doi: 10.32620/oikit.2025.104.12

UDC 629.7.014.16

D. V. Sokol, A. B. Zhukevych, H. A. Miroshnychenko

Application of Property Proving to Quadcopter Flight Control Software

National Aerospace University "Kharkiv Aviation Institute"

The object of the study is a computer model of an automatic control system for a quadcopter. The subject matter of the research encompasses the application of formal methods for analyzing and verifying requirements for quadcopter control systems, with a particular focus on the capabilities of the Simulink Design Verifier™ tool. The primary goal of the study is to enhance the overall reliability of the quadcopter model by applying formal analysis techniques at the design stage to uncover potential logical and structural errors before physical implementation. The tasks to be solved: conducting a comprehensive review of challenges associated with the operation of commercial quadcopter platforms; formulating safety-oriented requirements for quadcopter flight control; developing a detailed simulation model of the control system using the Simulink® environment; utilizing the Property Proving functionality to perform formal verification of critical system properties; analyzing and interpreting the verification reports generated by the tool; and identifying latent design flaws or inconsistencies that may compromise system safety. The study employed the following methods: formal verification using the Property Proving tool, simulation-based modeling, static code analysis, and model-based testing. As a result of the research, it was demonstrated how formal verification techniques, such as Property Proving, can be applied to validate safety-critical behaviors of quadcopter control systems. The use of Simulink Design Verifier™ proved effective in identifying design weaknesses early in the development cycle, reducing downstream risk and rework. Additionally, the generation of interactive diagnostic reports facilitated the visualization of failure scenarios and supported iterative debugging. Conclusions. The application of formal analysis tools such as Simulink Design Verifier™ represents a valuable approach to strengthening quadcopter control system reliability, if models are properly constructed and properties are carefully defined. Although limitations exist – particularly concerning compatibility with certain Simulink® blocks – the tool remains a powerful complement to traditional testing, especially when addressing high-assurance system requirements. The study underscores the necessity of integrating formal methods into standard verification workflows to balance theoretical rigor with practical validation.
***Keywords**:* UAV; quadcopter; control system; design; static analysis; property proving; computer model.

Introduction

Currently, the development of UAV control systems has significantly increased in value and importance. Quadcopters are now considered expensive expendable assets. Some individuals build their own models, while others upgrade existing popular products, as publicly available quadcopters are vulnerable to external disturbances (electronic warfare, firearms, and signal jammers, anti-drone actions, etc.) [1].

In the event of external interference with control, original models often cannot resist such disruptions and become uncontrollable. By assessing flight tasks, external factors, and the UAV's performance limits, relevant upgrade objectives are formulated, such as developing custom firmware or providing an alternative control channel.

At present, the main methods of minimizing decision-making risks using quadcopters include relying on predefined control rules, expert prior knowledge, and regularization constraints. However, these methodologies require quadcopters to meet strict preliminary conditions, such as acquiring extensive decision-making experience and establishing comprehensive rules.

Adhering to industrial standards and design protocols for quadcopters is often impractical due to significant limitations in both material and time resources. Nevertheless, abandoning preliminary design and a series of computer experiments is also unreasonable. Non-compliance with formal requirements often leads to frequent quadcopter crashes under uncertain conditions and subsequent mission failures. Thorough development of the computer model and the use of appropriate tools for its formal analysis are key to creating a more reliable physical prototype of quadcopter.

Upgrade – or especially development of a new quadcopter – is based on a list of current issues that hinder the operation of original models. For example, the paper [2] considers the implementation of a quadcopter automatic flight control system into an existing training model. As the authors note, the results confirm that such implementations can meet performance requirements under realistic conditions, underscoring the importance of software-level optimization and correctness.

Due to limited resources, developers conduct series of (iterative) tests and trials before a prototype can be finalized: verifying the operation of an algorithm in isolation from the full control system, evaluating the performance of an alternative control channel in combination with the main one. In any case, testing is a mandatory and inevitable stage in product development: the more diverse the tests, the more reliable the result will be.

In [3], the authors present a method for autonomous decision-making for UAVs based on safe reinforcement learning. The key idea lies in the adaptive selection of control strategies depending on the level of risk faced by the UAV in various flight situations. The proposed approach enables the system to dynamically switch between safety policies, thereby increasing resilience to environmental uncertainties and external disturbances. However, this work does not address the formal verification of system behavior correctness during transitions between strategies. There is no guarantee that, under any combination of risk factors, the UAV will always make a decision that ensures a safe outcome. This leaves the question of provable safety of such systems open, especially in the context of safety-critical scenarios.

In many cases, UAV control systems are designed heuristically or based on rigid logic, without formal proof of the correctness of transitions. The absence of formal guarantees that the system will always and correctly switch to a safe mode under all possible scenarios remains a significant issue.

As an example of a complex and testing-intensive control system, study [4] can be cited. It addresses the task of determining the safe flight envelope for UAVs based on the analysis of a variety of potential external destabilizing factors. The main contribution of the work lies in the development of a theoretical model that enables quantitative assessment of such factors' impact on flight stability and controllability. The authors emphasize that due to limitations in the experimental base and the lack of necessary conditions for field tests, they were forced to rely solely on mathematical modeling and simulation of flight scenarios. However, the question remains open as to how accurately the obtained results reflect the behavior of a real UAV under multi-component disturbances and environmental uncertainties. The simulation of isolated scenarios, in essence, covers only a narrow range of possible situations, leaving potentially critical combinations of external influences beyond the scope of analysis. Thus, despite the usefulness of the approach, the lack of guaranteed completeness in scenario coverage limits the applicability of the results for tasks requiring formal safety assurance, especially in systems with automatic return or mode-switching mechanisms. This underscores the need for formal verification methods capable of

proving correct system behavior under all possible conditions, not just those manually tested.

Study [5] presents the results of full-scale (field) experiments with UAVs, which undoubtedly adds practical value to the research. The authors demonstrate that even with limited modeling accuracy or assumptions in flight conditions, it is possible to obtain useful data on the actual system behavior. Moreover, field tests are particularly valuable in identifying hidden design flaws that are difficult to detect through numerical simulation. Nevertheless, despite its practical importance, this approach can be considered sufficient and complete only if the UAV control system has been tested across the entire range of operating conditions, including edge cases and emergency scenarios. Otherwise, the risk of incorrect system behavior in unforeseen situations remains, especially during automatic execution of critical decisions.

