Aerospace Science and Technology 127 (2022) 107725

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

A cooperative particle swarm optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme

Aristotelis Kapnopoulos, Alex Alexandridis [*]

Department of Electrical and Electronic Engineering, University of West Attica, Ancient Olive Grove Campus, Thivon 250 & P. Ralli, 12244, Aigaleo, Greece

A R T I C L E   I N F O

Article history:
Received 5 April 2021
Received in revised form 29 October 2021
Accepted 20 June 2022
Available online 27 June 2022
Communicated by Christian Circi

Keywords:
Cooperative methods
Model predictive control
Particle swarm optimization
Path tracking
Tuning
Quadcopter

A B S T R A C T

This paper presents a novel method for tuning a quadrotor trajectory-tracking model predictive control (MPC) framework, based on cooperative particle swarm optimization (PSO). The overall control strategy is decomposed into two different schemes; the first scheme, which is responsible for position control, is based on MPC, whereas the second one, which regulates the attitude of the quadrotor, makes use of three standard PID controllers. To optimize the trajectory tracking ability of the quadrotor, a cooperative PSO framework is introduced for tuning the large number of parameters of diverse nature, which are employed by the different controllers. The method makes use of two distinct swarms, with the first one containing the position controller parameters and the second one the attitude controller parameters. By exchanging information, the two swarms work together in a cooperative way in order to explore more efficiently the search space and discover tuning parameters that improve the trajectory tracking performance. Simulation results on a detailed quadcopter model over trajectories with different geometric characteristics, and comparisons to other tuning approaches verified by statistical testing, illustrate the efficiency of the proposed scheme in optimizing the control parameters. The scheme's robustness is also verified by testing the performance of the resulting controller on trajectories different than the ones used for tuning it.

1. Introduction

In the last decades great interest has been raised around vertical take-off and landing unmanned aerial vehicles (UAVs). One of the most important UAVs belonging to the family of rotary-wing aircrafts [1,2] is the quadrotor, or quadcopter. A quadrotor helicopter is a vehicle equipped with four propellers, which makes it possible to reduce the size of each rotor, while maintaining, or even increasing the total load capacity, compared to a helicopter with one main rotor. The advantages of the quadrotor are related to its high maneuverability, its precise movements in limited space, its stationary flight (hovering) and its ability for vertical take-off and landing. The ample set of abilities that the quadrotor possesses has led to a growing implementation in several industries such as surveillance, rescue, photography, forest patrolling, and farm quality inspection [3–6]. The escalating use of quadrotors in practical tasks, has steered researchers towards an in-depth study regarding their design, modeling and control [7,8].

The control of a quadrotor is not an easy task as it is a naturally unstable, underactuated, multiple-input-multiple-output (MIMO) nonlinear system, exhibiting strongly coupled dynamic terms. Early attempts to quadrotor flight control utilize techniques such as the proportional-integral-derivative (PID) controller, the linear quadratic regulator (LQR) and the H-infinity controller (H_∞). Due to their easy implementation and parameter adjusting, PID-based schemes were used by many researchers to produce autonomous flying quadrotors [9]. The LQR controller forms another standard control scheme for quadrotors, which has been applied for trajectory planning [10] and stability control [11]. In [12] a comparison between the performance of PID and LQR controllers is conducted, including experimental results. As far as the application of H_∞ for quadrotor control is concerned, Wang et al. [13] designed a tracking controller with the use of variation-based linearization to track the desired attitude trajectory. Raffo et al. [14,15] proposed a method for solving the path tracking problem by using an H_∞ controller that achieves path following in the presence of external disturbances and model uncertainties. Simulations from these two papers show the robustness of H_∞ controllers. While the aforementioned strategies display advantages in the domains of simplicity

[*] Corresponding author.
E-mail address: alexx@uniwa.gr (A. Alexandridis).

https://doi.org/10.1016/j.ast.2022.107725
1270-9638/© 2022 Elsevier Masson SAS. All rights reserved.

and implementation, they are too dependent to the linearized model of the system. Due to the fact that the quadrotor has an under-actuated nonlinear nature, linearization results to inaccurate models that only approximate its behavior in regions near the equilibrium points. As a result, nonlinear control methods were used to expand the stability and controllability capabilities of the quadrotor.

Feedback linearization comprises a common way in controlling nonlinear systems. The method involves a transformation that maps the nonlinear system to a linear one with a change in variables. Das et al. [16] designed a two-loop approach nonlinear controller with an outer PD loop and an inner feedback linearizing controller loop that deals with the coupling dynamics problem of the quadrotor. In [17] Voos presented a control system that combines control strategies including feedback linearization to cope with the nonlinear dynamics of the system showing satisfying attitude performance. In [18] a nonlinear controller responsible for position tracking and attitude stabilization is implemented successfully to a real quadrotor using a backstepping-feedback linearization scheme that takes into account physical parameter problems and outdoor experiments. While feedback linearization control overcomes the capabilities of the aforementioned linear controllers, it shares some limitations with them, as precise modeling is required, while it can't handle efficiently external disturbances.

Two methods comprising useful tools when dealing with quadcopter control and nonlinear systems in general, are the backstepping and sliding mode control schemes. The main idea of the backstepping algorithm design is to decompose the whole control system into several steps. Then, in each decomposed step, a virtual control is selected to make the former system stable until all the subsystems are stabilized. In [19] the backstepping approach is used for the design of a fully autonomous quadrotor with the capabilities of landing, obstacle avoidance and way-point following. In [20], a robust backstepping tracking controller is designed for a quadrotor subjected to disturbances and time-varying measurement delay, simultaneously. Although the backstepping controller can handle efficiently nonlinear systems and external uncertainties, it is sometimes associated with lack of robustness. Theoretically, the sliding mode control (SMC) framework is better equipped in this respect; in this case, the controlled system is forced from any point in space to move along a sliding surface. First applications of the method were studied by Sira-Ramirez [21] for the attitude control of a small helicopter. In [22] a model free controller is combined along with an SMC controller to solve the position and attitude trajectory problem for a quadrotor. Another interesting publication [23] showcases the advantages from the utilization of an adaptive SMC controller for attitude and position control, while adjusting its parameters using a neural network. In [24] a trajectory tracking designing procedure was combined with a cuckoo search algorithm for reducing the power consumption of a quadrotor that uses a terminal SMC. In [25] Tang et al. proposed a fault-tolerant terminal SMC for the robust trajectory tracking control of a quadrotor under external disturbances, parametric uncertainties and actuator faults. SMC is often used in conjunction with backstepping, e.g. Chen et al. achieved position trajectory tracking for a quadrotor along with a fault estimation observer by applying cooperation between a backstepping controller that deals with position control, and a sliding mode controller that deals with the attitude subsystem [26]. While the SMC algorithm can effectively handle the nonlinearity present in quadrotors, it is very sensitive to chattering phenomena which could be caused by unmodeled dynamics.

Model Predictive Control (MPC) [27] constitutes a different control method that can cope with some of the aforementioned disadvantages; during the last years the mathematical framework behind MPC has been well evolved [28], together with practical aspects regarding its implementation [29,30]. MPC relies on the simple idea of using an explicit dynamic model of the system to predict the effect of future actions on the output. The control actions are determined through an optimization procedure, with the objective of minimizing the predicted error. A plethora of works use the MPC strategy in quadrotors. In [31], an MPC control law is responsible for position control while a feed-forward controller guarantees the quadrotor's stabilization. In [32], a robust MPC controller is implemented in real flight scenarios. In [33] a state space error predictive controller is responsible for reference tracking, while an H_∞ controller stabilizes the attitude of the quadrotor. In [34], the control system is decoupled into two subsystems, where the path following problem is solved by a state-space predictive controller, while the attitude stabilization is executed by a non-model based active disturbance rejection controller (ADCR) with the use of linear extended state observer (LESO) strategy.

Regardless of the choice of control method, a task of paramount importance from a practical point of view is associated with parameter tuning, as a poor selection in controller parameters can hinder the quadrotor flight. Unfortunately, although MPC has been highly successful for quadrotor control, it is often associated with a cumbersome parameter tuning procedure. This can be attributed to the high number of operational parameters that need to be selected, which increases further when MPC is coupled with different techniques that carry their own parameters, as is often the case in quadrotor control. Under such circumstances, performing the tuning procedure by trial and error is not effective and a mathematical optimization problem needs to be formulated and solved. Unfortunately, when viewing MPC tuning as an optimization procedure, the high number of design variables is not the only obstacle to be encountered; the mixed integer and continuous nature of the design variables, combined with a complicated, nonlinear and multimodal objective function make the use of conventional optimization methods inadequate.

Metaheuristic search methods constitute a class of optimization algorithms that are better equipped to overcome such difficulties and provide better quality solutions to the tuning problem. By relying on stochastic search, metaheuristic search methods reduce the risk of getting trapped in local minima, while the generation and evolution of multiple solutions bestows increased exploration capabilities. PSO [35] is a simple yet effective metaheuristic optimization method which belongs to the family of swarm intelligence; it relies on simulating the social structure of flocking birds flying in formation, and has been used successfully for tuning diverse control schemes [36] and aerodynamic design optimization [37].

Not surprisingly, PSO has also been used extensively for solving the quadrotor control tuning optimization problem. In [38], a PSO tuning process was used in order to design four decentralized PID controllers achieving stabilization for the quadrotor's altitude and attitude. In [39], a PSO algorithm is used to tune radial basis function (RBF) neural networks responsible for adjusting a the parameters of a PID controller coupled to the quadrotor. Yacef F. et al. in [40], implement a PSO-based tuning strategy to an integral back-stepping controller in order to achieve height and angle stabilization. PSO has also been used to optimize the control parameters in quadrotor path tracking problems. In [41] a path planning solution is presented for the inspection of a photovoltaic farm based on a PSO - Bezier curve algorithm. In [42], the PSO algorithm optimizes the parameters of an ADCR strategy for the sake of following a desired path. In the particular work the PSO-ADCR methodology is compared with the simple ADCR showing the improvement in settling time, overshoot and desired tracking error. In [43], a heterogeneous comprehensive learning PSO is utilized to optimize the parameters of a quadrotor saturation-based controller in order to perform three-dimensional trajectories in space. Mahmoodabadi et al. in [44] used a multi-objective high exploration PSO algorithm to tune the membership functions of a quadrotor fuzzy controller based on the LQR methodology.

