Linear Controller Design with The Use of PSO Algorithm for UAV Trajectory Tracking

Piotr Pawłowski*, Stanisław Konatowski
Military University of Technology, Dept. of Electronics, Institute of Radioelectronics,
Gen. S. Kaliskiego 2, 00-908 Warsaw 46, Poland.

ABSTRACT

This paper uses the PSO (Particle Swarm Optimization) algorithm to linearly control a BSP unmanned aircraft, enabling accurate tracking of a predefined trajectory. In the process of control optimization, the PSO algorithm was implemented with a quadratic cost function. During the study, unstable algorithm behavior was observed, as a result of which a modification was made by introducing a coefficient suppressing the movement of particles in the search space. The main task of the coefficient is to ensure the convergence of the solution. The effect of the coefficient value on population behavior was tested. In the simulation research on tracking, the previously generated trajectory was used, which is a reference for the location, velocity, and course for the linear dynamic BSP model. The results of these tests were presented in the form of waveforms of control signals and waveforms of state variables. By selecting the appropriate parameters, the proposed algorithm enabled the repeatability of results, as well as the proper mapping of the trajectory at the time of the operation of the algorithm not exceeding 0.03 seconds.

Keywords: UAV, linear control, trajectory tracking, PSO.

1 INTRODUCTION

Due to the utility and growing popularity of unmanned aerial vehicles (UAV), especially in military field [1], various methods of control have been developed [2-5]. One important aspect of controlling these platforms is waypoint following, or trajectory tracking. The correct movement of a UAV to a designated destination [10] is one of the main requirements for its autonomy. On the other hand, trajectory tracking is a more detailed task, primarily focused on mapping individual flight parameters during its execution. Such tasks are carried out using appropriate controllers, the purpose of which is to select the right sequence of control signals so that the error resulting from the mismatch to the reference trajectory during the flight is as small as possible.

A classic approach in such control is the use of PID (Proportional Integral Derivative) linear controllers [6]. Their basic task is to regulate the control signal on the basis of the error resulting from the difference between the controlled value and reference value. The PID controller is a SISO (Single Input Single Output) system with feedback, which does not require knowledge of the controlled system model. Assuming a time-varying reference signal, one can implement a trajectory tracking problem [7] using, for example, a tracking system. In turn, works [7] and [8] present a method of trajectory tracking using an LQR – Linear Quadratic Regulator. The LQR controller implements optimal control (regulation) by minimizing the quadratic cost function [9]. Trajectory tracking, resulting from the difference between the target trajectory and the original state, can be obtained by converting the linear system into its equivalent in the error space, respectively. In [7], the authors present a direct comparison of PID and LQR controllers in the context of trajectory tracking, and demonstrate the superiority of the LQR controller. MPC (Model Predictive Control) is another widely used technique for designing controllers [10, 12]. It consists of determining an optimal sequence of control signals in parallel with the state prediction, minimizing the cost function determined by the designer (usually in a square form with selected weighting factors) in such a way that the UAV trajectory tends to the reference trajectory. The optimization takes place over a given time window, called the horizon. After performing the optimization at a given time, the system performs only the first step of the designated control sequence, then performs the optimization again. This process repeats continuously during system

*piotr.pawlowski@wat.edu.pl; phone +48 261 837 475; www.wat.edu.pl

Radioelectronic Systems Conference 2019, edited by Piotr Kaniewski, Jan Matuszewski, Proc. of SPIE
Vol. 11442, 114420Z · © 2020 SPIE · CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2565129

operation. MPC controllers are used with great success [13, 15-16], despite the requirements of significant computational load. Calculations are often carried out on separate platforms that send only designated optimization solutions to UAV platforms. Authors in [12] show MPC variants that can be used directly on UAVs.

This paper presents a method of suboptimal feedback control using PSO (Particle Swarm Optimization) [17]. PSO has many applications in various fields of science [14, 19], and also in controller design. The rest of the article presents the use of PSO to determine the suboptimal control that implements UAV trajectory tracking.

2 MATHEMATICAL MODEL

The paper assumes the UAV model as a rigid body with six degrees of freedom (6 DoF): three representing translational displacement, and three corresponding to spatial orientation. Movement is described in two coordinate systems: inertial $\boldsymbol{F}^e = \{\boldsymbol{X}_e, \boldsymbol{Y}_e, \boldsymbol{Z}_e\}$ associated with Earth, and non-inertial $\boldsymbol{F}^b = \{\boldsymbol{X}_b, \boldsymbol{Y}_b, \boldsymbol{Z}_b\}$ associated with the UAV (Fig. 1).

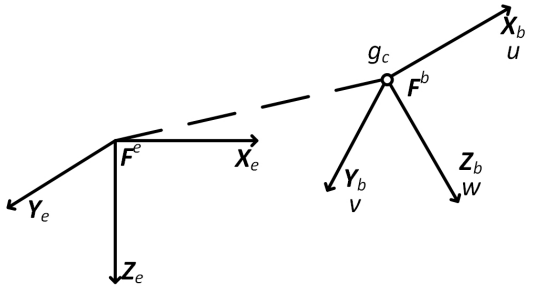

Fig 1. Reference coordinate systems: inertial Earth and non-inertial UAV.

An UAV moves by regulating angular speeds $\omega_{1\text{-}4}$ of four engines assembled with propellers generating thrusts $f_{t1\text{-}4}$. In the UAV design presented in Fig. 2, motors are located on opposite sides of the intersection of two perpendicular axes. These axes simultaneously coincide with the axes $\boldsymbol{X}_b$, $\boldsymbol{Y}_b$. Due to the difference in forces between the pairs of engines, appropriate torques τ_x and τ_y are generated, allowing rotation around the corresponding axes – rotation ϕ around the $\boldsymbol{X}_b$ axis and rotation θ around the $\boldsymbol{Y}_b$ axis (pitch). Propellers also cause torques $\tau_{1\text{-}4}$ acting in the plane $[\boldsymbol{X}_b\ \boldsymbol{Y}_b]$. The resultant net torque τ_z causes rotation ψ around the axis $\boldsymbol{Z}_b$ (yaw).

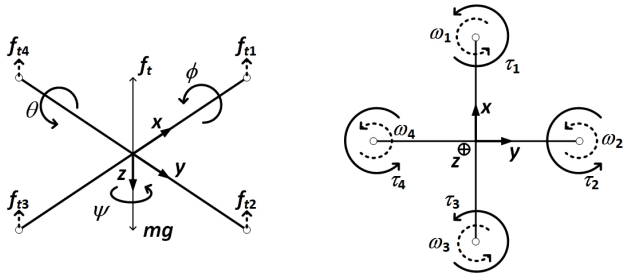

Figure 2. Arrangement of UAV engines and action of forces.

A mathematical model of the UAV was built based on the following assumptions:

- symmetric construction,
- rigid structure,
- the UAVs center of gravity g_c coinciding with the beginning of coordinate system $\boldsymbol{F}^b$.