In papers [6, 7], the authors demonstrate that when model simplifications or limitations on the complexity of described conditions are allowed during the system design stage, the use of fuzzy logic becomes a justified and effective solution. Based on a dataset reflecting both the internal state of the UAV and the external environmental conditions, a fuzzy controller can form an intuitively understandable and adaptive control strategy.

Fuzzy logic has proven to be a powerful tool for designing decision-making logic, especially in systems where it is impossible to formulate precise mathematical models or transition rules between modes. It shows good compatibility with soft, context-dependent requirements that are often imposed on autonomous UAV control systems. However, fuzzy logic cannot provide guarantees that, under all possible scenarios, the system will necessarily and correctly switch to the appropriate mode.

In cases where field experiments do not require complex preparation and the tests themselves are conducted quickly and relatively easily; an iterative design approach is especially effective. Its essence lies in accumulating a large body of experimental data for subsequent use in calibrating and validating mathematical models. This approach allows for model refinement as experience is gained and improves its reliability based on observed data.

For example, in study [8], the validation of lithium-polymer batteries for UAVs is carried out using an algorithm that analyzes results under repeatable conditions. Thanks to the possibility of isolating the battery from the rest of the system, the battery itself becomes the object of testing, simplifying data collection and analysis. The authors developed a results processing algorithm that accounts for experiment repeatability and provided extensive tables with the real-world results.

The proposed approach demonstrates high effectiveness when working with local subsystems, such as power modules, where direct verification of characteristics is possible. In such an iterative method, it is important to accumulate the maximum amount of data under as many diverse experimental conditions as possible. Only in this way can the correctness of the system be at least partially ensured in the absence of formal verification.

1. Objectives and Approach

Regular tests cover only a minimal range of potential scenarios, whereas the broader the coverage, the higher the probability of detecting a development error. In addition to testing, which provides specific information about the state of the model, it is recommended to perform static analysis using specialized tools. This approach allows for the identification of requirement violations and other issues that may affect

the quality of the control system – essentially by iterating through all possible combinations of input data and comparing them with the system's behavior.

However, even static analysis has its limits when dealing with highly nonlinear or state-dependent systems such as UAV controllers. This is where formal verification techniques, such as Property Proving, offer a significant advantage.

These methods do not rely on scenario sampling or manual test design but instead attempt to mathematically prove whether a given property holds under all possible conditions. This is particularly crucial in avionics, where any design error can result in mission failure or safety violations.

Figure 1 illustrates the areas of system functionality coverage using simulation-based testing, comprehensive analysis, and feasibility analysis.

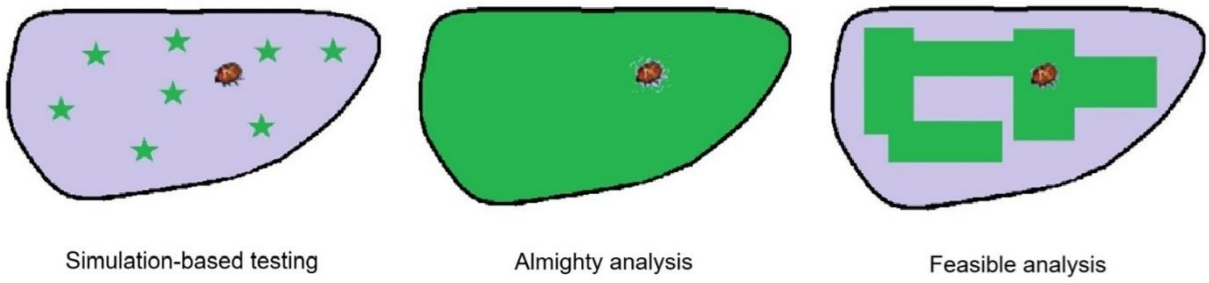

Fig. 1. Areas of analysis of the functioning of the control system

Fortunately, most technologies and tools for solving the described problem have either built-in functionality or are compatible with other software products that support industry standards. And then the developer's task at this stage will be to correctly represent the computer model, appropriately formulate the requirements and qualitatively analyze the results.

This paper focuses on formally verifying the return functionality as a critical safety scenario. However, the method is not limited to this single behavior; it is extensible to other failure modes, including loss of communication, battery discharge, sensor faults etc. So that, such flexibility allows comprehensive coverage of safety requirements and supports the development of more resilient UAV control systems.

The goal is not only to confirm the correctness of a given property but also to show how formal tools can be applied iteratively as new safety functions are introduced.

2. Materials and methods of research

In the field of control system development, the primary technology is Simulink® [9]. Therefore, this work focuses on the Simulink Design Verifier™ application, specifically its formal analysis tool, Property Proving, which supports industry standards [10].

The model for automatic control system of a quadcopter is represented in Figure 2 [11]. The top-level model served as a structural container, with its behavior implemented via subsystems, each responsible for distinct components.