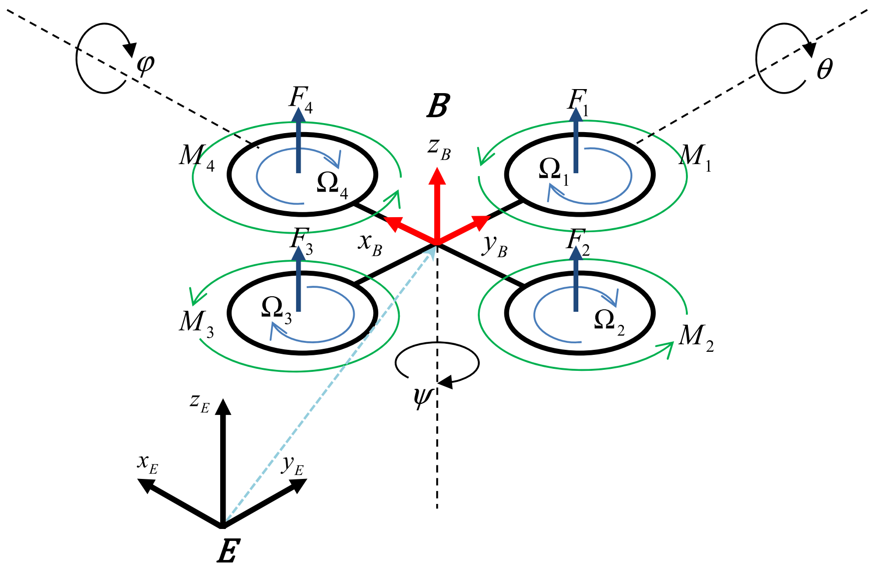

Fig. 1. Schematic depiction of the earth and body fixed frames for a quadrotor vehicle.

Though simple variants of PSO have been used successfully for tuning quadrotor control schemes, the modularity of the respective optimization problem could be better addressed by cooperative PSO techniques. The success of cooperative techniques dealing with complex optimization problems has been highlighted in many works [45–47]. However, to the authors' best knowledge there are no publications in the subject of tuning quadrotor control schemes with cooperative techniques. In this work, a framework based on MPC and PID control is designed for solving the quadrotor path tracking problem, along with a cooperative optimization algorithm that is tailored for tuning the parameters of the two different controllers. To be more specific, the employed control framework comprises two subsystems, namely an MPC controller for dealing with the path following problem and a PID scheme for dealing with attitude stabilization. The two different control subsystems require a large number of tuning parameters, which are optimized effectively by a cooperative PSO scheme, employing multiple swarms in order to optimize the different components of the solution vector. In this case, two different swarms are used for the MPC and PID tuning parameters, respectively; though each swarm controls a different set of parameters, they ultimately work together towards bestowing improved path tracking abilities to the integrated control framework. The proposed approach is evaluated through a series of experiments employing a number of different trajectories, while also testing the method's robustness by applying it in trajectories different than the ones used for tuning; performance is compared to different metaheuristic search methods through statistical testing.

The rest of this paper is organized as follows: In Section 2 the quadrotor's dynamic model is described. Section 3 presents the control framework for quadcopter trajectory tracking based on state space predictive control. The proposed cooperative PSO tuning framework is introduced in Section 4. In section 5, a series of experiments on various spatial trajectories is applied to evaluate the proposed strategy and comparisons are made with other methods. Finally, conclusions and plans for future research are drawn in Section 6.

2. Mathematical model

The quadrotor vehicle is a nonlinear system with under-actuation and strong coupling. Fig. 1 depicts an X-configuration quadrotor, where the four rotors are divided into two pairs of (1,3) and (2,4) which rotate in opposite directions in order to compensate for the interaction of the reaction torques generated. The right cooperation between the rotors speed ensures the quadrotor's basic movements in aerial space as follows: vertical motion is achieved by increasing or decreasing the speed of all rotors; the differential speeds of rotors (1,3) and (2,4) contributes into the roll and pitch motions coupled with forward motions respectively, while yaw motion is performed through the difference of counter-torques generated by each propeller.

This work utilizes a detailed dynamic model of the quadrotor which has been shown in many studies to provide realistic results when compared to real-world quadcopters [19,48]. The model is based on the assumptions that the structure is rigid and symmetrical, the center of gravity and the body fixed frame coincide, and the propellers are rigid. The translational and rotational dynamics occur using the second Newton's law of linear and angular conservation. To describe the kinematics of the quadrotor, two coordinate systems are defined, namely the earth-fixed frame $E = \{x_E, y_E, z_E\}$ which is considered fixed with respect to the earth, and the fuselage, or body-fixed frame $B = \{x_B, y_B, z_B\}$, linked to the rigid body of the quadrotor. The two frames are depicted in Fig. 1. In this case, the x_B axis is in the quadrotor's normal flight direction, y_B is orthogonal to x_B and positive to starboard in the horizontal plane, whereas z_B is oriented in the ascendant sense and orthogonal to the plane created from the vectors x_B and y_B. The linear position of the quadrotor in frame E is denoted with the vector $\xi = [x, y, z]^T \in \mathbb{R}^3$, while the orientation of the quadrotor is described in frame B by Euler's angles roll φ, pitch θ, and yaw ψ, thus forming the vector $\eta = [\varphi, \theta, \psi]^T \in \mathbb{R}^3$. Furthermore, let $\Omega = [p, q, r]^T \in \mathbb{R}^3$ and $V = [u, v, w]^T \in \mathbb{R}^3$ denote the angular and linear velocity of the quadrotor vehicle in frame B, respectively. Thus, the translational and rotational kinematic model is given by:

$$\begin{cases} \dot{\xi} = R_t V \\ \dot{\eta} = R_r \Omega \end{cases} \tag{1}$$

Where R_t and R_r are the transformation matrices between the two frames which are given by [25,49]:

$$R_t(\eta) = \begin{bmatrix} C_\theta C_\psi & S_\varphi S_\theta C_\psi - C_\varphi S_\psi & C_\varphi S_\theta C_\psi + S_\varphi S_\psi \\ C_\theta S_\psi & S_\varphi S_\theta S_\psi + C_\varphi C_\psi & C_\varphi S_\theta S_\psi - S_\varphi C_\psi \\ -S_\theta & S_\varphi C_\theta & C_\varphi C_\theta \end{bmatrix} \qquad R_r(\eta) = \begin{bmatrix} 1 & 0 & -S_\varphi \\ 0 & C_\varphi & S_\varphi C_\theta \\ 0 & -S_\varphi & C_\varphi C_\theta \end{bmatrix} \tag{2}$$

and $S_{(\bullet)}$, $C_{(\bullet)}$ are the abbreviations for $\sin(\bullet)$ and $\cos(\bullet)$, respectively.

Assuming null disturbances, the quadrotor's dynamics equation can be expressed in frame B by:

$$\begin{cases} m\dot{V} = -\Omega \times (mV) - F_{aero} - F_{grav} + T \\ I\dot{\Omega} = -\Omega \times (m\Omega) - M_{aero} - M_{gyro} + \tau \end{cases} \tag{3}$$

where $m \in \mathbb{R}$ and $I = diag(I_x, I_y, I_z) \in \mathbb{R}^{3\times 3}$ denote the mass and the inertia matrix of the quadrotor, $F_{grav} = mR_t^T G$ is the force due to gravity (where $G = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ is the gravity vector), $M_{gyro} = \begin{bmatrix} -J_r \dot{p}\Omega_r & J_r \dot{q}\Omega_r & 0 \end{bmatrix}$ is the gyroscopic moment formed by the rotations of the rotors around their axis (where J_r is the rotor inertia), Ω_r are the overall residual rotor speeds, and $F_{aero} = K_t V$ and $M_{aero} = K_r \Omega$ are the aerodynamical drag forces and moments, with $K_t = diag(K_x, K_y, K_z)$ and $K_r = diag(K_\varphi, K_\theta, K_\psi)$ [25,50] denoting the aero dynamical drag coefficient matrices. The rotation of the quadrotor's propellers generates the forces responsible for its movement in space. Each rotor produces a lift force F_i and a reactive moment M_i. The force $T \in \mathbb{R}^3$ and torque $\tau \in \mathbb{R}^3$ generated by each rotor can be expressed as $F_i = b\Omega_i^2$, $M_i = d\Omega_i^2$ [19,51], where b is the thrust coefficient, d is the drag coefficient and Ω_i is the angular velocity of rotor i, $i \in \{1, 2, 3, 4\}$.

Proper arrangement of forces and moments leads to a total thrust $T \in \mathbb{R}^3$ and a control torque $\tau \in \mathbb{R}^3$ given by:

$$T = \begin{bmatrix} 0 \\ 0 \\ b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \end{bmatrix} \tag{4}$$

$$\tau = \begin{bmatrix} lb\left(-\Omega_2^2 + \Omega_4^2\right) \\ lb\left(\Omega_1^2 + \Omega_3^2\right) \\ d\left(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2\right) \end{bmatrix} \tag{5}$$

Using the Newton Euler formalism [52], equation (3) can be expressed in the inertia frame E as:

$$\begin{aligned} m\ddot{x} &= (\cos\varphi \sin\theta \cos\psi + \sin\varphi \sin\psi) U_1 - K_x \dot{x} \\ m\ddot{y} &= (\cos\varphi \cos\theta \sin\psi - \sin\varphi \cos\psi) U_1 - K_y \dot{y} \\ m\ddot{z} &= \cos\varphi \cos\theta U_1 - mg - K_z \dot{z} \\ I_x \ddot{\varphi} &= \dot{\theta}\dot{\psi}\left(I_y - I_z\right) + lU_2 + J_r \dot{\theta}\Omega_r - K_\varphi \dot{\varphi} \\ I_y \ddot{\theta} &= \dot{\varphi}\dot{\psi}\left(I_z - I_x\right) + lU_3 - J_r \dot{\varphi}\Omega_r - K_\theta \dot{\theta} \\ I_z \ddot{\psi} &= \dot{\varphi}\dot{\theta}\left(I_x - I_y\right) + U_4 - K_\psi \dot{\psi} \end{aligned} \tag{6}$$

In order to control the flight mechanism of the quadrotor, a control input U is defined which consists of four control actions U_i, with $i \in \{1, 2, 3, 4\}$. U_1 is the control action related to the total thrust and is responsible for the change of altitude, while U_2, U_3 and U_4 are control actions related to the moments generated around the body axes x_B, y_B, z_B, respectively; more specifically, U_2, U_3 and U_4 control the desired rotation for the roll angle φ, pitch angle θ and yaw angle ψ, respectively. The control input vector U is given by:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \\ lb\left(-\Omega_2^2 + \Omega_4^2\right) \\ lb\left(\Omega_1^2 + \Omega_3^2\right) \\ d\left(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2\right) \end{bmatrix} \tag{7}$$

