Mohd Ariffanan Mohd Basri
Kumeresan A. Danapalasingam
Abdul Rashid Husain

ISSN 1333-1124
eISSN 1849-1391

DESIGN AND OPTIMIZATION OF BACKSTEPPING CONTROLLER FOR AN UNDERACTUATED AUTONOMOUS QUADROTOR UNMANNED AERIAL VEHICLE

UDC 681.5.015.23:629.7

Summary

The development of a high performance controller for a quadrotor unmanned aerial vehicle (UAV) is a challenging issue since a quadrotor is an underactuated and a highly unstable nonlinear system. In this paper, the contribution is focused on the design and optimization of a controller for an autonomous quadrotor UAV. Firstly, the dynamic model of the aerial vehicle is mathematically formulated. Then, an optimal backstepping controller (OBC) is proposed. Conventionally, control parameters of a backstepping controller (BC) are often chosen arbitrarily. To this end, it is necessary to invoke a well-established optimization algorithm in order to find the best parameters. Here, the particle swarm optimization (PSO) is utilized as a new key idea to determine the optimal values of the BC parameters. In the algorithm, the control parameters are computed by minimizing the fitness function defined by using the integral absolute error (*IAE*) performance index. Since the control law is derived based on the Lyapunov theorem, the asymptotical stability of the system can be guaranteed. Finally, the efficiency of the proposed OBC is illustrated by implementing several simulation experiments.

Key words: *quadrotor, underactuated system, backstepping control, particle swarm optimization*

1. Introduction

Unmanned Aerial Vehicles are being widely used in a number of applications involving surveillance of indoor or outdoor environments, remote inspection and monitoring of hostile environments. Among UAVs, quadrotors are emerging as a popular platform, due to their larger payload capability and their higher manoeuvrability with respect to single-rotor vehicles. Furthermore, quadrotors have a set of advantages over traditional helicopters in terms of structure, motion control and cost. Therefore, quadrotors have received increasing attention from scientists and engineers, and have also become a promising option for various unmanned military and civilian applications. However, the quadrotor UAV poses great scientific and engineering problems in terms of vertical take-off and landing (VTOL), autonomous hovering and manoeuvring due to its features including underactuation, strong coupling, multivariable

and unknown nonlinearities. Hence, the control of a quadrotor becomes quite a complex and difficult task mainly due to its underactuated properties and nonlinearities.

The backstepping control scheme is a nonlinear control method based on the Lyapunov theorem. The backstepping control design techniques have received great attention because of its systematic and recursive design methodology for nonlinear feedback control [1-5]. Unlike the feedback linearization method with the problems such as the precise model requirement and the cancellation of useful nonlinear terms, the backstepping approach offers a choice of design tools for accommodation of nonlinearities, and can avoid unwanted cancellations. The advantage of backstepping compared with other control methods lies in its design flexibility, due to its recursive use of Lyapunov functions. The key idea of the backstepping design is to select recursively some appropriate state variables as virtual inputs for lower dimension subsystems of the overall system and the Lyapunov functions are designed for each stable virtual controller [6]. Therefore, the designed final actual control law can guarantee the stability of the total control system.

In aviation technology, the backstepping technique has been used to solve the stabilization and trajectory tracking problems of quadrotor UAVs [7-11]. Although the backstepping method can provide a systematic process for controller design, it is not easy to get satisfactory performance because the controller parameters obtained by the backstepping method are chosen arbitrarily. It is necessary to select proper parameters to obtain a good response because an improper selection of the parameters leads to inappropriate responses or may even lead to instability of the system. This can also happen if the parameters are properly chosen, but it cannot be said that the optimal parameters are selected.

Genetic algorithm (GA) and particle swarm optimization (PSO) are well-established optimization algorithms that belong to the class of evolutionary algorithms (EA). These heuristics are routinely used to generate useful solutions to optimization and search problems in a wide range of engineering and computer sciences [12-17]. GA is a technique inspired by natural evolution, such as inheritance, mutation, selection, and crossover. GA has demonstrated the ability to reach near optimal solutions to big problems. However, it may require a long processing time to reach a near optimal solution. Similarly to GA, PSO is also a population based optimizer. The method has been motivated by the behaviour of organisms, such as fish schooling and bird flocking [18]. Unlike GA, PSO does not contain any crossover and mutation processes. Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. In addition, PSO has a memory, so the knowledge of good solutions is retained by all the particles and optimal solutions are found by the swarm particles if they follow the best particle. This swarm feature allows more of the space to be searched which decreases the chances of getting caught up in a local optimum. This is the major advantage of PSO as compared to the gradient descent and other convex optimization techniques. Thus, in this study, PSO is used to off-line compute the optimal parameters for the backstepping controller of quadrotor systems. So far, this technique has not been developed with the backstepping method in aviation applications.

