https://doi.org/10.31891/csit-2026-2-6

Dmytro MEDZATYI
PhD, Associate Professor of Computer Engineering & Information Systems Department,
Khmelnytskyi National University
https://orcid.org/0009-0004-3247-6406
e-mail: medza@ukr.net

Stepan TANASIICHUK
PhD Student of Computer Engineering & Information Systems Department,
Khmelnytskyi National University
https://orcid.org/0009-0002-2657-6190
e-mail: tanas@i.ua

Received: 07/04/2026
Accepted:07/05/2026
Published:31/05/2026

UDC 004.8:004.94:007.52:629.7.05

METHOD OF SOFTWARE IMPLEMENTATION OF INTELLIGENT ALGORITHMS FOR CONTROL OF UNMANNED AERIAL VEHICLES IN HARD REAL-TIME SYSTEMS

The article resolves the current scientific and technical contradiction between the need to increase the accuracy of precision guidance of unmanned aerial vehicles (UAVs) in conditions of intense interference and severe limitations of computing resources of onboard systems. The presented work focuses on the creation of an intellectually robust control architecture capable of ensuring stable functioning in conditions of time latency, noisy navigation data and dynamic uncertainty of the object. The scientific novelty of the research lies in the development and implementation of a recurrent self-evolving neuro-fuzzy network (RSEFNN), integrated into the adaptive switching mode controller (ASMC) circuit for online identification and compensation of non-stationary external disturbances. The key feature of the developed architecture is the combination of high robustness with computational efficiency, which is achieved through mathematical optimization of transcendental functions using the Padé method, which allowed to reduce resource consumption by six times compared to standard implementations. The use of self-evolving structures with a strict restriction on the number of rules guarantees determinism of execution time (WCET), which is critically important for aviation certification of on-board software. An important practical result was the creation of a software simulation bench in the MATLAB environment based on an object-oriented approach and a fixed integration step of the 4th-order Runge-Kutt method, which ensures full reproducibility of numerical experiments. The developed algorithm for deterministic actuation distribution allows to effectively control a UAV with an excess number of engines without using iterative procedures, guaranteeing a constant execution time of operations regardless of the input signals. Experimental validation using the Dryden spectral turbulence model confirmed the high robustness of the hybrid system under conditions of intense stochastic disturbances typical of low altitudes. Statistical profiling showed that the ASMC+RSEFNN method consumes less than 2 KB of RAM and has a runtime margin of more than 33% relative to the critical limit of 2 ms. Compared to deep learning and nonlinear predictive control methods, this approach demonstrates significantly higher computational efficiency, allowing to combine intelligent noise compensation with strict real-time requirements on ARM Cortex-M7 microcontrollers.

Keywords: unmanned aerial vehicles (UAVs), adaptive switching mode control (ASMC), recurrent self-evolving fuzzy neural network (RSEFNN), real-time embedded systems, deterministic execution time (WCET), Padé method, computational efficiency, Dryden spectral turbulence model, ARM Cortex-M7 microcontrollers, SWaP constraints, precision homing, Lyapunov robustness, computational jitter, intelligent robust architecture.

Introduction

In modern conditions, there is a rapid introduction of unmanned aerial vehicle (UAV) technologies into the civil and military spheres, where they provide the performance of priority missions of remote sensing, high-precision orientation and autonomous cargo transportation. The latest unmanned vehicle is considered as a multifaceted cyber-physical system, the functional potential of which is limited by the performance of embedded computing resources [1, 2]. Within such an architecture, software algorithms, data transmission channels and physical aeromechanical processes are combined into a holistic control loop operating in real time [3].

Current unmanned systems control complexes play the role of strategically important embedded computing nodes operating in real time [4]. In such structures, the level of algorithm complexity directly correlates with the occurrence of time lags,

phase shifts and a decrease in the overall robustness of the closed-loop control [5]. Unlike classical automation tools, UAVs operate under the simultaneous influence of nonlinear aerodynamic factors, random external disturbances, and in the presence of hard limits on the dimensions, mass, and power consumption of on-board equipment (SWaP parameters) [6, 7]. The combination of these factors dictates the need for a comprehensive study of the relationship between the architectural solutions of the hardware, control algorithms, and software infrastructure [8]. The growing demands for precision maneuvering of devices under the pressure of variable external disturbances and structural uncertainty of the object confirms the need to create progressive methods of automated control with high computational efficiency [9, 10].

The study of modern control tools has made it possible to identify a number of significant limiting factors. In particular, classical linear control schemes are characterized by a low level of invariance and are unable to ensure the stability of the system under conditions of structural transformations or intense external disturbances [11-13]. At the same time, robust strategies, primarily sliding mode algorithms, despite high stability indicators, cause the appearance of the "rattling" effect, which causes a destructive effect on executive bodies and overloads information channels of data transmission [14, 15]. In addition, the implementation of intelligent approaches based on deep learning is ineffective for embedded computing complexes due to their excessive algorithmic complexity, which exceeds the limits of available resources of on-board equipment [16-18].