3. MPC controller design

3.1. Error state-space position control

In this section a control framework that deals with the path tracking problem of an autonomous discrete-time nonlinear quadrotor system is given. To be more specific, a linear error-based MPC strategy is constructed in order to control the quadrotor position. The error model derived is divided into two different position controllers. The first one controls the altitude of the quadrotor via the input U_1, while the second one is responsible for the control in the x-y plane through the inputs u_x, u_y, which are derived as follows. The position system (6) can be rewritten in a state space form as $\dot{\bar{\xi}} = f\left(\bar{\xi}(t), u_\xi(t)\right)$, where $\bar{\xi}(t) = \begin{bmatrix} x(t) & u(t) & y(t) & v(t) & z(t) & w(t) \end{bmatrix}$ stands for the augmented system state space vector, and $u(t)$, $v(t)$, $w(t)$ are the linear velocities of the quadrotor's mass center:

$$\dot{\bar{\xi}} = f\left(\bar{\xi}(t), u_{\xi}(t)\right) = \begin{bmatrix} u(t) \\ u_x(t)\dfrac{U_1(t)}{m} \\ v(t) \\ u_y(t)\dfrac{U_1(t)}{m} \\ w(t) \\ -g + \cos\theta(t) cos\varphi(t)\dfrac{U_1(t)}{m} \end{bmatrix} \tag{8}$$

where

$$\begin{aligned} u_x(t) &= \cos\psi(t)\sin\theta(t) cos\varphi(t) + sin\psi(t)\sin\varphi(t) \\ u_y(t) &= \sin\psi(t)\sin\theta(t) cos\varphi(t) - \cos\psi(t)\sin\varphi(t) \end{aligned} \tag{9}$$

The reference trajectory is provided off-line and a virtual reference vehicle with the same dynamics as the quadrotor is proposed along with the real one on the same track. Assuming that there is no external disturbance to the virtual reference quadrotor, the dynamics can be written in the form:

$$\dot{\bar{\xi}}_{ref} = f\left(\bar{\xi}_{ref}(t), u_{\xi ref}(t)\right) \tag{10}$$

where $\bar{\xi}_r(t) = \begin{bmatrix} x_r(t) & u_r(t) & y_r(t) & v_r(t) & z_r(t) & w_r(t) \end{bmatrix}$ and $u_{\xi ref}(t) = \begin{bmatrix} u_{xref} & u_{yref} & U_{1ref} \end{bmatrix}$ are the reference state vector and the reference input control, respectively.

Therefore, by subtracting the virtual system (10) from the position system (8) and using the forward Euler method for altitude z, the error model can be obtained as a discrete time linear equation:

$$\tilde{x}_{\xi}(k+1) = A \cdot \tilde{x}_{\xi}(k) + B(k) \cdot \tilde{u}_{\xi}(k) \tag{11}$$

where $\tilde{x}_{\xi}(k) = \bar{\xi}(k) - \bar{\xi}_r(k)$ is the position error vector and $\tilde{u}_{\xi}(k) = u(k) - u_{\xi ref}(k)$ is the control input error vector. The error model (11) can be divided into two discrete state-space subsystems [15], namely the altitude and attitude one.

For the altitude subsystem the height position error model takes the form:

$$\tilde{x}_z(k+1) = A_z \cdot \tilde{x}_z(k) + B_z(k) \cdot \tilde{u}_z(k) \tag{12}$$

where matrices A_z and $B_z(k)$ are given by:

$$A_z = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \qquad B_z(k) = \begin{bmatrix} 0 \\ \frac{\Delta t}{m}\cos\theta(k)\cos\varphi(k) \end{bmatrix} \tag{13}$$

and Δt is the sampling time.

Successful path tracking for a desired trajectory is possible by finding the suitable control inputs. In order to achieve this, a control law needs to be designed, minimizing the tracking error through a cost function defined by:

$$\min_{\bar{u}_z} J_z = \bar{x}_z^T Q_z \bar{x}_z + \bar{u}_z^T R_z \bar{u}_z + \bar{x}_z\left(k+N_p/k\right)^T G_{zf} \bar{x}_z\left(k+N_p/k\right) \tag{14}$$

where $\bar{x}_z = \hat{\tilde{x}}_z - \hat{\tilde{x}}_{zr}$, $\bar{u}_z = \hat{\tilde{u}}_z - \hat{\tilde{u}}_{zr}$ and $\bar{x}_z\left(k+N_p/k\right) = \tilde{x}_z\left(k+N_p/k\right) - \tilde{x}_{zr}\left(k+N_p/k\right)$. Matrices Q_z, R_z and G_{zf} are positive definite and penalize divergences from the reference state, input and terminal state, respectively. The predictions of the plant output $\hat{\tilde{x}}_z(k+j/k)$ are computed using a linear time-varying state-space model of the vehicle using (12) and (13), giving:

$$\hat{\tilde{x}}_z = P_z(k/k) \cdot x_z(k/k) + H_z(k/k) \cdot \hat{\tilde{u}}_z \tag{15}$$

where $\tilde{u}_z(k/k) = U_1(k) - u_{zref}(k)$ and $\hat{\tilde{x}}_z$ is the height state space vector. The reference height and input state error vectors are:

$$\hat{\tilde{x}}_{zr} \triangleq \begin{bmatrix} \tilde{x}_{zr}(k+1/k) - \tilde{x}_{zr}(k/k) \\ \vdots \\ \tilde{x}_{zr}\left(k+N_p/k\right) - \tilde{x}_{zr}(k/k) \end{bmatrix}, \qquad \hat{\tilde{u}}_{zr} \triangleq \begin{bmatrix} \tilde{u}_{zr}(k/k) - \tilde{u}_{zr}(k-1/k) \\ \vdots \\ \tilde{u}_{zr}(k+N_C-1/k) - \tilde{u}_{zr}(k-1/k) \end{bmatrix} \tag{16}$$

where N_p is the prediction horizon, showing how far the controller predicts to the future, and N_C is the control horizon, which constitutes the number of consecutive moves to be manipulated by the controller for minimizing the cost function J_z. The control moves $\hat{\tilde{u}}_z$ at time k are evaluated by solving the minimization problem (14); however, only the first control move is implemented on the system, while the remaining ones are rejected. After applying the control move $\hat{\tilde{u}}_z(k/k)$, the system moves to a new position at time $k+1$ and the minimization problem is solved again for the new current state, yielding a new control action.

Therefore, the control input applied to the quadrotor is:

$$U_1(k) = \tilde{u}_z(k/k) + u_{zref}(k) \tag{17}$$

In a similar way the x-y motion position error model takes the following form:

$$\hat{\tilde{x}}_{xy}(k+1) = A_{xy} \cdot x_{xy}(k) + B_{xy}(k) \cdot \hat{\tilde{u}}_{xy}(k) \tag{18}$$

where matrices A_{xy} and $B_{xy}(k)$ are given by:

$$A_{xy} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_{xy}(k) = \begin{bmatrix} 0 & 0 \\ \frac{\Delta t}{m} U_1(k) & 0 \\ 0 & 0 \\ 0 & \frac{\Delta t}{m} U_1(k) \end{bmatrix} \tag{19}$$

Following the same procedure as above, a cost function J_{xy} is set to minimize the tracking error in the x-y direction forming the quadratic problem:

$$\min_{\bar{u}_{xy}} J_{xy} = \bar{x}_{xy}^T Q_{xy} \bar{x}_{xy} + \bar{u}_{xy}^T R_{xy} \bar{u}_{xy} + \bar{x}_{xy}\left(k+N_p/k\right)^T G_{xyf} \bar{x}_{xy}\left(k+N_p/k\right) \tag{20}$$

where Q_{xy}, R_{xy} and G_{xyf} are positive definite matrices. The rest of the matrices are computed in similar manner to the height predictive control scheme.

By minimizing the cost function J_{xy}, the control input $\tilde{u}_{xy}$ is obtained, where $\tilde{u}_{xy}(k/k) = \left[\, \tilde{u}_x(k/k) \quad \tilde{u}_y(k/k) \,\right]^T$ and:

$$u_{xy}(k) = \tilde{u}_{xy}(k/k) + u_{xyr}(k) \tag{21}$$

The error reference state-space vector $\hat{\tilde{x}}_{xyr}$ and the reference input $\hat{\tilde{u}}_{xyr}$ are obtained in the same way as in the height control mechanism.

Writing equation (9) for time instance k, we obtain:

$$\begin{aligned} u_x(k) &= cos\varphi_{ref}(k)\sin\theta_{ref}(k)\cos\psi(k) + \sin\varphi_{ref}(k)\, sin\psi(k) \\ u_y(k) &= cos\varphi_{ref}(k)\sin\theta_{ref}(k)\sin\psi(k) - \sin\varphi_{ref}(k)\cos\psi(k) \end{aligned} \tag{22}$$

By setting $\psi_{ref} = 0$ and using (21), (22), the reference values for the roll angle φ_{ref} and pitch angle θ_{ref} are computed; these are given as set points to the attitude control loop which will be described in the next subsection.

3.2. PID attitude control

The position controller only generates U_1 and the desired values for φ_{ref}, θ_{ref} by taking as input the trajectory's current reference position in space. However, in order for the quadrotor to track the desired trajectory, the control inputs U_2, U_3, U_4 should be calculated along with the thrust U_1. This task is accomplished by the inner attitude controller which regulates the orientation angles φ, θ and ψ using as reference inputs the desired values provided by the outer position controller. In this work, the development of the inner attitude loop is based on standard PID controllers.

The rotational dynamic model in (3) is used in order to form the attitude controller, which can be expressed as:

$$I\dot{\Omega} = \tau \tag{23}$$

where (23) can be rewritten using the notation of equations (6) and (7) as:

$$\begin{aligned} I_x \cdot \ddot{\varphi} &= U_2 \\ I_y \cdot \ddot{\theta} &= U_3 \\ I_z \cdot \ddot{\psi} &= U_4 \end{aligned} \tag{24}$$

The objective of the attitude controller is to ensure that the Euler's angles φ, θ, ψ track the desired trajectory values $\varphi_{ref}, \theta_{ref}, \psi_{ref}$ asymptotically, by applying the suitable control signals U_2, U_3, U_4 to the quadrotor. Three independent PID controllers are used in this work to control the angular accelerations of the quadrotor, one for each Euler angle.

By applying forward Euler method on continuous PID controllers, each controller can be expressed in velocity form as follows [53]:

$$\begin{aligned} u_j(k) = {} & \left(K_{Pj} + K_{Dj} \cdot N_j\right) e_j(k) + \left(N_j \cdot \Delta t \cdot K_{Pj} + K_{Ij} \cdot \Delta t - 2 \cdot K_{Dj} \cdot N - 2 \cdot K_{Pj}\right) e_j(k-1) \\ & + \left(K_{Pj} - K_{Pj} \cdot N_j \cdot \Delta t - K_{Ij} \cdot \Delta t + K_{Ij} \cdot N_j \cdot \Delta t^2 + K_{Dj} \cdot N_j\right) e_j(k-2) \\ & - \left(N_j \cdot \Delta t - 2\right) u_j(k-1) - \left(1 - N_j \cdot \Delta t\right) u_j(k-2) \end{aligned} \tag{25}$$

The index j with $j = 1, 2, 3$ describes the Euler's angles φ, θ, ψ, while r_j describes the reference angles $\varphi_{ref}, \theta_{ref}, \psi_{ref}$ of the *jth* channel. The control signal is denoted by $u_j(k)$ and the error signal by $e_j(k)$ at each discrete time instant k. The error signal is the difference between the actual value $\eta_j(k)$ and reference r_j. $K_{Pj}, K_{Ij} K_{Dj}$ are the controller proportional, integral and derivative gains and N_j is the low pass filter coefficient of the derivative term of the PID controller.