2.1 System dynamics

For the simulation, a model derived from the Newton and Euler equations presented in [20] was used. Assuming the vector $[x\ y\ z\ \phi\ \theta\ \psi]^{\mathrm{T}}$ as the position and orientation vector in the $\boldsymbol{F}^e$ system, as well as the vector $[u\ v\ w\ p\ q\ r]^{\mathrm{T}}$ for the vector of linear and angular velocities in the $\boldsymbol{F}^b$ system, the dynamic UAV model takes the following form:

$$\begin{cases} \dot{\phi} = p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ \dot{\theta} = q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} = r\frac{c(\phi)}{c(\theta)} + q\frac{s(\phi)}{c(\theta)} \\ \dot{p} = qr\frac{I_y - I_z}{I_x} + \frac{\tau_x}{I_x} - \frac{k_p p}{I_x} \\ \dot{q} = pr\frac{I_z - I_x}{I_y} + \frac{\tau_y}{I_y} - \frac{k_q q}{I_y} \\ \dot{r} = pq\frac{I_x - I_y}{I_z} + \frac{\tau_z}{I_z} - \frac{k_r r}{I_z} \\ \dot{u} = rv - qw - g[s(\theta)] - \frac{k_u u}{m} \\ \dot{v} = pw - ru + g[s(\phi)c(\theta)] - \frac{k_v v}{m} \\ \dot{w} = qu - pv + g[c(\theta)c(\phi)] - \frac{k_w w}{m} - \frac{f_t}{m} \\ \dot{x} = w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] + u[c(\theta)s(\psi)] \\ \dot{z} = w[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)] \end{cases} \quad (1)$$

with the following definitions: $s(\alpha) := sin(\alpha)$, $c(\alpha) := cos(\alpha)$, $t(\alpha) := tan(\alpha)$, and m as the UAV's mass. Inertia matrix $\boldsymbol{I}$, due to platform construction symmetry, takes the diagonal form of

$$diag\ (\boldsymbol{I}) = [I_x \quad I_y \quad I_z]. \quad (2)$$

Minor damping reflecting aerodynamic drag has been added to the model [22]. Diagonal matrices of displacement and rotation aerodynamic coefficients are expressed by the following relationships:

$$diag\ (\boldsymbol{D_t}) = [k_u \quad k_v \quad k_w], \quad (3)$$

$$diag\ (\boldsymbol{D_r}) = [k_p \quad k_q \quad k_r]. \quad (4)$$

By organizing individual variables, a state vector is obtained:

$$\boldsymbol{x} = [\phi\ \theta\ \psi\ p\ q\ r\ u\ v\ w\ x\ y\ z]^T \quad (5)$$

along with the control input vector

$$\boldsymbol{u} = [f_t\ \tau_x\ \tau_y\ \tau_z]^T. \quad (6)$$

This vector contains, respectively, the value of the resultant lift generated by the engines set f_t, torques τ_x and τ_y in the Ox and Oy axes of the UAV platform, and torque τ_z in the Oz UAV axis. Finally, the form of a nonlinear dynamic system is expressed as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \quad (7)$$

2.2 Linearization of nonlinear model

The four-rotor model is a highly non-linear system with many coupled variables. In order to enable linear analysis of the system, the model should be linearized, assuming an appropriate operation point. The most frequently chosen operation point is the equilibrium condition obtained when the four-rotor model hovers. This means no UAV movement, so its dynamics are described by the following differential equation:

$$\dot{\boldsymbol{x}}^* = \boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*) = 0, \quad (8)$$

where $\boldsymbol{x}^* = [0^{1\times9}\ x\ y\ z]$ and $\boldsymbol{u}^* = [mg\ 0\ 0\ 0]$ are nominal values for the given operation point. This condition occurs when all

four engines generate a net thrust equal to the weight *mg* of the UAV in the opposite direction. Under ideal conditions, there are no other forces or moments present.

The system was linearized using the Control System Toolbox of MATLAB/Simulink software, thanks to which the LTI (Linear Time Invariant) state space system described by two vector-matrix equations was obtained:

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu}, \;\; \boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du}, \tag{9}$$

where $\boldsymbol{A} \in \Re^{12\times12}$ is the system matrix, $\boldsymbol{B} \in \Re^{4\times12}$ is the control matrix, $\boldsymbol{C} \in \Re^{12\times12}$ is the output matrix, and $\boldsymbol{D} \in \Re^{12\times4}$ is the transmission matrix. For the calculations, the parameters contained in Table 1 were used and the following results were obtained:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5.10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5.10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3.79 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -9.81 & 0 & 0 & 0 & 0 & -1.25 & 0 & 0 & 0 & 0 & 0 \\ 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & -1.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{10}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 51.02 & 0 & 0 \\ 0 & 0 & 51.02 & 0 \\ 0 & 0 & 0 & 37.88 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2.50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{11}$$

$$\boldsymbol{C} = \boldsymbol{I}^{12\times12}, \tag{12}$$

$$\boldsymbol{D} = 0^{12\times4}. \tag{13}$$

The parameters used for UAV modeling are presented in Table 1.

Table 1. Parameters used to create UAV model.

Parameter		**Value**
Inertia matrix		$deg(\boldsymbol{I}) = [0{,}0196 \quad 0{,}0196 \quad 0{,}0264]$
Aerodynamic coefficients	translation	$deg(\boldsymbol{D_t}) = [0{,}5 \quad 0{,}5 \quad 0{,}5]$
	rotation	$deg(\boldsymbol{D_r}) = [0{,}1 \quad 0{,}1 \quad 0{,}1]$
UAV mass		$m = 0.4\ kg$

3 CONTROLER DESIGN

3.1 Model discretization

This application uses a discrete controller with a sampling time of T_c = 0.1s using the PSO algorithm for cost function minimalization. The task of the algorithm, considering the assumed quality indexes, is to select such a value of control vector $\boldsymbol{u}(k)$ at a given moment k that the difference between the predicted state $\boldsymbol{x}(k+1|k)$ and corresponding reference point $\boldsymbol{x}_{ref}(k+1)$ is minimal. Due to its discrete nature, the dynamic model (9-13) has also been transformed to the discrete form:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_d\boldsymbol{x}(k) + \boldsymbol{B}_d\boldsymbol{u}, \;\; \boldsymbol{y}(k) = \boldsymbol{C}_d\boldsymbol{x}(k) + \boldsymbol{D}_d\boldsymbol{u}(k), \tag{14}$$

with *c2d*() command in MATLAB. For sampling time *Ts* = 0.05s, the following results were obtained:

$$\boldsymbol{A}_d = \begin{bmatrix} 1 & 0.0783 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6004 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0832 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6847 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0783 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6004 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0471 & 0.0014 & 1 & 0.0940 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9222 & 0.0399 & 0 & 0.8825 & 0 & 0 & 0 & 0 \\ -0.0471 & -0.0014 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.0940 & 0 & 0 \\ -0.9222 & -0.0399 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8825 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.0940 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8825 \end{bmatrix}, \tag{15}$$

$$\boldsymbol{B}_d = \begin{bmatrix} 0 & 0.0488 & 0 & 0 \\ 0 & 0.8992 & 0 & 0 \\ 0 & 0 & 0 & 0.1676 \\ 0 & 0 & 0 & 3.1531 \\ 0 & 0 & 0.0488 & 0 \\ 0 & 0 & 0.8992 & 0 \\ 0 & 0 & -0.0004 & 0 \\ 0 & 0 & -0.0161 & 0 \\ 0 & -0.0004 & 0 & 0 \\ 0 & -0.0161 & 0 & 0 \\ -0.0120 & 0 & 0 & 0 \\ -0.2350 & 0 & 0 & 0 \end{bmatrix}, \tag{16}$$

$$\boldsymbol{C}_d = \boldsymbol{I}^{12\times12}, \tag{17}$$

$$\boldsymbol{D}_d = 0^{12\times4}. \tag{18}$$

3.2 PSO algorithm for control optimization

The PSO algorithm searches for solutions in the n-dimensional space of the function to be optimized [18]. This is accomplished by iteratively moving the population of p particles to find coordinates for which the value of a given cost function is minimal. Each i-th particle has the following known parameters:

- position $\boldsymbol{x}_i \in \Re^n$,
- velocity $\boldsymbol{v}_i \in \Re^n$,
- actual best own position $\boldsymbol{p\_best}_i \in \Re^n$.