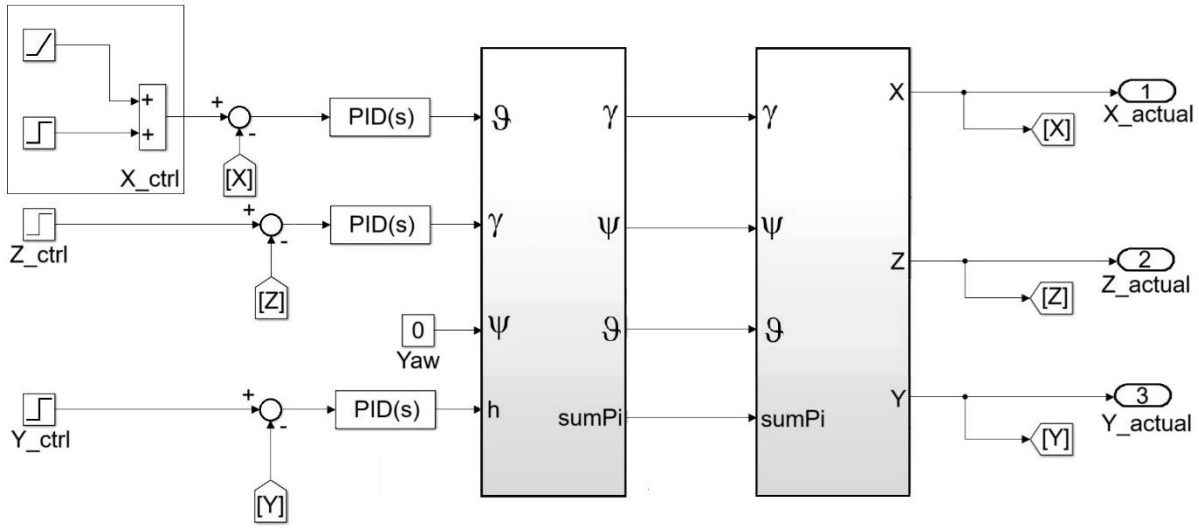

Fig. 2. Computer model of the original quadcopter control system

The quadcopter's motion is controlled by changing the motor rotation speed. To describe the quadcopter's motion dynamics, two coordinate systems were introduced: the normal terrestrial coordinate system and the principle (moving) coordinate system linked to the quadcopter's center of mass. The quadcopter's angular position is specified by Euler angles based on angular velocity information. A feature of quadcopter control is the generation of control signals for all six degrees of freedom using four motors that generate lift. In this regard, it is advisable to consider the quadcopter's position in space not by Euler angles, but by coordinates relative to the normal terrestrial coordinate system.

The original model describes complex movement across three control channels. The input to the model consists of desired displacement values along three axes (X_ctrl, Y_ctrl, Z_ctrl), which are then used to calculate Euler angles, angular velocities, and torques. The output signals represent the actual position of the quadcopter along the three axes (X_actual, Y_actual, Z_actual).

As part of the study, the following requirement is imposed on the original control system: if the quadcopter moves beyond the boundary distance from the pilot (1000 m), the control system must automatically switch to an autonomous mode and return to the pilot.

Figure 3 shows the computer model of the upgraded control system. The subject of the analysis is an automatic control subsystem of the quadcopter that implements the "Return-to-Home" behavior when a defined distance threshold is exceeded. This subsystem was modeled in Simulink® using a combination of chart-based logic and signal processing blocks. In the Figure 3 X_rth, Y_rth, Z_rth are expected coordinates from the "Return-to-Home" operation.

Figure 4 shows the "Return-to-Home" function of the upgraded quadcopter automatic control system.

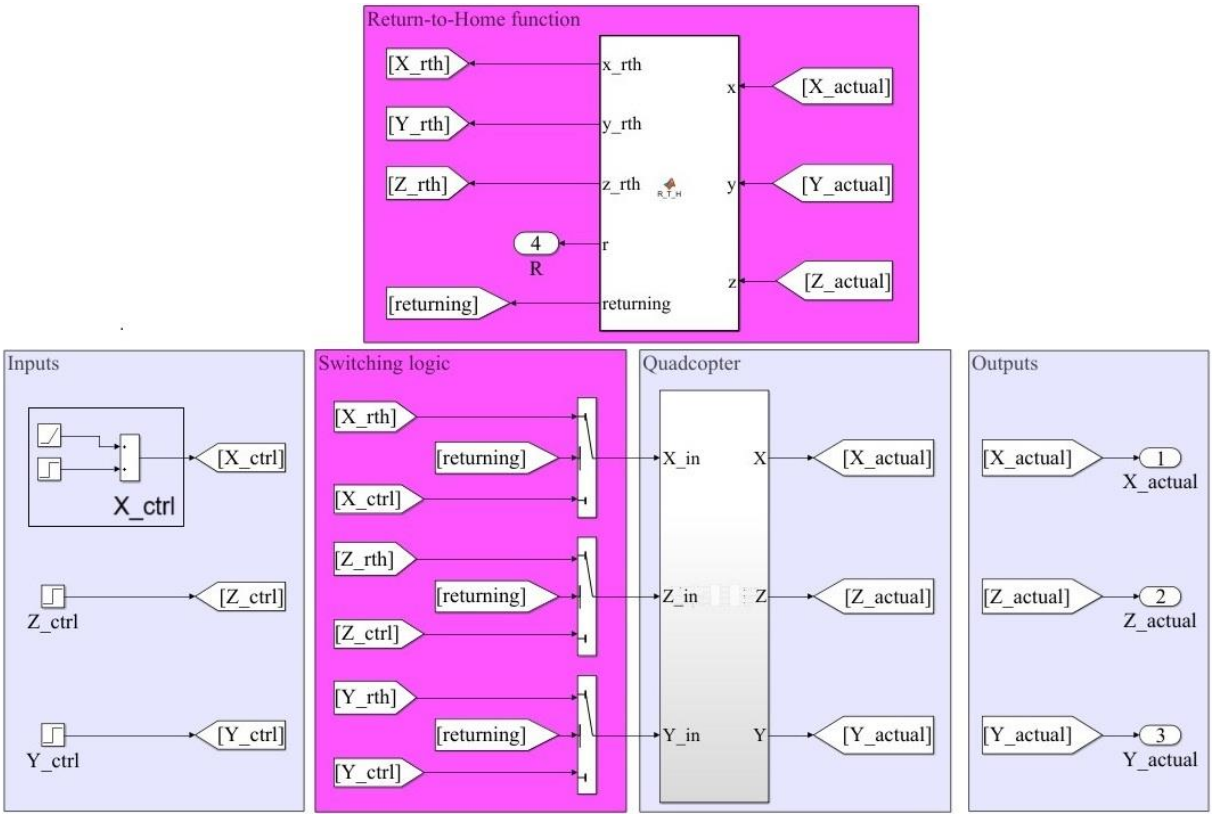

Fig. 3. Computer model of the upgraded quadcopter control system

```
function[x_rth,y_rth,z_rth,r,returning]=R_T_H(x,y,z)
pos = [x; y; z];
r = norm(pos);
threshold = 1000;
T_return = 100;
persistent t_now
if isempty(t_now), t_now = 0; end
if r > threshold
    returning = true;
    t_now = t_now + 1;
    scale = max(0, 1 - t_now / T_return);
    x_rth = pos(1) * scale;
    y_rth = pos(2) * scale;
    z_rth = pos(3) * scale;
else
    returning = false;
    x_rth = x;
    y_rth = y;
    z_rth = z;
end
end
```

Fig. 4. “Return-to-Home” function of the upgraded quadcopter control system

The function is applied to the model operating in three-dimensional space. It monitors the spatial coordinates of the quadcopter and determines whether it has exceeded a predefined operational boundary.