2. Quadrotor system modelling

In order to develop the model of the quadrotor, reasonable assumptions are made in order to accommodate the controller design. The assumptions are as follows [19]:

Assumption 1: Quadrotor is a rigid body and has a symmetric structure.

Assumption 2: Aerodynamic effects can be ignored at low speed.

Assumption 3: The rotor dynamics is relatively fast and thus can be neglected.

Assumption 4: The quadrotor centre of mass and the body-fixed frame origin coincide.

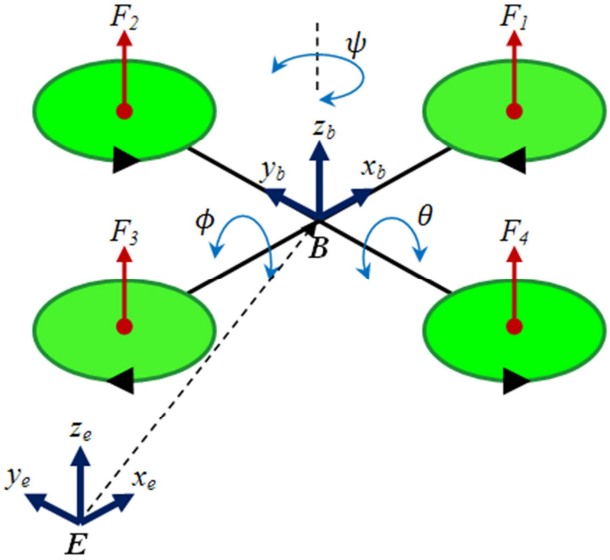

Fig. 1 Configuration of a quadrotor UAV

2.1 Quadrotor kinematic model

Let us consider the earth fixed frame $E = \{x_e, y_e, z_e\}$ and the body fixed frame $B = \{x_b, y_b, z_b\}$, as seen in Fig. 1. Let $q = (x, y, z, \phi, \theta, \psi) \in R^6$ be the generalized coordinates for the quadrotor, where (x, y, z) denote the absolute position of the rotorcraft and (ϕ, θ, ψ) are the three Euler angles (roll, pitch, and yaw) that describe the orientation of the aerial vehicle. Therefore, the model could be divided into two coordinate subsystems: translational and rotational. They are defined respectively by:

$$\xi = (x, y, z) \in R^3 \tag{1}$$

$$\eta = (\phi, \theta, \psi) \in R^3 \tag{2}$$

The kinematic equations of the translational and rotational movements are obtained by means of the rotation R and transfer T matrices respectively. The expression of the rotation R and transfer T matrices can be defined accordingly by (3) and (4) [20]:

$$R = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \tag{3}$$

$$T = \begin{pmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{pmatrix} \tag{4}$$

where $s(\cdot)$, $c(\cdot)$ and $t(\cdot)$ are abbreviations for $sin(\cdot)$, $cos(\cdot)$ and $tan(\cdot)$, respectively.

The translational kinematic can be written as:

$$\dot{\xi} = RV \tag{5}$$

where $\dot{\xi} = (\dot{x}, \dot{y}, \dot{z})$ and $V = (u, v, w)$ are respectively the linear velocity vector w.r.t. the earth fixed frame E and the body fixed frame B.

The rotational kinematics can be defined as follows:

$$\dot{\eta} = T\omega \tag{6}$$

where $\dot{\eta} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$ and $\omega = (p, q, r)$ are the angular velocity vector w.r.t. the earth fixed frame E and the body fixed frame B, respectively.

2.2 Quadrotor dynamic model

The dynamic model of a quadrotor is derived from the Newton-Euler approach. It can be useful to express the translational dynamic equations w.r.t. the earth fixed frame *E* and the rotational dynamic equations w.r.t. the body fixed frame *B*.

Therefore, the translational dynamic equations of a quadrotor can be written as follows:

$$m\ddot{\xi} = -mge_z + u_T R e_z \tag{7}$$

where m denotes the quadrotor mass, g the gravity acceleration, $e_z = (0,0,1)^T$ the unit vector expressed in the frame *E* and u_T the total thrust produced by the four rotors.

$$u_T = \sum_{i=1}^{4} F_i = b\sum_{i=1}^{4} \Omega_i^2 \tag{8}$$

where F_i and Ω_i denote respectively, the thrust force and the speed of the rotor i and b is the thrust factor.