Based on (23) and (24), the control inputs U_2, U_3, U_4 can be estimated through the following equations:

$$\begin{aligned} U_2 &= I_x \cdot u_\varphi \\ U_3 &= I_y \cdot u_\theta \\ U_4 &= I_z \cdot u_\psi \end{aligned} \tag{26}$$

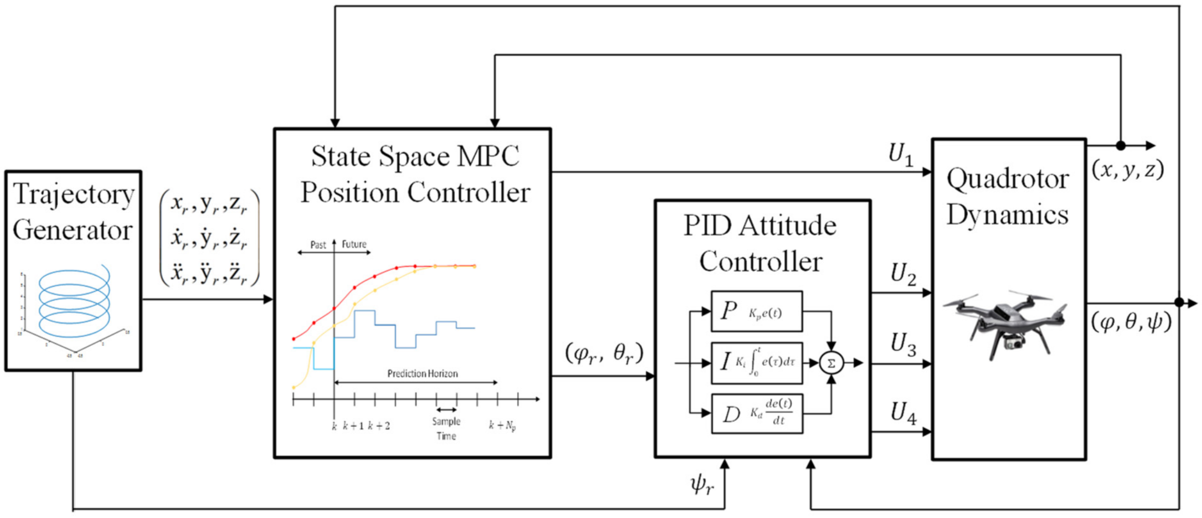

Fig. 2. Overall quadrotor control structure.

An important addition to the proper design of the control strategy takes into consideration the saturation bounds for each one of the control signals. With the primary objective of limiting and controlling the quadrotor's thrust force, many saturation-constraint based methodologies have been developed [54]. The improvement of the quadcopter's control design brought on by introducing saturated limits for U_1, U_2, U_3, U_4 control signals can be explained as follows. First, both the total trust T and the control torque τ are bounded in range due to physical constraints imposed by the electrical motors. Second, the control inputs must be in a range of values that allows the creation of a smooth and continuous track. Finally, incorporating boundaries in the input signals helps to avoid flight scenarios that can lead to loss of support for the quadcopter. Based on the above, eight input saturation limits are imposed to the four control inputs U_1, U_2, U_3, U_4, i.e. four upper $U_{i,\text{upper}}$ and four lower bound $U_{i,\text{lower}}$ values, where $i = 1, 2, 3, 4$. The overall quadrotor position and attitude control strategy is depicted in Fig. 2.

As shown above, the development of an efficient framework that delivers the proper control inputs to the quadrotor, requires cooperation between the position and attitude controllers. Due to the fact that these two subsystems comprise a number of 28 tuning parameters, a trial-and-error attempt to design an operational quadrotor is a cumbersome task. Moreover, the difference in the design of each control subsystem, further complicates the overall controller tuning task. While many techniques have been implemented [55–57] for the sake of standard MPC tuning, the design of an automated tuning procedure that considers the different structure of each control subsystem is yet unanswered in the domain of quadrotor control. In order to surpass this obstacle, a new cooperative optimization process is designed, taking into account the unique nature of each controller, and at the same time their combined effects on the quadrotor behavior.

4. Control scheme tuning using a cooperative PSO-based optimizer

4.1. Standard particle swarm optimization

PSO is a stochastic search optimization technique attributed to Kennedy and Eberhart [35]. The concept of PSO is based on the simulation of bird flocking social behavior. To be more specific, the algorithm encodes a population of possible solutions known as particles, which 'fly' through the search space by taking into account the personal best location they have visited, as well as the global best solution achieved.

During each iteration, the particles move towards the direction of their own personal best solution found so far, as well as in the direction of the global best position found by the entire swarm. Each particle of the swarm is characterized by its position $x_i(t)$, its velocity $v_i(t)$ and its best previous position in the search space $y_i(t)$, where $\hat{y}(t)$ denotes the best known position of the entire swarm.

For a fitness function f to be minimized and a swarm consisting of s particles of dimensionality equal to n, the following equations describe how the particles update their current position and velocity:

$$v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 \cdot r_{1,i}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 \cdot r_{2,i}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right] \quad (27)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (28)$$

where $v_{i,j}$, $i = 1, 2, \ldots s$, $j = 1, 2, \ldots n$ is the velocity for the jth dimension of the ith particle, c_1, c_2 denote the acceleration coefficients, w the inertia coefficient and $r_{1,i}, r_{2,i}$ are two random values sampled from a uniform distribution in the range [0,1].

Mathematical expression (27) consists of three parts. The first is the momentum or inertia part, that reflects the particle's tendency to maintain its current motion. The second is the cognitive part which represents the ability of the particle to reflect on its behavior and follow the best personal position found in the past. The last part is the social one, which depicts the particles' tendency to follow the optimal position found by their neighbors.

The personal best position of each particle is updated using the following equation,

$$y_i(t+1) = \begin{cases} y_i(t), & if\ f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1), & if\ f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad (29)$$

and the global best position of the swarm is defined as:

$$\underbrace{Q_{z_1}\ Q_{z_2}\ R_z\ \ G_{z_1}\ G_{z_2}\ Q_{xy_1}\ Q_{xy_2}\ R_{xy}\ \ G_{xy_1}\ G_{xy_2}\ N_c\ N_p}_{\text{MPC control parameters}}\ \underbrace{K_{P_\theta}\ K_{I_\theta}\ K_{D_\theta}\ N_\theta\ K_{P_\psi}\ K_{I_\psi}\ K_{D_\psi}\ N_\psi\ K_{P_\varphi}\ K_{I_\varphi}\ K_{D_\varphi}\ N_\varphi\ U_{1_B}\ U_{2_B}\ U_{3_B}\ U_{4_B}}_{\text{PID control parameters}}$$

Fig. 3. Overall particle structure in the case of standard PSO.

$$\hat{y}_i(t+1) = \arg\min_{y_i} f(y_i(t+1)), \quad 1 \le i \le s \tag{30}$$

4.2. Cooperative PSO for quadrotor control scheme tuning

The standard PSO algorithm could be applied for tuning the proposed quadrotor control scheme, but in this case, the elements of each particle should contain all the controller parameters to be tuned, which would be optimized concurrently; Fig. 3 shows the corresponding particle structure in this case.

However, as discussed in section 3, there is a high number of parameters involved, pertaining to the two control subsystems. From an optimization perspective, the increased dimensionality of the search space makes it difficult to discover even a good suboptimal solution. On the other hand, a distinction can be made between two different parts in each particle, where each part controls a different mechanism of the integrated control scheme. To be more specific, the first part contains the parameters from the position control strategy of the MPC controller, while the second one the parameters from the attitude control through the implementation of PID controllers. The existence of two different groups of parameters within the particle, leads to the idea of using two cooperative swarms, each containing the parameters from the MPC and PID controllers, respectively.

Agent cooperation is an important tool for solving optimization problems. Potter and De Jong applied this idea to genetic algorithms [58] by dividing the solution vector into more than one chromosomes, with each belonging to different populations. A corresponding framework can be implemented to the PSO algorithm, in which different families of particles cooperate to form the solution vector. The first cooperative PSO framework implemented, named CPSO-S [59], splits the n-dimensional solution vector space into n1-D swarms, with each one of them undergoing through a distinct optimization procedure, also making use of the global best particles of the other swarms. In this work, a modification of the $\mathrm{CPSO}-\mathrm{S}_K$ algorithm [60] is proposed, where the solution vector is allowed to decompose into two parts for simultaneously optimizing the MPC and PID parameters. The two distinct swarms evolve independently while sharing information during the optimization procedure. The particles for the MPC swarm which contain the position control tuning parameters are denoted as P_1x_i, while the particles of the PID swarm containing the attitude control tuning parameters are denoted as P_2x_i. Equations (31) and (32) display the parameters controlled by each one of the swarms P_1 and P_2:

$$P_1x_i = \begin{bmatrix} Q_{z_1} & Q_{z_2} & R_z & G_{z_1} & G_{z_2} & Q_{xy_1} & Q_{xy_2} & R_{xy} & G_{xy_1} & G_{xy_2} & N_c & N_p \end{bmatrix} \tag{31}$$

$$P_2x_i = \begin{bmatrix} K_{P_\theta} & K_{I_\theta} & K_{D_\theta} & N_\theta & K_{P_\psi} & K_{I_\psi} & K_{D_\psi} & N_\psi & K_{P_\varphi} & K_{I_\varphi} & K_{D_\varphi} & N_\varphi & U_{1_B} & U_{2_B} & U_{3_B} & U_{4_B} \end{bmatrix} \tag{32}$$

For each swarm the position and velocity vector are updated according to the following equations:

$$P_kv_{i,j}(t+1) = w \cdot P_kv_{i,j}(t) + c_1 \cdot r_{1,i}(t)\left[P_ky_{i,j}(t) - P_kx_{i,j}(t)\right] + c_2 \cdot r_{2,i}(t)\left[P_k\hat{y}_j(t) - P_kx_{i,j}(t)\right] \tag{33}$$

$$P_kx_{i,j}(t+1) = P_kx_{i,j}(t) + P_kv_{i,j}(t+1) \tag{34}$$

where the ith particle in dimension j is represented as $P_kx_{i,j}(t)$ for the swarm k, the speed for particle i in dimension j is represented as $P_kv_{i,j}(t)$, the personal best position of the ith particle in dimension j is denoted as $P_ky_{i,j}(t)$, and $P_k\hat{y}_j(t)$ represents the best particle position among all particles for the jth dimension.