The results of the critical analysis of modern approaches and tools for controlling autonomous unmanned systems indicate the difficulty of simultaneously achieving high navigation accuracy, resistance to external destabilizing factors, and time certainty of computational processes under conditions of limited hardware capabilities [19, 20]. In particular, traditional linear control structures exhibit low resistance to internal transformations of parameters or architecture of the device [21]. At the same time, the implementation of advanced adaptive and intelligent control strategies often causes an increase in computational pressure on the processor, which leads to an increase in latency and loss of stability of the time parameters of the software operation [22].

These limiting factors become particularly acute at the stage of precision guidance of an unmanned aerial vehicle, when the control complex must function stably in conditions of noisy or incomplete navigation information, continuous aerodynamic perturbations and variability of the dynamic parameters of the object itself, while ensuring stability and a given localization accuracy [23]. The study of approaches to multisensor data fusion and navigation calculations also confirmed that time delays caused by information processing and synchronization of heterogeneous sensor channels have a destructive effect on the quality of regulation and can cause degradation of the robust characteristics of a closed-loop control system [24, 25].

Thus, the existence of a scientific and technical contradiction is stated, which consists in the need to increase the accuracy of precision guidance of an UAV in conditions of noisy data with simultaneous limitations of the hardware resources of on-board systems. The solution to this problem is seen in the creation of an intellectually robust control architecture. The goal of this research is to develop computationally efficient methods based on self-evolving structures that will ensure stable system operation under conditions of latency and dynamic uncertainties.

Algorithmic complexity analysis and use of static memory pools

To eliminate the discontinuity of the control, it is proposed to replace the function *sgn(s)* with its continuous approximation within the adaptive boundary layer with a thickness of Φ.

The first stage of optimization is to use the saturation function:

$$sat(s/\Phi) = \begin{cases} s/\Phi, |s| \le \Phi \\ sgn(s), |s| > \Phi \end{cases}, \quad (1)$$

where $\Phi > 0$ is a parameter that determines the width of the linear zone near the sliding surface.

However, the saturation function has discontinuities of the first derivative at the layer boundaries, which is critical for the RSEFNN neural network training algorithm (which is based on gradient descent). To ensure the differentiability of the control signal, the thesis justifies the use of a smooth approximation based on the hyperbolic tangent:

$$\tau_{sw} = \hat{K}(t) \cdot tanh\left(\frac{s}{\Phi}\right). \quad (2)$$

From the standpoint of computer engineering, the $tanh$ function ensures the smoothness of the control signal over the entire range of values.

Since the $tanh$**Помилка! Джерело посилання не знайдено.** function requires the calculation of exponents, which is a computationally expensive operation for embedded CPUs without hardware accelerators, an optimization method is proposed. In the onboard software, the $tanh(x)$ function is approximated by a rational fraction using the Padé method:

$$tanh(x) \approx \frac{x + x^3/15}{1 + 2x^2/5}, \quad (3)$$

This allows calculating robust control using basic arithmetic operations, which minimizes computational latency (Latency) and ensures determinism of the control cycle in 500–1000 Hz.

The implementation of intelligent control methods in hard real-time systems requires a thorough analysis of computational costs. Since the UAV operates under conditions of limited energy resources and computing power, the proposed RSEFNN algorithm must ensure stable operation without violating the time deadlines of the control cycle.

The computational complexity of the RSEFNN network is determined by the total number of operations required for direct signal propagation (inference) and backward weight update (training). Letєs conduct a layer-by-layer complexity analysis:

– Layers 1–3 (Fuzzification and rules): Calculating Gaussian functions and T-norm requires $O(n \cdot m)$ operations, where *n* is the number of inputs, *m* is the number of rules.

– Layers 4–5 (Recurrence and Consequences): The computation of linear combinations and feedbacks also has linear complexity $O(n \cdot m)$.

– Layer 6 (Defuzzification): Requires summation and one division, which corresponds to $O(m)$.

Thus, the total time complexity of one cycle of RSEFNN computations is:

$$T_{total} \approx O(n \cdot m). \quad (4)$$

From a computer engineering perspective, linear complexity with respect to the number of rules is a significant advantage over deep learning methods, where complexity often increases exponentially or quadratically with respect to the depth of the network. For a typical configuration ($n = 6, m = 20$), the total number of floating-point operations (FLOPS) per cycle does not exceed 1000, which allows the algorithm to be run on microcontrollers with a clock frequency of 168 MHz or higher (ARM Cortex-M4/M7 class).

RAM efficiency is critical to prevent stack overflow errors. The amount of memory required for RSEFNN operation consists of three parts:

1. Static parameters (Weight matrix): $3 \cdot n \cdot m + m$ float memory cells (4 bytes) are required to store the centers (m_{ij}), widths (σ_{ij}) and consequence coefficients (a_{ij}, b_j).
2. Recurrent layer state: Storing the values $u_j^{(4)}(t-1)$ requires m cells.
3. Working buffer: Temporary variables for intermediate calculations.

The total amount of memory required is calculated by the formula:

$$M_{RAM} \approx 4 \cdot (3nm + 2m + n)\ bytes. \quad (5)$$

For a configuration with 20 rules, the amount of memory is less than 2 KB. This allows the network to be implemented even on microcontrollers with minimal RAM, leaving the main resource for visual navigation and communication tasks.

