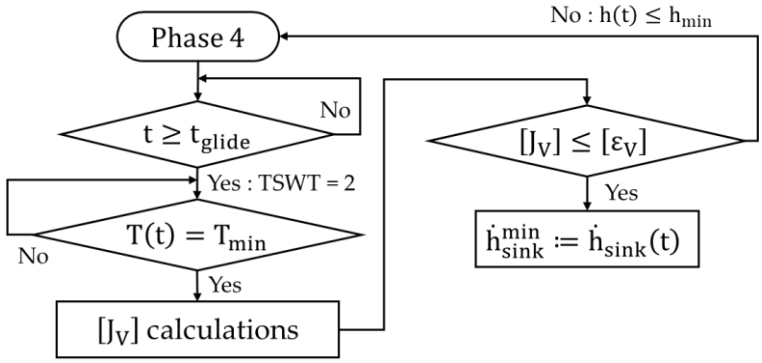


Figure 6. Phase 5 Processing Flowchart.

2.4.5. Phase 6: Determination of the Cruise (Trim) Throttle

In Phase 6, we determine the trim throttle rate T_{trim} under steady level flight. Using V_{min}, V_{max}, $\dot{h}_{max}$, and $\dot{h}_{sink}^{max}$ determined in the preceding phases, we set a safe cruise-flight condition. The UAV then performs steady level flight while maintaining the target airspeed V^{ref}. Next, we select airspeed $V(t)$, longitudinal acceleration $\dot{V}(t)$, rate of climb $\dot{h}(t)$, and altitude $h(t)$ for steady-state evaluation. As in the previous phases, we determine the throttle rate T(t) that satisfies Equation (8) for the selected state variables as the trim throttle rate T_{trim}. T_{trim} functions as the reference point for the zero net energy balance in the TECS algorithm. In practical flights, when airspeed estimation using a pitot tube cannot be performed or when sensor information from the inertial sensor (IMU) cannot be obtained, speed control is performed with reference to T_{trim}. Therefore, determining T_{trim} is expected to contribute to improving the overall safety of autonomous flight. Here, the processing flow of Phase 6 is shown in Figure 7.

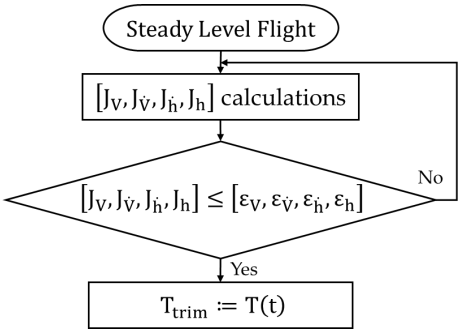


Figure 7. Phase 6 Processing Flowchart.

2.4.6. Correspondence Between the TECS-TG Procedure and the Phase-Based Procedure

Table 2 shows the correspondence between the contents of the TECS-TG, which is publicly available as a manual procedure for setting TECS parameters, and each Phase procedure implemented as described in Sections 2.4.1–2.4.5. As described in Section 2.1, some setup items addressed in the TECS-TG are outside the scope of automatic determination in this study; therefore, we omit them from Table 2.

Table 2. Correspondence between the TECS-TG manual tuning steps and the proposed phase-based procedure.

Parameter	Unit	Key Point in TECS-TG Manual Procedure	Corresponding Phase
V_{min}	m/s	Set as the minimum safe airspeed to avoid stall in level flight.	1
V_{max}	m/s	Set slightly below the maximum level-flight speed at maximum throttle.	2
θ_{max}	deg.	Adjust as the maximum pitch in climb while maintaining adequate airspeed.	4
$\dot{h}_{max}$	m/s	Measure and set as the best steady climb rate at maximum throttle.	4
θ_{min}	deg.	Adjust as the minimum pitch in descent without overspeeding.	4
$\dot{h}_{sink}^{max}$	m/s	Set as a sink rate that remains feasible without overspeed.	4
$\dot{h}_{sink}^{min}$	m/s	Measure and set as the steady sink rate in a minimum-throttle glide.	5
T_{trim}	%	Adjust and set as the throttle for level flight near the cruise speed.	6

2.5. Safety Design and Abnormal-Condition Recovery Logic

The proposed method ensured in-flight safety by combining an operator-initiated interruption operation with automatic recovery actions that assume abnormalities during Phase execution. In this subsection, we describe the definition of safe states, the procedures for interruption and resumption, the abnormal-condition recovery logic, and the parameter cleanup process.

First, the safe state defined in this study was defined, as described in the Introduction, as a state in which the attitude-angle controller performance was ensured in advance by using ArduPilot in conjunction with the SITL simulation [18], and a state in which the aircraft stably continued orbiting flight (holding orbit) in ArduPilot AUTO mode along the Way Point (WP) / Turn Point (TP) specified in the Flight Plan. Note that WP was defined, in general, as a point that an aircraft passes through for setting a flight route. In contrast, TP was implemented by our team. TP is a point that has the center coordinates, turn radius, and turn direction as parameters, and the circumference specified by the TP becomes the target path. The designed Lua script maintained the stability of the aircraft attitude and flight path by returning to this holding orbit. In addition, it was designed to operate only in AUTO mode.

Next, we describe the interruption and resumption by the user. As stated at the end of Section 2.2, we designed the Lua script so that it can be started and stopped using an RC switch. When the switch is turned OFF, the script saves the current Phase number and stops, and we designed it so that it resumes from the stopped Phase when the switch is turned ON the next time. For processes that have the number of attempts within a Phase, such as Phase 1, Phase 2, and Phase 4, we also designed the script to save the number of attempts so that the search can be continued under the same conditions upon resumption.

As the recovery logic for abnormal conditions, when a steady state is not obtained during the execution of each Phase (e.g., a timeout process), or when there is a possibility of violating safety constraints (e.g., the upper limit of climb altitude in Phase 4), the current Phase is interrupted and the aircraft is returned to the holding orbit. As an example, in Phase 5, when the minimum sink rate cannot be determined despite reaching the lower altitude limit, the glide is interrupted once with safety prioritized. After that, we incorporated a process to transition to the holding orbit, recover altitude, and then perform a retry. In this manner, under abnormal conditions, we adopted the sequence of interruption, stabilization in the holding orbit, recovery of required conditions, and retry as the basic procedure.

In addition, in the proposed method, we temporarily rewrote multiple TECS-related parameters and throttle-related parameters for Phase execution. When the script is stopped, a cleanup process is always executed to return these parameters to safety-side values. Specifically, the switch is turned OFF during the progress of an arbitrary Phase. At this time, after saving the current state, the process to return the parameters to the safety side is executed, and we designed it so that the aircraft can reliably return to the normal AUTO-mode flight without intervention by the script.

Furthermore, to ensure the reproducibility of this method, we make the Lua script that implements the proposed method and the related source-code modifications described in Section 2.2 publicly available in a GitHub repository (see the Data Availability Statement). We also provide a manual in Markdown format that describes the design intent and the setup procedure of the Lua script. Finally, Figure 8 shows the overall processing flow, including the Lua-script execution trigger, Phase progression, and abnormal-condition recovery.

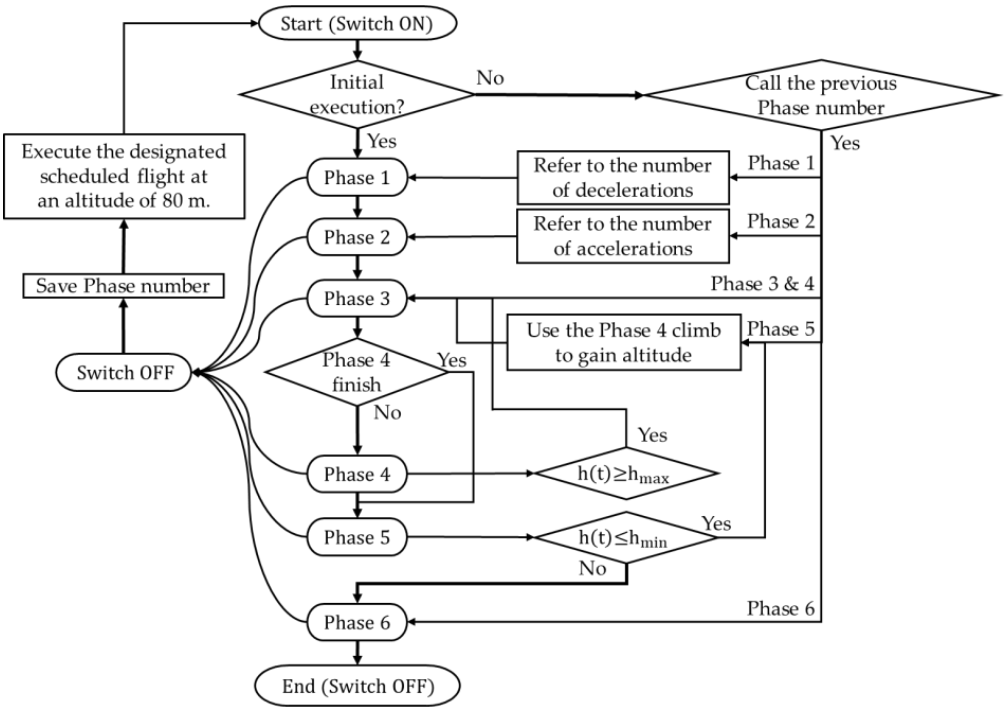


Figure 8. Overall processing flow of the proposed method, including trigger, Phase progression, and recovery.

3. Verification of the Proposed Method Using Software-in-the-Loop (SITL) Simulation

In this section, we verify the operation of the Lua-script-based TECS parameter automatic determination algorithm designed in Section 2 by using the Software-in-the-Loop (SITL) simulation environment of ArduPilot Plane v4.6.1 [7,18,19].

Software-in-the-Loop (SITL) simulation enables the same autopilot software as that installed on the real aircraft to be built and executed on a host PC. In this environment, sensor outputs and actuator commands are exchanged between the running software and a virtual aircraft model, allowing evaluation of an automatic flight control system without using a real aircraft [8]. As the flight simulator (FS) for running the virtual aircraft model, we selected XPlane-12 [24], and we selected Mission Planner (MP) as the ground control station (GCS) for creating the flight plan [25]. This configuration provides the advantage that the functions used in this study—such as parameter overwriting via Lua Scripts and mission control within the flight plan—can be verified under system conditions equivalent to those of real flight. For both SITL and real-flight firmware, we used ArduPilot Plane 4.6.1. In addition, based on the ArduPilot-4.6 branch on GitHub [19], we applied only the minor source-code modifications described in Section 2.2. Consequently, the longitudinal control system operates using the same TECS implementation as in real flight [14,15], and TECS parameters overwritten by the Lua script are reflected in ArduPilot immediately. The logging rate and the main-loop frequency were also set to be identical to those in real flight, enabling steady-state evaluation based on the sampling interval Δt and evaluation window length T_{obs} defined in Section 2.3. Here, Figure 9 shows the relationship among the flight controller (FC) running ArduPilot in SITL, the GCS, and the FS that runs the virtual aircraft model.

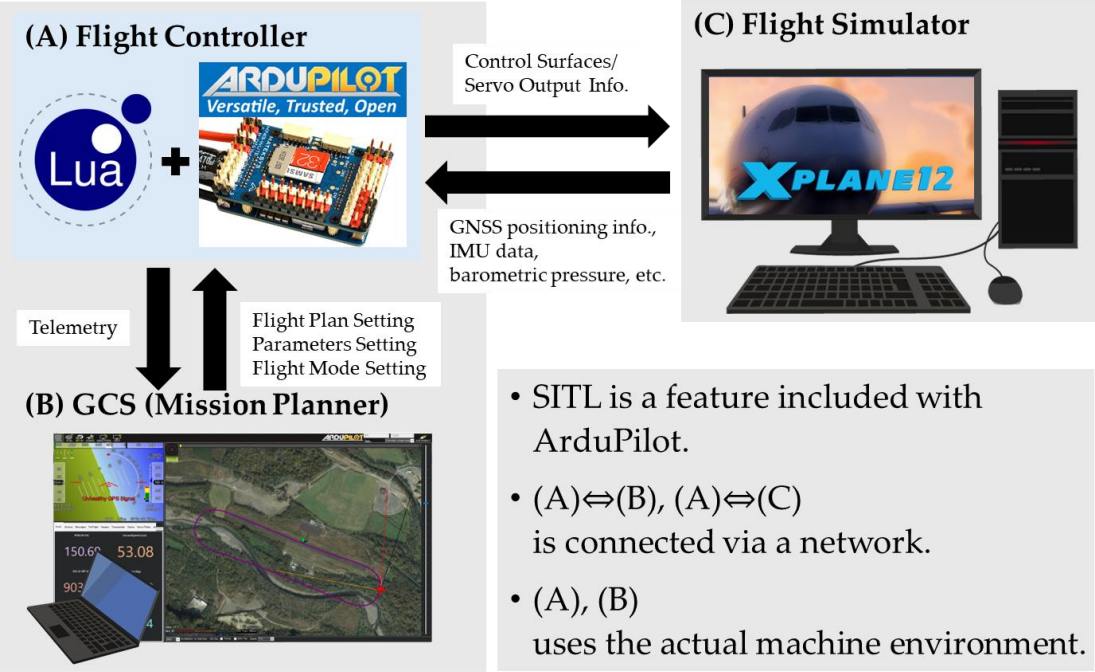


Figure 9. Relationship with FC, GCS, and FS on the SITL.

3.1. Target Aircraft

The effectiveness of the proposed method was verified using the piston-engine-powered, propeller-driven model aircraft shown in Figure 10, whose maximum level-flight speed is approximately 30 m/s (Kyosho Calmato 40 Sports). This aircraft has a low-wing configuration with tricycle landing gear and is equipped with an FC and various sensors, including a differential-pressure airspeed sensor, an acceleration sensor, a gyroscope, a compass, a GNSS sensor, and a barometric altitude sensor. In addition, as shown in Figure 10, we also created an XPlane-12 model of the aircraft for use in the SITL simulation. Note that the aircraft model used in the SITL simulation was obtained from a publicly available file distributed via the official X-Plane forum [26]. Therefore, the differences between the aircraft in the simulation and the flight experiment are the presence or absence of the FC and various sensors.

In the simulation, the model mass was set so that the all-up weight matched that in the flight experiment. Figure 11 shows the center-of-gravity (CG) location range and the three-view drawing of the target aircraft, and Table 3 shows the main aircraft specifications and characteristics. For the set of characteristic values in Table 3, we estimated the allowable CG range and the moments of inertia (I_{xx}, I_{yy}, I_{zz}) to clarify the dynamic characteristics of the aircraft used in the simulation and the flight demonstration. The center-of-gravity (CG) location was managed as the distance from the main-wing leading edge, expressed as a percentage of the mean aerodynamic chord (MAC%), and was set within a range that allows stable flight. The moments of inertia were estimated using a geometric model based on the aircraft geometry and mass distribution. Specifically, we assumed that 80% of the all-up weight was uniformly distributed in the fuselage and 20% was uniformly distributed in the main wing; the fuselage was modeled as a homogeneous rectangular prism, and the main wing was modeled as a homogeneous rectangular plate with a dihedral angle. In calculating the moments of inertia, we approximated that the centroid of each component coincides with the CG of the entire aircraft and summed the moments of inertia about each axis. In addition, the maximum and minimum load factors were presented as physical constraints of the aircraft, with reference to the values in the U.S. Federal Aviation Regulations (14 CFR Part 23.337) [27].