The velocity of each particle $P_kv_{i,j}(t)$ is controlled by a clamping constant P_kv_{max} that regulates the maximum velocity update to a defined range of $[-P_kv_{max}, P_kv_{max}]$. The addition of the clamping constant is used to control the exploration-exploitation trade-off, by affecting the particles' ability to explore a small, or big part of the search space.

As each swarm is connected to a specific part of the solution vector of the optimization procedure, the right cooperation between the agents is essential in order to calculate the fitness function. The implementation of this task is feasible by using a context vector. This vector provides a suitable context in which the individuals from each swarm can be evaluated. To form the context vector, the global best particle $\hat{y}$ from one swarm, is combined with each particle $x_i, i = 1, \dots s$ from the other swarm. Therefore, in order to calculate the fitness functions for swarm P_k, all the respective particles should be concatenated with the global best particle of the complementary swarm $P_{(2k)^{2-k}}\hat{y}$. This concatenation leads to the full set of parameters for both controllers, which is used to simulate the quadrotor flight. The fitness function is calculated as the sum of the Euclidean distances (SED) between the reference trajectory and the actual quadrotor trajectory during the simulation flight, based on the following equation:

$$f = \sum_{t_s=1}^{t_{end}} \sqrt{\sum_{i=1}^{3} \left(q_i(t_s) - p_i(t_s)\right)^2} \tag{35}$$

where $t_s = 1, \dots, t_{end}$ is the running time of the flight simulation, $q(t_s) = \left[x_{ref}, y_{ref}, z_{ref}\right]$ is the vector containing the Cartesian coordinates of the reference altitude and attitude of the quadcopter at time t_s, $p(t_s) = [x, y, z]$ is the vector containing the Cartesian coordinates of the actual altitude and attitude achieved by the quadcopter at time t_s, and t_{end} is the end time of the simulation.

The personal best position $P_ky_i(t)$ for each particle i of each swarm k depicts the best result found for this particle up to iteration t:

$$P_ky_i(t+1) = \begin{cases} P_ky_i(t), & \text{if } f\left(P_kx_i(t+1), P_{(2k)^{2-k}}\hat{y}(t)\right) \ge f\left(P_ky_i(t), P_{(2k)^{2-k}}\hat{y}(t-1)\right) \\ P_kx_i(t+1), & \text{if } f\left(P_kx_i(t+1), P_{(2k)^{2-k}}\hat{y}(t)\right) \le f\left(P_ky_i(t), P_{(2k)^{2-k}}\hat{y}(t-1)\right) \end{cases} \tag{36}$$

The global best position of each particle i for each swarm is updated using:

$$P_k\hat{y}(t) = \arg\min_{P_k y_i(t)} f\left(P_k y_i(t), P_{(2k)^{2-k}}\hat{y}(t-1)\right), i = 1, \ldots s \tag{37}$$

Fig. 4 depicts a schematic overview of the MPC and PID swarms working together towards optimizing the tuning parameters:

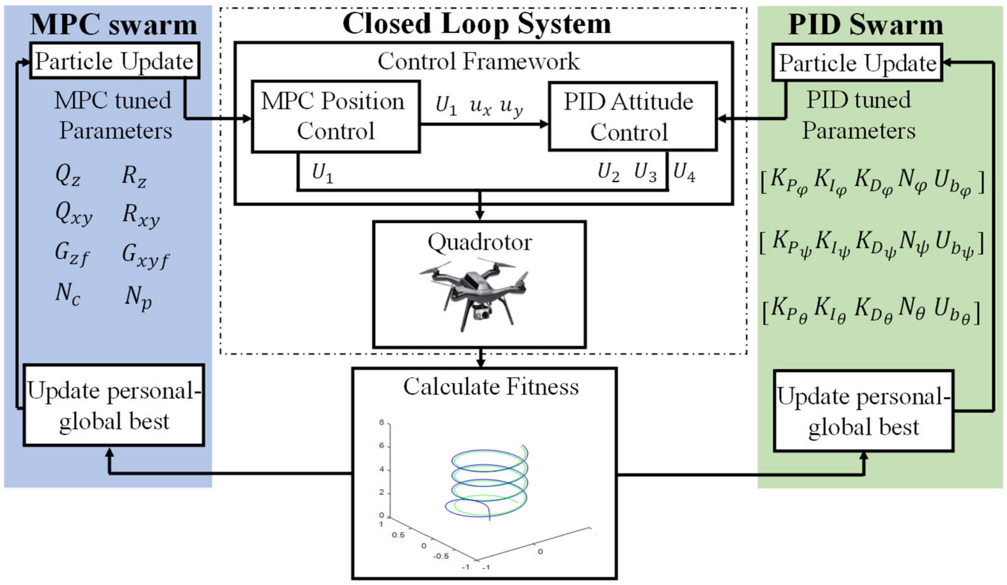

Fig. 4. Overview of the two cooperative swarms, working towards optimizing the tuning parameters.

The described algorithm has two important advantages with respect to the optimization procedure when used for tuning the controller parameters. The first one is associated with the fitness function evaluation after each separate group of controller parameters has been updated. This leads to a finer-grained assignment avoiding the classic two steps forward - one step back problem often encountered in standard PSO. The second advantage is related to the increased combinations of different individuals from different swarms, which boost the solution diversity.

Another important problem often encountered by the PSO algorithm is the trapping of particles in suboptimal solutions, a phenomenon known as stagnation. This happens because over time, the particles tend to cluster around the current global optimum solutions; this prevents the swarms from exploring the area thoroughly and further improving their global best positions. In order to avoid the stagnation problem, the swarm particles can be reset into new random positions when one of the swarms converges in a small region of the search-space while the global best position of the complementary swarm changes. The reset for swarm k occurs when the normalized sum of distances to the global best of the swam becomes smaller than a convergence constant ε, as described in [47].

A pseudo-code for the proposed cooperative PSO (COOP-PSO) tuning technique is given in Algorithm 1.

5. Case studies

5.1. Experimental set-up

This section aims to demonstrate the effectiveness of the proposed tuning methodology through simulated experiments. The proposed control scheme was fully programmed in MATLAB® environment and the simulations were performed by solving the system of ODEs (5) and (6) at each time instant to calculate the quadrotor's position and attitude.

In order to perform the simulations, the values shown in Table 1 were assigned to the quadrotor parameters.

Table 1
Quadrotor parameters.

Symbol	Description	Value	Unit
l	Arm length	0.24	m
m	Mass of the quadrotor	1	kg
I_x	Body moment of inertia around x axis	$8\ 10^{-3}$	N m s^2
I_y	Body moment of inertia around y axis	$8\ 10^{-3}$	N m s^2
I_z	Body moment of inertia around z axis	$14.2\ 10^{-3}$	N m s^2
J_r	Rotational moment of inertia	$1.08\ 10^{-6}$	N m s^2
b	Thrust coefficient	$54.2\ 10^{-6}$	N s^2
d	Drag coefficient	$1.1\ 10^{-6}$	N m s^2
K_t	Aero dynamical damping translational matrix	diag(0.048,0.11,0.046)	N m s^2
K_r	Aero dynamical damping rotational matrix	diag(0.03,0.03,0.01)	N m s^2
g	Acceleration due to gravity	9.81	m s^2

Algorithm 1 COOP-PSO Algorithm.

```
Input:   $\varepsilon$: swarm convergence parameter
         $c_1, c_2, w, P_k v_{max}$: PSO operational parameters
         $s$: Swarm size
         $Gen$: Maximum generations set for each COOP run
Output:  Optimized quadrotor control parameters

1:  For $i$ =1 to the number of particles $s$
2:    Initialize particles $P_1 x_i$ and $P_2 x_i$ randomly
3:    Compute fitness $f(P_1 x_i(0), P_2 x_i(0))$ and initialize $P_1 y_i(0)$, $P_2 y_i(0)$ accordingly
4:  End for
5:  Initialize global best positions $P_1 \hat{y}(0)$ and $P_2 \hat{y}(0)$
6:  For $t$ =1 to the maximum number of generations $Gen$
7:    Select the MPC swarm $k = 1$
8:    If $t > 1$ and $P_2 \hat{y}(t-1) \neq P_2 \hat{y}(t-2)$ and normalized sum of distances for swarm 1 $< \varepsilon$
9:    Then reset $P_1 x_i(t), \quad i = 1, \ldots s$ to random positions
10:   For $i$ =1 to the number of particles $s$
11:     Compute fitness $f\left(P_1 x_i(t), P_2 \hat{y}_i(t-1)\right)$ and update $P_1 y_i(t)$ if needed
12:   End for
13:   Update global best $P_1 \hat{y}(t)$ if needed
14:   For $i$ =1 to the number of particles $s$
15:     For $j$ =1 to the number of dimensions $n$
16:       Calculate velocity $P_1 v_{ij}(t+1)$
17:       Calculate position $P_1 x_{ij}(t+1)$
18:     End for
19:   End for
20:   Select the PID swarm $k = 2$
21:   If $t > 1$ and $P_1 \hat{y}(t) \neq P_1 \hat{y}(t-1)$ and normalized sum of distances for swarm 2 $< \varepsilon$
22:     Then reset $P_2 x_i(t), \quad i = 1, \ldots s$ to random positions
23:   For $i$ =1 to the number of particles $s$
24:     Compute fitness $f\left(P_2 x_i(t), P_1 \hat{y}_i(t)\right)$ and update $P_2 y_i(t)$ if needed
25:   End for
26:   Update global best $P_2 \hat{y}(t)$ if needed
27:   For $i$ =1 to the number of particles $s$
28:     For $j$ =1 to the number of dimensions $n$
29:       Calculate velocity $P_2 v_{ij}(t+1)$
30:       Calculate position $P_2 x_{ij}(t+1)$
31:     End for
32:   End for
33: End for
```

Table 2
Operational parameters for the proposed cooperative algorithm.

Parameter	Symbol	Value
Population	P	25
Inertia coefficient	w	0.5
Nostalgia coefficient	c_1	1.65
Envy coefficient	c_2	1.65
Velocity clamping coefficient	$P_k v_{max}$	0.2
Maximum number of generations	Gen	400
Convergence constant	ε	0.2

The proposed cooperative-PSO algorithm was tested in tuning the quadrotor control framework for two scenarios, by employing two distinct reference spatial trajectories. As these trajectories display different geometrical characteristics in the 3-D space, each one may require a different tuning approach in order for the vehicle to track the given path in an optimal way. A robustness test was also performed by first tuning the controller on a given trajectory and then applying it to a different, unknown trajectory, so as to examine the optimizer's ability to deliver robust quadrotor control parameters.