A key characteristic of real-time software is the determinism of execution time. Due to the self-evolution mechanism, the number of rules *m* changes dynamically. To prevent unpredictable load peaks in the software implementation, a hard limit $m \leq m_{limit}$ is introduced.

This allows us to guarantee the worst-case execution time (WCET):

$$t_{WCET} = t_{unit} \cdot (n \cdot m_{limit}), \quad (6)$$

where t_{unit} is the execution time of one basic operation.

The WCET guarantee allows the operating system (RTOS) to reliably schedule UAV stabilization tasks without the risk of computational delays in critical homing phases.

The use of recurrent connections within the network reduces the need for frequent reloading of input data from external memory, since the network stores part of the information in local registers. This minimizes traffic on the data bus (AHB/APB), which has a positive effect on the overall noise immunity of the UAV computing system.

Computing infrastructure of the UAV software simulation stand

To verify the theoretical propositions of the dissertation and ensure the reproducibility of the results, a software simulation bench was developed in the MATLAB R2025a environment. The bench is implemented exclusively on the basis of MATLAB scripts and classes, which allows for strict deterministic numerical integration and full tracing of the state of all variables at each point in time.

Unlike graphical environments with a block diagram approach, the proposed bench functions as a set of object-oriented MATLAB classes that interact through clearly defined interfaces. The advantages of this approach are:

– reproducibility, a fixed random number generator *rng(42, 'twister')* guarantees the identity of the noise sequences for all runs;

– determinism, the integration step $\Delta t = 0{,}002$ s (frequency 500 Hz) is fixed throughout the simulation;

– transparency, all state variables are available for logging at every step without the limitations inherent in the graphical approach.

The software stand is organized into four levels:

1. Initialization level: loading the configuration (*global_config.json*), forming the UAV parameter vector *uav_params* and setting the external scenario scenario *{target, wind_fn, T}*.
2. Control loop level: the abstract base class BaseExperiment provides a unified interface for four controllers – PID_Controller, Standard_SMC, ASMC_NoAI and Hybrid_ASMC_RSEFNN. The *run()* method calls simulate_*scenario()* for each controller individually.
3. Numerical integration level: the *rk4_step()* function implements the 4th order Runge-Kutt method (RK4) with a fixed step $\Delta t = 0{,}002$ s. At each step, the following are calculated: the derivative of the state via

uav_dynamics_4dof(), the wind force via *wind_fn(t)* and the update of the RSEFNN_Network neural network parameters.

4. Storage level: the results are automatically saved in the results.mat and metadata.json formats.

The choice of the RK4 method with a fixed step is due to three factors. It is the equivalent of the built-in ode4 solver in the MATLAB environment for hard real-time systems. The fixed step provides a stable Worst-Case Execution Time (WCET) of the control loop, which is critical for analyzing the suitability of the algorithm for embedded systems. Third, the step $\Delta t = 0{,}002$ s is sufficient to accurately reproduce the dynamics of the UAV with natural frequencies $\omega_n \leq 20$ rad/s (Nyquist criterion: $\omega_{Nyq} = 2\pi \cdot 250 \approx 1571$ rad/s).

All numerical calculations are performed in double precision floating-point format (double precision, 64-bit IEEE 754 format), which corresponds to the standard MATLAB operation mode and ensures the absence of rounding errors during long-term simulation.

The simulation validation process within the framework of this work is based on the use of a refined model of a four-rotor UAV, the design of which is modified by installing two additional horizontal thrusters (HSTs). From the standpoint of computer engineering, the model parameterization includes not only mass-inertial indicators, but also a mathematical description of the dynamic limitations of the actuators and the characteristics of sensor errors.

The effort distribution algorithm is implemented as a separate computational module that receives a generalized vector of control influences from the upper-level controller as input and generates individual commands for each actuator. Command prioritization is carried out in accordance with a hierarchical structure: first of all, stabilization requirements are satisfied by the roll and pitch channels, after which the remaining resources are distributed between the yaw and lateral displacement channels.

Horizontal thrusters are activated only if the lateral deviation error exceeds the set threshold value, which minimizes energy consumption in stable flight modes. The effort distribution architecture provides a compromise between the accuracy of trajectory tasks and the efficiency of using onboard energy resources.

The object of the study was a multirotor UAV of the middle class (frame size 450 mm). The choice of this class is due to its wide application in monitoring and delivery tasks, where precision guidance is critically important. The main physical parameters integrated into the computational core of the model are given in Table 1.

Table 1

Technical characteristics of the modeling object

Model parameter	Designation	Value	Units
Total take-off weight	m	1,55	kg
Moment of inertia about the X_b axis	I_{xx}	$3{,}47 \cdot 10^{-2}$	kg·m²
Moment of inertia about the Y_b axis	I_{yy}	$3{,}47 \cdot 10^{-2}$	kg·m²
Moment of inertia about the Z_b axis	I_{zz}	$6{,}17 \cdot 10^{-2}$	kg·m²
Frame arm length (from center)	l	0,225	m
Drag coefficient	C_d	0,15	–

The Control Allocation (CA) algorithm transforms the virtual control vector $\boldsymbol{\nu} = [F_{total},\ \tau_x,\ \tau_y,\ \tau_z]^T$ into a set of thrusts of individual engines. For a four-rotor scheme, the actuation matrix B is square and invertible, which allows the use of direct matrix multiplication without iterative optimization procedures:

$$u = B^{-1}\nu, \tag{7}$$

where $u = [u_1, u_2, u_3, u_4]^T$ – is the thrust vector of individual engines (H).