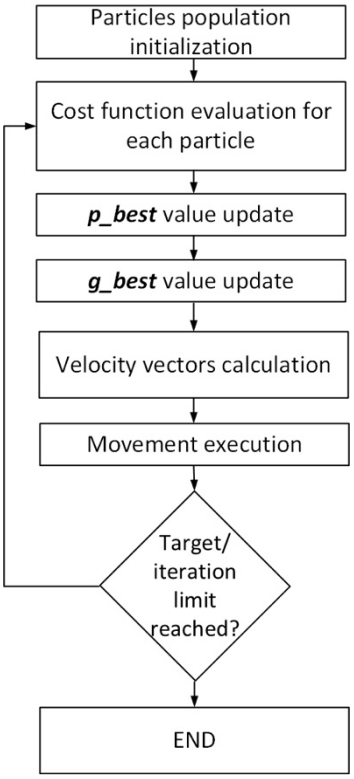

Figure 3. PSO algorithm block diagram.

The particle with best global value $\boldsymbol{g\_best} \in \Re^n$ is also determined for the entire population. The evolution of the algorithm is described by expressions:

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 rand()[\boldsymbol{p}_{\boldsymbol{best}_i} - \boldsymbol{x}_i(t)] + c_2 rand()[\boldsymbol{g}_{\boldsymbol{best}} - \boldsymbol{x}_i(t)], \tag{19}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1), \tag{20}$$

where w is the inertia weight, c_1 is the value of the so-called personal coefficient, c_2 is the value of the social coefficient, and *rand*() is a pseudo-random function which selects a number from a range [0, 1]. The movement of each particle depends on its current velocity, its best own position so far, and the global best position for time (iteration) t. The c_1 and c_2 coefficients determine the tendency to choose the direction of particle movement. The *rand*() function gives the algorithm a pseudo-random character, while increasing its searching performance.

Due to its stochastic properties, the PSO is a metaheuristic algorithm [18, 23]. It does not guarantee an optimal solution, but only determines a sub-optimal variant that meets given assumptions, as close as possible to the optimal solution. This allows for the reduction of the computational load of the solution, at the expense of its accuracy. This is especially useful when reaction speed is more important than precision. The algorithm ends its execution after reaching the iteration limit or meeting the given criterion, returning the best solution so far ***g_best***.

3.3 Search space and cost function optimization.

During the algorithm execution, particles explore the $U \subset \Re^4$ space of the control vector $\boldsymbol{u}$ values. At the beginning of each iteration (after the movement of the entire population), a cost function is calculated for each particle, considering its current position. In the proposed application, the function being optimized takes a quadratic form:

$$J_i^t = \left[\boldsymbol{x}_{ref}(k+1) - \boldsymbol{x}_i(k+1)\right]^T \boldsymbol{Q}\left[\boldsymbol{x}_{ref}(k+1) - \boldsymbol{x}_i(k+1)\right] +$$
$$+[\boldsymbol{u}_{nom} - \boldsymbol{u}_i(k)]^T \boldsymbol{R}[\boldsymbol{u}_{nom} - \boldsymbol{u}_i(k)] + [\boldsymbol{u}(k) - \boldsymbol{u}_i(k+1)]^T \boldsymbol{R}_{\Delta u}[\boldsymbol{u}(k) - \boldsymbol{u}_i(k+1)], \tag{21}$$

where J^t_i is the cost function value for the i-th particle in the iteration t, $\boldsymbol{u}_i(k+1)$ means the coordinates of the i-th particle (selected values for control signals), $\boldsymbol{x}_i$ means the state that the UAV would achieve by applying the control determined by the coordinates of the particle i, and $\boldsymbol{u}_{nom}$ is the nominal value of the control vector (for the equilibrium point). Disturbances and measurement errors are not considered [21]. For known initial values of the state vector, the values of $\boldsymbol{x}_i(k+1)$ are determined directly from the expression (14).

Diagonal weighting matrices $\boldsymbol{Q} \geq 0 \in \Re^{12\times12}$ and $\boldsymbol{R}$, $\boldsymbol{R}_{\Delta u} \geq 0 \in \Re^{4\times4}$ determine the impact of each factor in total value of the cost function. The $\boldsymbol{Q}$ matrix is responsible for the influence of error magnitude between state variables and their reference values (fitness criterion). The $\boldsymbol{R}$ matrix determines the reaction to the difference between the nominal values of the control signals and the values selected in a given solution (energy criterion). The $\boldsymbol{R}_{\Delta u}$ matrix is responsible for the impact of the difference between the selected control and the current signal level (signal dynamics criterion).