For comparison purposes, two different metaheuristic search techniques were also tested in tuning the MPC-PID framework, along with the cooperative PSO algorithm. The first one is a non-cooperative PSO algorithm as described in section 4, while the second one involves a genetic algorithm (GA) utilizing binary encoding, scattered crossover and gaussian mutation. Furthermore, the performance of the MPC-PID framework was also compared to a standard PID-based control scheme for trajectory tracking [61]. This scheme also utilizes two control loop parts, namely the inner loop control (ILC) and the outer loop control (OLC), where ILC is responsible for the Euler angle control that regulates the attitude of the vehicle, while OLC regulates the position in the x-y-z plane; however, in this case both control subsystems are based on PID control, while tuning is performed manually.

The operational parameters used by COOP-PSO on all experiments, are given in Table 2. Parameter selection for all comparison approaches was based on suggestions in literature, along with trial and error.

5.2. Results and discussion

The first trajectory chosen to evaluate the proposed cooperative optimizer, namely the spiral trajectory (38), displays simple geometrical characteristics and constitutes a standard benchmark to demonstrate a quadrotors' ability to fly automatically.

Table 3
Tuning performance metrics for the spiral trajectory.

	Fitness average	Fitness standard deviation	Best Fitness	p-value
COOP-PSO	36.59	3.111	32.17	–
GA	39.80	2.564	34.16	0.001
PSO	38.47	2.052	35.06	0.029
PID	-	-	40.86	6.67E-06

$$\begin{aligned} x_{ref}(t) &= 0.5\cos 0.5t \\ z_{ref}(t) &= 0.5\sin 0.5t \\ y_{ref}(t) &= 1 + t/10 \end{aligned} \tag{38}$$

The second trajectory, given by (39), is named composite, as it consists of three distinct sub-trajectories, namely a spiral [62], a helical cone [63], and a linear dependent [34] trajectory. Combining trajectories with different characteristics, makes the tuning process a demanding task for the optimizer.

$$\begin{aligned} x_{ref}(t) &= \begin{cases} 0.8\cos 0.6t, & t\in[0,22] \\ 6.52 - 0.266t, & t\in(22,38] \\ -6 + (6.2-0.1t)\cos 0.5t, & t\in[38,65] \\ -6.166, & t\in(65,80] \end{cases}, \\ y_{ref}(t) &= \begin{cases} 0.8\sin 0.6t, & t\in[0,22] \\ 0.367t - 7.62, & t\in(22,38] \\ 6 + (6.2-0.1t)\sin 0.5t, & t\in[38,65] \\ 5.73, & t\in(65,80] \end{cases}, \\ z_{ref}(t) &= \begin{cases} 0.1t + 2, & t\in[0,22] \\ 0.22t - 0.68, & t\in(22,38] \\ 12.5 - 0.125t, & t\in[38,65] \\ 8.7 - 0.066t, & t\in(65,80] \end{cases} \end{aligned} \tag{39}$$

The simulated results also contain a robustness test for the proposed cooperative optimization tuning technique. In this case, the tuning parameters generated for the spiral trajectory simulation runs, were used to guide the quadrotor so as to follow a new trajectory (40), namely the complex helical [43]. The geometry of this trajectory exhibits an elliptic trail with complex characteristics.

$$\begin{aligned} x_{ref}(t) &= \cos 0.6t - \cos(0.5t)^3 \\ z_{ref}(t) &= \sin 0.2t - \sin(0.4t)^3 \\ y_{ref}(t) &= 0.3t \end{aligned} \tag{40}$$

Due to the stochastic nature of all metaheuristic optimizers, which produce a different result for each run, 20 runs were performed for each trajectory for the cases of COOP-PSO, GA and PSO. To validate the results, a t-test between the proposed cooperative strategy and each of the comparison methods was implemented. In the cases of GA and PSO the null hypothesis defines that the results from COOP-PSO and the rivaling method are generated from normal distributions with equal means, while in the case of PID that the results from COOP-PSO are generated from a normal distribution with a mean equal to the fitness generated by the PID approach.

Based on above, Tables 3–5 present the results for all methods, including the mean fitness values and standard deviations, the best fitness values, and p-values resulting from the t-tests for the spiral, composite and complex helical trajectories, respectively. The response for the position variables x, y, z and the attitude angles φ, θ, ψ for the case of the spiral trajectory is shown in Fig. 5, while the flight of the quadrotor in the 3-D space is visually depicted in Fig. 6. Figs. 7–8 and 9–10 present the corresponding results for the cases of the composite trajectory and the complex helical trajectory where the robustness case was tested, respectively. For each one of the three different trajectories, a randomly selected simulation out of the 20 runs is depicted in the figures.

As it can be seen by the figures, the attitude and position control schemes satisfy their respective goals and the overall control framework manages to track accurately the reference trajectory in all cases. The efficiency of the proposed scheme is highlighted by the scores in Tables 3–5 where the cooperative PSO-tuned controller results to superior performance metrics over its rivals in all the tested trajectories.

To be more specific, as far as the spiral trajectory is concerned, the proposed approach outperforms all the methods used for comparison, producing the best run, which results to the minimum offset error from the reference trajectory. More importantly, it also outperforms its rivals in terms of the average result produced from the 20 runs, with a statistical significance higher than 97% according the produced p-values.

The composite trajectory forms a more demanding test compared to the spiral one, as it essentially comprises three different sub-trajectories, and thus provides a strong indication of the ability of the tuning method to cope with reference signals that exhibit different geometrical characteristics. In this respect, the COOP-PSO algorithm produces superior results compared to the remaining methods, considering either the best produced run, or the average from the 20 runs. Regarding the later, the superiority of the proposed approach is confirmed with a statistical significance of 99%.

In the last tested trajectory, namely the complex helical one, the objective was to evaluate the robustness of the different tuning methods; this can be assessed by asking the controllers to follow a trajectory different than the one used for tuning their parameters.

Table 4
Tuning performance metrics for the composite trajectory.

	Fitness average	Fitness standard deviation	Best Fitness	*p*-value
COOP-PSO	89.47	4.075	82.49	–
GA	97.12	5.076	84.86	5.99E-06
PSO	94.97	6.082	83.94	0.002
PID	-	-	94.47	2.72E-05

Table 5
Robustness metrics for the complex helical trajectory.

	Fitness average	Fitness standard deviation	Best Fitness	p-value
COOP-PSO	106.81	4.777	98.75	–
GA	112.25	5.960	101.89	0.003
PSO	110.51	3.356	100.19	0.007
PID	-	-	10E+05	0

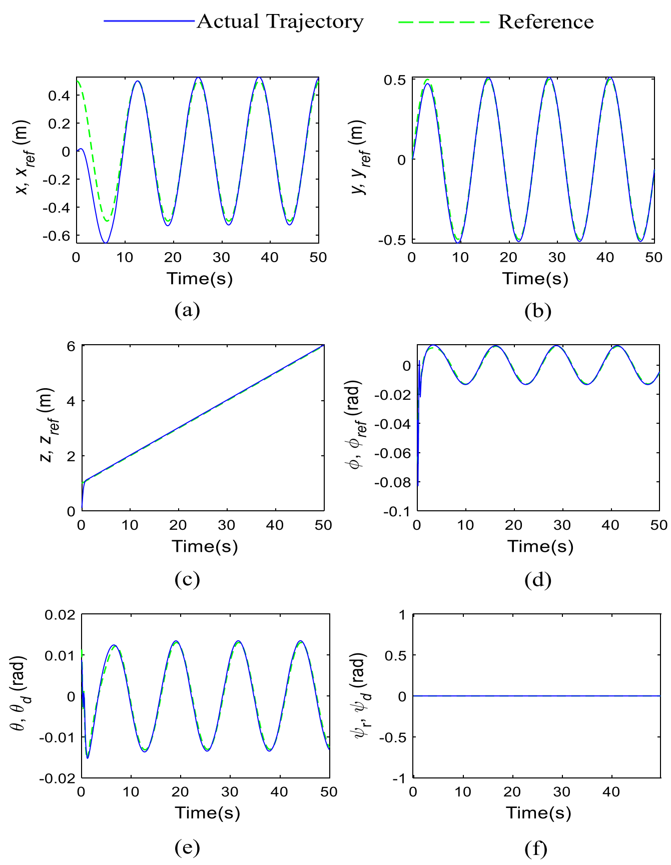

Fig. 5. Spiral trajectory simulation results for (a) position x, (b) position y, (c) position z, (d) roll angle φ, (e) pitch angle θ, (f) yaw angle ψ. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

To this end, the control parameters that were generated by tuning on the spiral trajectory, were then applied to a situation where the quadrotor was asked to follow the complex helical trajectory. The results confirm the superiority of the proposed cooperative technique in delivering robust design parameters, both in terms of the best run and the average of 20 runs. The p-values produced from the t-test show that this conclusion is statistically significant with a confidence level of 99% or higher. From a practical point of view, this is an important result, as in a real-world situation it is not expected that the quadcopter will be tuned beforehand for all possible trajectories that it may be asked to track – most probably at some point during its operation the quadcopter will need to perform moves that are not part of the trajectories used for tuning it.

It should be noted that, besides producing controllers with better tracking abilities and lower offset error, the proposed approach also manages to produce consistent results, as indicated by the standard deviation values from the average of the 20 runs, which are kept within a reasonable range throughout all the scenarios. This outcome comes in contrast with the metrics provided by the PSO and GA algorithms, which exhibit variations in standard deviation values, depending heavily on the geometry of the respective trajectory. This is a strong indication that the proposed approach yields consistent results, even for spatial trajectories with different geometrical attributes.

Among the rest of the competing methods, PSO seems to produce slightly better results compared to standard GA on average, albeit noticeably worse than the proposed approach. The standard PID control scheme on the other hand produces a tracking error which is considerably higher than the rest of the methods for all three tested cases; especially in the third case evaluating the robustness of the

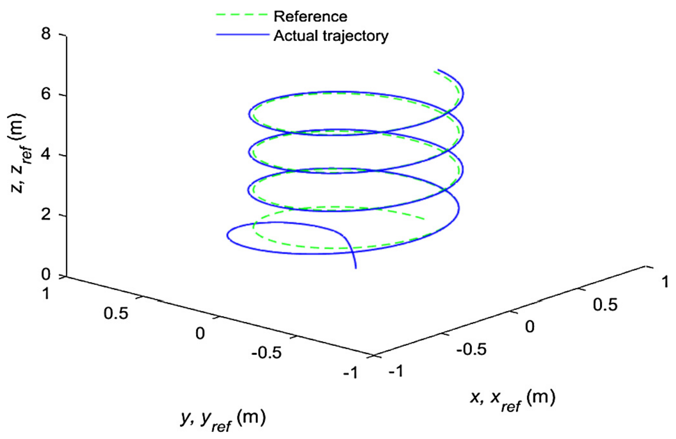

Fig. 6. 3-D simulation results for the spiral trajectory.