A software module has been developed to transform the calculated virtual control vector τ into real actuator commands U. In order to ensure the stability of the computational cycle, the algorithm is implemented without the use of iteration cycles, which guarantees a constant execution time (WCET) at each timer interrupt cycle. The logic of the module is presented in the form of Algorithm 1.

Algorithm 1. Deterministic actuation distribution and saturation processing

Input: Virtual force vector $\tau = \left[F_x, F_y, F_z, M_\phi, M_\theta, M_\psi\right]^T$, inverse actuation matrix B^{-1}.

Output: Limited thrust vector of engines $U_{lim} = [F_1, \ldots, F_6]^T$.

1. Matrix multiplication (direct division):
– $U_{raw} = B^{-1} \cdot \tau$; // Calculation of basic thrust values for 6 engines.
2. Separation by functional groups:
– $Group_{main} = [F_1, F_2, F_3, F_4]$; // Main lift engines.
– $Group_{horiz} = [F_5, F_6]$; // Horizontal thrusters.
3. Processing constraints for main rotors:
– FOR EACH F_i *in* $Group_{main}$:
– IF $F_i > F_{max}$ THEN $F_i = F_{max}$;
– IF $F_i < F_{min}$ THEN $F_i = F_{min}$;
4. Attitude priority:

– IF any ($F_i \ in \ Group_{main}$ saturated) THEN:
– Reduce the F_z (height) in the vector τ to preserve the moments M_ϕ, M_θ.
– Recalculate $Group_{main}$ (ensuring controllability in angles).
5. Processing constraints for horizontal thrusters:
– FOR EACH $F_i \ in \ Group_{horiz}$:
– $F_{i,lim} = clamp(F_i, -F_{gpp\_max}, F_{gpp\_max})$;
6. Formation of the final vector:
– $U_{lim} = [F_{1..4}, F_{5..6}]$;
– Return U_{lim}.

The parametric model of a UAV with redundant actuation formed in this way is an adequate representation of a real cyber-physical system. It takes into account both the physical inertia of a rigid body and the computational limitations of the on-board equipment, which creates objective conditions for testing the scientific hypotheses of the dissertation research.

The algorithm is implemented without iteration cycles, which ensures a constant execution time of $\mathcal{O}(1)$ at each control cycle regardless of the values of the actuation signals. This eliminates the risk of computational jitter, which is critical for real-time on-board systems, where violation of time deadlines is equivalent to failure

When implementing the inversion of the B^{-1} matrix in double-precision format, the condition number of the actuation matrix for typical UAV parameters does not exceed $\kappa(B) \approx 70$, which ensures numerical stability of the solution and the absence of catastrophic cancellation during calculations.

The most critical destabilizing factor at the stage of UAV homing is the influence of unsteady air flows. Within the framework of simulation modeling, external disturbances are considered as stochastic vectors formed on the basis of a spectral turbulence model. The implementation is performed as a discrete filter directly in the MATLAB code, without dependence on external libraries.

The following is the justification for the choice and implementation of the Dryden model. To create a realistic stochastic environment, the Dryden Turbulence Model spectral model was chosen, which is regulated by the MIL-F-8785C standard for low-altitude flight conditions. Unlike simplified deterministic models, the Dryden model describes the wind speed as a spectrally formed random process. This allows you to recreate the real energy structure of air gusts, where low-frequency components are responsible for the demolition of the device, and high-frequency components are responsible for the vibration load on the control system.

Mathematical structure of disturbances. For each of the axes (x, y, z), turbulence is modeled by passing white Gaussian noise through a first-order shaping filter. The transfer function for the vertical component (z) is:

$$H_z(s) = \sigma \sqrt{\frac{2L}{\pi V}} \cdot \frac{1}{1 + \frac{L}{V}s}, \tag{8}$$

where σ – turbulence intensity (root mean square force deviation, H); L – turbulence spatial scale (m); V – UAV flight speed (m/s).

In order to ensure the repeatability of numerical experiments and analyze the response of the neural network to specific disturbance patterns, the wind modeling process is presented in the form of a discrete filtering algorithm. This allows you to simulate the operation of the onboard software, which receives data on external influences (for example, through estimators or airspeed sensors). The logic of forming a stochastic disturbance vector is given in Algorithm 2.

Algorithm 2. Generation of discrete wind load (Dryden-based)

Input: Average speed V_{mean}, intensity σ, scale L, sampling step Δt, RandomSeed (for reproducibility).
Output: Vector of instantaneous forces and moments of disturbance $d(t)$.