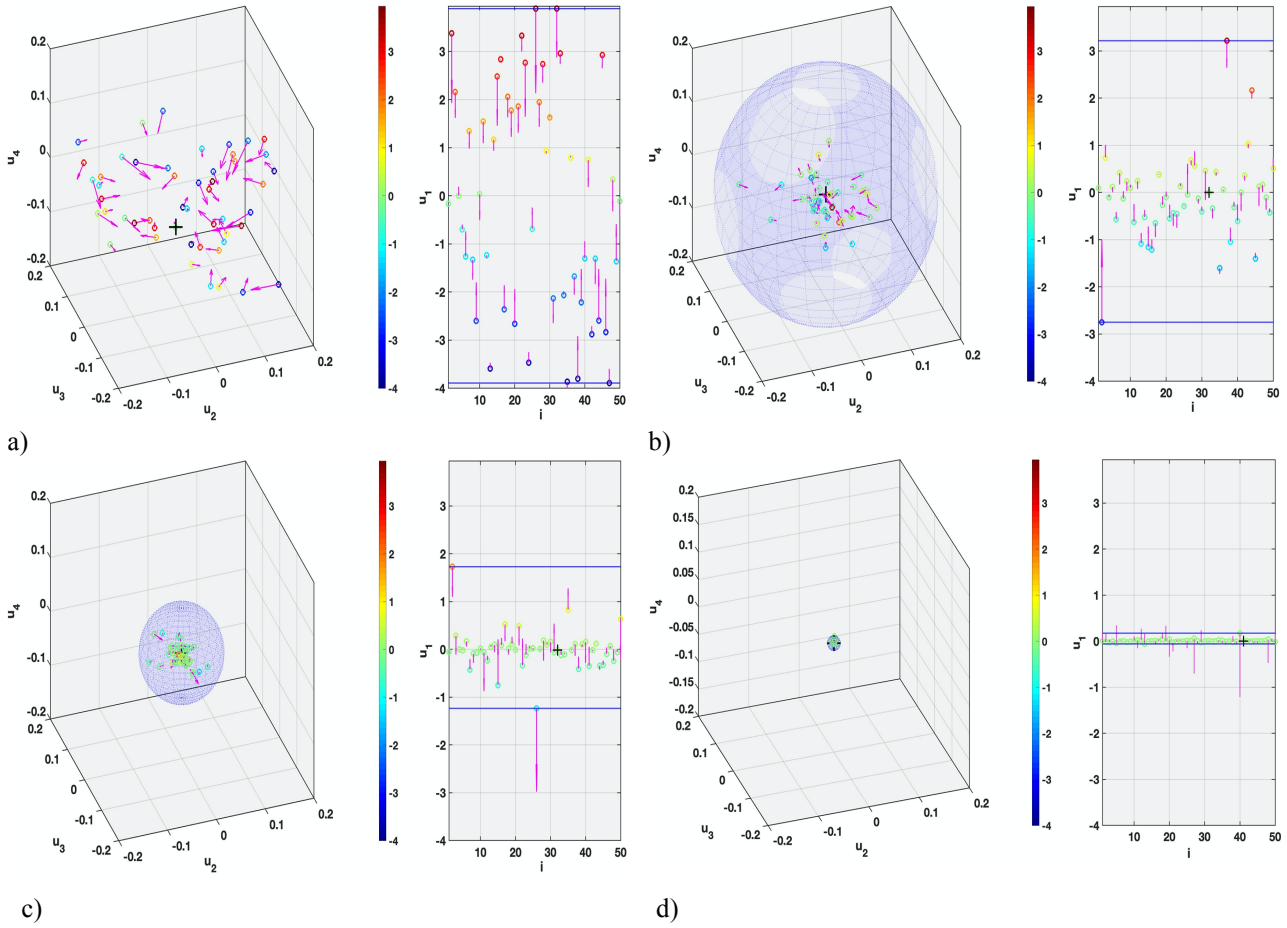

Figure 4. a-d) PSO solution searching process.

The algorithm allows for the explicit limitation of the values of the control signals. By imposing a condition on the allowed positions of particles in the search space

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}_i \leq \boldsymbol{u}_{max} \tag{22}$$

it can be ensured that no control value outside the specified range is selected.

4 SIMULATION RESULTS

4.1 Initial parameter configuration

The following parameters for the PSO algorithm were adopted for the simulation tests: number of particles $p = 40$, iterations limit $it\_lim = 150$, inertia coefficient $w = 0.9298$, and personal and social coefficients $c_1 = c_2 = 1.4986$. The diagonals of the weight matrices were selected as follows:

$$diag(\boldsymbol{Q}) = [1 \quad 1 \quad 5 \quad 1 \quad 1 \quad 1 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5], \tag{23}$$

$$diag(\boldsymbol{R}) = [0{,}0057 \quad 1{,}25 \quad 1{,}25 \quad 5], \tag{24}$$

$$diag(\boldsymbol{R}_{\Delta u}) = [1 \quad 1 \quad 1 \quad 1]. \tag{25}$$

The search space of the PSO algorithm has been limited by the following ranges:

$$\begin{bmatrix} -4 \\ -0.2 \\ -0.2 \\ -0.1 \end{bmatrix} \leq \boldsymbol{u}_i \leqslant \begin{bmatrix} 3.942 \\ 0.2 \\ 0.2 \\ 0.1 \end{bmatrix}, \tag{26}$$

where this range refers to the nominal control $\boldsymbol{u}_{nom}$ in which the value of the first coordinate mg has been compensated, so $\boldsymbol{u}_{nom} = [0\ 0\ 0\ 0]$.

In order to perform the simulations, a test trajectory was generated containing references for position, velocity and course (Figs. 5-6).

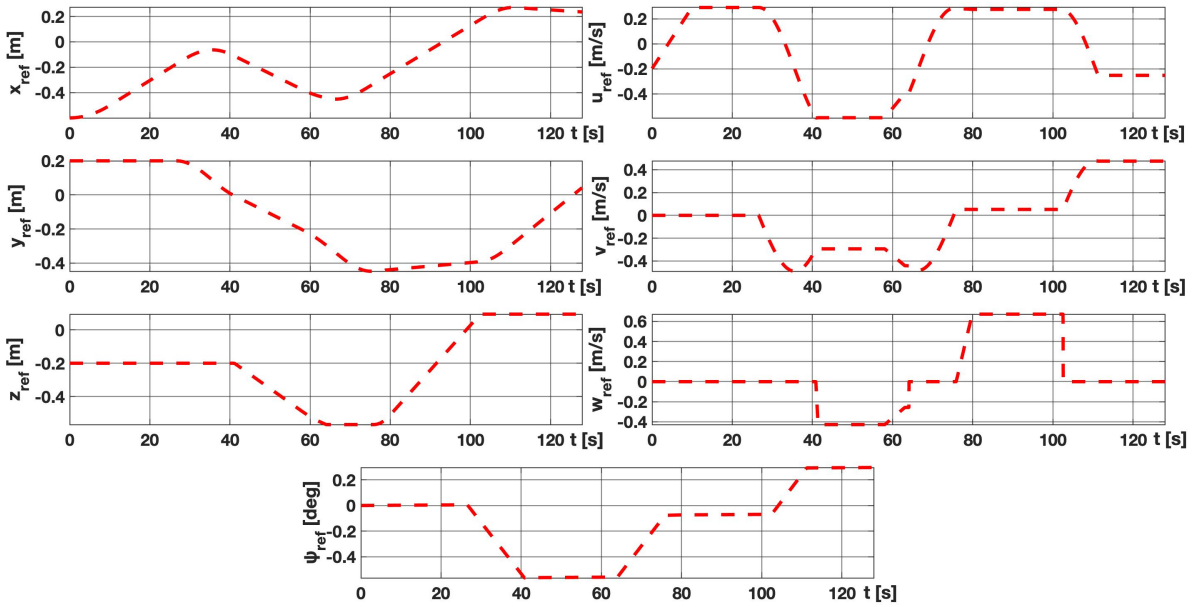

Figure 5. Reference values for state variables of UAV.

Other state variables, without reference values, were compared with their equilibrium values $\boldsymbol{x}^*$, so

$$\boldsymbol{x}_{ref}(k) = [0\ 0\ \psi_{ref}(k)\ 0\ 0\ 0\ u_{ref}(k)\ v_{ref}(k)\ w_{ref}(k)\ x_{ref}(k)\ y_{ref}(k)\ z_{ref}(k)]^T. \tag{27}$$

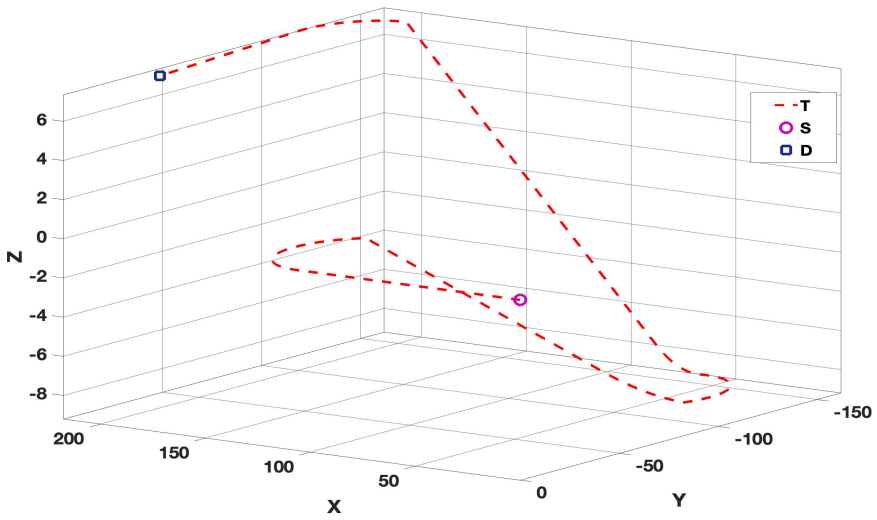

Figure 6. 3D path profile (T): S – starting point, D – destination point.