Specifically, the function receives the current position vector and calculates the Euclidean distance from the origin, which is assumed to represent the “home”. If this distance exceeds a defined threshold (1000 m), the function activates a return mechanism.

A persistent counter keeps track of how many steps the quadcopter has been outside the safe boundary. During each step, the position vector is scaled down linearly, moving the quadcopter progressively closer to the origin. As a result, the quadcopter is smoothly guided back within the acceptable operational range, rather than abruptly repositioned. Thus, this mechanism simulates a practical safety protocol for autonomous systems, ensuring controlled return behavior when the system moves beyond acceptable limits, thereby enhancing robustness and mission reliability.

The simulation results of the model with the original “Return-to-Home” function are shown in Figure 5.

As can be seen the return home trigger is indeed set to “1”, however, as soon as the distance is reduced to less than 1000 m, it resets to “0”.

Over the interval [290 360] sec the function works as expected, gradually reducing the quadcopter’s distance from the origin. However, once the quadcopter returns within the threshold and the return behavior deactivates. If the quadcopter later exceeds the threshold again, the function resumes using the old which causes the scaling factor to be immediately small or even negative. As a result, the “Return-to-Home” mechanism does not function correctly on subsequent violations because the system behaves as if it is already far along or even finished with the return sequence.

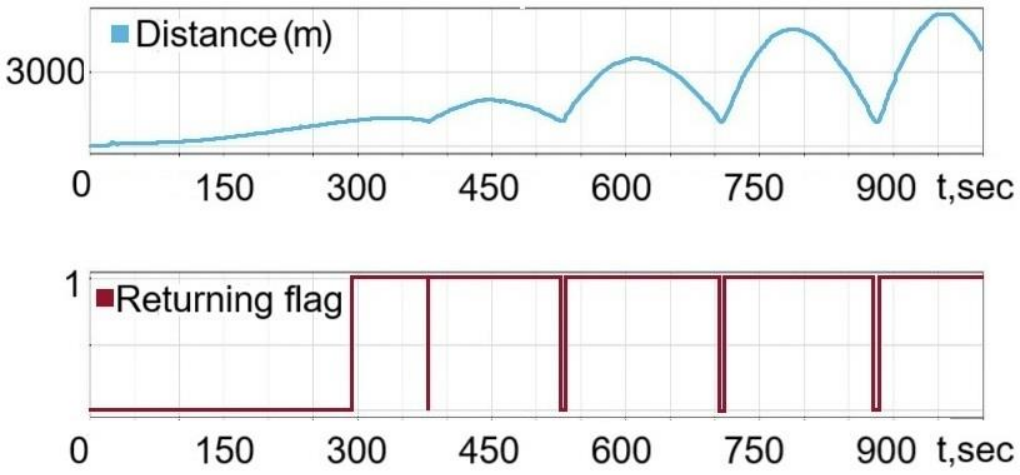

Fig. 5. Simulation results with the original function

The designer has two strategies to choose from:

- verify that a specified property (requirement) always holds for the model under all possible conditions. If the property can be proven, the designer gains assurance that the system behaves as expected with respect to that property. If it cannot be proven, it means that a formal proof could not be established – however, this does not necessarily imply that the property is false;
- identify a specific scenario or counterexample where the property does not hold. The tool searches for input conditions or sequences of events that lead to a violation of the property.

The verification direction gets explored by applying the Property Proving tool to

a quadcopter control model, identifying critical safety properties, and integrating formal analysis into the existing model-based design workflow.

To support traceability and reduce false positives, the results of the verification process were automatically parsed, and undecided or unreachable objectives may be either resolved through model updates or justified manually.

The main task of the described formal analysis is the proper interpretation of the established requirements in the form of a computer model. After all, it is the verbal and computer forms of requirements that are the standard for analyzing the control system. Thus, it is impractical to analyze the entire system, and then the model under analysis makes sense to form from the supplemented functionality (the area highlighted in red in Figure 3).

So that, the model under analysis is shown in Figure 6.

From the stated requirement, two properties can be formulated for analysis:

- if the quadcopter moves away from the pilot to a limiting distance (1000 m), the control system must be forced to switch to automatic mode – Property A;

- when the control system switches to automatic mode, the quadcopter should return to the pilot – Property B.