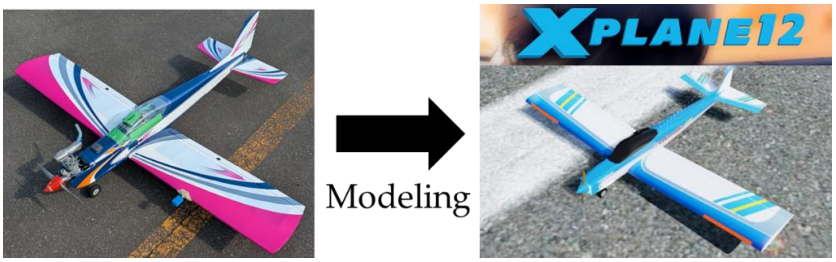


Figure 10. The experimental appearance of the model aircraft and its modeled appearance in XPlane-12.

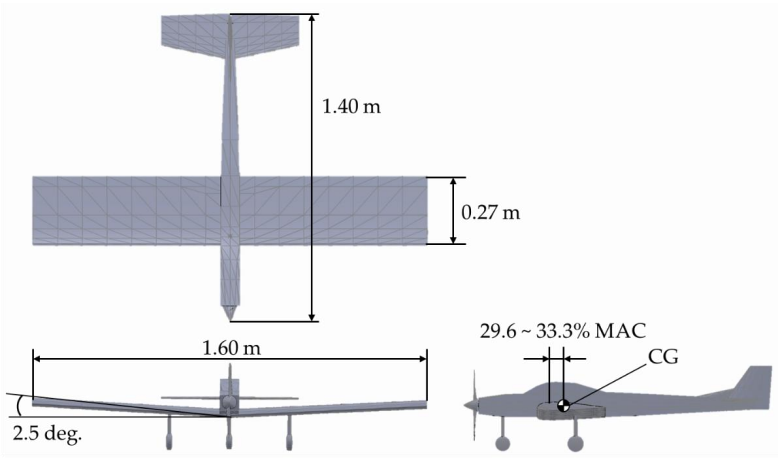


Figure 11. Three-view diagram of the target Aircraft.

Table 3. Specifications and Performance Values of the Target Aircraft.

Category	Parameter	Unit	Value
Specifications	Name	-	Kyosho Calmato α40 Sports
	Dry weight	kg	2.45
	Full weight	kg	3.19
	Overall length	m	1.40
	Wingspan	m	1.60
	Chord length	m	0.270
	Dihedral angle	deg.	2.5
	Wing loading	kg/m^2	5.57~5.79
	Aspect Ratio (AR)	-	5.81
	Maximum deflection of aileron/elevator/rudder	deg.	24
Characteristic values	CG	% MAC	29.6~33.3
	I_{xx}	kg·m^2	0.144
	I_{yy}	kg·m^2	0.426
	I_{zz}	kg·m^2	0.560
	$n_{z,\ max}$	G	3.8 [27]
	$n_{z,\ min}$	G	−1.52 [27]

3.2. Simulation Conditions

3.2.1. SITL Implementation of the Lua Script and Flight Plan

We implemented the Lua script for automatic determination of TECS parameters as described in Section 2.2 and loaded it into the SITL environment using ArduPilot's Lua Scripts feature. The script was executed in parallel with the flight controller at a period of 20 ms (50 Hz). In addition, to emulate an RC transmitter on the host PC for simulation, we used a software tool called vJoy v2.1.8 [28]. We then configured Mission Planner (MP),

used as the ground control station (GCS), to recognize this virtual RC input and assigned it to the script execution trigger.

The flight plan was configured to have the same structure as that used in the real-flight experiments, so that identical Phase transition conditions could be realized in both SITL and flight experiments. The coordinates were also matched to the test site, Shiraoi Gliding Field in Hokkaido (42° 32′ 12.7″ N, 141° 15′ 17.5″ E). Figure 12 shows the flight plan used in this study. In the flight plan shown in Figure 12a, TP1 and TP2 are set, and autonomous flight is conducted by tracking a closed circuit composed of the two TP circles connected by their common tangents. Based on this configuration, the Lua script operates in coordination with the flight plan in Figure 12. First, as shown in Figure 12a, the aircraft loiters through TP1 and TP2, and Phases 1–2 proceed when the script is triggered at an arbitrary timing. After Phase 2 is completed, the aircraft heads to WP1 and transitions to the flight segment shown in Figure 12b. After passing WP2, Phases 3–4 proceed. After Phase 4 is completed, Phase 5 proceeds as shown in Figure 12c; once it is completed, the aircraft transitions to the nominal loitering flight shown in Figure 12d, and then Phase 6 proceeds. Note that Figure 12a,d use essentially the same flight plan, which enables reuse of the plan across different Phases. Finally, when Phase 6 is completed, a completion message is sent to Mission Planner (MP), and the operator turns the trigger switch off after confirming the message. Figure 13 shows the exterior of Shiraoi Gliding Field, and Table 4 lists the coordinates and altitude of each WP and TP; the ordering in Table 4 generally follows the sequence in which the aircraft passes them.

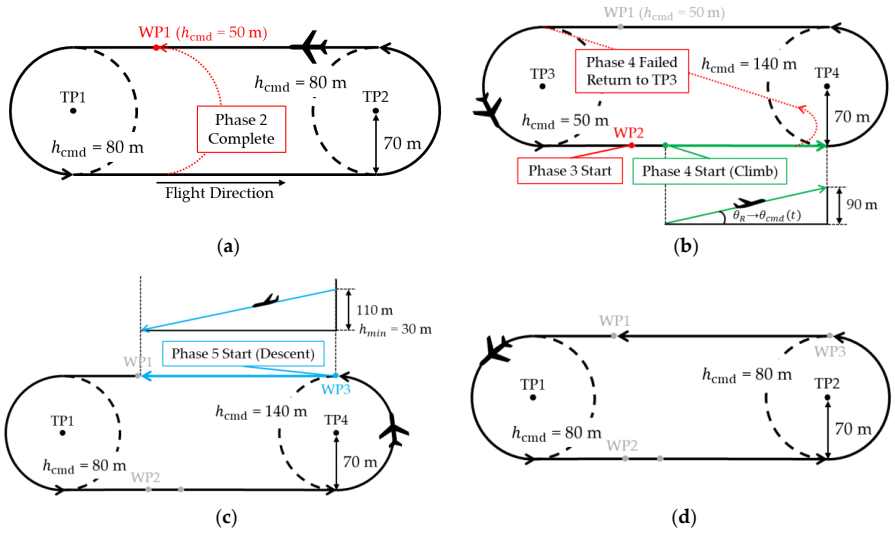


Figure 12. Flight plans used in the automatic sequence: (**a**) Phases 1–2; (**b**) Phases 3–4; (**c**) Phase 5; (**d**) Phase 6.

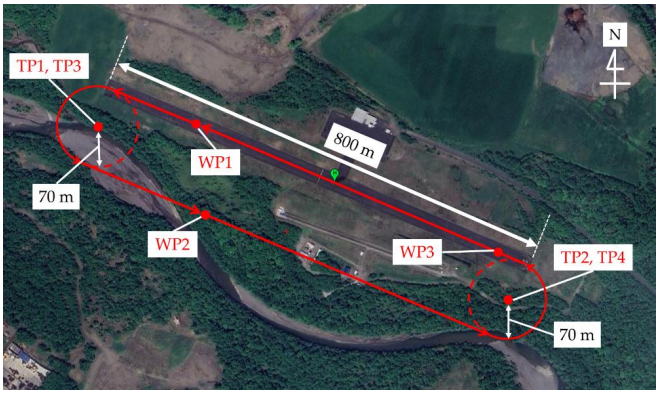


Figure 13. Overview of Shiraoi Gliding Field.

Table 4. Coordinates and Altitude of each WP and TP.

WP/TP	Latitude	Longitude	Altitude
TP1	42° 53′ 75.1″ N	141° 25′ 04.3″ E	80 m
TP2	42° 53′ 49.9″ N	141° 25′ 85.9″ E	80 m
WP1	42° 53′ 76.0″ N	141° 25′ 23.6″ E	50 m
TP3	42° 53′ 75.1″ N	141° 25′ 04.3″ E	50 m
WP2	42° 53′ 62.4″ N	141° 25′ 23.3″ E	50 m
TP4	42° 53′ 49.9″ N	141° 25′ 85.9″ E	140 m
WP3	42° 53′ 57.8″ N	141° 25′ 83.0″ E	140 m

3.2.2. Steady-State Evaluation Criteria and Phase-Specific Settings

In the SITL simulation, sensor outputs are idealized, and measurement noise, bias, and delays (e.g., actuator delays) that occur in real flights may not be sufficiently reflected [29]. Therefore, threshold-based steady-state judgement such as that based on the mean absolute error (MAE) may be more readily satisfied than in real flights. To address this issue, we introduced an effective disturbance (flight-data-derived effective noise) that equivalently represents the multiple factors described above. The effective disturbance was represented by adding zero-mean Gaussian white noise to the state variables used for the MAE evaluation (steady-state evaluation) described in Section 2.3. The standard deviation of the noise was set based on the fluctuations of each state variable observed during straight-and-level segments in real-flight data. The Gaussian white noise was generated using the Box–Muller transform, which generates normally distributed random numbers from uniformly distributed random numbers. In this implementation, only one of the two normally distributed random numbers obtained simultaneously was used, and the other was not used. The Box–Muller transform can generate normally distributed random numbers concisely using only basic mathematical functions, and we adopted it because it is widely used [30,31]. Note that the addition of the effective disturbance was applied only to the steady-state judgement inside Lua and was not recorded in the simulation flight log.

Steady-state evaluation was performed in principle in accordance with Section 2.3. The allowable MAE thresholds ε_i were set based on the MAE values calculated from past flight-experiment data obtained during steady level flight at an altitude of 80 m over a straight-flight segment of approximately 400 m. Table 5 shows the sampling time interval Δt_{past}, the number of samples N_{past}, and ε_i for each state variable used in the calculation. Note that the aircraft used in the past flight experiment was the same as that shown in Figures 10 and 11, and Table 3. In addition, Table 5 shows the standard deviation σ_i of the Gaussian white noise added as the effective disturbance. The sampling time interval and the number of samples for σ_i were also the same as those used to calculate ε_i.

Table 5. Tolerable MAE Threshold.

Item	Unit	Value
Δt_{past}	s	0.02
N_{past}	-	470
ε_V	m/s	0.52
$\varepsilon_{\dot{V}}$	m/s^2	0.55
$\varepsilon_{\dot{h}}$	m/s	0.76
ε_h	m	0.71
σ_V	m/s	0.367
$\sigma_{\dot{V}}$	m/s^2	0.388
$\sigma_{\dot{h}}$	m/s	0.538
σ_h	m	0.500

Next, Table 6 lists the sampling interval Δt used for the steady-state evaluation in the proposed method, the evaluation window T_{obs} for each phase, and the time difference between the evaluation start time t_0 and the forced termination time t_f. Table 7 lists the state variables $\mathbf{x}(t)$ referenced in each phase described in Section 2.4, the MAEs J_i, and the parameter settings.

Table 6. Steady-state evaluation time in the proposed methodology.

Item	Unit	Value	Note
Δt	s	0.02	-
$T_{obs,\,P1}$, $T_{obs,\,P2}$, $T_{obs,\,P6}$	s	4.0	Phase 1, Phase 2, Phase 6
$T_{obs,\,P4}$	s	3.5	Phase 4
$T_{obs,\,P5}$	s	3.0	Phase 5
$t_f - t_0$ (Time out)	s	20	Only use Phase 1, Phase 2

Table 7. State Variables Referenced in Each Phase, MAE Metrics, and Parameter Settings.

Phase	Item	Unit	Value
Common	V^{ref}	m/s	25
1	$\mathbf{x}(t)$	$[\mathrm{m/s\ m/s^2\ m/s}]^T$	$\left[V(t) \quad \dot{V}(t) \quad \dot{h}(t)\right]^T$
	$\mathbf{x}^{ref}$	$[\mathrm{m/s\ m/s^2\ m/s}]^T$	$\left[V_{dec}^{N_{dec}} \quad 0 \quad 0\right]^T$
	J_i	$\{\mathrm{m/s, m/s^2, m/s}\}$	$\left\{J_V, J_{\dot{V}}, J_{\dot{h}}\right\}$
	$V_{dec}^{N_{dec}}$	m/s	$\{24,\ 23, \ldots, 10\}$
2	$\mathbf{x}(t)$	$[\mathrm{m/s\ m/s^2\ m/s}]^T$	$\left[V(t) \quad \dot{V}(t) \quad \dot{h}(t)\right]^T$
	$\mathbf{x}^{ref}$	$[\mathrm{m/s\ m/s^2\ m/s}]^T$	$\left[V_{acc}^{N_{acc}} \quad 0 \quad 0\right]^T$
	J_i	$\{\mathrm{m/s, m/s^2, m/s}\}$	$\left\{J_V, J_{\dot{V}}, J_{\dot{h}}\right\}$
	$V_{acc}^{N_{acc}}$	m/s	$\{26,\ 27, \ldots, 40\}$
3&4	$\mathbf{x}(t)$	$[\mathrm{m/s\ m/s^2}]^T$	$\left[V(t) \quad \dot{V}(t)\right]^T$
	$\mathbf{x}^{ref}$	$[\mathrm{m/s\ m/s^2}]^T$	$\left[25 \quad 0\right]^T$
	J_i	$\{\mathrm{m/s, m/s^2}\}$	$\{J_V, J_{\dot{V}}.\}$
	V_R	m/s	28
	θ_{R0}	deg.	23
	$\theta_{min,climb}$	deg.	5
	h_{max}	m	140
	$\Delta\theta_{margin}$	deg.	5
5	$\mathbf{x}(t)$	[m/s]	$[V(t)]$
	$\mathbf{x}^{ref}$	[m/s]	[25]
	J_i	{m/s}	$\{J_V\}$
	T_{min}	%	10
	h_{min}	m	30
6	$\mathbf{x}(t)$	$[\mathrm{m/s\ m/s^2\ m/s\ m}]^T$	$\left[V(t) \quad \dot{V}(t) \quad \dot{h}(t) \quad h(t)\right]^T$
	$\mathbf{x}^{ref}$	$[\mathrm{m/s\ m/s^2\ m/s\ m}]^T$	$\left[25 \quad 0 \quad 0 \quad 80\right]^T$
	J_i	$\{\mathrm{m/s, m/s^2, m/s, m}\}$	$\{J_V, J_{\dot{V}}, J_{\dot{h}}, J_h.\}$

3.3. Simulation Results

Figure 14 presents top and bird's-eye views of the simulated flight trajectory for the proposed method with the designed flight plan, together with the placement of each Phase and the corresponding starting points. The trajectory was plotted using UAV Log Viewer, which was developed for ArduPilot [32]. As shown in Figure 14, transitions between the Phases were executed correctly, confirming that the designed Lua script and the flight plan operated in coordination. In the following subsubsections, we examine whether the steady-state evaluation of each state variable was performed properly in each Phase.