4.2 PSO algorithm improvements

The first attempts showed unstable operation of the algorithm. Control signal waveforms generated by the algorithm were characterized by strong distortion (Fig. 7).

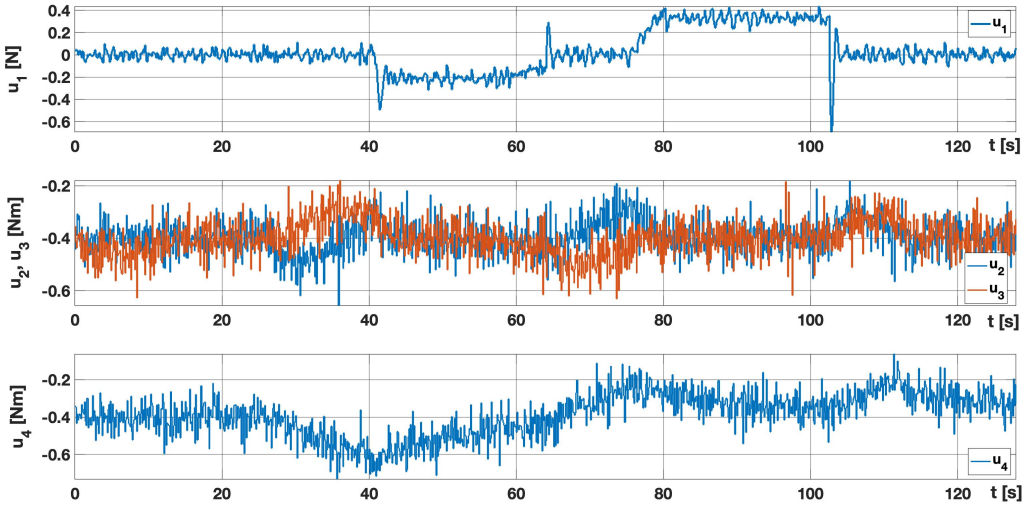

Figure 7. Control signal waveforms generated in initial configuration.

Despite the occurrence of undesirable distortion of control signals, the algorithm showed the ability to track the given trajectories of individual state variables (Figs. 8-9). Therefore, an improvement in the process of determining signal values could be achieved by the additional modification and tuning of algorithm parameters.

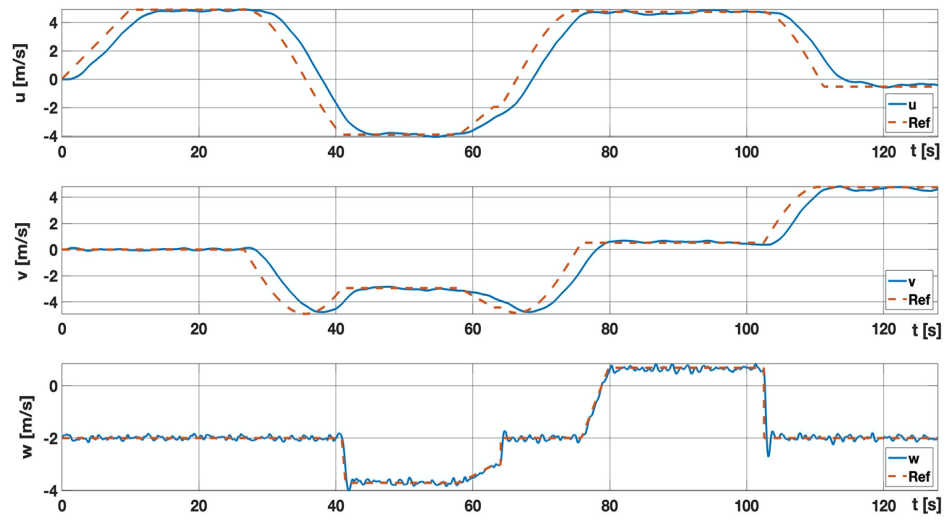

Figure 8. State variable waveforms [u, v, w] for initial configuration.

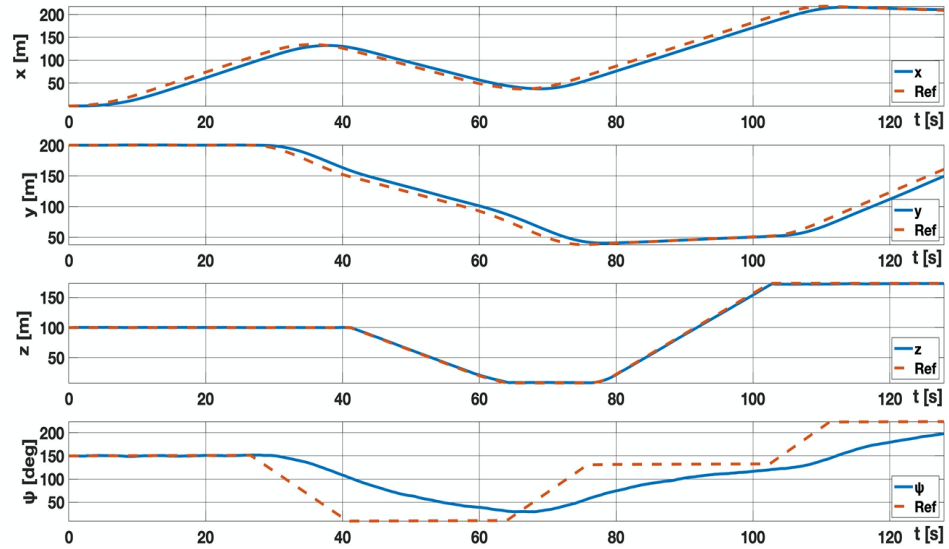

Figure 9. State variables waveform [x, y, z, ψ] for initial configuration.

After observing the execution of the PSO algorithm, it was noticed that the problem with stability is due to the difficulty in achieving convergence. By design, each particle should gradually approach the global best solution, while reducing its velocity in subsequent iterations. Thus, it is possible to exploit more closely neighborhood of the best solution. This effect was not achieved in the basic algorithm configuration (Fig. 10). The velocities of the particles did not decrease sufficiently, resulting in long-distance motion. The end result was a large spread around the globally best solution. Let $S \subset \Re^+$ be a set of Euclidean distances s_i of all particles from the position ***g_best***, so

$$s_i = \|\boldsymbol{g_{best}} - \boldsymbol{x}_i\|. \tag{28}$$

The average distance between the particles and ***g_best*** in the last iteration was assumed as the measure of population spread:

$$\varepsilon = \frac{\sum_{i=1}^{p} s_i}{p}. \tag{29}$$

In order to improve the convergence of the algorithm, expression (19) was modified by implementing a factor λ that scales the velocity of particles in each iteration:

$$\boldsymbol{v}_i(t+1) = \lambda\{w\boldsymbol{v}_i(t) + c_1 rand()[\boldsymbol{p_{best}}_i - \boldsymbol{x}_i(t)] + c_2 rand()[\boldsymbol{g\_best} - \boldsymbol{x}_i(t)]\}. \quad (30)$$

Table 2 presents the simulation results for the factor λ in the range <0.3, 0.8>. Without considering λ, the algorithm obtained the following results: average value of the dispersion index $\varepsilon_{avr} = 1.3944$, minimum value $\varepsilon_{min} = 0.9380$, maximum value $\varepsilon_{max} = 2.0023$, and total path cost $J_{cl} = 8.5574e + 05$.