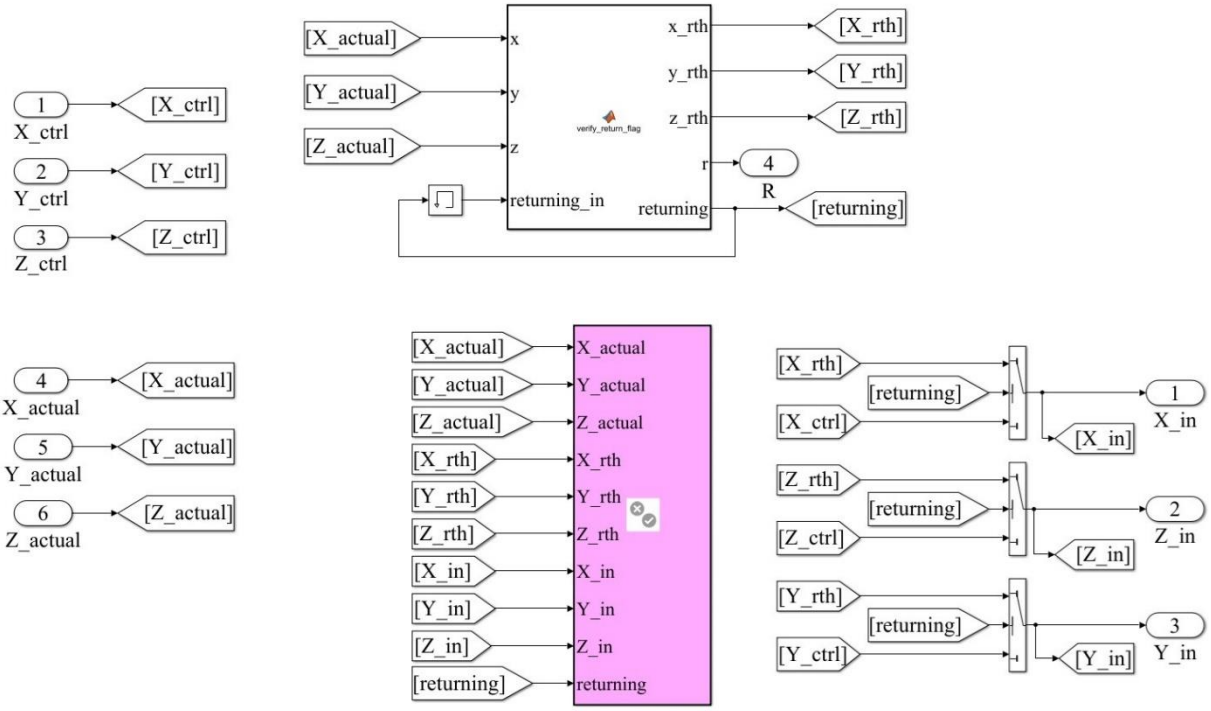

Fig. 6. Model under analysis

To formally verify the expected behavior, the Property Proving tool is used. It allowed the definition and checking of properties that guarantee the quadcopter, after exceeding a distance threshold, must eventually reduce its position vector below the threshold again, regardless of how many times the condition is triggered.

The implementation diagram of these two properties is shown in Figure 7.

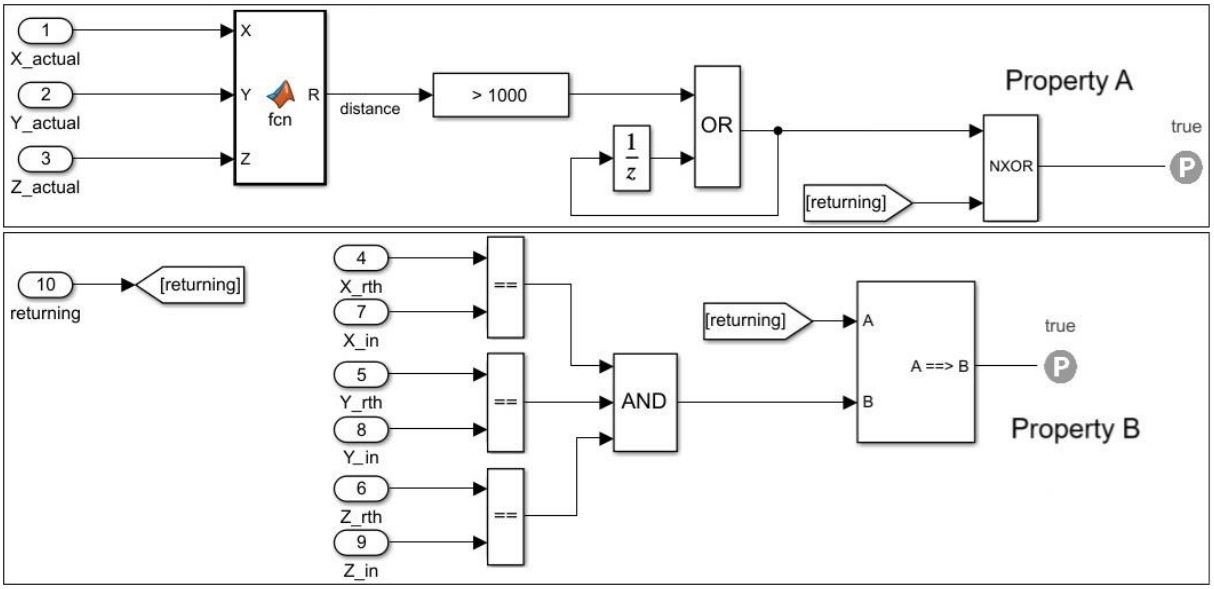

Fig. 7. Implementation of the properties

As can be seen from the Property Proving Report in Figures 8 and 9, the presented upgraded quadcopter control system does not meet the properties. Also, upon completion of the analysis, the report provides a counterexample with input values, for which the output parameters are false relative to the properties in Figure 8. At the same time, the correctness of the implementation of property B is confirmed by Figure 9.

Chapter 3. Proof Objectives Status

Content

Objectives Falsified with Counterexamples

Objectives Falsified with Counterexamples

#	Type	Model Item	Description	Analysis Time (sec)	Counterexample
1	Proof objective	Verification Subsystem/Proof Objective	Objective: T	12	1

Chapter 4. Properties

Content

Verification Subsystem/Proof Objective

Verification Subsystem/Proof Objective

Summary

Model Item: Verification Subsystem/Proof Objective

Property: Objective: T

Status: Falsified

Counterexample

Time	**0**	**0.001-0.999**	**1**	**1.001-1.999**	**2**
Step	**1**	**2-1000**	**1001**	**1002-2000**	**2001**
X_ctrl	-	-	-	-	-
Y_ctrl	-	-	-	-	-
Z_ctrl	-	-	-	-	-
X_actual	1	0	1001	0	1000
Y_actual	1	-	1	-	1
Z_actual	1	-	1	-	1

Fig. 8. An excerpt from the report of the upgraded control system, indicating non-compliance with property A

Chapter 3. Proof Objectives Status

Content

Objectives Valid

Objectives Valid

#	Type	Model Item	Description	Analysis Time (sec)	Counterexample
1	Proof objective	Verification Subsystem/Proof Objective1	Objective: T	4	n/a

Chapter 4. Properties

Content

Verification Subsystem/Proof Objective1

Verification Subsystem/Proof Objective1

Summary

Model Item: Verification Subsystem/Proof Objective1

Property: Objective: T

Status: Valid

Fig. 9. An excerpt from the report of the upgraded control system, indicating compliance with property B

3. Results and Discussion

Per the Figure 8, using the simplest case as a counterexample, Property Proving provides with the distances of 1001 and 1000 by X_actual, and the 1000 is the sample of time when the returning flag must be set, but it is not. The report results (see Figure 8) correspond to the simulation results (see Figure 5).