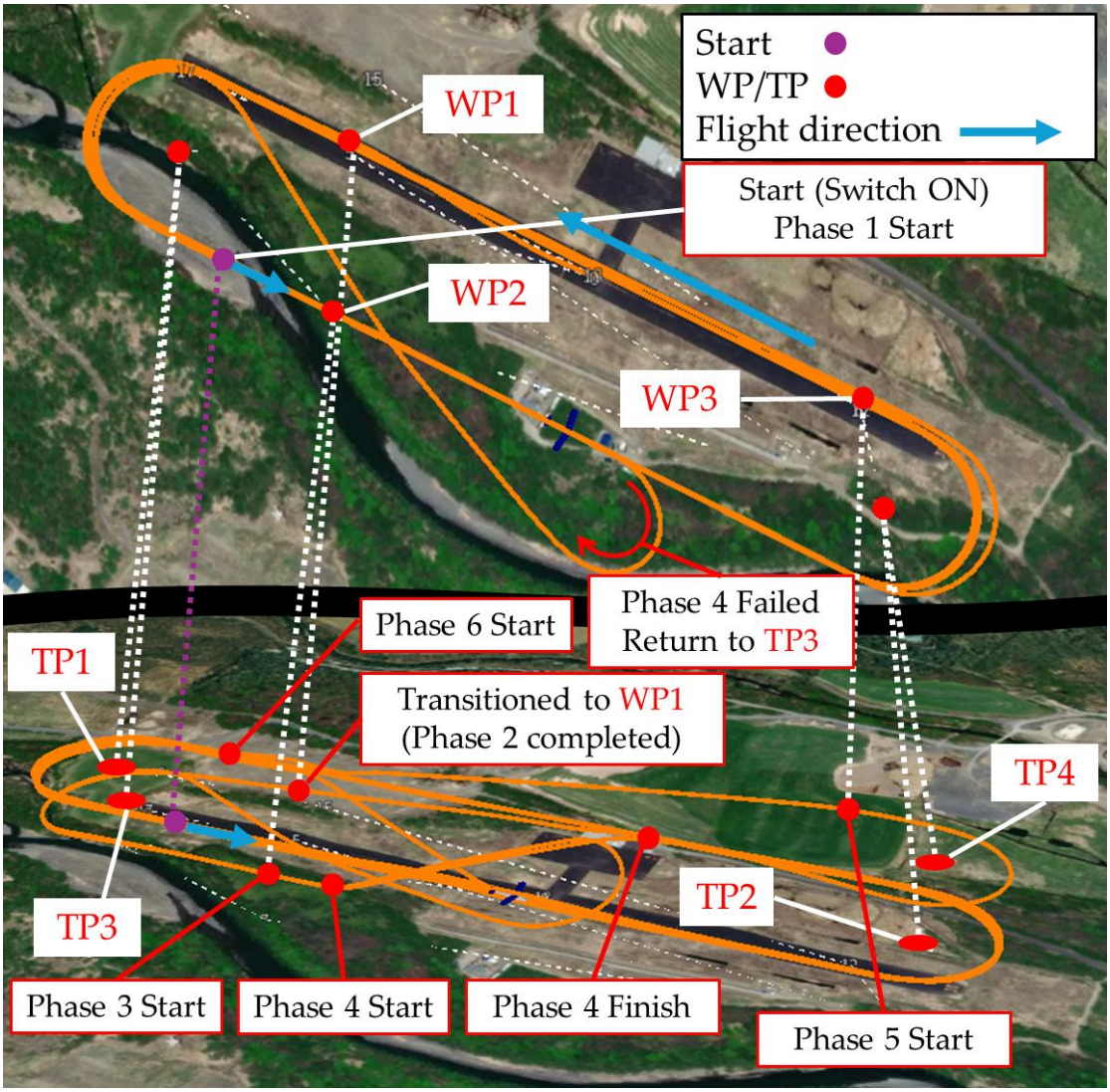


Figure 14. Top view and bird's-eye view of the entire flight path generated by simulation.

3.3.1. Phases 1 and 2: Automatic Determination Results for V_{min} and V_{max} (Simulation Results)

Figure 15 shows the time histories of the state variables $\mathbf{x}(t)$ used for steady-state evaluation in Phases 1 and 2. Here, the black vertical line indicates the start of the Phase, the steady-state judgment interval (green band) is defined as the Steady-State Measurement Range (SSMR) ($T_{obs,\ P1}$, $T_{obs,\ P2}$ = 4.0 s), and the blue vertical line indicates the end of the Phase. As shown in Figure 15a, the commanded airspeed decreased stepwise as the logic described in Section 2.4.1 operated. When the commanded airspeed reached 12 m/s, TECS judged that it could not maintain the demanded altitude, and the airspeed failed to track the command of 12 m/s. Consequently, the steady-state evaluation timed out (after 20 s), and V_{min} was automatically determined as 13 m/s. Next, as shown in Figure 15b, the com-

manded airspeed increased stepwise as the logic described in Section 2.4.2 operated. When the commanded airspeed reached 37 m/s, TECS did not track the demanded longitudinal acceleration and rate of climb. Consequently, the steady-state evaluation again timed out (after 20 s), and V_{max} was automatically determined as 36 m/s.

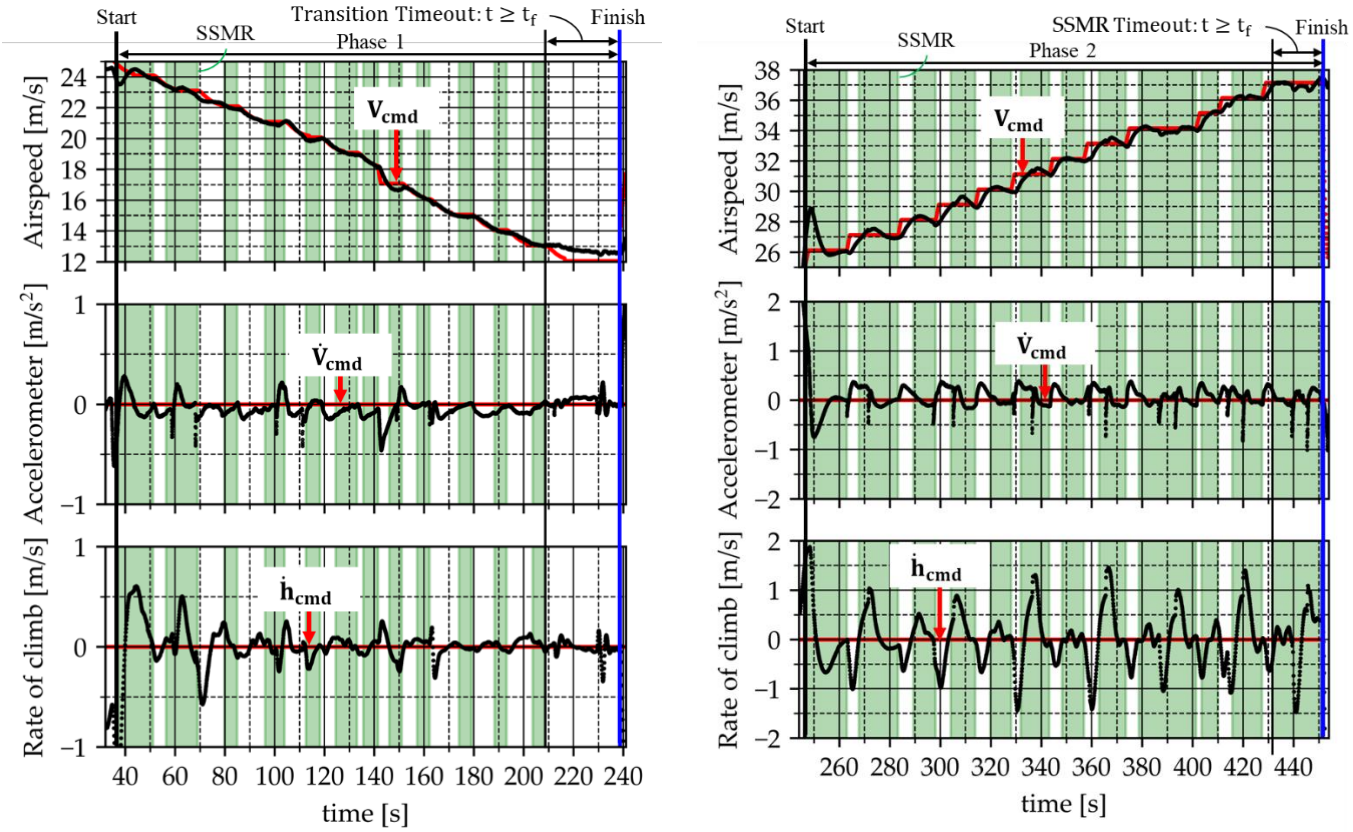
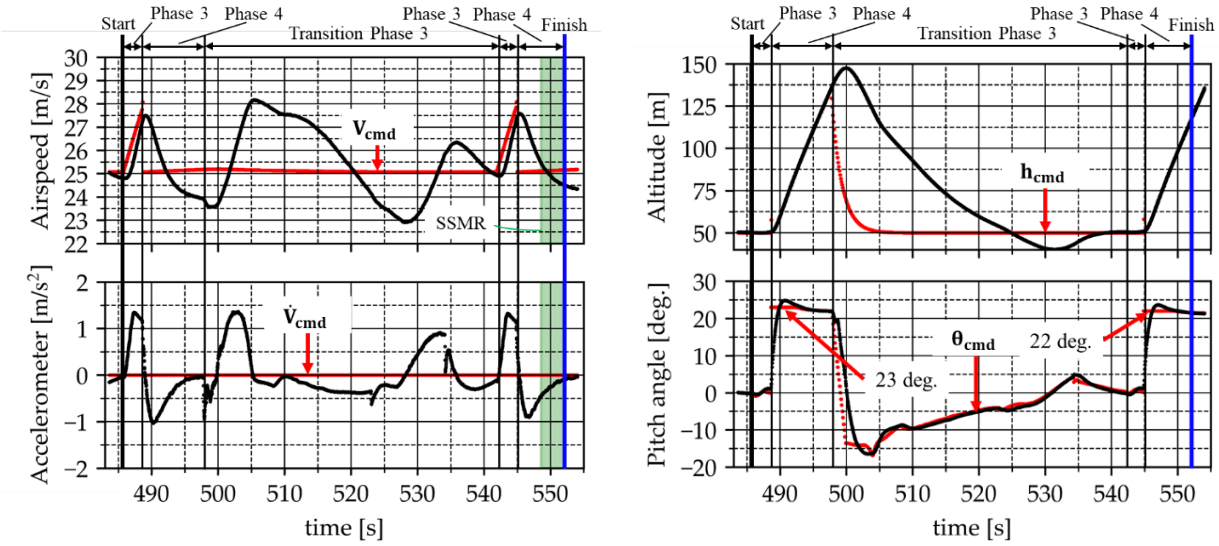


Figure 15. Time histories of state variables used for steady-state evaluation for automatic determination results in Phases 1 and 2 in simulation: (**a**) Phase 1; (**b**) Phase 2.

3.3.2. Phase 3 and Phase 4: Automatic Determination Results for $\dot{h}_{max}$, θ_{max}, θ_{min} and $\dot{h}^{max}_{sink}$ (Simulation Results)

Figure 16 shows the time histories of the state variables $\mathbf{x}(t)$ used for steady-state evaluation in Phases 3 and 4. As supplementary information, Figure 16b also shows the altitude demand $h_{cmd}(t)$ and the altitude $h(t)$, as well as the pitch angle $\theta(t)$ that follows the climb pitch-angle command $\theta_{cmd}(t)$ generated by applying Equation (12). The meaning of the legends in the figure is the same as in Section 3.3.1. As shown in Figure 16, the logic designed in Section 2.4.3 enabled seamless repetition of Phases 3 and 4, and the UAV transitioned to climbing flight after each acceleration. In addition, as shown in Figure 16b, θ_{R0} in the second climb changed from 23 deg. to 22 deg., indicating that the designed Equation (12) was effective. Finally, in accordance with Section 2.4.3, $\dot{h}_{max}$, θ_{max}, θ_{min} and $\dot{h}^{max}_{sink}$ in Phase 4 were automatically determined as 9.10 m/s, 21.5 deg., −16.5 deg., and −10.2 m/s, respectively.

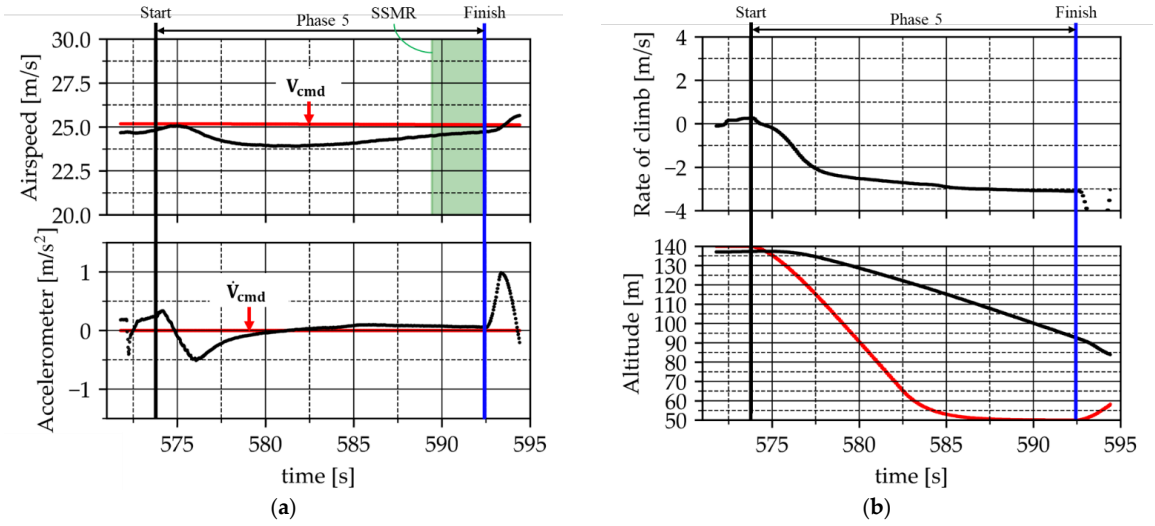


(**a**) (**b**)

Figure 16. Automatic determination results for Phases 3 and 4 in simulation: (**a**) Time histories of state variables used for steady-state evaluation. (**b**) Supplementary information ($h_{cmd}(t)$, $h(t)$, $\theta_{cmd}(t)$, and $\theta(t)$).

3.3.3. Phase 5: Automatic Determination Results for $\dot{h}^{min}_{sink}$ (Simulation Results)

Figure 17 shows the time histories of the state variables $\mathbf{x}(t)$ subject to the steady-state evaluation in Phase 5. In addition, Figure 17b provides supplementary information, including the rate of climb $\dot{h}(t)$, the altitude demand $h_{cmd}(t)$, and the altitude $h(t)$. The meaning of the legends in the figure is the same as in Section 3.3.1. As shown in Figure 17a, the speed-prioritized control was achieved by Equation (17) in the logic designed in Section 2.4.4, and the airspeed was confirmed to track the commanded value. In addition, the altitude time history in Figure 17b confirms that the UAV descended at an approximately constant angle while ignoring the altitude demand. Although not shown in the figure, the throttle was also confirmed to remain constant at the minimum throttle rate of 10%, indicating that the intended steady gliding condition was achieved. Consequently, a steady-state judgment was made based on the airspeed time history in Figure 17a, and the minimum sink rate in Phase 5, $\dot{h}^{min}_{sink}$, was automatically determined as -3.10 m/s.