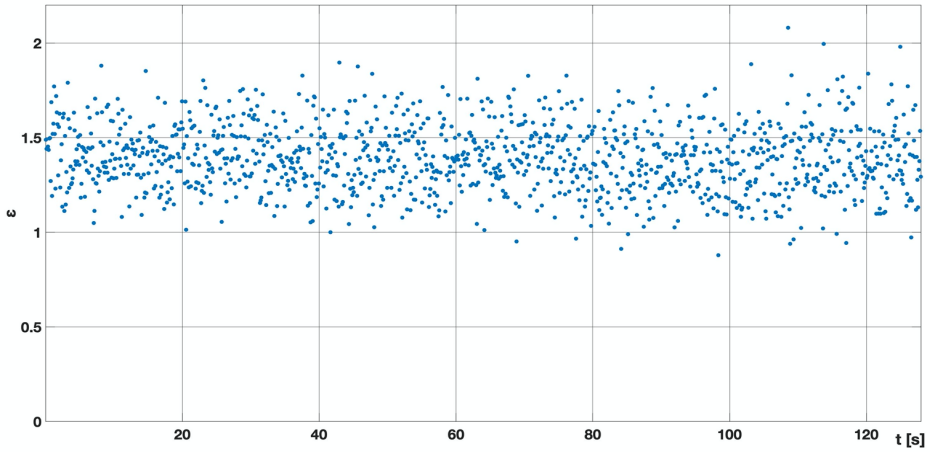

Figure 10. Values of ε for initial configuration.

Table 2. Comparison of experiment results for different values of λ.

λ	$\varepsilon_{śr}$	ε_{min}	ε_{max}	$J_{c1} - J_{c\lambda}$
0.3	1.6238e-11	9.3965e-19	2.8786e-09	3.4817e+03
0.35	6.1628e-10	5.8451e-18	2.6602e-08	1.5645e+03
0.4	8.0896e-09	6.2658e-17	6.9009e-08	5.4511e+03
0.45	1.7173e-08	6.8815e-15	9.4198e-08	1.2267e+04
0.5	1.9109e-08	4.4003e-11	8.4363e-08	1.3897e+04
0.55	2.1086e-08	3.8774e-11	9.8508e-08	1.3913e+04
0.6	2.4056e-08	5.0962e-11	1.1915e-07	1.3913e+04
0.65	2.7744e-08	4.7023e-11	1.6079e-07	1.3914e+04
0.7	3.4030e-08	8.8582e-11	2.0271e-07	1.3913e+04
0.75	6.1766e-08	4.1638e-09	6.8558e-07	1.3913e+04
0.8	5.1780e-06	3.5249e-07	9.3663e-04	1.3913e+04

Based on the simulation results, it can be concluded that the best variant is λ equal to 0.65. Fig. 11 shows a graph of changes in the index ε for the selected value λ. The difference can be clearly seen compared to the initial version of the algorithm – the magnitude order of the average value ε_{avr} is several times smaller, which coincides with the original assumptions.

The total cost J_c also improved, mainly due to the decrease in the cost of control signal dynamics, whose waveforms are shown in Fig. 12.

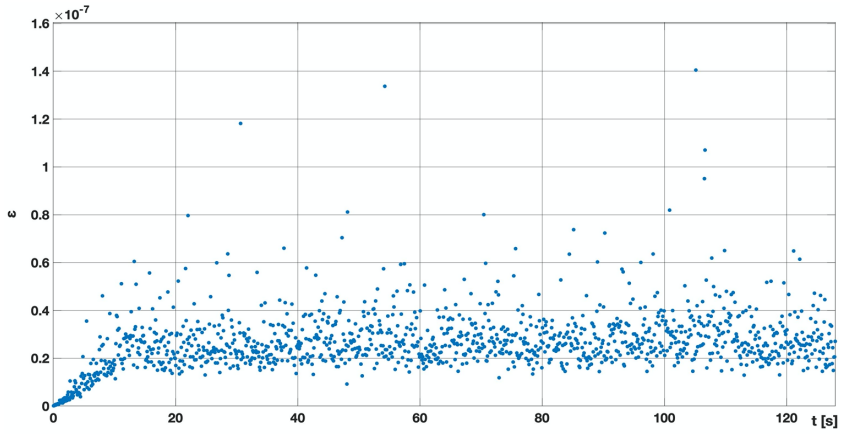

Figure 11. Values of ε for $\lambda = 0.65$.

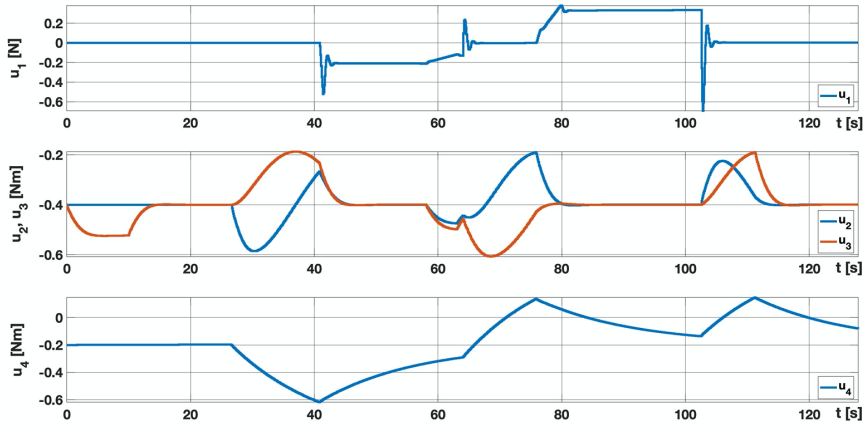

Figure 12. Control signals generated for $\lambda = 0.65$.

After the stabilization of the control signals, the trajectories were smoothened (Figs. 13-14) but their shape and fitness changed only slightly. This indicates that the algorithm's instability has no effect on its efficiency (in the context of matching to reference trajectories). The sum of squares of differences between real values and reference state variables $[u, v, w, x, y, z, \psi]$ at all discrete moments k was taken as fitness criterion μ_e:

$$\mu_e = e_u + e_v + e_w + e_x + e_y + e_z + e_\psi, \tag{31}$$

$$e_\alpha := \sum_k [\alpha_{ref}(k) - \alpha(k)]^2. \tag{32}$$

Table 3 compares the individual components of the indicator μ_e for $\lambda_{opt} = 0.65$ and $\lambda_0 = 1$.

Table 3. Comparison of error values for the basic and modified variants of the PSO algorithm

	e_u	e_v	e_w	e_x	e_y	e_z	e_ψ	μ_e
λ_0	1.8193e03	1.0054e03	3.2405	2.4148e05	1.4808e05	41.9107	5.2868e06	5.6792e06
λ_{opt}	1.8307e03	1.0497e03	2.1436	2.1246e05	1.3312e05	32.6742	5.0034e06	5.3519e06
$e_{opt} \div e_0$	1.0060	1.0441	0.6615	0.8798	0.8990	0.7796	0.9464	0.9424

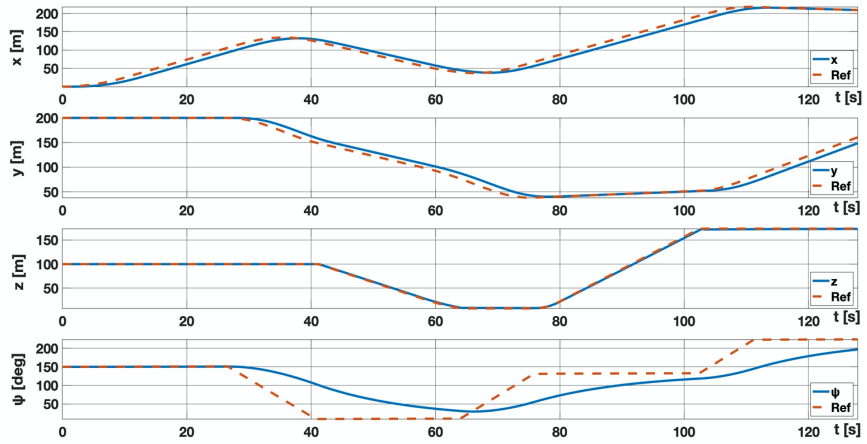

Figure 13. State variable waveforms [u, v, w] for $\lambda = 0.65$.