The rotational dynamic equations of a quadrotor can be written as follows:

$$I\dot{\omega} = -\omega \times I\omega - G_a + \tau \tag{9}$$

where I is the inertia matrix, $-\omega \times I\omega$ and G_a are the gyroscopic effect due to rigid body rotation and the propeller orientation change, respectively, while τ is the control torque obtained by varying the rotor speeds. G_a and τ are defined as:

$$G_a = \sum_{i=1}^{4} J_r(\omega \times e_z)(-1)^{i+1}\Omega_i \tag{10}$$

$$\tau = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} lb(\Omega_4^2 - \Omega_2^2) \\ lb(\Omega_3^2 - \Omega_1^2) \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{pmatrix} \tag{11}$$

where J_r is the rotor inertia, l represents the distance from the rotors to the centre of mass and d is the drag factor.

Then, by recalling (7) and (9), the dynamic model of a quadrotor in terms of position (x, y, z) and rotation (ϕ, θ, ψ) is written as:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \frac{1}{m}\begin{pmatrix} c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\phi s_\theta s_\psi - s_\phi c_\psi \\ c_\phi c_\theta \end{pmatrix} u_T \tag{12}$$

$$\begin{pmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} qr\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right) \\ pr\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right) \\ pq\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right) \end{pmatrix} - \begin{pmatrix} \frac{J_r}{I_{xx}} q\Omega_d \\ -\frac{J_r}{I_{yy}} p\Omega_d \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{I_{xx}}\tau_\phi \\ \frac{1}{I_{yy}}\tau_\theta \\ \frac{1}{I_{zz}}\tau_\psi \end{pmatrix} \tag{13}$$

Consequently, a quadrotor is an underactuated system with six outputs $(x, y, z, \phi, \theta, \psi)$ and four control inputs ($u_T, \tau_\phi, \tau_\theta, \tau_\psi$).

Finally, the quadrotor dynamic model can be written in the following form:

$$\begin{aligned}
\ddot{x} &= \dot{u} = \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)\frac{1}{m}u_1 \\
\ddot{y} &= \dot{v} = \left(c_\phi s_\theta s_\psi - s_\phi c_\psi\right)\frac{1}{m}u_1 \\
\ddot{z} &= \dot{w} = -g + \left(c_\phi c_\theta\right)\frac{1}{m}u_1 \\
\ddot{\phi} &= \dot{p} = qr\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right) - \frac{J_r}{I_{xx}}q\Omega_d + \frac{l}{I_{xx}}u_2 \\
\ddot{\theta} &= \dot{q} = pr\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right) + \frac{J_r}{I_{yy}}p\Omega_d + \frac{l}{I_{yy}}u_3 \\
\ddot{\psi} &= \dot{r} = pq\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right) + \frac{1}{I_{zz}}u_4
\end{aligned} \tag{14}$$

with the renaming of the control inputs as:

$$\begin{aligned} u_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ u_2 &= b(\Omega_4^2 - \Omega_2^2) \\ u_3 &= b(\Omega_3^2 - \Omega_1^2) \\ u_4 &= d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{aligned} \tag{15}$$

and the definition of disturbance as:

$$\Omega_d = \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3 \tag{16}$$

3. Backstepping control system for a quadrotor

The dynamic model (14) can be represented by a nonlinear dynamic equation described as follows:

$$\ddot{X} = f(X) + g(X)u \tag{17}$$

where u and X are respectively the input and the state vector given as follows:

$$u = [u_1\ u_2\ u_3\ u_4]^T \tag{18}$$

$$X = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ ]^T = [x\ y\ z\ \phi\ \theta\ \psi\ ]^T \tag{19}$$

From (14) and (19), the nonlinear dynamic function $f(X)$ and nonlinear control function $g(X)$ matrices can be written accordingly as:

$$f(X) = \begin{pmatrix} f_1(X) \\ f_2(X) \\ f_3(X) \\ f_4(X) \\ f_5(X) \\ f_6(X) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -g \\ qra_1 - qa_2\Omega_d \\ pra_3 + pa_4\Omega_d \\ pqa_5 \end{pmatrix} \tag{20}$$

$$g(X) = \begin{pmatrix} g_1(X) & 0 & 0 & 0 \\ g_2(X) & 0 & 0 & 0 \\ g_3(X) & 0 & 0 & 0 \\ 0 & g_4(X) & 0 & 0 \\ 0 & 0 & g_5(X) & 0 \\ 0 & 0 & 0 & g_6(X) \end{pmatrix} = \begin{pmatrix} u_x\frac{1}{m} & 0 & 0 & 0 \\ u_y\frac{1}{m} & 0 & 0 & 0 \\ u_z\frac{1}{m} & 0 & 0 & 0 \\ 0 & b_1 & 0 & 0 \\ 0 & 0 & b_2 & 0 \\ 0 & 0 & 0 & b_3 \end{pmatrix}$$