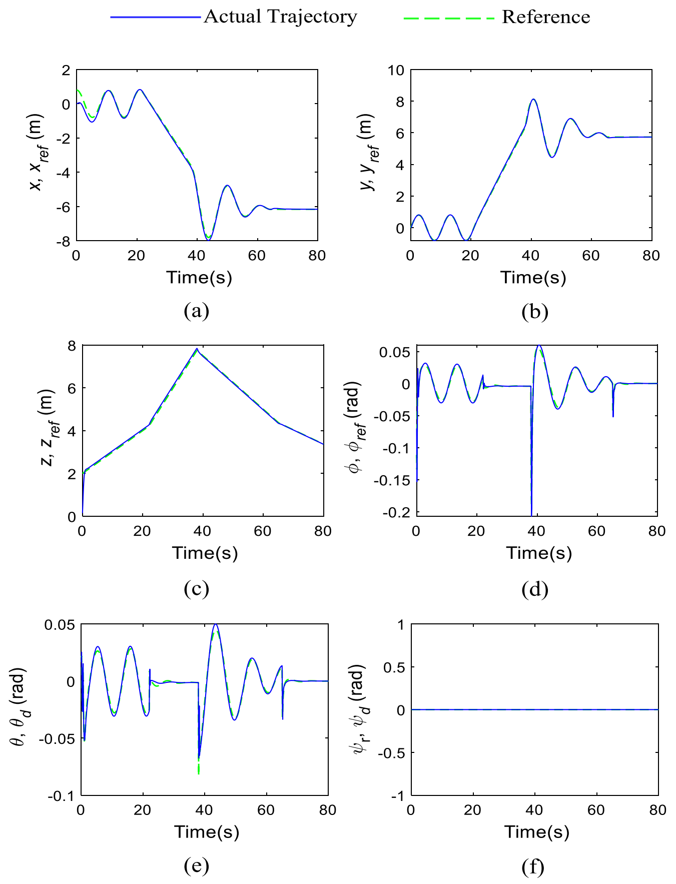

Fig. 7. Composite trajectory simulation results for (a) position x, (b) position y, (c) position z, (d) roll angle φ, (e) pitch angle θ, (f) yaw angle ψ.

methods, the tracking error of the PID scheme is orders of magnitude higher, essentially failing to follow the reference trajectory. This can be explained by the superiority of the MPC scheme employed by the rest of the methods for position control, over the standard PID technique.

The success of the proposed scheme can be attributed to the cooperation between the two different swarms which exchange information while evolving simultaneously. Some insight into this cooperation can be obtained through Fig. 11, which depicts the respective fitness value per generation for the MPC and PID swarms, in the spiral and composite trajectories. It can be seen that the lead in terms of lower fitness value between the two swarms may change many times, as each swarm helps the other to evolve through their mutual cooperation.

6. Conclusions

In this paper a cooperative PSO approach is presented for fully tuning the parameters of a control framework comprising two different control schemes, with the objective of solving the path tracking problem for a quadrotor; to be more specific, the two control schemes include an MPC controller responsible for position control, and a PID one that adjusts the attitude of the quadrotor. Both schemes employ

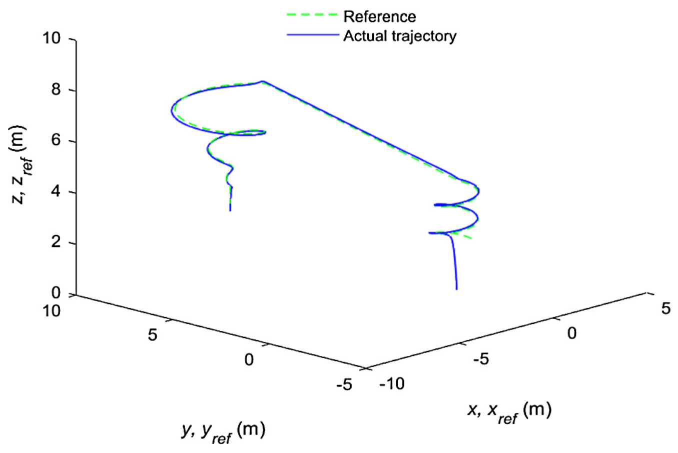

Fig. 8. 3-D simulation results for the composite trajectory.

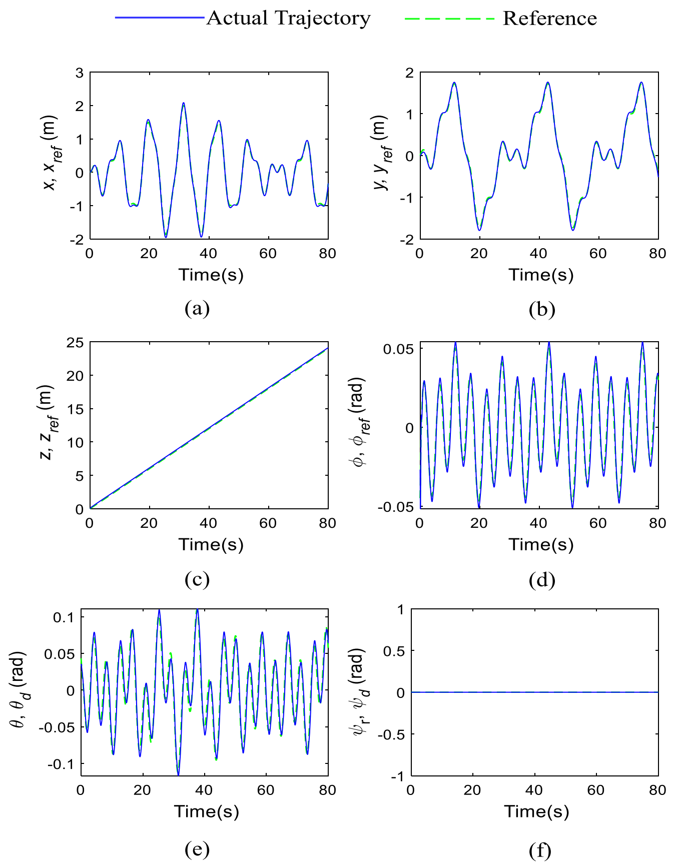

Fig. 9. Complex helical simulation results for (a) position x, (b) position y, (c) position z, (d) roll angle φ, (e) pitch angle θ, (f) yaw angle ψ.

a large number of design parameters, which render the tuning procedure of the overall framework a cumbersome task. To cope with the complexity of the optimization problem corresponding to tuning the method, a cooperative PSO approach is proposed, that separates the parameters to be tuned into two distinct swarms, where the first swarm contains the parameters of the MPC position controller, whereas the second takes care of the parameters used by the PID attitude controller. The two swarms cooperate by exchanging information towards discovering improved solutions.

The presented tuning method is evaluated on 3 different trajectory cases. In the first two cases the proposed approach is used for tuning the controllers on a given spatial trajectory, and then the quadrotor tracking ability is evaluated on the same trajectory, whereas the third one focuses on assessing the method's robustness, by evaluating the quadrotor tracking ability to a new trajectory, different than the one used for tuning. The performance of the proposed methodology is validated through comparison tests against the GA and PSO algorithms, as well as a standard PID control scheme. Statistical testing on the simulated results verifies that the cooperative scheme provides tuned parameters with better trajectory tracking ability and robustness compared to its rivals.

Future research will be directed towards modifying the proposed tuning approach for application in other schemes that incorporate different control structures with an increased number of tuning parameters of diverse nature, like in combinations of LQR and backstepping [64], or sliding mode and H_∞ control schemes [65], or even in multiple cooperating MPC schemes [66].

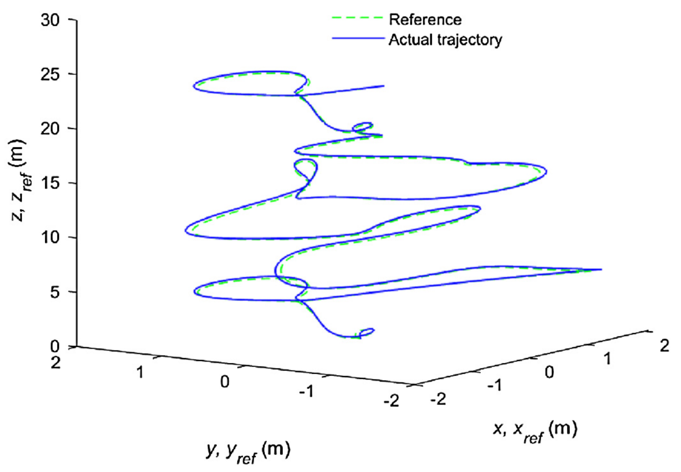

Fig. 10. 3-D simulation results for the complex helical trajectory.