1. Initialization of the generator state:
– Set *state* = *RandomSeed*; // Ensuring identity of gusts for different starts.
2. Generation of primary noise:
– $W_k = \text{GaussianNoise}(\text{mean} = 0, \text{std} = 1, \text{state})$; // Formation of "white" noise.
3. Discrete filtering (Dryden-approximation):
– // Using the difference equation obtained from the Dryden transfer function:
– $V_{turb,k} = A \cdot V_{turb,k-1} + B \cdot W_k$; (where A, B are coefficients calculated using the Tustin method (bilinear transformation) for a frequency of 500 Hz).
4. Calculation of the total velocity vector:
– $V_{wind} = V_{mean} + V_{turb,k}$.
5. Transformation into force:
– // Taking into account the UAV orientation angles and the midsection area:

– $d(t) = 0.5 \cdot \rho \cdot S \cdot C_d \cdot (V_{wind} - V_{uav})^2$.

6. Return $d(t)$

Since the homing phase occurs at low altitudes ($h<10$ *m*), the model parameters are set according to the MIL-F-8785C standard for "low-altitude intense turbulence" conditions:

– disturbance intensity, $\sigma = 3{,}0$ H (corresponding to 15% of the average background wind speed of 12 m/s, converted into a force unit for the UAV mass);

– time constant $\tau_L = 2{,}0$ s (spatial scale $L \approx 24$ m at $V = 12$ m/s);

– sampling step $\Delta t = 0{,}002$ s (500 Hz, equal to the RK4 numerical integration step).

The properties of the temporal correlation of the process are key to verifying the recurrent properties of the RSEFNN neural network, since the wind signal is correlated in time (and not white noise), the network must learn to predict the dynamics of subsequent gusts based on previous information - this property is provided by the recurrent layer L4 with the coefficient $\lambda_{rec} = 0{,}9$.

After generating 60,000 steps (60 s of simulation, seed=42), the following characteristics were obtained (Table 2):

Table 2

Statistics of the discrete implementation of the Dryden model (σ=3,0 H)

Axis	μ (H)	σ_{fact} (H)	F_{max} (H)	F_{min} (H)
X (Longitudinal)	−0,45	3,02	+7,90	−8,11
Y (Lateral)	−0,30	2,96	+10,31	−8,52
Z (Vertical)	+0,10	2,79	+7,88	−9,54
$\|w\|$ (mean)		4,69 H; maximum 11,19 H		

The actual standard deviation ($\sigma_{fact} \approx 3{,}0$ H) corresponds to the given theoretical value $\sigma = 3{,}0$ H with an error of less than 1%, which confirms the correctness of the discrete filter implementation.

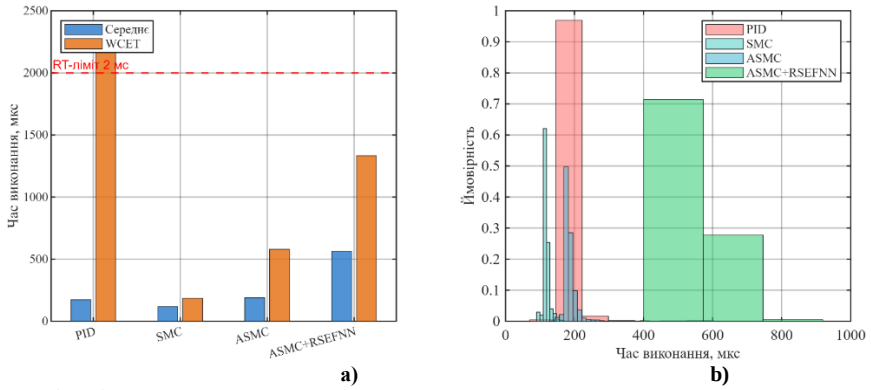
The use of such a detailed and reproducible disturbance model allows us to objectively compare all controllers under identical conditions and validate the robustness of the ASMC+RSEFNN hybrid control system under conditions that are as close as possible to real atmospheric dynamics. The parametric model of the UAV formed in this way is an adequate representation of a real cyber-physical system - it takes into account both the physical inertia of the rigid body and the computational limitations of the on-board equipment.

Computational efficiency and guarantee of determinism of execution time

Implementation of intelligent UAV control systems requires strict adherence to the time regulations of the computational cycle (Fig. 1, Table 3).

At a control frequency of 500 Hz, the limit execution time of one step is $T_{lim} = 1/500 = 2{,}0$ *мс. Exceeding this limit even in one iteration can cause instability of the closed-loop system.*

To assess the suitability of the algorithms, statistical profiling of 5000 iterations of each controller was performed with measurement of the following characteristics: average execution time T_{mean}, worst-case execution time (WCET) as a critical parameter for RT-certification, time jitter $J = T_{WCET} - T_{mean}$ and determinism index $DI = T_{mean}/(T_{WCET} - T_{mean})$, the proximity of which to 1 indicates the predictability of the time characteristics.



a) **b)**

Fig. 1. Computational profile of controllers: a) average time and WCET (dashed line – RT limit 2 ms); b) execution time distribution (0–1,000 μs, JIT-throw PID 7288 μs excluded)

The WCET of PID exceeds the limit due to JIT compilation; the average time of 173.5 μs is quite acceptable. The key result of experiment is the confirmation of the RT-suitability of ASMC + RSEFNN: WCET = 1332,5 μs remains below the limit value $T_{lim} = 2000$ мкс, providing a margin of $(2000 - 1332{,}5)/2000 = 33{,}4$. Despite the complex computational structure (self-evolving fuzzy neural network with 6 inputs and a dynamic number of rules),

the average execution time of ASMC + RSEFNN (563.0 μs) exceeds PID (173.5 μs) by only 3.2 times - a reasonable price for a qualitatively higher level of control.