(**a**) (**b**)

Figure 17. Automatic determination results for Phase 5 in simulation: (**a**) Time histories of state variables used for steady-state evaluation. (**b**) Supplementary information ($\dot{h}(t)$, $h_{cmd}(t)$ and $h(t)$).

3.3.4. Phase 6: Automatic Determination Results for T_{trim} (Simulation Results)

Figure 18 shows the time histories of the state variables **x**(t) subject to steady-state evaluation in Phase 6. In addition, the meaning of the legends in the figure is the same as in Section 3.3.1. As shown in Figure 18, within the steady-state judgment range SSMR ($T_{obs,P6}$ = 4.0 s), the test aircraft was confirmed to be in steady level flight, and steady state was judged. Accordingly, following the logic designed in Section 2.4.5, we defined the throttle rate at the time Phase 6 ended as the trim throttle rate T_{trim}. Therefore, T_{trim} in Phase 6 was automatically determined as 37.1%.

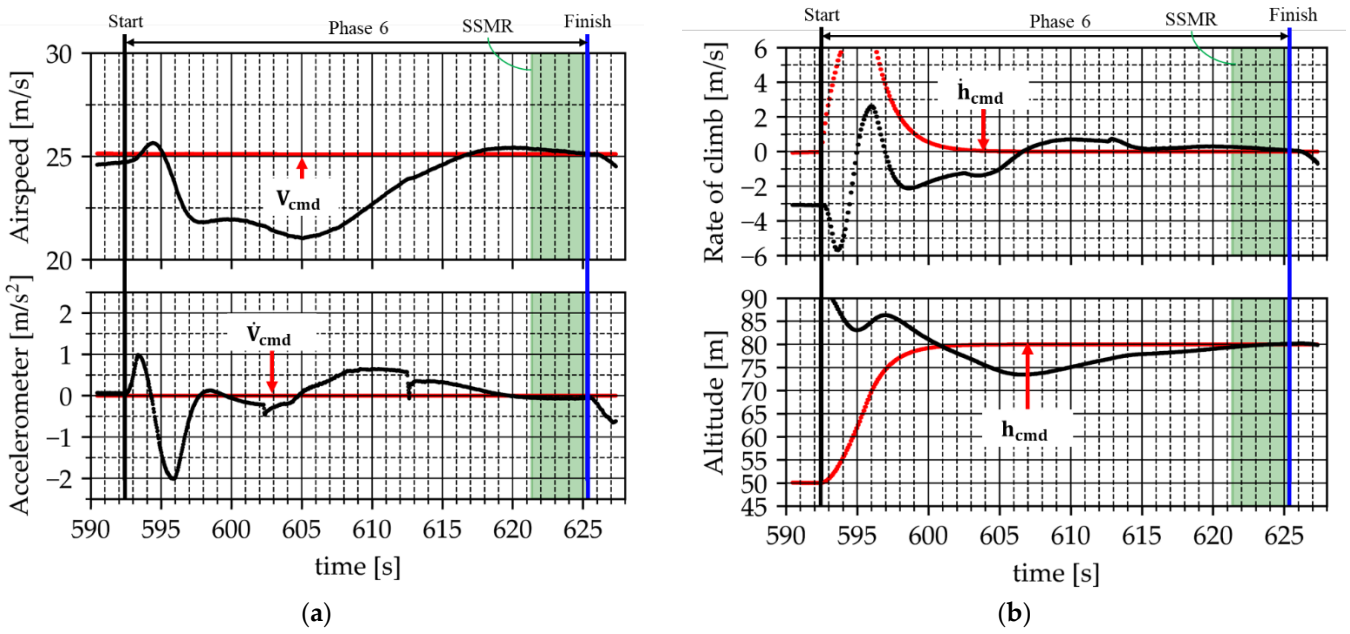


Figure 18. Automatic determination results for Phase 6 in simulation: (**a**) Airspeed and longitudinal accelerometer data subject to steady-state evaluation. (**b**) Rate of climb and altitude subject to steady-state evaluation.

3.4. Robustness Evaluation of the Proposed Method against Wind Disturbances

We evaluated, using SITL simulation, the reproducibility of the automatically determined TECS parameters under changes in wind conditions. As shown in Table 8, we conducted the evaluation using 1 + 18 cases based on the design of experiments (DOE). Because CaseID = 1 was used to compute the Baseline, we excluded it from the evaluation. We set the factors as the mean wind speed (WindSpd), wind direction (WindDir), and gust (Gust and its magnitude). In each case, we executed the same automatic-determination sequence (Phase 1–Phase 6). In the proposed method, the flight-altitude region is approximately 50–140 m above ground level (AGL). Therefore, in X-Plane 12 [24], we set a wind layer so that it included this altitude band. We set the wind to be uniform with altitude within the target wind layer. We defined wind direction as a relative angle with respect to the takeoff heading (details are shown in Table 8). As a note, the errors treated in this subsection do not represent differences from true (ideal) values. Because this study aims to determine unknown parameters, we cannot define a single unique correct value to be reached. Therefore, as stated above, we defined the determined values obtained in Case 1 (no wind) as the Baseline and defined the difference for each case as "determined value − Baseline". The Baseline values are identical to those obtained in Section 3.3. Here, we show the error distributions using box-and-whisker plots and also report the RMSE as a representative value in the figures.

In summary, in this subsection, we first evaluate the error distributions and reproducibility using box-and-whisker plots. Next, we organize the influence of each factor on

the mean error using factor effect plots. Finally, to explain the statistical properties of the errors, we show the error distribution shape and the dominant components of the errors using Q–Q plots and Theil's inequality coefficient decomposition.

Table 8. The 1 baseline case and 18 cases based on the design of experiments (DOE).

CaseID	WindSpd [m/s]	WindDir [deg.]	Gust [m/s]
1 (Baseline)	0	0	0
2	4	0	0
3	4	45	0
4	4	90	0
5	4	0	2
6	4	45	2
7	4	90	2
8	8	0	0
9	8	45	0
10	8	90	0
11	8	0	2
12	8	45	2
13	8	90	2
14	10	0	0
15	10	45	0
16	10	90	0
17	10	0	2
18	10	45	2
19	10	90	2

3.4.1. Assessment of Error Distributions Based on Box-and-Whisker Plots

We show the error distributions for all 18 cases (CaseID = 2–19) in Figure 19. We present the distributions separately for the no-gust (No Gust) and gust (Gust) conditions. To examine the behavior in the low-wind-speed region, we also show the same distributions for the subset of cases that belong to the low-wind-speed row in Figure 20.

Across all 18 cases, the error scale and distribution width differed among parameters. In particular, V_{min} showed a large RMSE of 6.90 m/s under the Gust condition. This result indicated that the determined V_{min} could vary substantially under strong wind disturbances. The RMSE of Ttrim was also large, at 5.42% for No Gust and 4.82% for Gust, respectively. Note that Ttrim is expressed in %, and its numerical scale differs from those of airspeed [m/s], vertical velocity [m/s], and angles [deg.]. Therefore, we interpreted the results not by directly comparing RMSE magnitudes across parameters, but as changes in the sign and width of the error distribution. By contrast, $\dot{h}_{max}$ and $\dot{h}^{min}_{sink}$, which are related to vertical motion, showed small RMSE in both conditions (e.g., the Gust-condition RMSE of $\dot{h}^{min}_{sink}$ was 0.30 m/s). Therefore, under this setting, the effect of changes in wind conditions on parameters related to vertical motion is inferred to be limited. Among the eight TECS parameters evaluated, the main text shows representative examples with clearly distinguishable characteristics—those with larger errors (e.g., V_{min}, T_{trim}) and those with smaller errors (e.g., $\dot{h}_{max}$, $\dot{h}^{min}_{sink}$). Next, as stated above, to clarify the behavior in the low-wind-speed region shown in Figure 20, we also performed the same evaluation for the subset $2 \leq$ CaseID ≤ 7. In this subset, the RMSE of V_{min} was 0.82 m/s for No Gust and 0.58 m/s for Gust. These values were substantially smaller than the RMSE of V_{min} across all 18 cases. Therefore, in the low-wind-speed region, the gust-induced error was minor, and the determined values were found to be reproducible.

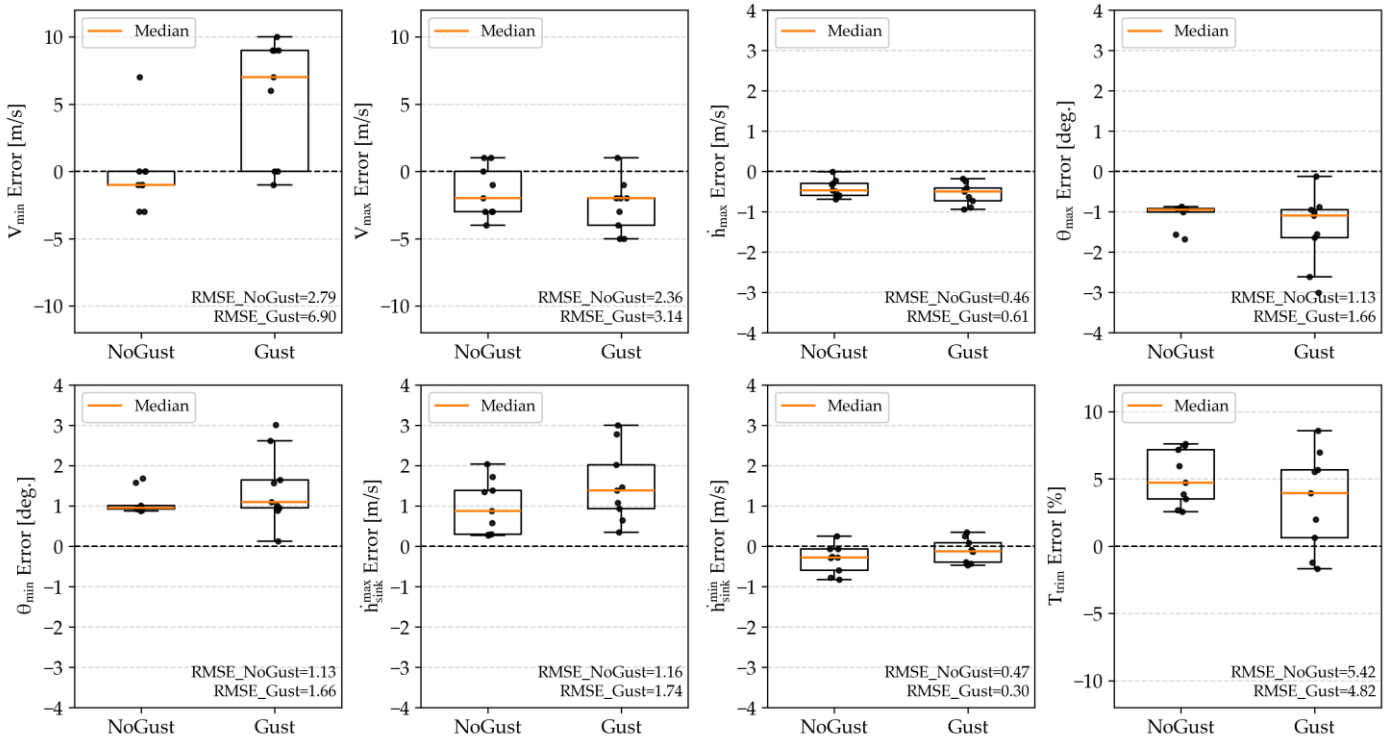


Figure 19. Evaluation of Error Distribution Based on Boxplots.

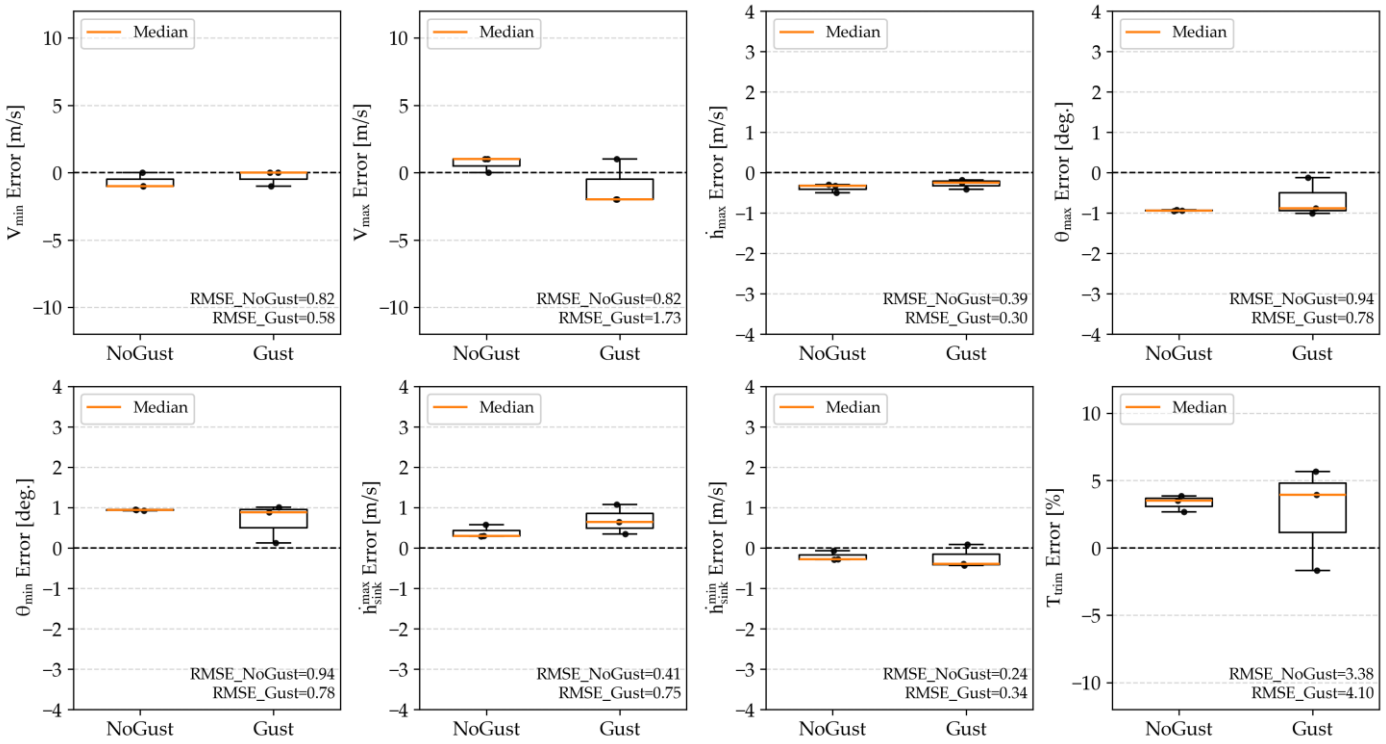


Figure 20. Error distribution of the subset of $2 \leq$ CaseID ≤ 7 in the low wind speed region.