The data from the counterexample may be used for simulation testing with further debugging, manual inspection. The generated by Property Proving harness for the counterexample (see Figure 8) is shown in Figure 10.

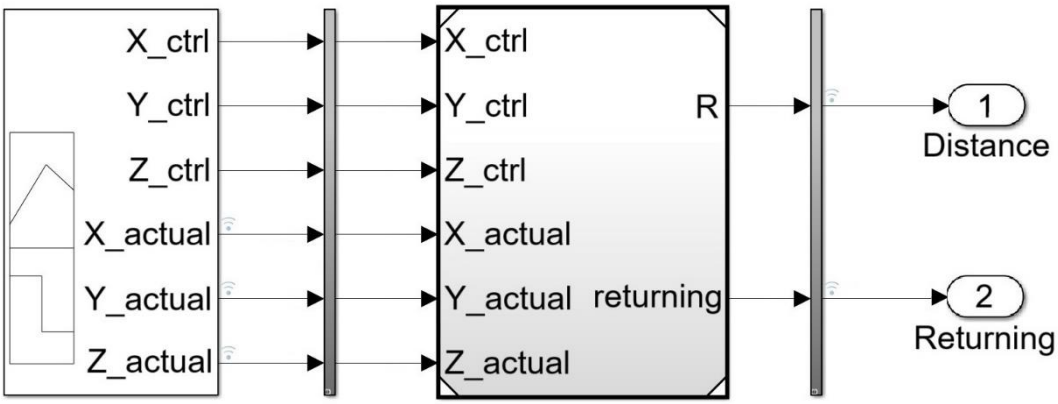

Fig. 10. Harness model for the model under analysis

If run its simulation, the logs of counterexample may be visualized, as presented in Figure 11 in form of graphs. Since the formal analysis is applied not to the quadcopter's behavior but to the logic of the "Return-to-Home" function, the distance graph is valid. However, according to Property A (see Figure 7), the returning flag should be latched at the 1st sec when the 1000 m threshold has been reached or exceeded.

The corresponding graph of the flag in Figure 10 indicates violation of this logic, as does the graph in Figure 5.

The process of debugging and correction is iterative: with subsequent modifications of the computer model, the Property Proving may generate some other counterexamples.

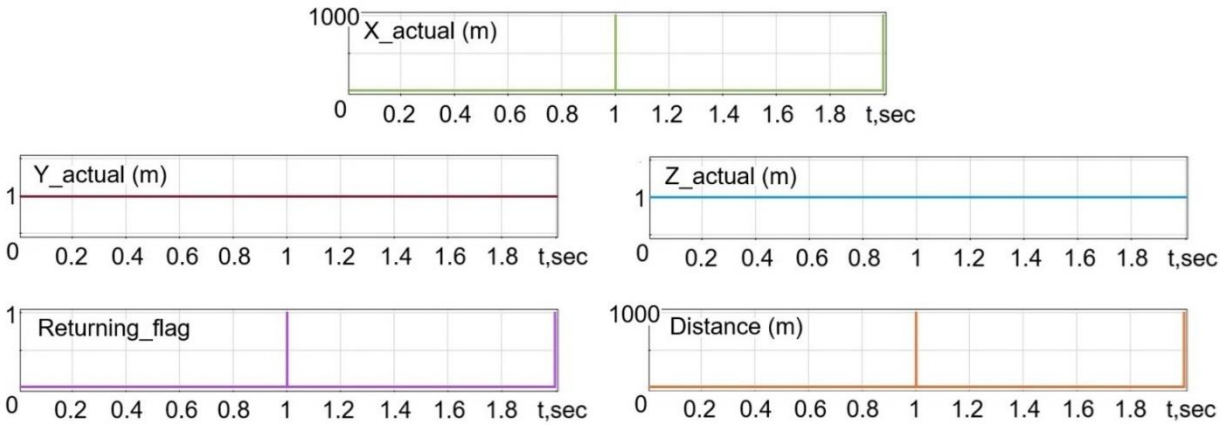

Fig. 11. Results of the harness model simulation

In the end, Figure 12 shows the updated “Return-to-Home” function of the upgraded quadcopter automatic control system.

```
function[x_rth,y_rth,z_rth,r,returning]=R_T_H(x,y,z)
persistent t_now
persistent returning_flag
persistent start_pos
if isempty(t_now),t_now = 0;end
if isempty(returning_flag),returning_flag=false;end
if isempty(start_pos),start_pos = zeros(3,1);end
threshold = 1000,T_return = 100,Ts = 0.1;
pos = [x; y; z];
r = norm(pos);
pos_cmd = pos;
if r > threshold,returning_flag = true;end
if returning_flag
    returning = true;
    if t_now == 0,start_pos = pos;end
    t_now = t_now + Ts;
    scale = max(0, 1 - t_now / T_return);
    pos_cmd = start_pos * scale;
    x_rth = pos_cmd(1);
    y_rth = pos_cmd(2);
    z_rth = pos_cmd(3);
else
    returning = false;
    x_rth = pos(1);
    y_rth = pos(2);
    z_rth = pos(3);
end
end
```

Fig. 12. Updated “Return-to-Home” function of the upgraded quadcopter control system

The results of the simulation of the updated computer model of the upgraded quadcopter control system are shown in Figure 13.