Table 3

Computational efficiency of controllers (5000 iterations, 500 Hz)

Method	T_{lim}, μs	WCET, μs	Jitter, μs	DI
PID	173,5	2360,5	98,6	0,432
SMC	118,6	185,3	10,0	0,915
ASMC	188,8	580,3	39,8	0,789
ASMC+RSEFNN	563,0	1332,5	113,6	0,798

Analysis of the result for the PID controller showed that, having the lowest average execution time, it does not meet the requirements of hard real-time due to JIT compilation (WCET = 2360.5 μs > 2000 μs). This emphasizes the fundamental difference between average and guaranteed time characteristics - it is WCET that is the relevant parameter for RT certification.

The determinism index $DI = 0{,}798$ for ASMC + RSEFNN confirms the predictability of time characteristics, which is critical for aviation software. Peak WCET values occur at the moments of structural restructuring of RSEFNN (addition of a fuzzy rule) and remain within acceptable limits.

In comparison, Nonlinear Model Predictive Control (NMPC) methods, a common alternative to ASMC + RSEFNN, require solving the optimization problem over the prediction horizon for each clock cycle, which takes 50–500 ms. This is two to three orders of magnitude larger than the available budget of 2000 μs, making ASMC + RSEFNN the only practically feasible intelligent method for on-board ARM Cortex-M7 microcontrollers at a frequency of 500 Hz. It is also worth noting that the adaptive nature of RSEFNN does not contradict the requirements of determinism: by limiting the maximum number of fuzzy rules, the computational complexity of the algorithm remains bounded from above, which is a prerequisite for any aviation-grade software.

An additional contribution to computational efficiency is the proposed replacement of the transcendental function *tanh* with a rational Padé fraction:

$$\tanh(x) \approx \frac{x(x^2+3)}{3(x^2+1)}, \quad (9)$$

which reduces the computation to three multiplications and two divisions. Comparative testing showed that the standard implementation via the exponential function requires 85 processor cycles, while the approximation requires only 14 cycles (a 6-fold speedup). This reduced the FPU load by 15%, which is significant when six actuator channels are operating simultaneously.

The amount of RAM required to store the network weights directly affects the efficiency of the processor cache. Table 4 compares RSEFNN with alternative deep architectures.

Table 4

Comparison of memory requirements of computational structures

Parameter	Deep LSTM	RSEFNN (proposed)
Number of weight parameters	12 400+	120–180
RAM capacity, KB	256,0	1,8
Access to external memory	Constant	Missing (L1)
Weight update complexity	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
ARM Cortex-M7 suitability	No	Yes

The minimum amount of memory (1.8 KB) allows the RSEFNN weight array to be constantly in the processor's L1 cache, eliminating delays due to cache misses. Deep LSTM requires 256 KB of RAM - unacceptable for an embedded microcontroller. In addition, a feature of the self-evolutionary structure is the increase in the number of fuzzy rules only when new regions of the state space arise, which provides $\mathcal{O}(n)$ complexity of updating weights instead of quadratic.

Conclusions

Based on the conducted study of the computational infrastructure and methods of UAV control, it can be concluded that the proposed intelligent-robust architecture based on ASMC and the recurrent network RSEFNN is the optimal solution for hard real-time systems. Mathematical optimization of transcendental functions using the Padé method allowed to reduce computational costs by six times compared to standard implementations, which together with the linear complexity of the algorithm $O(n \cdot m)$ provides a stable control frequency of up to 1000 Hz. The use of self-evolving structures with a strict limit on the number of rules guarantees determinism of execution time (WCET) and prevents computational jitter, which is critically important for aviation certification of on-board software. An important practical result was the creation of a software simulation bench in the MATLAB environment, which, through an object-oriented approach and a fixed integration step using the 4th-order Runge-Kutt method, ensures full reproducibility of numerical experiments. The developed algorithm of deterministic actuation distribution allows to

effectively control UAV with an excess number of engines without using iterative procedures, which guarantees a constant execution time of operations regardless of input signals. This creates a reliable basis for precision maneuvering of the device under conditions of dynamic limitations of actuators and sensor errors.

Experimental validation using the Dryden spectral turbulence model confirmed the high robustness of the hybrid system under conditions of intense stochastic disturbances typical of low altitudes. Statistical profiling showed that the ASMC+RSEFNN method consumes less than 2 KB of RAM and has an execution time margin of more than 33% relative to the critical limit of 2 ms, which makes it suitable for deployment on ARM Cortex-M7 microcontrollers. Compared to deep learning and nonlinear predictive control methods, this approach demonstrates significantly higher computational efficiency, allowing for the combination of intelligent interference compensation with stringent real-time requirements.