3.4.2. Sensitivity Analysis Using Factor Effect Plots

As a sensitivity analysis, we show the factor effect plots using all 18 cases (CaseID = 2–19) in Figure 21. The factor effect plots show the mean signed error at each factor level, such as WindSpd, WindDir, and the presence or absence of Gust. We define the error in the same manner as in Section 3.4. From Figure 21, among the three factors, WindSpd had the largest effect on the mean error. As a representative example, as WindSpd increased,

the mean error of V_{min} increased toward the positive side, and the mean error of V_{max} decreased toward the negative side. This result was consistent with the tendency that, as WindSpd increased, V_{min} and V_{max} shifted to a conservative side from the viewpoints of stall avoidance and overspeed suppression, respectively. Note that this shift was not a result intentionally corrected in post-processing. All automatically determined values obtained by the proposed method were derived from the closed-loop responses formed by ArduPilot's automatic flight control system under disturbances (thrust demand and attitude margin). Therefore, for V_{min} and V_{max}, as the disturbance increased, additional speed margin was required to satisfy steady-state conditions within the control limits. As a result, the segments that satisfied the steady-state judgement described in Section 2.3 were biased toward a speed region away from stall, and the mean error of V_{min} appeared on the positive side. In contrast, in the high-speed region, acceleration and speed holding were constrained, and the segments that satisfied the steady-state judgement were biased toward the lower-speed side; therefore, the mean error of V_{max} appeared on the negative side.

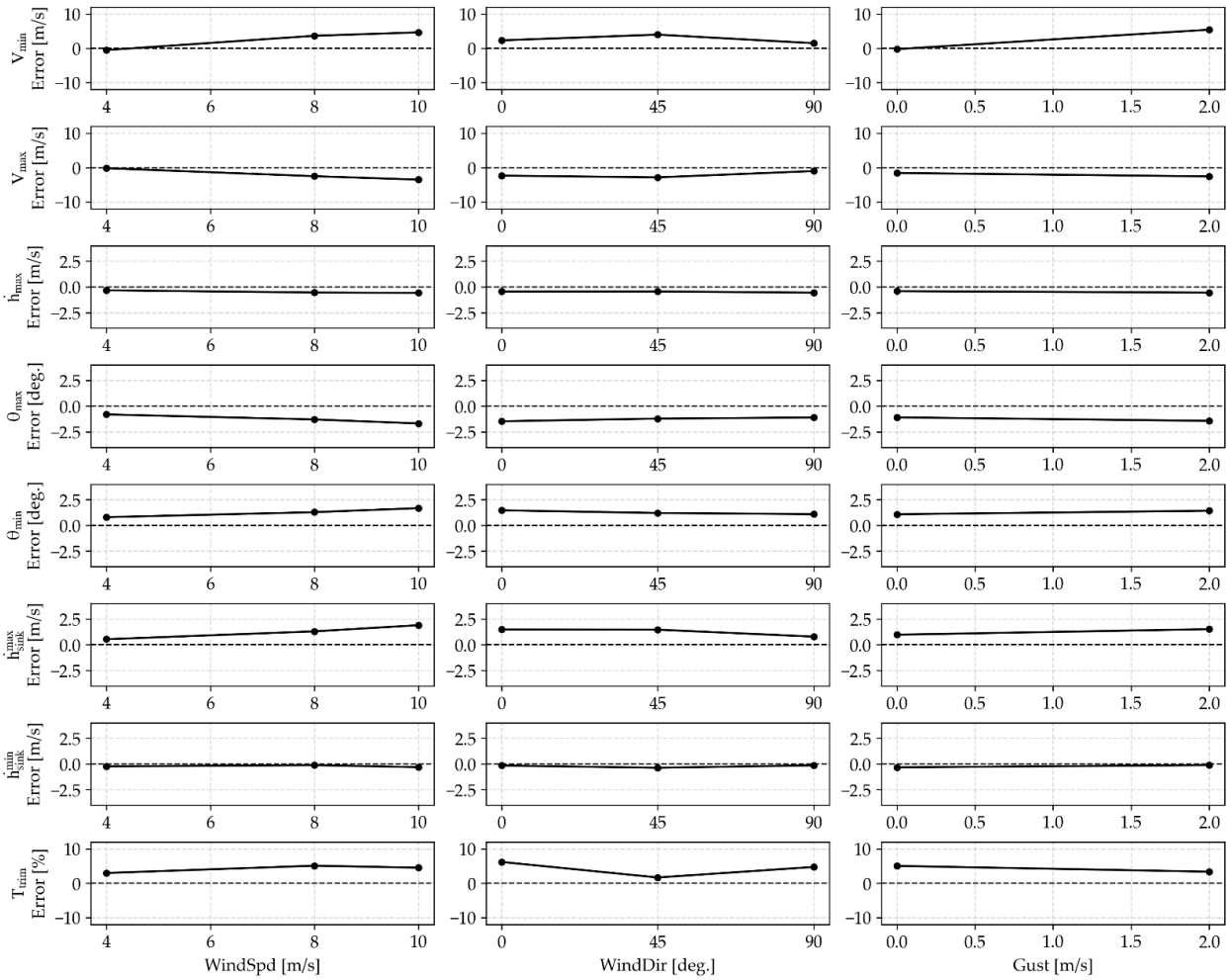


Figure 21. Cause-and-Effect Diagram for All 18 Cases.

Next, the change in the mean error due to WindDir was small compared with that due to WindSpd. In addition, we did not confirm monotonic increases or decreases for many parameters. Here, for V_{min}, the mean error became relatively large around WindDir = 45 deg., whereas it became small at WindDir = 90 deg. This reflected that, because wind direction was defined as a relative angle with respect to the takeoff heading, the effective headwind and crosswind components varied and the influence on the closed-loop responses (thrust

demand and attitude margin) was not uniform. Therefore, within this evaluation range, WindDir was a secondary factor and WindSpd was the dominant factor.

Finally, the main effect of Gust was also small compared with that of WindSpd. However, for V_{min}, as Gust increased, the mean error increased toward the positive side. This was consistent with the tendency that, when gusts existed, the segments corresponding to the steady-state judgement were biased toward a safer speed region. For the other parameters, the change in the mean error was small, or the tendency was not clear.

3.4.3. Error-Factor Analysis Based on Q–Q Plots and Theil's Inequality Coefficient Decomposition

Up to Section 3.4.2, we showed changes in the mean error (main effects). However, we need to distinguish whether the errors originate from random variability or from a shift in a certain direction. Therefore, in this subsubsection, we organized the dominant components of the errors using Q–Q plots to confirm the shape of the error distributions and Theil's inequality coefficient decomposition to quantify the components of the MSE. As stated at the end of Section 3.4, because no reference value exists as a true value, we used these analyses to identify whether the errors are dominated by systematic shifts or by random variability. Based on this background, in the Theil's inequality coefficient decomposition, we set the covariance component U^C to zero and treated it as a two-component decomposition consisting of the systematic bias component U^M and the variance component U^S.

First, we show the Q–Q plots in Figure 22. The horizontal axis represents the theoretical quantiles of a normal distribution. The vertical axis represents the ordered errors, i.e., the errors of each parameter sorted in ascending order. The closer the points are to a straight line, the closer the error distribution is to a normal distribution. In contrast, deviations from the straight line at the tails suggest the presence of outliers or the effects of skewness and kurtosis. From Figure 22, for many parameters (V_{max}, $\dot{h}_{max}$, $\dot{h}^{max}_{sink}$, $\dot{h}^{min}_{sink}$, and T_{trim}), the points approximately followed a straight line. This result indicated a tendency that the error distributions can be approximated by a simple normal distribution. In contrast, V_{min} showed a relatively large deviation from the straight line, indicating that its error distribution tends to be difficult to approximate as a simple normal distribution. This reflects that, because the feasibility and the speed region of segments that satisfy the steady-state judgement changed depending on the disturbance conditions, the errors are not uniform noise and can become a condition-dependent mixture distribution.

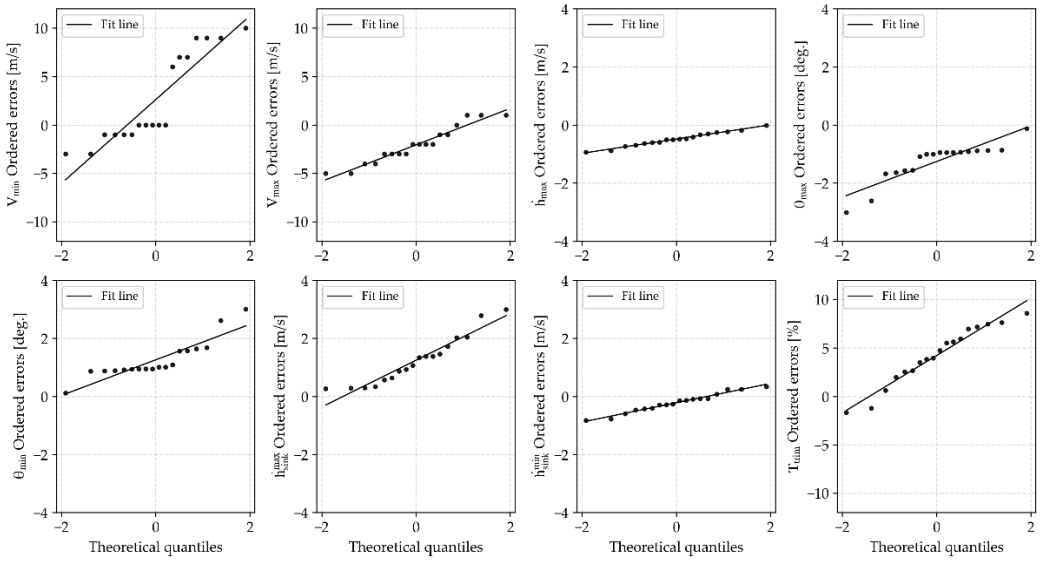


Figure 22. Q–Q plots of parameter errors (relative to the baseline case).

Next, we show the breakdown of error factors based on Theil's inequality decomposition (MSE decomposition) in Figure 23. Here, we decomposed the mean squared error (MSE) into the systematic bias component U^M, which originates from the deviation of the mean error, and the variance component U^S, which originates from the variability of the errors, and evaluated each component as a ratio to the MSE. As a result, for V_{max}, $\dot{h}_{max}$, θ_{max}, θ_{min}, $\dot{h}^{min}_{sink}$, and T_{trim}, U^M was dominant, and we found that the main cause of the errors was not random variability but a one-sided systematic shift associated with the wind conditions. This was consistent with the tendency shown in the factor effect plots in Section 3.4.2, in which the mean error changed monotonically with factors such as WindSpd. Therefore, as also described in Section 3.4.2, we again confirmed that the errors did not originate from post-processing corrections or incidental noise, but from changes in the locations of the steady-state segments formed by ArduPilot's closed-loop responses under disturbances. In contrast, for V_{min} and $\dot{h}^{min}_{sink}$, we confirmed that the contribution of U^S was relatively large. This suggested that the errors for each condition cannot be explained only by systematic shifts (i.e., the influence of variability is large). Therefore, although robustness against wind disturbances is generally maintained, especially V_{min} tends to undergo changes in the distribution shape and increases in variance under disturbance conditions. For this reason, in reproducibility evaluation, it is reasonable to use nonparametric indices such as quantiles and box-and-whisker plots in combination.

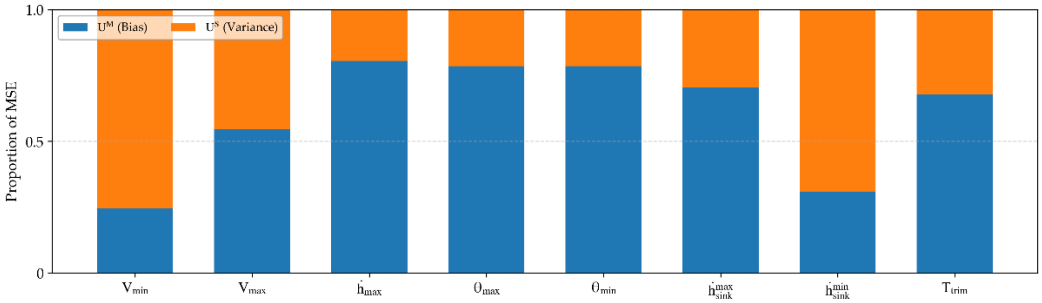


Figure 23. Proportions of MSE attributed to systematic bias and variance for each parameter.

3.5. Sensitivity Analysis on the Tolerable MAE Threshold

In this subsection, we evaluate how the settings of the allowable MAE threshold components ε_i for each TECS parameter affect the feasibility of the Phase procedures and the determined values. We evaluated the effects of wind disturbances in Section 3.4. In this subsection, we conduct the evaluation under the no-wind condition.

As a condition, for the reference values ε_i shown in Table 5, we defined $\varepsilon_i(\lambda) = \lambda\varepsilon_i$. We varied the coefficient λ from 0.70 to 1.30 in increments of 0.05. We defined the convergence condition as such that each Phase was feasible and the parameters were determined. We defined the determination rate as the number of parameters that were not NaN divided by the total of eight items. Here, Table 9 shows the determination rate and the determined parameters for each λ. In addition, we set the fifth column of Table 9 to "All" when all items were determined. Otherwise, we listed only the names of the parameters that were determined.

Next, we evaluated the sensitivity of the determined values to changes in λ. To restrict the analysis to the region where all items were determined, we considered $\lambda \geq 0.90$. Here, we introduce the normalized deviation $\Delta p(\lambda)$ and express it as in Equation (18).

$$\Delta p(\lambda) = \frac{p(\lambda) - p(1.0)}{p(1.0)} \cdot 100 \tag{18}$$

Next, we show the results of the sensitivity evaluation in Figure 24. From Figure 24, at λ = 0.90 and 0.95, the deviation of V_{min} reached a maximum of approximately + 46%.

In addition, $\dot{h}_{sink}^{max}$ reached a maximum deviation of approximately + 14%. In contrast, for $\lambda \geq 1.05$, the maximum deviation of all parameters was within approximately ± 8%. These results confirmed that, near the lower bound of λ, some determined parameter values tended to fluctuate because the conditions were close to the feasibility boundary. In addition, when λ was sufficiently close to 1.0, the variations in the determined values were limited. Therefore, the allowable MAE threshold components ε_i used in the proposed method (λ = 1.0) were sensitive to tight conditions, whereas we did not observe large deviations under loose conditions. For this reason, we judged that the determined values were stable for $\lambda \geq 1.05$.

Table 9. Determination rate of the eight TECS parameters for each scaling factor λ.

λ	Count	Rate	Determined Param.
0.70	1	0.125	$\dot{h}_{sink}^{min}$
0.75	1	0.125	$\dot{h}_{sink}^{min}$
0.80	1	0.125	$\dot{h}_{sink}^{min}$
0.85	5	0.625	θ_{max}, $\dot{h}_{max}$, θ_{min}, $\dot{h}_{sink}^{max}$, $\dot{h}_{sink}^{min}$
0.90	8	1	All
1.0	8	1	All
1.05	8	1	All
1.10	8	1	All
1.15	8	1	All
1.20	8	1	All
1.25	8	1	All
1.30	8	1	All

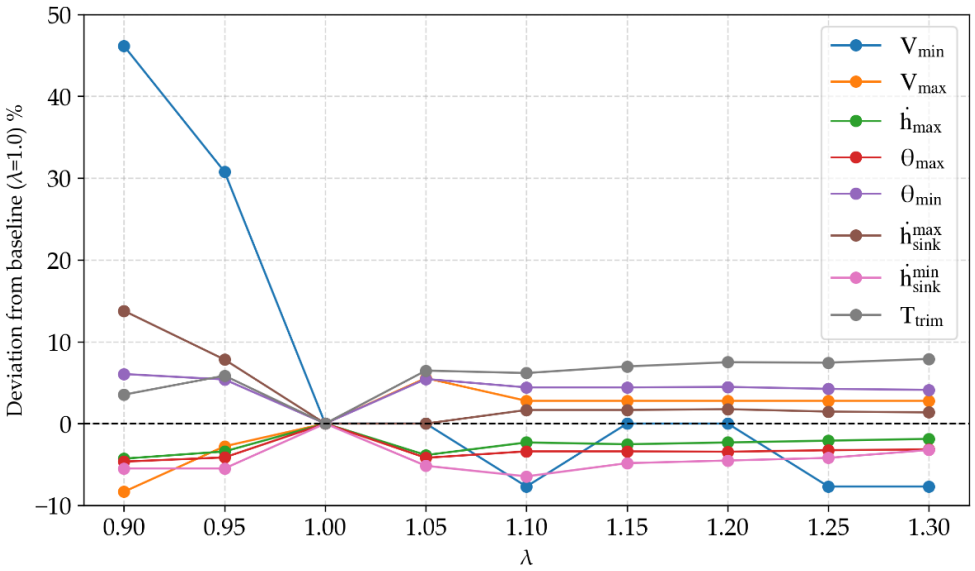


Figure 24. Sensitivity of determined TECS parameters to λ (normalized).