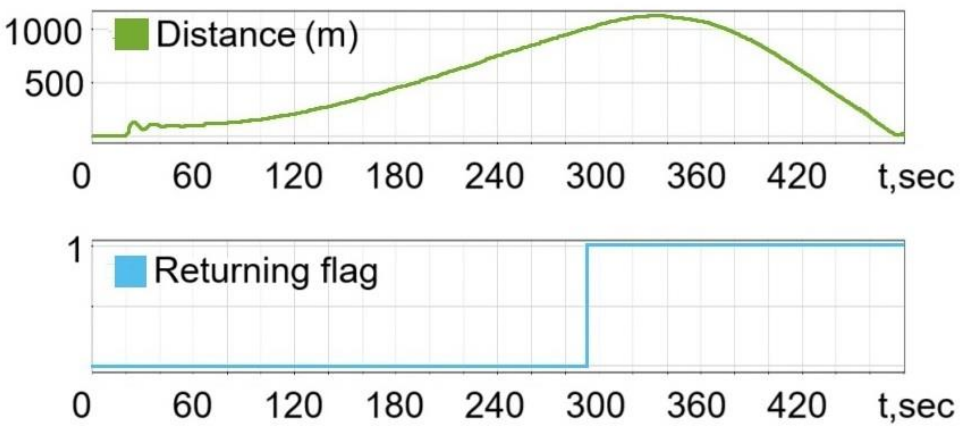

Fig. 13. Results of the simulation of the updated computer model of the upgraded quadcopter control system

The presented graphs confirm that the “Return-to-Home” function works correctly – the trigger is permanently set to “1”, despite the reduction in distance. At the same time, the desired coordinates in the lower three graphs correspond to the described behavior. So according to the graphs, the behavior is valid.

As can be seen from the new Property Proving report in Figure 14, the updated model of the quadcopter control system meets the properties and there are no counterexamples.

Chapter 3. Proof Objectives Status

Content

Objectives Valid

Objectives Valid

#	Type	Model Item	Description	Analysis Time (sec)	Counterexample
1	Proof objective	Verification Subsystem/Proof Objective	Objective: T	4	n/a
2	Proof objective	Verification Subsystem/Proof Objective1	Objective: T	4	n/a

Chapter 4. Properties

Content

Verification Subsystem/Proof Objective

Verification Subsystem/Proof Objective1

Verification Subsystem/Proof Objective

Summary

Model Item: Verification Subsystem/Proof Objective

Property: Objective: T

Status: Valid

Verification Subsystem/Proof Objective1

Summary

Model Item: Verification Subsystem/Proof Objective1

Property: Objective: T

Status: Valid

Fig. 14. Extract from the analysis results indicating compliance with the properties A and B

The resulting reports can be used for formal analysis, documentation of the execution of the design stage of the control system. And then, in combination with tests of the desired behavior, it is possible to approve the design stage and proceed to design and production. Design errors are identified, which are often made due to the human factor. With such detailed information (see Figures 8, 9 and 14), it is possible to obtain a computer model that fully meets the properties in an iterative manner.

The formal verification confirmed that the “Return-to-Home” logic, as implemented, consistently satisfies the expected properties under all admissible conditions. Notably, that the Property Proving Reports have shown that the earlier version of the “Return-to-Home” function exhibited faulty behavior due to improper handling of the time counter. This was corrected by ensuring the persistent state reset conditions aligned with spatial thresholds, which was verified by rerunning the property checks.

Conclusions

The study demonstrates a structured application of formal verification techniques to the control logic of a quadcopter system using the Property Proving tool in Simulink Design Verifier™. Specifically, the correctness of the “Return-to-Home” behavior was validated against predefined requirements through exhaustive model checking. Unlike conventional simulation testing, which only explores selected scenarios, Property Proving examined all reachable states of the control logic, guaranteeing that the desired safety property holds within the given model structure.

The effectiveness of the method lies in its ability to detect logical inconsistencies without relying on manually created test cases or real-world experiments. In one instance, faulty logic in the temporal return function was identified and resolved. The approach proved useful not only in confirming the intended behavior but also in supporting safe model evolution by verifying updates in a traceable manner. This demonstrates the tool’s potential for reducing risk in the early stages of quadcopter software development.

The key approach of this research is to integrate formal analysis directly into the quadcopter control software development cycle. By modeling the required scenarios as formal properties, the verification pipeline can be extended in future work. Since the method is not tied to specific scenarios, behavior, or functionality, it can be applied to all requirements, if needed, to establish a more comprehensive safety of the quadcopter system.

A comprehensive analysis of the system based on all properties takes about 12 seconds for falsified results (see Figure 8) and 4 seconds for valid results (see Figure 14). During this time, interactive reports are generated with detailed information, and there is an option to visualize and debug counterexamples (if any are found) (see Figures 10 and 11).

One drawback of formal analysis is the complete reliance on the formulated properties and their implementation. If errors in the model due to human factors are identified during the analysis, errors in the testing and analysis strategy require a more rigorous and in-depth verification.

A second drawback of using the Simulink Design Verifier™ application is its incompatibility with many blocks [12]. This complicates the implementation of properties for Property Proving analysis, which can lead to significant time expenditure and render the use of formal analysis tools inefficient.

However, it should be remembered that neither analysis nor testing

demonstrates the absence of errors or defects [13]. It is important not to overoptimize the product, as the model object always idealizes the original object.

The analysis was applied to a specific safety function; however, the methodology is extensible to other failure scenarios such as signal loss, actuator faults, or battery depletion. Future work will involve expanding the set of formally specified requirements to build a more comprehensive safety profile for the entire flight control system.

References

1. Fedorovich, O., Krytskyi, D., Lukhanin, M., Prokho-rov, O., & Leshchenko, Y. Modeling of strike drone missions for conducting wave attacks in conditions of enemy anti-drone actions. Radioelectronic and Computer Systems, 2025, vol. 2025, no, 1, pp. 29-43. DOI: 10.32620/reks.2025.1.02.

2. Pogudin, A., Pohudina, O., Bykov, A., & Plastun, T. Simulation of the automatic flight of a small unmanned aerial vehicle over a marker line. Open Information and Computer Integrated Technologies, 2022, no. 95, pp. 71-82. DOI: 10.32620/oikit.2022.95.06.

3. Xiao W. SRAD: Autonomous Decision-Making Method for UAV Based on Safety Reinforcement Learning / W. Xiao, X. Luo, S. Xie // Expert Systems. – 2025. – Vol. 42, Iss. 5. – 18 p. DOI: 10.1111/exsy.70004.

4. Ma B. Safety flight envelope calculation and protection control of UAV based on disturbance observer / B. Ma, M. Chen // Security and Safety. – 2023. – Vol. 2(2023020). – 17 p. DOI: 10.1051/sands/2023020.

5. Hung I.-K. Assessing Drone Return-to-Home Landing Accuracy in a Woodland Landscape / I.-K. Hung, D. Unger, Y. Zhang, D. Kulhavy // Drones and Autonomous Vehicles. – 2024. – Vol. 1(3). DOI: 10.35534/dav.2024.10005.