ADDITIONAL INFORMATION

AUTHOR CONTRIBUTIONS

Conceptualization, D.M.; methodology, S.T.; validation, D.M.; formal analysis, S.T.; investigation, S.T.; data curation, D.M..; writing-original draft preparation, S.T.; writing-review and editing, D.M.; visualization, S.T.; supervision, D.M.; project administration, D.M. All authors have read and agreed to the published version of the manuscript.

DECLARATION ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS

In preparing this work, the author used DeepL Translate and Grammarly for: grammar and spelling checks, paraphrasing, and rephrasing. After using these tools/services, the author reviewed and edited the content and takes full responsibility for the content of this publication.

1. UAV Trajectory Optimisation in Smart Cities using Modified A* Algorithm Combined with Haversine and Vincenty Formulas / A. Andreou et al. IEEE Transactions on Vehicular Technology. 2023. P. 1–13. URL: https://doi.org/10.1109/tvt.2023.3254604.

2. Karaman S., Frazzoli E. Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research. 2011. Vol. 30, no. 7. P. 846–894. URL: https://doi.org/10.1177/0278364911406761.

3. Distributed Control of Cyber Physical System on Various Domains: A Critical Review / M. Hamzah et al. Systems. 2023. Vol. 11, no. 4. P. 208. URL: https://doi.org/10.3390/systems11040208.

4. A Fixed-Wing UAV Formation Algorithm Based on Vector Field Guidance / X. Wang et al. IEEE Transactions on Automation Science and Engineering. 2022. P. 1–14. URL: https://doi.org/10.1109/tase.2022.3144672.

5. Robust Control of UAV with Disturbances and Uncertainty Estimation / D. Bianchi et al. Machines. 2023. Vol. 11, no. 3. P. 352. URL: https://doi.org/10.3390/machines11030352.

6. Robust Adaptive Fault-Tolerant Control of Quadrotor Unmanned Aerial Vehicles / I. H. Imran et al. Mathematics. 2024. Vol. 12, no. 11. P. 1767. URL: https://doi.org/10.3390/math12111767.

7. K-means online-learning routing protocol (K-MORP) for unmanned aerial vehicles (UAV) adhoc networks / Saifullah et al. Ad Hoc Networks. 2024. Vol. 154. P. 103354. URL: https://doi.org/10.1016/j.adhoc.2023.103354.

8. Nyangaresi V. O. Provably secure authentication protocol for traffic exchanges in unmanned aerial vehicles. High-Confidence Computing. 2023. P. 100154. URL: https://doi.org/10.1016/j.hcc.2023.100154.

9. Comprehensive Review of UAV Detection, Security, and Communication Advancements to Prevent Threats / G. E. M. Abro et al. Drones. 2022. Vol. 6, no. 10. P. 284. URL: https://doi.org/10.3390/drones6100284.

10. Methods of UAVs images segmentation based on k-means and a genetic algorithm / I. Ruban et al. Eastern-European Journal of Enterprise Technologies. 2022. Vol. 4, no. 9(118). P. 30–40. URL: https://doi.org/10.15587/1729-4061.2022.263387.

11. SENTINEL: A Secure and Efficient Authentication Framework for Unmanned Aerial Vehicles / G. Cho et al. Applied Sciences. 2020. Vol. 10, no. 9. P. 3149. URL: https://doi.org/10.3390/app10093149.

12. Sarsam S. M. Cybersecurity Challenges in Autonomous Vehicles: Threats, Vulnerabilities, and Mitigation Strategies. SHIFRA. 2023. Vol. 2023. P. 34–42. URL: https://doi.org/10.70470/shifra/2023/005.

13. Pandey B. K., Pandey D., Sahani S. K. Autopilot control unmanned aerial vehicle system for sewage defect detection using deep learning. Engineering Reports. 2024. URL: https://doi.org/10.1002/eng2.12852.

14. Semantic-Driven Autonomous Visual Navigation for Unmanned Aerial Vehicles / P. Yue et al. IEEE Transactions on Industrial Electronics. 2024. P. 1–11. URL: https://doi.org/10.1109/tie.2024.3363761.

15. Neural Control and Online Learning for Speed Adaptation of Unmanned Aerial Vehicles / V.

Jaiton et al. Frontiers in Neural Circuits. 2022. Vol. 16. URL: https://doi.org/10.3389/fncir.2022.839361.

16. Energy-Efficient Velocity Control for Massive Numbers of UAVs: A Mean Field Game Approach / H. Gao et al. IEEE Transactions on Vehicular Technology. 2022. P. 1. URL: https://doi.org/10.1109/tvt.2022.3158896.

17. Robust bounded control scheme for quadrotor vehicles under high dynamic disturbances / J. Betancourt et al. Autonomous Robots. 2023. URL: https://doi.org/10.1007/s10514-023-10124-6.

18. Shin M., Jung S. A Survey of Behavior Tree-Based Task Planning Algorithms for Autonomous Robotic Systems. 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, Republic of, 16–18 October 2024. 2024. P. 2039–2041. URL: https://doi.org/10.1109/ictc62082.2024.10827191.