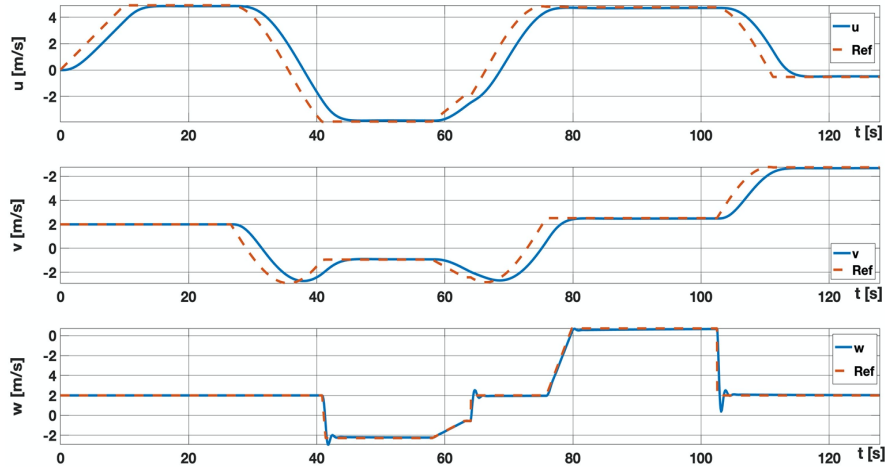

Figure 14. State variable waveforms [x, y, z, ψ] for $\lambda = 0.65$.

4.3 Selection of weight matrices elements

Due to the lack of significant improvement in the adjustment of state variable trajectories to their reference values (despite the stabilization of the PSO algorithm), attempts have been made to improve the fitness indicator by changing the values of the weight matrices. Analyzing the results from Table 3, it can be seen that the largest errors are coupled with variables u, v, x, y, ψ, with error e_ψ an order of magnitude greater than other errors. However, weighting factors for individual matrices were intuitively modified, considering that the weighting factor for e_ψ should be greater than the weighting factors corresponding to the remaining state variables being tracked. Finally, the weighting matrices took the following values:

$$diag(\boldsymbol{Q}) = [1 \quad 1 \quad 25 \quad 1 \quad 1 \quad 1 \quad 20 \quad 20 \quad 5 \quad 20 \quad 20 \quad 5], \tag{33}$$

$$diag(\boldsymbol{R}) = [0{,}00057 \quad 0{,}125 \quad 0{,}125 \quad 0{,}5], \tag{34}$$

$$diag(\boldsymbol{R}_{\Delta u}) = [1 \quad 1 \quad 1 \quad 1]. \tag{35}$$

The simulation results for the new values of the weight coefficients are presented in Figs. 15-17. Table 4 presents a comparison of individual components of the index μ_e for the modified values of $\boldsymbol{R}_{\text{opt}}$ and the unmodified $\boldsymbol{R}_0$.

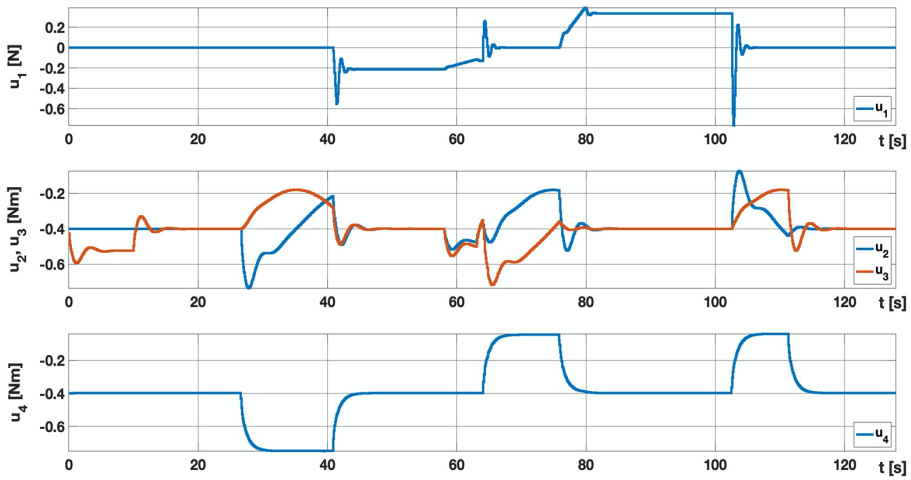

Figure 15. Control signal waveforms for modified weight matrices.

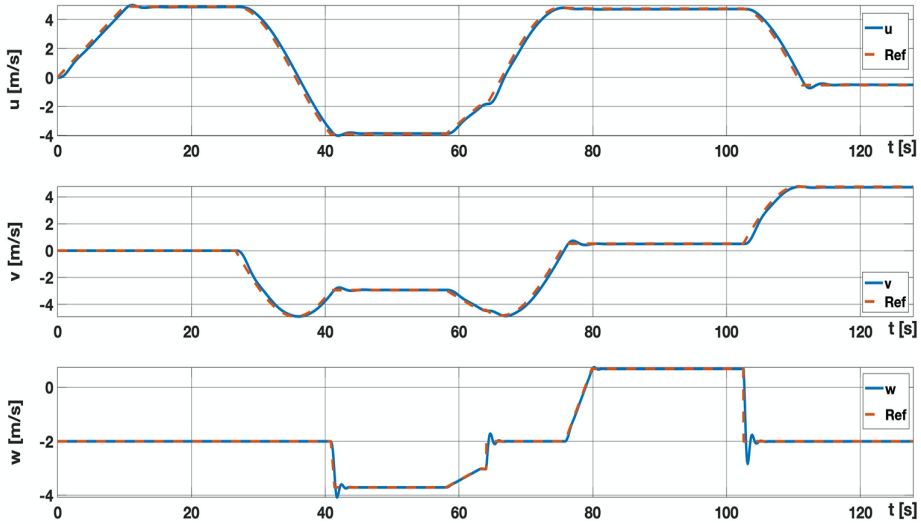

Figure 16. State variable waveforms [*u, v, w*] for modified weight matrices.