4. Validation Through Flight Experiments

To confirm the effectiveness of the proposed method, we conducted flight experiments at Shiraoi Gliding Field in Hokkaido, Japan, using a model aircraft (Figure 10, Table 3). Table 10 shows the environmental conditions on the day of the flight experiment. We extracted the weather conditions corresponding to the experiment time from the data provided by the Japan Meteorological Agency [33]. In this subsection, as in the SITL simulation, we confirmed that the designed Lua script automatically determines the TECS parameters and that the associated processing in each Phase can be executed normally.

Table 10. Meteorological conditions on the flight-test day (based on JMA data).

Item	Unit	Value	Note
Air temperature	°C	−5.1	-
Humidity	%	71	-
Vapor pressure	hPa	2.8	-
Mean wind speed	m/s	5.2	-
Mean wind direction	deg.	WNW (292.5)	Wind direction is given as the direction from which the wind blows.
Maximum instantaneous wind speed (gust)	m/s	9.0	-
Gust direction	deg.	NW (315)	Direction from which the gust blows.
Snow depth	cm	1	-

4.1. Overview of the Experimental Aircraft and Onboard System

As described at the beginning of Section 3, the communication configuration between the ground control station (GCS) and the flight controller (FC) is the same as in SITL, except that the "Flight Simulator" component in Figure 9C is replaced by the actual aircraft. Therefore, in this subsection, we focus on the FC and peripheral devices installed on the experimental aircraft. Figures 25 and 26 and Table 11 show the appearance, system configuration, and detailed names of the major onboard hardware installed on the experimental aircraft.

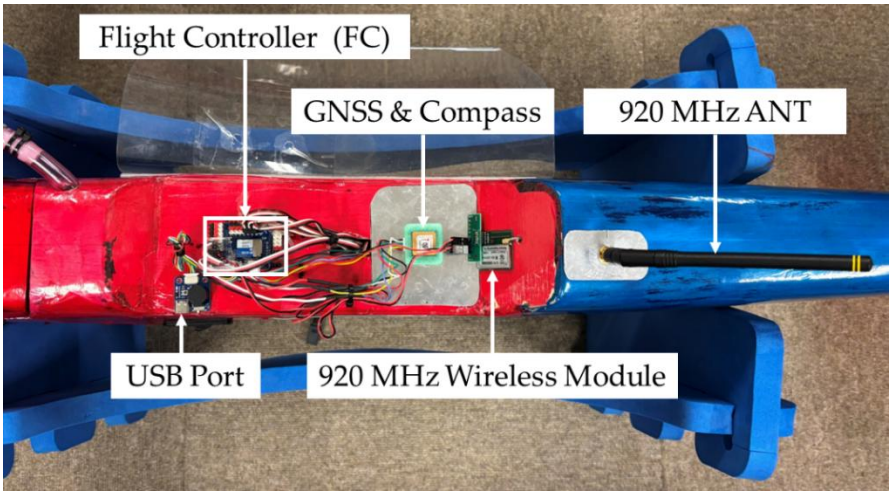


Figure 25. Appearance of the major onboard hardware installed on the experimental aircraft.

Table 11. Detailed names of the major onboard hardware installed on the experimental aircraft.

Subsystem/Function	Manufacture (City, State, Country)	Model/Product Name
Flight Controller (FC)	Matek Systems (Hong Kong, China)	H743-WLITE
GNSS & Compass		M10Q-5883
920 MHz radio module	Nomura Engineering (Yamato, Kanagawa, Japan)	TS92 EZmdm
920 MHz ANT	-	Whip antenna
Airspeed sensor	TE Connectivity (Berwyn, PA, USA)	MS4525DO series
Propulsion (Engine)	O.S. Engines (Osaka, Japan)	MAX-46AX II

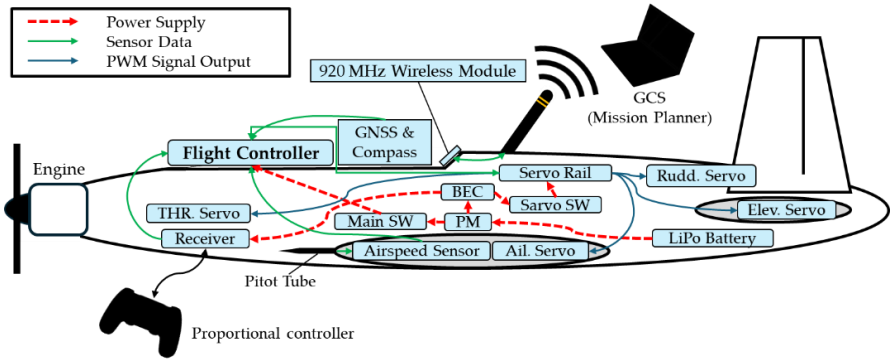


Figure 26. Onboard hardware system diagram of the experimental aircraft.

4.2. Experimental Results

Figure 27 presents the experimental flight trajectory of the proposed method with the designed flight plan, together with the placement and starting points of each Phase, in top and bird's-eye views in the same manner as the simulation result in Figure 14. The trajectory was plotted using the same procedure as that used for Figure 14. As shown in Figure 27, the transitions between the Phases were executed correctly, and we confirmed that the designed Lua script operated in coordination with the flight plan, as in the simulation. In the following subsubsections, we examine whether the steady-state evaluation of each state variable $\mathbf{x}(t)$ was performed properly in each Phase.

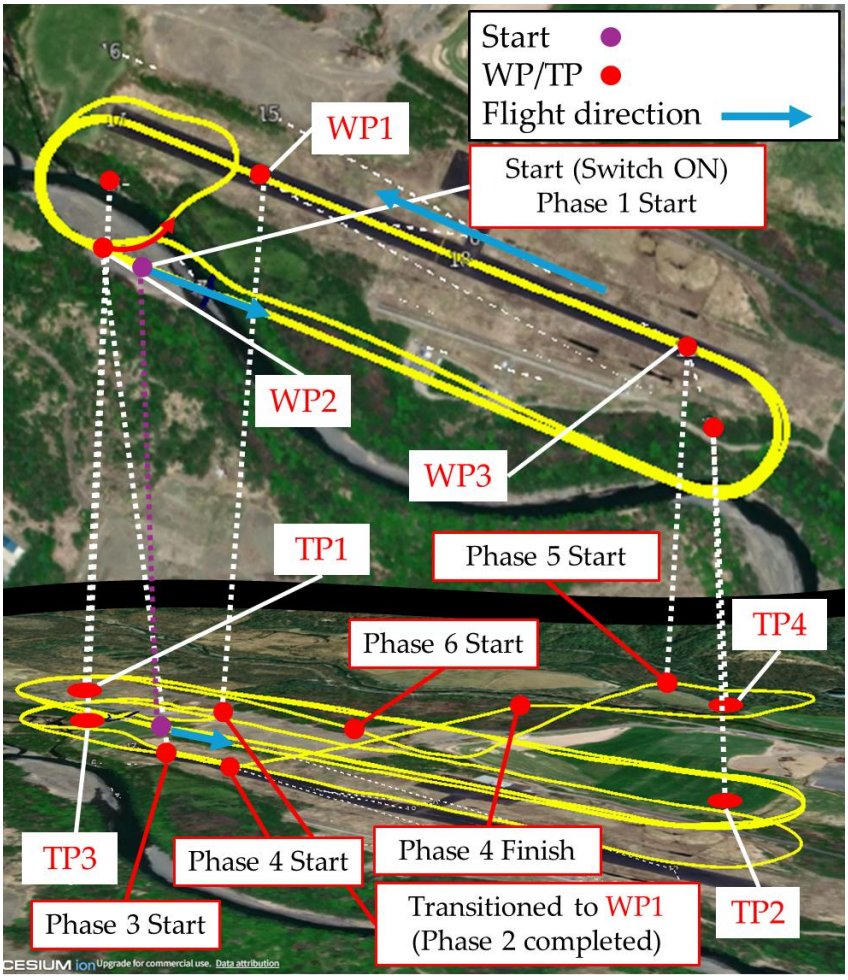
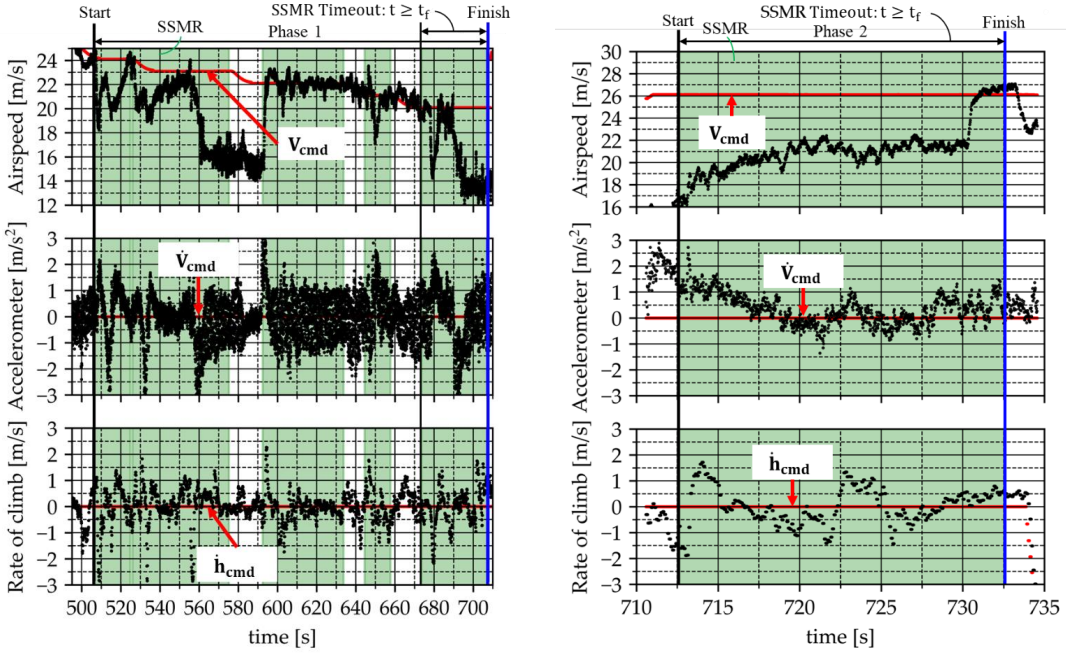


Figure 27. Top view and bird's-eye view of the entire flight path generated by experimentation.

4.2.1. Phases 1 and 2: Automatic Determination Results for V_{min} and V_{max} (Flight Experiments Results)

We plot the time histories of the state variables **x**(t) used for steady-state evaluation in Phases 1 and 2 in Figure 28. In this figure, the black vertical line indicates the start of the phase, the steady-state judgement interval (green band) is denoted as the Steady-State Measurement Range (SSMR) ($T_{obs,\ P1}$, $T_{obs,\ P2}$ = 4.0 s), and the blue vertical line indicates the end of the phase. As shown in Figure 28, similarly to the simulation results in Figure 15, we confirmed that the logic described in Sections 2.4.1 and 2.4.2 operated as intended. Consequently, as shown in Figure 28b, V_{min} in Phase 1 was automatically determined as 20 m/s. Next, as shown in Figure 28b, V_{max} in Phase 2 was automatically determined as 26 m/s. However, because the airspeed sensor measurement was lost due to adverse weather conditions, the value of V_{max} automatically determined in Phase 2 involves uncertainty. Since this value resulted from a timeout triggered by the sensor outage, we supplement the Phase 2 result later using an experimental dataset acquired on a different day.

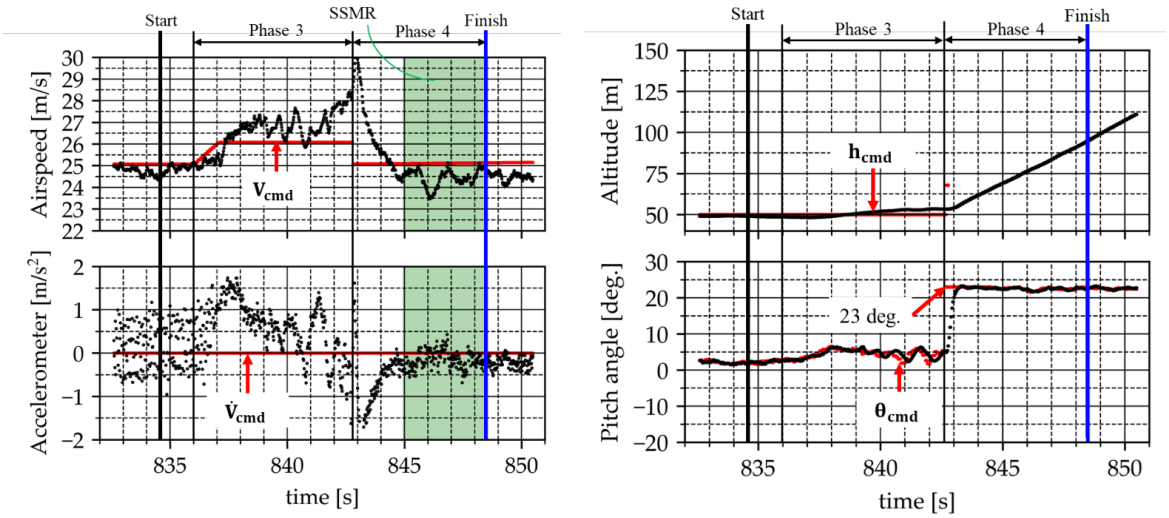


(**a**) (**b**)

Figure 28. Time histories of state variables used for steady-state evaluation for automatic determination results in Phases 1 and 2 in flight experiments: (**a**) Phase 1; (**b**) Phase 2.

4.2.2. Phases 3 and 4: Automatic Determination Results for $\dot{h}_{max}$, θ_{max}, θ_{min} and $\dot{h}_{sink}^{max}$ (Flight Experiments Results)

We show the time histories of the state variables **x**(t) used for steady-state evaluation in Phases 3 and 4 in Figure 29. In addition, we present supplementary information in Figure 29b in the same manner as in Figure 16b. The meaning of the legends in the figure is the same as in Section 4.2.1. As shown in Figure 29, similarly to the simulation results in Figure 16, we confirmed that the logic designed in Section 2.4.3 operated as intended: Phases 3 and 4 were repeated seamlessly, and the aircraft transitioned to climbing flight after acceleration. Moreover, as shown in Figure 29b, steady state was judged after only one climb, in contrast to the simulation. Finally, following Section 2.4.3, we automatically determined $\dot{h}_{max}$, θ_{max}, θ_{min} and $\dot{h}_{sink}^{max}$ in Phase 4 as 7.87 m/s, 22.5 deg, −17.5 deg, and −7.81 m/s, respectively. Note that $\dot{h}_{sink}^{max}$ is coupled with V_{max} determined in Phase 2; therefore, the determined value still leaves room for doubt.