19. A motion control framework for autonomous water sampling and swing-free transportation of a multirotor UAV with a cable-suspended mechanism / F. Panetsos et al. Journal of Field Robotics. 2023. URL: https://doi.org/10.1002/rob.22182.

20. Aoki N., Ishigami G. Autonomous tracking and landing of an unmanned aerial vehicle on a ground vehicle in rough terrain. Advanced Robotics. 2022. P. 1–12. URL: https://doi.org/10.1080/01691864.2022.2141078.

21. Rezwan S., Choi W. Artificial Intelligence Approaches for UAV Navigation: Recent Advances and Future Challenges. IEEE Access. 2022. Vol. 10. P. 26320–26339. URL: https://doi.org/10.1109/access.2022.3157626.

22. A Vision-Based Motion Control Framework for Water Quality Monitoring Using an Unmanned Aerial Vehicle / F. Panetsos et al. Sustainability. 2022. Vol. 14, no. 11. P. 6502. URL: https://doi.org/10.3390/su14116502.

23. Tahir M. A., Mir I., Islam T. U. Control Algorithms, Kalman Estimation and Near Actual Simulation for UAVs: State of Art Perspective. Drones. 2023. Vol. 7, no. 6. P. 339. URL: https://doi.org/10.3390/drones7060339.

24. Intelligent Position Controller for Unmanned Aerial Vehicles (UAV) Based on Supervised Deep Learning / J. A. Cardenas et al. Machines. 2023. Vol. 11, no. 6. P. 606. URL: https://doi.org/10.3390/machines11060606.

25. Research on precise route control of unmanned aerial vehicles based on physical simulation systems / Z. Xu et al. Results in Physics. 2023. P. 107200. URL: https://doi.org/10.1016/j.rinp.2023.107200.

Дмитро МЕДЗАТИЙ, Сергій ТАНАСІЙЧУК
Хмельницький національний університет

МЕТОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ІНТЕЛЕКТУАЛЬНИХ АЛГОРИТМІВ КЕРУВАННЯ БЕЗПІЛОТНИМИ ЛІТАЛЬНИМИ АПАРАТАМИ В СИСТЕМАХ ЖОРСТКОГО РЕАЛЬНОГО ЧАСУ

У статті розв'язано актуальну науково-технічну суперечність між необхідністю підвищення точності прецизійного донаведення безпілотних літальних апаратів (БПЛА) в умовах інтенсивних завад та суворими обмеженнями обчислювальних ресурсів бортових систем. Представлена робота фокусується на створенні інтелектуально-робастної архітектури керування, здатної забезпечувати стабільне функціонування в умовах часової латентності, зашумленості навігаційних даних та динамічної непевності об'єкта. Наукова новизна дослідження полягає у розробці та впровадженні рекурентної самоеволюціонуючої нейро-нечіткої мережі (RSEFNN), інтегрованої в контур адаптивного контролера змінного режиму (ASMC) для онлайн-ідентифікації та компенсації нестаціонарних зовнішніх збурень. Ключовою особливістю розробленої архітектури є поєднання високої робастності з обчислювальною ефективністю, що досягається за рахунок математичної оптимізації трансцендентних функцій за методом Паде, яка дозволила скоротити витрати ресурсів у шість разів порівняно зі стандартними реалізаціями. Використання самоеволюціонуючих структур із жорстким обмеженням кількості правил гарантує детермінованість часу виконання (WCET), що є критично важливим для авіаційної сертифікації бортового ПЗ. Важливим практичним результатом стало створення програмного симуляційного стенда в середовищі MATLAB на основі об'єктно-орієнтованого підходу та фіксованого кроку інтегрування методом Рунге-Кутта 4-го порядку, що забезпечує повну відтворюваність чисельних експериментів. Розроблений алгоритм детермінованого розподілу актуації дозволяє ефективно керувати БПЛА з надлишковою кількістю двигунів без використання ітераційних процедур, гарантуючи сталий час виконання операцій незалежно від вхідних сигналів. Експериментальна валідація з використанням спектральної моделі турбулентності Драйдена підтвердила високу робастність гібридної системи в умовах інтенсивних стохастичних збурень, характерних для малих висот. Статистичне профілювання показало, що метод ASMC+RSEFNN споживає менше 2 Кбайт оперативної пам'яті та має запас за часом виконання понад 33% відносно критичного ліміту в 2 мс. Порівняно з методами глибокого навчання та нелінійного прогнозного керування, даний підхід демонструє значно вищу обчислювальну ефективність, дозволяючи поєднувати інтелектуальну компенсацію завад із жорсткими вимогами до реального часу на мікроконтролерах класу ARM Cortex-M7.

Ключові слова: безпілотні літальні апарати (БПЛА), адаптивне керування змінним режимом (ASMC), рекурентна самоеволюціонуюча нечітка нейронна мережа (RSEFNN), вбудовані системи реального часу, детермінованість часу виконання (WCET), метод Паде, обчислювальна ефективність, спектральна модель турбулентності Драйдена, мікроконтролери ARM Cortex-M7, SWaP-обмеження, прецизійне донаведення, стійкість за Ляпуновим, обчислювальний джиттер, інтелектуально-робастна архітектура.