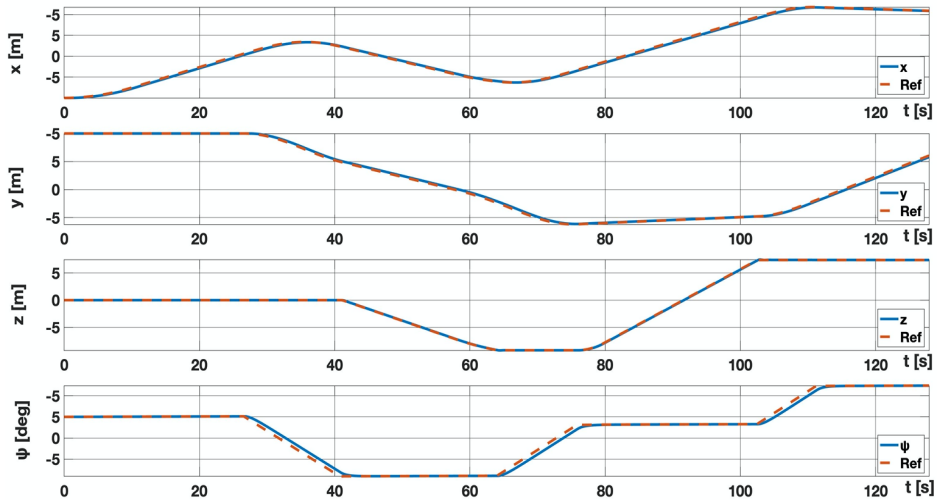

Figure 17. State variable waveforms [*x, y, z, ψ*] for modified weight matrices.

Table 4. Comparison of error values for the basic and modified variants of the PSO algorithm.

	e_u	e_v	e_w	e_x	e_y	e_z	e_ψ	μ_e
R_0	1.8307e03	1.0497e03	2.1436	2.1246e05	1.3312e05	32.6742	5.0034e06	5.3519e06
R_{opt}	82.4018	56.4724	2.1201	9.3949e03	6.4715e03	2.9794	7.1436e04	8.7447e04
$e_{Ropt} \div e_{R0}$	0.0450	0.0538	0.6615	0.0442	0.0486	0.0911	0.0142	0.0163

5 CONCLUSIONS

The applied PSO algorithm presents accurate results in the discrete process of controlling an unmanned aerial vehicle. Despite the difficulties associated with the unstable operation of the algorithm, the implemented factor λ suppressing the motion of the population of particles allowed for the improvement of the average population dispersion by almost 8 orders of magnitude. Application of the coefficient $\lambda = 0.65$ resulted in the reduction of ε_{avr} from 1.3944 to 2.7744e-08. Thus, much greater precision of optimization was achieved, while slightly increasing the total cost of the route.

The stabilization of the PSO algorithm did not bring significant benefits in terms of matching to the given trajectory, reducing the value of μ_e by only 5.8%. In order to improve the fitness index, weight matrices determining the cost function were modified. The selection of new weighting coefficients allowed for improvement of μ_e by over 98%, which was illustrated in the plot of individual state variables.

In subsequent stages of work on the application of the PSO algorithm to optimize the tracking of UAV trajectories, one can consider the application of integration term for fitness error into cost function. This would eliminate visible steady-state error. Another modification may be the introduction of requirements on the output values of state variables, thus limiting the system dynamics.

REFERENCES

[1] Kaniewski, P., Leśnik, C., Serafin, P. and Łabowski, M., "Chosen Results of Flight Tests of WATSAR System," 17th International Radar Symposium IRS 2016, Kraków, pp. 1-5 (2016).

[2] Gopalakrishnan, E. M., "Quadcopter Flight Mechanics and Control Algorithms," Diploma Thesis Assignment, Czech Technical University in Prague, Prague (2016).

[3] Zulu, A. and John, S., "A Review of Control Algorithms for Autonomous Quadrotors," Open Journal for Applied Sciences, 4, pp. 547-556 (2014).

[4] Kaniewski, P., Kazubek, J. and Kraszewski T., "Application of UWB Modules in Indoor Navigation System," 2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS) (2017).

[5] Kraszewski T. and Czopik G., "Tracking of land vehicle motion with the use of distance measurements," Proc. SPIE 11055, XII Conference on Reconnaissance and Electronic Warfare Systems, 110550Y; DOI: 10.1117/12.2524954 (2019).

[6] Holejko, D. and Kościelny, W. J., "Automatyka procesów ciągłych," Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa (2012).

[7] Changlong, L., Jian, P. and Yufang, C., "PID and LQR Trajectory Tracking Control for An Unmanned Quadrotor Helicopter: Experimental Studies," Proceedings of the 35th Chinese Control Conference (2016).

[8] Xie, H., Cabecinhas, D. and Cunha, R., "A trajectory tracking LQR controller for a quadrotor: Design and experimental evaluation," TENCON 2015 - 2015 IEEE Region 10 Conference (2015).

[9] Zhou, K. and Doyle, J. C., "Essentials of Robust Control," Prentice Hall (1999).

[10] Abdolhosseini, M., "Model Predictive Control of an Unmanned Quadrotor Helicopter: Theory and Flight Tests," A Thesis in The Department of Mechanical and Industrial Engineering, Concordia University, Canada (2012).

[11] Matuszewski, J. and Dikta, A., "Emitter location errors in electronic recognition system," Proceedings of SPIE - The International Society for Optical Engineering, 10418, art. no. 104180C, pp. C1-C8. DOI: 10.1117/12.2269295 (2017).

[12] Bouffard, P., "On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation and Experiments," Electrical Engineering and Computer Sciences, University of California at Berkeley (2012).

[13] Ganga, G. and Dharmana, M. M., "MPC Controller for Trajectory Tracking Control of Quadcopter," 2017 International Conference on Circuits Power and Computing Technologies (2017).
[14] Matuszewski, J., "Radar signal identification using a neural network and pattern recognition methods," 14th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, TCSET 2018 - Proceedings, pp. 79-83, DOI: 10.1109/TCSET.2018.8336160 (2018).
[15] Falanga, D., Foehn, P., Lu, P. and Scaramuzza, D., "PAMPC: Perception-Aware Model Predictive Control for Quadrotors," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018).
[16] Kamel, M., Burri, M. and Siegwart, R., "Linear vs Nonlinear MPC for Trajecotry Tracking Applied to Rotary Wing Micro Aerial Vehicle," IFAC (International Federation of Automatic Control) Hosting by Elsevier (2017).
[17] Kennedy J. and Eberhart, R., "Particle Swarm Optimization," Proceedings of IEEE International Conference on Neural Networks. IV, pp. 1942–1948 (1995).
[18] Konatowski, S. and Pawłowski, P., "PSO algorithm for UAV autonomous path planning with threat and energy cost optimization," Proc. SPIE 11055, XII Conference on Reconnaissance and Electronic Warfare Systems, 110550T; DOI: 10.117/12.2524886 (2019).
[19] Poli, R., "Analysis of the Publications on the Applications of Particle Swarm Optimisation," Hindawi Publishing Corporation, Journal of Artificial Evolution and Applications (2008).
[20] Sabatino, F., "Quadrotor control: modeling, nonlinear control design, and simulation," Master's Degree Project, KTH Electrical Engineering, Sweden (2015).
[21] Kraszewski, T. and Czopik, G., "Nonlinear Kalman filtering in the presence of additive noise," Proc. SPIE 10418, XI Conference on Reconnaissance and Electronic Warfare Systems, 104180N; DOI: 10.1117/12.2269355 (2017).
[22] Elkholy, M. H., "Dynamic Modeling and Control of a Quadrotor using Linear and Nonlinear Approaches," Thesis, American University in Cairo (2014).
[23] Konatowski, S. and Pawłowski, P., "Application of the ACO algorithm for UAV path planning," Przeglad Elektrotechniczny, Vol. 95, Issue 7, pp. 115-118, DOI:10.15199/48.2019.07.24 (2019).