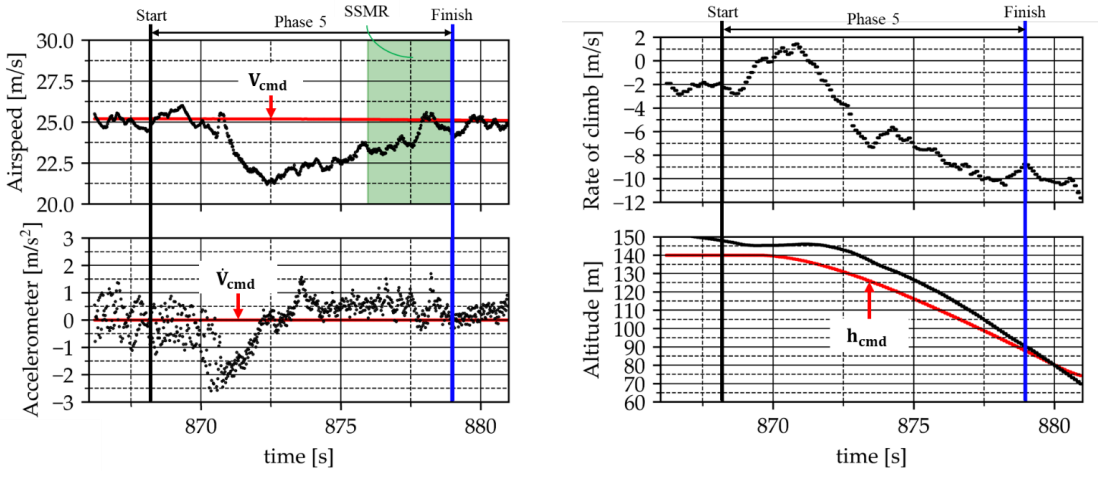


(**a**) (**b**)

Figure 29. Automatic determination results for Phases 3 and 4 in flight experiments: (**a**) Time histories of state variables used for steady-state evaluation. (**b**) Supplementary information ($h_{cmd}(t)$, h(t), $\theta_{cmd}(t)$, and $\theta(t)$).

4.2.3. Phase 5: Automatic Determination Results for $\dot{h}^{min}_{sink}$ (Flight Experiments Results)

We show the time histories of the state variables **x**(t) subject to steady-state evaluation in Phase 5 in Figure 30. We also provide supplementary information in Figure 30b in the same manner as in Figure 17b. The meaning of the legends in the figure is the same as in Section 4.2.1. As shown in Figure 30a, similarly to the simulation results in Figure 17, speed-prioritized control was achieved by Equation (17) in the logic designed in Section 2.4.4, and we confirmed that the airspeed tracked the commanded value. In addition, the altitude time history in Figure 30b confirms that the aircraft descended at an approximately constant angle while ignoring the altitude demand. Although not shown in the figure, we also confirmed that the throttle rate remained constant at 10%, i.e., the minimum throttle rate, indicating that the intended steady gliding condition was achieved. Therefore, a steady-state judgement was made based on the airspeed time history in Figure 30a, and the minimum sink rate in Phase 5, $\dot{h}^{min}_{sink}$, was automatically determined as -10.5 m/s.

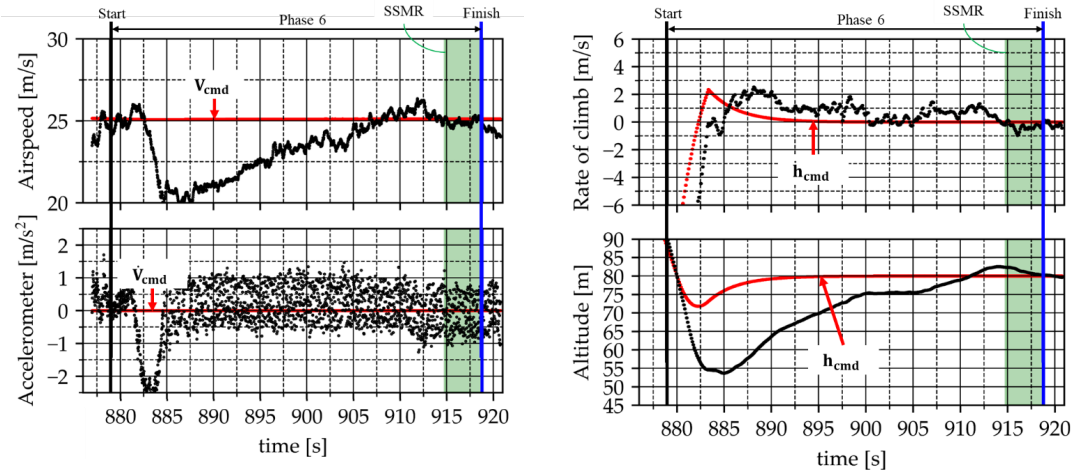


(**a**) (**b**)

Figure 30. Automatic determination results for Phase 5 in flight experiments: (**a**) Time histories of state variables used for steady-state evaluation. (**b**) Supplementary information ($\dot{h}(t)$, $h_{cmd}(t)$ and h(t)).

4.2.4. Phase 6: Automatic Determination Results for T_{trim} (Flight Experiments Results)

We show the time histories of the state variables **x**(t) subject to steady-state evaluation in Phase 6 in Figure 31. The meaning of the legends in the figure is the same as in Section 4.2.1. As shown in Figure 31, similarly to the simulation results in Figure 18, we confirmed that the test aircraft maintained steady level flight within the steady-state judgement range SSMR ($T_{obs,\ P6}$ = 4.0 s). Since steady state was judged, we defined the throttle rate at the end of Phase 6 as the trim throttle rate T_{trim} in accordance with the logic designed in Section 2.4.5. Consequently, T_{trim} in Phase 6 was automatically determined as 41.8%.



(**a**) (**b**)

Figure 31. Automatic determination results for Phase 6 in flight experiments: (**a**) Airspeed and longitudinal accelerometer data subject to steady-state evaluation. (**b**) Rate of climb and altitude subject to steady-state evaluation.

4.3. Compensation for Underestimation in Phase 2

As shown in Figure 28b, in the flight-experiment dataset adopted in this study, the maximum airspeed V_{max} automatically determined in Phase 2 was 26 m/s. Compared with 32 m/s that was automatically determined in a previous experiment conducted under the same conditions to validate the proposed method, the desired value was not automatically determined. During AUTO-mode flight, the airspeed sensor status was temporarily judged as Fail, and the standard ArduPilot calibration process was executed. While this process was running, the Phase 2 logic timed out, and the maximum airspeed was underestimated. Consequently, V_{max} was fixed at 26 m/s as V^1_{acc}.

As can also be confirmed in the flight-test video provided in the Supplementary Materials, the adverse weather accompanied by snowfall during the experiment likely contributed to this outcome. In general, pitot-tube blockage or icing is known to cause erroneous airspeed indications and can lead to airspeed anomalies [34]. In addition, even in rainfall, if water ingresses and cannot be discharged due to blockage of the drain section, temporary pitot-tube blockage can also result in erroneous airspeed readings [35]. For this reason, we did not use the Phase 2 result shown in Figure 28b as is; instead, we adopted 32 m/s, which was obtained as the automatically determined V_{max} in the previous experiment conducted under the same conditions. ArduPilot also provides a function that automatically adjusts the airspeed scale factor in flight, and it can update the parameter called ARSPD_RATIO through repeated flight paths and multiple turns [36].

4.4. Consistency of $\dot{h}_{sink}^{max}$ in Phase 4 Under the Compensated V_{max}

Using the maximum airspeed V_{max} = 32 m/s supplemented in 4.3 and $\theta_{min} = -17.5$ deg. automatically determined in Phase 4, we derived the maximum sink rate $\dot{h}_{sink}^{max}$ as −9.62 m/s. However, for this supplemented value, the minimum sink rate $\dot{h}_{sink}^{min}$ automatically determined in Phase 5 became −10.47 m/s, and thus $\dot{h}_{sink}^{max} < \dot{h}_{sink}^{min}$ occurred. Here, $\dot{h}_{sink}^{max}$ is given by Equations (14)–(16). Therefore, because the adopted experimental dataset approximated the flight-path angle γ using the minimum pitch angle θ_{min}, it did not account for the angle of attack α during descent. Accordingly, we corrected $\dot{h}_{sink}^{max}$ using the maximum value of the angle of attack observed in the steady gliding segment of Phase 5, α_{sink}^{max}. Note that this correction did not use Equation (16); instead, we used Equation (19), derived from Equations (14) and (15). We adopted the maximum value in order to provide a conservative (safety-side) upper bound for the resulting value.

$$\dot{h}_{sink}^{max} = V_{max} \cdot \sin(\theta_{min} - \alpha_{sink}^{max}) \tag{19}$$

Additionally, the flight experiment observed α_{sink}^{max} = 4.74 deg. From the above, using V_{max} = 32 m/s and $\theta_{min} = -17.5$ deg., the correction value $\dot{h}_{sink}^{max}$ was derived as −12.1 m/s.

4.5. First-Order Effect of Angle-of-Attack Variations on Maximum Sink Rate Estimation

In this subsection, we evaluated, using a first-order approximation, the effect of variations in the angle of attack α used in Equation (19) on the correction of the maximum sink rate $\dot{h}_{sink}^{max}$. The first-order approximation is based on linearization by performing a first-order Taylor expansion of the output around the input and neglecting higher-order terms. This concept is organized as the definition of sensitivity coefficients in the uncertainty propagation law [37].

It is recommended that $\dot{h}_{sink}^{max}$ be set so that it does not exceed the vertical speed that can be maintained under the conditions of minimum throttle rate, minimum pitch angle θ_{min}, and maximum airspeed V_{max} [7]. Therefore, in this subsection, we treat V_{max} = 32 m/s as a constant. We also treat $\theta_{min} = -17.5$ deg., which was determined in Phase 4, as a constant.

Using the relationships in Equations (14) and (15) in Section 2.4.3 and applying first-order linearization around the reference point α_0, we can approximate the sink-rate variation $\Delta\dot{h}$ with respect to the angle-of-attack variation $\Delta\alpha = \alpha - \alpha_0$ by Equations (20) and (21) as follows.

$$\Delta\dot{h} \approx -V_{max} \cdot \cos\gamma_0 \cdot \Delta\alpha \tag{20}$$

$$\gamma_0 = \theta_{min} - \alpha_0 \tag{21}$$

Here, because $|\cos\gamma_0| \leq 1$, we can evaluate the upper bound of the effect using Equation (22) as follows.

$$\left|\Delta\dot{h}\right| \leq V_{min} \cdot |\Delta\alpha| \tag{22}$$

As described in Section 4.4, the angle of attack observed during the descent in Phase 5 had a maximum of 4.74 deg. From the measured values in the flight experiment, the minimum and mean were −0.50 deg. and 2.69 deg., respectively. When we set α_0 = 2.69 deg., the deviation on the maximum side was 2.05 deg., and the deviation on the minimum side was 3.19 deg. Substituting these values into Equation (22), the first-order effect due to variations in the angle of attack ranged from approximately 1.15 m/s to 1.78 m/s. Therefore, we confirmed that the treatment of the angle of attack can have a non-negligible effect on the correction of $\dot{h}_{sink}^{max}$. As described in Section 4.4, in this study, we used α_{sink}^{max} = 4.74 deg. to

maintain $\dot{h}_{sink}^{max} < \dot{h}_{sink}^{min}$ and to avoid underestimation. In contrast, in a high-speed descent, the angle of attack may be smaller than that in steady glide. Therefore, this correction value may estimate $\dot{h}_{sink}^{min}$ conservatively.

4.6. Comparison Between SITL Simulation and Flight-Experiment Results

SITL simulation and the flight experiment showed an approximately 3.4-fold difference in the minimum sink rate $\dot{h}_{sink}^{min}$ in Phase 5 of the flight experiment compared with the simulation value. We consider that this difference originated from the limited fidelity of the SITL simulation in reproducing the real aircraft's propulsion system and aerodynamic characteristics. Specifically, in the aircraft model used in X-Plane, the propulsion system was implemented as an electric motor, whereas the real aircraft used an internal combustion engine. Because this inherently changes the relationship among throttle input, rotational speed, and thrust, we referred to the rated power of the real engine reported in the catalog [38] and adjusted the X-Plane settings so that the maximum power approximately matched. However, even if the maximum power is matched, it is difficult to reproduce, with the same accuracy as the real aircraft, the torque–rotational-speed characteristics, the rotational-speed changes due to propeller loading, and the transient response. As a result, even under the same throttle command, not only the absolute thrust level but also the thrust gradient can differ from those of the real aircraft. In addition, in the low-thrust regime during descent, previous studies reported that the drag can vary depending on the thrust level [39]. This implies that the interaction between the propulsion system (including the propeller slipstream) and the airframe can be readily reflected in the difference in the determined $\dot{h}_{sink}^{min}$. Moreover, icing on the airframe due to snowfall during the experiment may also have contributed to changes in the aerodynamic characteristics.

In contrast, in Section 3.4, we statistically verified and evaluated, using SITL simulation results obtained under varied wind-disturbance conditions, the effects of wind on the feasibility of steady-state segments and on the error distributions. However, to examine whether the large differences observed between the SITL simulation and the experiment originated from the presence or absence of wind disturbances, we need to compare the two under disturbance conditions close to those on the experiment day. Therefore, using the Japan Meteorological Agency weather data [33], we matched the date, time, and location of the experiment with the in-flight wind disturbances and compared them with the information corresponding to WindSpd, WindDir, and Gust used in Table 8. As a result, in terms of the total wind speed, the condition in Table 8 that was closest to the wind disturbances on the experiment day was CaseID = 17. Here, Table 12 shows CaseID = 1 (no wind disturbance) and CaseID = 17 in Table 8, together with the wind disturbances on the experiment day. In addition, initially, approximating the angle of attack α as zero caused an inconsistency in the magnitude relationship between the maximum and minimum sink rates. We resolved this issue by discarding the approximate calculation and considering α strictly. Finally, Table 13 shows the comparison between the corrected flight-experiment results and the SITL simulation results.

Table 12. Comparison of wind disturbance conditions.

Conditions	WindSpd [m/s]	WindDir [deg.]	Gust [m/s]	Total WindSpd [m/s]
SITL Simulation (CaseID = 1)	0	0	0	0
SITL Simulation (CaseID = 17)	10	0	2	12
Experiment	5.2	0	9	14.2

Table 13. Comparison of corrected flight-experiment and SITL simulation results for automatically determined TECS parameters.