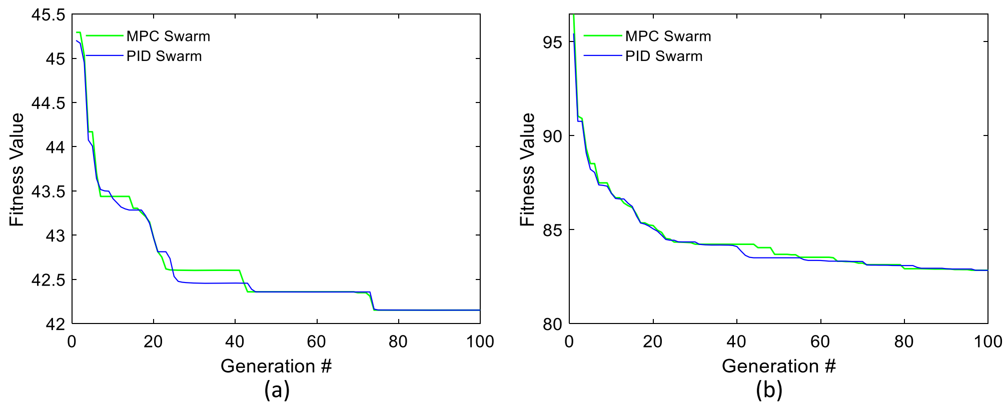

Fig. 11. Change of best fitness value per generation for the MPC and PID swarms in the case of (*a*) spiral trajectory, (*b*) composite trajectory.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Mahony, V. Kumar, P. Corke, Multirotor aerial vehicles: modeling, estimation, and control of quadrotor, IEEE Robot. Autom. Mag. 19 (3) (2012) 20–32.
[2] C.S. Sharp, O. Shakernia, S.S. Sastry, A vision system for landing an unmanned aerial vehicle, in: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, IEEE, 2001.
[3] E. Camci, et al., An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm, Swarm Evol. Comput. 41 (2018) 1–8.
[4] F. Fraundorfer, et al., Vision-based autonomous mapping and exploration using a quadrotor MAV, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012.
[5] T. Tomic, et al., Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue, IEEE J. Robot. Autom. 19 (3) (2012) 45–56.
[6] K.A. Ghamry, Y. Zhang, Cooperative control of multiple UAVs for forest fire monitoring and detection, in: 2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), 2016.
[7] S. Bouabdallah, Design and Control of Quadrotors with Application to Autonomous Flying, Ecole Polytechnique federale de Lausanne, 2007.
[8] E. Capello, et al., Mini quadrotor UAV: design and experiment, J. Aerosp. Eng. 25 (4) (2012) 559–573.
[9] B.T.M. Leong, S.M. Low, M.P. Ooi, Low-cost microcontroller-based hover control design of a quadcopter, Proc. Eng. 41 (2012) 458–464.
[10] I.D. Cowling, et al., A prototype of an autonomous controller for a quadrotor UAV, in: 2007 European Control Conference (ECC), 2007.
[11] M.L. Dat, H. Cheolkeun, Modeling and control of quadrotor MAV using vision-based measurement, in: International Forum on Strategic Technology, 2010.
[12] S. Bouabdallah, A. Noth, R. Siegwart, PID vs LQ control techniques applied to an indoor micro quadrotor, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 2004.
[13] W. Haibo, et al., Robust H∞ attitude tracking control of a quadrotor UAV on SO(3) via variation-based linearization and interval matrix approach, ISA Trans. 87 (2019) 10–16.
[14] G.V. Raffo, M.G. Ortega, F.R. Rubio, Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle, in: American Control Conference, 2008.
[15] G.V. Raffo, M.G. Ortega, F.R. Rubio, An integral predictive/nonlinear H∞ control structure for a quadrotor helicopter, Automatica 46 (1) (2010) 29–39.
[16] A. Das, K. Subbarao, F. Lewis, Dynamic inversion with zero-dynamics stabilisation for quadrotor control, IET Control Theory Appl. 3 (3) (2009) 303–314.
[17] H. Voos, Nonlinear control of a quadrotor micro-UAV using feedback-linearization, in: IEEE International Conference on Mechatronics, 2009.
[18] Y. Choi, H. Ahn, Nonlinear control of quadrotor for point tracking: actual implementation and experimental tests, IEEE/ASME Trans. Mechatron. 20 (3) (2015) 1179–1192.
[19] S. Bouabdallah, R. Siegwart, Full control of a quadrotor, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 2007, pp. 153–158.
[20] H. Liu, et al., Robust backstepping-based trajectory tracking control for quadrotors with time delays, IET Control Theory Appl. 13 (12) (2019) 1945–1954.
[21] H. Sira-Ramirez, M. Zribi, S. Ahmad, Dynamical sliding mode control approach for vertical flight regulation in helicopters, IEE Proc., Control Theory Appl. 141 (1) (1994) 19–24.

[22] H. Wang, et al., Model-free–based terminal SMC of quadrotor attitude and position, IEEE Trans. Aerosp. Electron. Syst. 52 (5) (2016) 2519–2528.
[23] H. Razmi, S. Afshinfar, Neural network-based adaptive sliding mode control design for position and attitude control of a quadrotor UAV, Aerosp. Sci. Technol. 91 (2019) 12–27.
[24] R. Miranda-Colorado, L.T. Aguilar, J.E. Herrero-Brito, Reduction of power consumption on quadrotor vehicles via trajectory design and a controller-gains tuning stage, Aerosp. Sci. Technol. 78 (2018) 280–296.
[25] P. Tang, et al., An integral TSMC-based adaptive fault-tolerant control for quadrotor with external disturbances and parametric uncertainties, Aerosp. Sci. Technol. 109 (2021) 106415.
[26] F. Chen, et al., Robust backstepping sliding-mode control and observer-based fault estimation for a quadrotor UAV, IEEE Trans. Ind. Electron. 63 (8) (2016) 5044–5056.
[27] E.F. Camacho, C. Bordons, Model Predictive Control, 2 ed., 2007.
[28] David Q. Mayne, Model predictive control: recent developments and future promise, Automatica 50 (12) (2014) 2967–2986.
[29] M. Stogiannos, A. Alexandridis, H. Sarimveis, Model predictive control for systems with fast dynamics using inverse neural models, ISA Trans. 72 (2018) 161–177.
[30] A. Alexandridis, H. Sarimveis, K. Ninos, A radial basis function network training algorithm using a non-symmetric partition of the input space – application to a model predictive control configuration, Adv. Eng. Softw. 42 (10) (2011) 830–837.
[31] K. Alexis, G. Nikolakopoulos, A. Tzes, Constrained optimal attitude control of a quadrotor helicopter subject to wind-gusts: experimental studies, in: Proceedings of the 2010 American Control Conference, IEEE, Baltimore, MD, USA, 2010.
[32] K. Alexis, G. Nikolakopoulos, A. Tzes, Model predictive quadrotor control: attitude, altitude and position experimental studies, IET Control Theory Appl. 6 (12) (2012) 1812–1827.
[33] G.V. Raffo, M.G. Ortega, F.R. Rubio, MPC with nonlinear $\mathcal{H}\infty$ control for path tracking of a quad-rotor helicopter, IFAC Proc. Vol. 41 (2) (2008) 8564–8569.
[34] M. Dailiang, et al., Active disturbance rejection and predictive control strategy for a quadrotor helicopter, IET Control Theory Appl. 10 (17) (2016) 2213–2222.
[35] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. 6th Int. Symp. Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[36] M.J. Mahmoodabadi, State-varying optimal decoupled sliding mode control for the Lorenz chaotic nonlinear problem based on HEPSO and MLS, Int. J. Model. Simul. (2020) 1–10.
[37] J. Tao, G. Sun, Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization, Aerosp. Sci. Technol. 92 (2019) 722–737.
[38] H. Boubertakh, S. Bencharef, S. Labiod, PSO-based PID control design for the stabilization of a quadrotor, in: 3rd International Conference on Systems and Control, IEEE, Algiers, Algeria, 2013.
[39] W. Yaonan, et al., Fuzzy radial basis function neural network PID control system for a quadrotor UAV based on particle swarm optimization, in: 2015 IEEE International Conference on Information and Automation, IEEE, Lijiang, China, 2015.
[40] F. Yacef, et al., PSO optimization of integral backstepping controller for quadrotor attitude stabilization, in: International Conference on Systems and Control, IEEE, Algiers, Algeria, 2013.
[41] L. Xuejing, et al., Optimal path planning for UAV based inspection system of large-scale photovoltaic farm, in: Chinese Automation Congress (CAC), IEEE, Jinan, China, 2017.
[42] P. Cheng, et al., ADRC trajectory tracking control based on PSO algorithm for a quad-rotor, in: 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), IEEE, Melbourne, VIC, Australia, 2013.
[43] J.J. Wang, G.Y. Liu, Saturated control design of a quadrotor with heterogeneous comprehensive learning particle swarm optimization, Swarm Evol. Comput. 46 (2019) 84–96.
[44] M.J. Mahmoodabadi, N. Rezaee Babak, Robust fuzzy linear quadratic regulator control optimized by multi-objective high exploration particle swarm optimization for a 4 degree-of-freedom quadrotor, Aerosp. Sci. Technol. 97 (2020) 105598.
[45] F.B. de Oliveira, et al., A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem, Expert Syst. Appl. 43 (2016) 117–130.
[46] S. Ono, et al., User-system cooperative evolutionary computation for both quantitative and qualitative objective optimization in image processing filter design, Appl. Soft Comput. 15 (2014) 203–218.
[47] A. Alexandridis, E. Chondrodima, H. Sarimveis, Cooperative learning for radial basis function networks using particle swarm optimization, Appl. Soft Comput. 49 (2016) 485–497.
[48] S. Bouabdallah, Design and control of quadrotors with application to autonomous flying, PhD Thesis, in: STI School of Engineering, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2007.
[49] X. Li, et al., Finite-time control for quadrotor based on composite barrier Lyapunov function with system state constraints and actuator faults, Aerosp. Sci. Technol. 119 (2021) 107063.
[50] P. Tang, et al., Observer based finite-time fault tolerant quadrotor attitude control with actuator faults, Aerosp. Sci. Technol. 104 (2020) 105968.
[51] A. Tayebi, S. McGilvray, Attitude stabilization of a VTOL quadrotor aircraft, IEEE Trans. Control Syst. Technol. 14 (3) (2006) 562–571.
[52] P. Castillo, R. Lozano, A. Dzul, Stabilization of a mini rotorcraft with four rotors, IEEE Control Syst. Mag. 25 (6) (2005) 45–55.
[53] M.A. Johnson, et al., PID Control: New Identification and Design Methods, 2005, pp. 1–543.
[54] C. Ning, A.F. Lynch, Inner–outer loop control for quadrotor UAVs with input and state constraints, IEEE Trans. Control Syst. Technol. 24 (5) (2016) 1797–1804.
[55] J.L. Garriga, M. Soroush, Model predictive control tuning methods: a review, Ind. Eng. Chem. Res. 49 (2010) 2243–2248.
[56] W. Wojsznis, et al., Practical approach to tuning MPC, ISA Trans. 42 (2003) 149–162.
[57] G. Shah, S. Engell, Tuning MPC for desired closed-loop performance for SISO systems, in: 18th Mediterranean Conference on Control and Automation, MED'10 - Conference Proceedings, San Francisco, CA, USA, 2010, pp. 628–633.
[58] M.A. Potter, K.A.D. Jong, A cooperative coevolutionary approach to function optimization, in: International Conference on Parallel Problem Solving from Nature, 1994, pp. 249–257.
[59] F. Van den Bergh, A.P. Engelbrecht, Cooperative learning in neural networks using particle swarm optimizers, South African Comput. J. 26 (2000) 84–90.
[60] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004).
[61] D. Mellinger, Trajectory generation and control for quadrotors, in: Mechanical Engineering & Applied Mechanics, University of Pennsylvania, 2012.
[62] M. Islam, M. Okasha, M.M. Idres, Dynamics and control of quadcopter using linear model predictive control approach, IOP Conf. Ser., Mater. Sci. Eng. 270 (2017) 012007.
[63] H. ZeFang, Z. Long, Internal model control /backstepping sliding model control for quadrotor trajectory tracking, in: 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2017.
[64] Z. Liu, C. Luo, D. Hu, Active suspension control design using a combination of LQR and backstepping, in: 2006 Chinese Control Conference, 2006.
[65] B. Kharabian, H. Mirinejad, Hybrid sliding mode/H-infinity control approach for uncertain flexible manipulators, IEEE Access 8 (2020) 170452–170460.
[66] X. Xiaowen, X. Lili, M. Hongmin, Cooperative energy management optimization based on distributed MPC in grid-connected microgrids community, Int. J. Electr. Power Energy Syst. 107 (2019) 186–199.