6. Shcherban A. UAV Flight Safety System Based on Fuzzy Logic / A. Shcherban, V. Ieremenko // Transactions on Aerospace Research. – 2020. – Vol. 4. – P. 71–80. DOI: 10.2478/tar-2020-0022.

7. Motwakela A. Fuzzy logic in real-time decision making for autonomous drones / A. Motwakela et al. // International Journal of Data and Network Science. – 2024. – Vol. 9, Iss. 2. – P. 335–344. DOI: 10.5267/j.ijdns.2024.7.008.

8. Shcherban A. Validating Lithium-Polymer Battery Discharge Models to Ensure UAV Flight Safety and Performance/ A. Shcherban, V. Eremenko // Transactions on Aerospace Research. – 2024. – Vol. 4. – P. 80–100. DOI: 10.2478/tar-2024-0024.

9. Trauth M. H. Introduction to MATLAB. In: MATLAB® Recipes for Earth Sciences / M. H. Trauth // Springer Textbooks in Earth Sciences, Geography and Environment / Springer, Cham. – 2025. – 567 p. DOI: 10.1007/978-3-031-57949-3_2.

10. What Is Property Proving? / The MathWorks, Inc., 2024. – https://www.mathworks.com/help/sldv/ug/what-is-property-proving.html (23.04.2025).

11. Zhukevych A. Study of mutual influence between control channels of the quadcopter due to the low maneuverability of the UAV / A. Zhukevych, V. Dzhulgakov, O. Zhukevych // Aerospace technic and technology. – 2022. – Vol. 5, Iss. 183. – P. 68–81. DOI: 10.32620/aktt.2022.5.06.

12. Supported and Unsupported Simulink Blocks in Simulink Design Verifier / The MathWorks, Inc., 2024. – https://www.mathworks.com/help/sldv/ug/simulink-block-support.html (23.04.2025).

13. Yorkston K. Performance Testing. An ISTQB Certified Tester Foundation

Level Specialist Certification Review / K. Yorkston // Apress Berkeley, CA. – 2021. – 394 p. DOI: 10.1007/978-1-4842-7255-8.

Надійшла в редакцію 16.05.2025, розглянута на редколегії 16.05.2025

Застосування Property Proving до програмного забезпечення для управління польотом квадрокоптера

Об’єктом дослідження є комп’ютерна модель системи автоматичного управління квадрокоптером. Предметом дослідження є застосування формальних методів для аналізу та перевірки вимог до систем управління квадрокоптерами, з особливим акцентом на можливості інструменту Simulink Design Verifier™. Основна мета дослідження полягає в тому, щоб підвищити загальну надійність розробленої моделі квадрокоптера шляхом застосування формальних методів аналізу на етапі проєктування, щоб виявити потенційні логічні та структурні помилки перед фізичним впровадженням. Задачі дослідження включають: проведення комплексного огляду проблем, пов’язаних з експлуатацією комерційних квадрокоптерних платформ; формулювання вимог безпеки до управління польотом квадрокоптера; розробка детальної імітаційної моделі системи управління з використанням середовища Simulink®; використання функціональних можливостей Property Proving для виконання формальної перевірки критичних властивостей системи; аналіз та інтерпретація верифікаційних звітів, створених інструментом; а також виявлення прихованих недоліків або невідповідностей, які можуть поставити під загрозу безпеку системи. Застосовано такі методи: формальна перевірка з використанням модуля Property Proving, імітаційне моделювання, статичний аналіз коду та тестування на основі моделі. Отримано такі результаті: продемонстровано, як формальні методи перевірки, такі як Property Proving, можуть бути застосовані для перевірки важливих для безпеки поведінок систем управління квадрокоптерами, доведено, що використання Simulink Design Verifier™ виявилося ефективним у виявленні слабких місць проєктування на ранніх стадіях циклу розробки, зменшуючи подальший ризик і переробку. Крім того, створення інтерактивних діагностичних звітів сприяє візуалізації сценаріїв збою та підтримує ітеративне покращення. Висновки. Застосування формальних інструментів аналізу, таких як Simulink Design Verifier™, представляє цінний підхід до посилення надійності системи управління квадрокоптером за умови, що моделі правильно побудовані та властивості ретельно визначені. Незважаючи на наявність обмежень цей інструмент залишається потужним доповненням до традиційного тестування, особливо під час вирішення вимог до системи з високим рівнем надійності. Дослідження підкреслює необхідність інтеграції формальних методів у стандартні робочі процеси верифікації, щоб збалансувати теоретичну точність із практичною перевіркою.

Ключові слова: БПЛА; квадрокоптер; система управління; проєктування; статичний аналіз; property proving; комп’ютерна модель.

Відомості про авторів

Сокол Дмитро Вадимович, PhD, доцент кафедри систем управління літальних апаратів, Національний аерокосмічний університет «Харківський

авіаційний інститут», Харків, Україна. E-mail: d.sokol@khai.edu. ORCID: 0000-0003-0847-350X.

Жукевич Аркадій Борисович, кандидат технічних наук, доцент, доцент кафедри систем управління літальних апаратів, Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна. E-mail: a.zhukevych@khai.edu. ORCID: 0000-0001-5355-8495.

Мірошниченко Галина Анатоліївна, кандидат технічних наук, доцент кафедри систем управління літальних апаратів, Національний аерокосмічний університет «Харківський авіаційний інститут», Харків, Україна. E-mail: h.miroshnychenko@khai.edu. ORCID: 0000-0003-2914-0974.

About the authors

Sokol Dmytro – PhD, associate professor of Aircraft Control Systems Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine. E-mail: d.sokol@khai.edu. ORCID: 0000-0003-0847-350X.

Zhukevych Arkadii – Candidate of Science (Engineering), associate professor of Aircraft Control Systems Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine. E-mail: a.zhukevych@khai.edu. ORCID: 0000-0001-5355-8495.

Miroshnychenko Halyna – Candidate of Science (Engineering), associate professor of Aircraft Control Systems Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine. E-mail: h.miroshnychenko@khai.edu. ORCID: 0000-0003-2914-0974.