Phase	Parameter	Unit	SITL Simulation (CaseID = 1)	SITL Simulation (CaseID = 17)	Experiment
1	V_{min}	m/s	13	22	20
2	V_{max}	m/s	36	31	32
3&4	$\dot{h}_{max}$	m/s	9.10	8.16	7.87
	θ_{max}	deg.	21.5	18.5	22.5
	θ_{min}	deg.	−16.5	−13.5	−17.5
	$\dot{h}_{sink}^{max}$	m/s	−10.1	−13.0	−12.1
5	$\dot{h}_{sink}^{min}$	m/s	−3.10	−2.85	−10.5
6	T_{trim}	%	37.1	44.1	41.8

From Table 13, when we compare the SITL simulation results with wind disturbances applied (CaseID = 17) with the flight experiment, we confirmed that the differences in V_{min} and V_{max} became smaller than those in the SITL simulation results without wind disturbances (CaseID = 1). However, the difference in $\dot{h}_{sink}^{min}$ still did not improve even when the wind-disturbance condition changed. These results confirmed, also in the flight experiment, that V_{min} and V_{max} shift toward a safer speed region as wind disturbances increase, as described in Section 3.4.2. In addition, as described in Section 3.4.3, because the influence of wind disturbances on the determination of $\dot{h}_{sink}^{min}$ is considered to be small, model mismatch in the propulsion system and differences in aerodynamic characteristics are likely to contribute to the difference in $\dot{h}_{sink}^{min}$.

5. Conclusions

We focused on the TECS parameters used for longitudinal control of a fixed-wing UAV. The objective of this study was to automatically determine the TECS parameters stepwise during flight and to reduce reliance on conventional manual tuning. The proposed method sequentially executed multiple flight phases; in each Phase, it acquired onboard state variables, evaluated steady-state conditions, and determined the TECS parameter values.

In the implementation of the proposed method, we used the Lua Scripts feature of ArduPilot Plane 4.6. Without substantially modifying the core TECS control law written in C++, we integrated state acquisition, steady-state judgement, parameter overwriting, and phase transitions on the Lua side. We also designed the system to handle start, interruption, and resumption by the operator, and we additionally leveraged an existing flight plan and climb flight using the TAKEOFF command. With this configuration, we consider that the proposed method reduces reliance on individual expertise in determining TECS parameters and lowers the time cost required for TECS-parameter adjustment. The proposed method does not estimate performance using aerodynamic coefficients or a thrust model. Therefore, we detected steady-state segments from in-flight state variables and empirically determined the TECS parameters with respect to the performance limits of each aircraft. Accordingly, even if aircraft size, wing loading, and propulsion type differ, we consider that the Phase segmentation and the determination logic itself can be applied within the same framework. In TECS operation as well, it is recommended to measure and set V_{min}, V_{max}, and $\dot{h}_{max}$, among others, in actual flight for the target aircraft [15]. This study positions itself as automating that procedure.

In contrast, the allowable MAE threshold components ε_i used for the steady-state judgement depend on the effective disturbances of the real aircraft, including sensor

noise and actuator delays. Therefore, when porting the method to a different aircraft, it is necessary to recalculate ε_i from straight-and-level steady-flight data of the target aircraft. In addition, for aircrafts with slower propulsion-system responses, we consider it desirable to adjust the observation window length and the timeout settings related to the steady-state judgement.

From the above, this study establishes stepwise automatic determination of TECS parameters on a real aircraft in real time through a lightweight Lua-based extension. In addition, we show that the method can be realized without substantially modifying the core control logic and that the determined values are consistent with flight dynamics. Therefore, the proposed method provides a framework to automate TECS adjustment in a form that is easy for operators to understand.

Supplementary Materials: The following supplementary video is available at Zenodo: Video S1: Inflight automatic determination of TECS parameters. Available at: https://doi.org/10.5281/zenodo.18183655.

Author Contributions: Conceptualization, R.F.; methodology, R.F.; software, R.F.; validation, R.F., K.H.; formal analysis, R.F.; investigation, R.F.; resources, K.H., M.H.; writing—original draft, R.F.; writing—review and editing, K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC were funded by the Aerospace Plane Research Center (APReC), Muroran Institute of Technology; funding number: not applicable.

Data Availability Statement: The Lua scripts and associated source-code modifications implementing the proposed method are publicly available on GitHub and are archived on Zenodo (DOI: 10.5281/zenodo.18227833; Available at: https://doi.org/10.5281/zenodo.18227833).

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

Nomenclature

V	Airspeed, m/s	T_{trim}	Cruise (trim) throttle setting, %
$V(t)$	Airspeed time history, m/s	T_{ref}	Reference throttle used in evaluation, %
$\dot{V}(t)$	Longitudinal acceleration (time derivative of V), m/s^2	$TSWT(t)$	Switching rule for TECS_SPDWEIGHT, —
V_{min}	Minimum airspeed, m/s	t_{glide}	Switching time to the gliding segment in Phase 5, s
V_{max}	Maximum airspeed, m/s	$\mathbf{x}(t)$	State vector used for steady-state evaluation, component-dependent
V_{cruise}	Cruise airspeed, m/s	$\mathbf{x}[k]$	Sampled state at $t_0 + k\Delta t$, component-dependent
V^{ref}	Reference airspeed used for steady-state evaluation, m/s	$\mathbf{e}[k]$	Error vector $\mathbf{e}[k] = \mathbf{x}[k] - \mathbf{x}_{ref}$, component-dependent
V_R	Reference/threshold airspeed used inPhases 3–4, m/s	$e_i[k]$	i-th component of $\mathbf{e}[k]$, component-dependent
h	Altitude, m	J_i	MAE of the i-th component over the observation window, component-dependent
$h(t)$	Altitude time history, m	J_V	MAE associated with airspeed, m/s
$\dot{h}(t)$	Rate of climb, m/s	$J_{\dot{V}}$	MAE associated with longitudinal acceleration, m/s^2
$\dot{h}_{max}$	Maximum rate of climb, m/s	$J_{\dot{h}}$	MAE associated with climb rate, m/s
$\dot{h}_{sink}(t)$	Sink rate time history, m/s	J_h	MAE associated with altitude, m
$\dot{h}_{sink}^{max}$	Maximum sink rate, m/s	$\boldsymbol{\varepsilon}$	Allowable MAE threshold vector, component-dependent
$\dot{h}_{sink}^{min}$	Minimum sink rate, m/s	ε_V	Allowable MAE threshold for airspeed, m/s
$h_{cmd}(t)$	Altitude demand (command), m	$\varepsilon_{\dot{V}}$	Allowable MAE threshold for longitudinal acceleration, m/s^2
θ	Pitch angle, deg.	$\varepsilon_{\dot{h}}$	Allowable MAE threshold for climb rate, m/s

$\theta(t)$	Pitch angle time history, deg.	ε_h	Allowable MAE threshold for altitude, m
θ_{max}	Maximum pitch angle, deg.	ε_i	Allowable MAE threshold for i-th component, component-dependent
θ_{min}	Minimum pitch angle, deg.	T_{obs}	Observation time for MAE computation, s
$\theta_{cmd}(t)$	Commanded pitch angle, deg.	Δt	Sampling interval, s
θ_{R0}	Initial climb pitch angle used for scheduling, deg.	t	Time, s
$\theta_{min,climb}$	Lower bound for $\theta_{cmd}(t)$ during climb, deg.	t_0, t_f	Start/end times of the evaluation window, s
$\Delta\theta_{margin}$	Safety margin, deg.	k	Discrete-time sample index, —
α	Angle of attack, deg.	i	State-component index, —
$\alpha_{sin\,k,max}$	Estimated AoA atmaximum-sink condition, deg.	$V_{dec}^{N_{dec}}$	Reference airspeed at deceleration step N_{dec}, m/s
γ	Flight-path angle, deg.	$V_{acc}^{N_{acc}}$	Reference airspeed at acceleration step N_{acc}, m/s
T	Throttle command, %	N_{dec}, N_{acc}	Step indices for deceleration/acceleration tables, —
T_{min}	Minimum throttle setting, %	l, m	Total numbers of steps in deceleration/acceleration tables, —

References

1. Unmanned Aircraft. "LEVEL 4" Flight Web Portal. Available online: https://www.mlit.go.jp/koku/level4/en/ (accessed on 20 January 2026).
2. Colomina, I.; Molina, P. Unmanned Aerial Systems for Photogrammetry and Remote Sensing: A Review. *ISPRS J. Photogramm. Remote Sens.* **2014**, *92*, 79–97. [CrossRef]
3. Cai, G.; Dias, J.; Seneviratne, L. A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends. *Unmanned Syst.* **2014**, *2*, 175–199. [CrossRef]
4. Watts, A.C.; Ambrosia, V.G.; Hinkley, E.A. Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. *Remote Sens.* **2012**, *4*, 1671–1692. [CrossRef]
5. Liu, Y.; Wang, Y.; Li, H.; Ai, J. Runway-Free Recovery Methods for Fixed-Wing UAVs: A Comprehensive Review. *Drones* **2024**, *8*, 463. [CrossRef]
6. Delavarpour, N.; Koparan, C.; Nowatzki, J.; Bajwa, S.; Sun, X. A Technical Study on UAV Characteristics for Precision Agriculture Applications and Associated Practical Challenges. *Remote Sens.* **2021**, *13*, 1204. [CrossRef]
7. ArduPilot. Available online: https://ardupilot.org (accessed on 30 January 2026).
8. Aliane, N. A Survey of Open-Source UAV Autopilots. *Electronics* **2024**, *13*, 4785. [CrossRef]
9. Kaizu, Y.; Kimura, M. Development of Unmanned Surface Vehicle Using ArduPilot [Translated Japanese]. *J. Jpn. Soc. Agric. Mach. Food Eng.* **2021**, *83*, 145–149.
10. Li, P.; Liu, D.; Baldi, S. Plug-and-Play Adaptation in Autopilot Architectures for Unmanned Aerial Vehicles. In *Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society*; IEEE: Toronto, ON, Canada, 2021; pp. 1–6.
11. Automatic Tuning with AUTOTUNE—Plane Documentation. Available online: https://ardupilot.org/plane/docs/automatic-tuning-with-autotune.html (accessed on 30 January 2026).
12. Matt, J.J.; Flanagan, H.; Chao, H. Evaluation and Analysis of ArduPilot Automatic Tuning Algorithm for the Roll Tracking Controller of a Small UAS. In Proceedings of the AIAA Scitech 2021 Forum, Virtual Event, 11–15, 19–21 January 2021; pp. 11–15.
13. Faleiro, L.F.; Lambregts, A.A. Analysis and Tuning of a "Total Energy Control System" Control Law Using Eigenstructure Assignment. *Aerosp. Sci. Technol.* **1999**, *3*, 127–140. [CrossRef]
14. Jimenez, P.; Lichota, P.; Agudelo, D.; Rogowski, K. Experimental Validation of Total Energy Control System for UAVs. *Energies* **2019**, *13*, 14. [CrossRef]
15. TECS (Total Energy Control System) for Speed and Height Tuning Guide—Plane Documentation. Available online: https://ardupilot.org/plane/docs/tecs-total-energy-control-system-for-speed-height-tuning-guide.html (accessed on 30 January 2026).
16. Plane: Autotune for TECS · Issue #4821 · ArduPilot/Ardupilot. Available online: https://github.com/ArduPilot/ardupilot/issues/4821https://github.com/ArduPilot/ardupilot/issues/4821 (accessed on 30 January 2026).
17. Lua Scripts—Copter Documentation. Available online: https://ardupilot.org/copter/docs/common-lua-scripts.html (accessed on 30 January 2026).
18. SITL Simulator (Software in the Loop)—Dev Documentation. Available online: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html (accessed on 30 January 2026).
19. ArduPilot at ArduPilot-4.6 GitHub Repository. Available online: https://github.com/ArduPilot/ardupilot/tree/ArduPilot-4.6 (accessed on 30 January 2026).

20. Kephart, R.; Plamowski, S.; Domański, P.D. Effects of Leading Signals on Metrics of Control Quality Indicators. *Appl. Sci.* **2023**, *13*, 5720. [CrossRef]
21. Watanabe, K.; Shibata, T.; Ueba, M. Derivation and Flight Test Validation of Maximum Rate of Climb during Takeoff for Fixed-Wing UAV Driven by Propeller Engine. *Aerospace* **2024**, *11*, 233. [CrossRef]
22. Lammen, W.; Dewitte, P.-J.; Scheers, E. Retrofitted Hydrogen-Electric Propulsion Aircraft: Performance Simulation of Critical Operating Conditions. *Aerospace* **2025**, *12*, 95. [CrossRef]
23. Panagiotopoulos, I.; Sakellariou, L.; Hatziefremidis, A. Design, Construction, and Flight Performance of an Electrically Operated Fixed-Wing UAV. *Drones* **2024**, *8*, 217. [CrossRef]
24. Flight Simulator | X-Plane 12: Flight Simulation Done Right. Available online: https://www.x-plane.com/ (accessed on 30 January 2026).
25. ArduPilot/MissionPlanner. Available online: https://github.com/ArduPilot/MissionPlanner (accessed on 30 January 2026).
26. RC Calmato Alpha 40 Sport UAV. Available online: https://forums.x-plane.org/files/file/21159-rc-calmato-alpha-40-sport-uav/ (accessed on 30 January 2026).
27. 14 CFR 23.337 (29 August 2017)—Limit Maneuvering Load Factors. Available online: https://www.ecfr.gov/on/2017-08-29/title-14/part-23/section-23.337 (accessed on 30 January 2026).
28. vJoySerialFeeder. Available online: https://github.com/Cleric-K/vJoySerialFeeder/releases (accessed on 30 January 2026).
29. Shen, T.-J.; Chen, C.-L. Model-in-the-Loop Design and Flight Test Validation of Flight Control Laws for a Small Fixed-Wing UAV. *Drones* **2025**, *9*, 624. [CrossRef]
30. Golder, E.R.; Settle, J.G. The Box-Müller Method for Generating Pseudo-Random Normal Deviates. *J. R. Stat. Soc. Ser. C. Appl. Stat.* **1976**, *25*, 12–20. [CrossRef]
31. Umeda, T. Generation of Normal Distributions Revisited. *Comput. Stat.* **2024**, *39*, 3907–3921. [CrossRef]
32. ArduPilot/UAVLogViewer. Available online: https://github.com/ArduPilot/UAVLogViewer (accessed on 30 January 2026).
33. Japan Meteorological Agency. Available online: https://www.jma.go.jp/jma/indexe.html (accessed on 1 February 2026).
34. Unreliable Airspeed Indication and Stall Warning | ATSB. Available online: https://www.atsb.gov.au/media/news-items/2019/unreliable-airspeed-indication (accessed on 30 January 2026).
35. Ratvasky, T.P.; Strapp, J.W.; Lilie, L.; Bansemer, A.; Chen, R.-C. Air Data Probe Anomalies in Flight Through Measured High Ice Water Content Conditions. In *Proceedings of the AIAA AVIATION FORUM AND ASCEND 2024*; American Institute of Aeronautics and Astronautics: Las Vegas, NV, USA, 2024.
36. Calibrating an Airspeed Sensor—Plane Documentation. Available online: https://ardupilot.org/plane/docs/calibrating-an-airspeed-sensor.html (accessed on 5 February 2026).
37. NIST TN 1297: Appendix A. Law of Propagation of Uncertainty. *NIST* 2015. Available online: https://www.nist.gov/pml/nist-technical-note-1297/nist-tn-1297-appendix-law-propagation-uncertainty (accessed on 1 February 2026).
38. MAX-46AX II. Available online: https://www.os-engines.co.jp/line_up/engine/air/aircraft/air_catalog/15490.html (accessed on 1 February 2026).
39. Norris, J.; Bauer, A.B. Zero-Thrust Glide Testing for Drag and Propulsive Efficiency of Propeller Aircraft. *J. Aircr.* **1993**, *30*, 505–511. [CrossRef]