with the abbreviations $a_1 = (I_{yy} - I_{zz})/I_{xx}$, $a_2 = J_r/I_{xx}$, $a_3 = (I_{zz} - I_{xx})/I_{yy}$, $a_4 = J_r/I_{yy}$, $a_5 = (I_{xx} - I_{yy})/I_{zz}$, $b_1 = l/I_{xx}$, $b_2 = l/I_{yy}$, $b_3 = 1/I_{zz}$, $u_x = (c_\phi s_\theta c_\psi + s_\phi s_\psi)$, $u_y = (c_\phi s_\theta s_\psi - s_\phi c_\psi)$, $u_z = (c_\phi c_\theta)$

The control objective is to design a suitable control law so that the state trajectory X of the quadrotor system can track a desired reference trajectory $X_d = [x_{1d}\ x_{2d}\ x_{3d}\ x_{4d}\ x_{5d}\ x_{6d}\ ]^T$. Since the description of the control system design of the quadrotor is similar for each one of the six controllable degrees of freedom (DOF), for simplicity only one DOF is considered.

The design of backstepping control is described step-by-step as follows:

Step 1: Define the tracking error:

$$e_1 = x_{1d} - x_1 \tag{21}$$

where x_{1d} is a desired trajectory specified by a reference model. Then the derivative of the tracking error can be represented as:

$$\dot{e}_1 = \dot{x}_{1d} - \dot{x}_1 \tag{22}$$

The first Lyapunov function is chosen as:

$$V_1(e_1) = \frac{1}{2}e_1^2 \tag{23}$$

The derivative of V_1 is:

$$\dot{V}_1(e_1) = e_1\dot{e}_1 = e_1(\dot{x}_{1d} - \dot{x}_1) \tag{24}$$

$\dot{x}_1$ can be viewed as a virtual control. The desired value of virtual control known as a stabilizing function can be defined as follows:

$$\alpha_1 = \dot{x}_{1d} + k_1 e_1 \tag{25}$$

where k_1 is a positive constant and should be determined by the PSO algorithm.

By substituting the virtual control by its desired value, Eq. (24) then becomes:

$$\dot{V}_1(e_1) = -k_1 e_1^2 \leq 0 \tag{26}$$

Step 2: The deviation of the virtual control from its desired value can be defined as:

$$e_2 = \dot{x}_1 - \alpha_1 = \dot{x}_1 - \dot{x}_{1d} - k_1 e_1 \tag{27}$$

The derivative of e_2 is expressed as:

$$\begin{aligned} \dot{e}_2 &= \ddot{x}_1 - \dot{\alpha}_1 \\ &= f_1(X) + g_1(X)u_1 - \ddot{x}_{1d} - k_1\dot{e}_1 \end{aligned} \tag{28}$$

The second Lyapunov function is chosen as:

$$V_2(e_1, e_2) = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 \tag{29}$$

Finding derivative of (29) yields:

$$\begin{aligned} \dot{V}_2(e_1, e_2) &= e_1\dot{e}_1 + e_2\dot{e}_2 \\ &= e_1(\dot{x}_{1d} - \dot{x}_1) + e_2(\ddot{x}_1 - \dot{\alpha}_1) \\ &= e_1(-e_2 - k_1 e_1) + e_2(f_1(X) + g_1(X)u_1 - \ddot{x}_{1d} - k_1\dot{e}_1) \\ &= -k_1 e_1^2 + e_2(-e_1 + f_1(X) + g_1(X)u_1 - \ddot{x}_{1d} - k_1\dot{e}_1) \end{aligned} \tag{30}$$

Step 3: For satisfying $\dot{V}_2(e_1, e_2) \leq 0$, the control input u_1 is selected as:

$$u_1 = \frac{1}{g_1(X)}(e_1 + k_1\dot{e}_1 + \ddot{x}_{1d} - f_1(X) - k_2 e_2) \tag{31}$$

where k_2 is a positive constant and should be also determined by the PSO algorithm. The term $k_2 e_2$ is added to stabilize the tracking error e_1.

Substituting (31) into (30), the following equation can be obtained:

$$\dot{V}_2(e_1, e_2) = -k_1 e_1^2 - k_2 e_2^2 = -E^T K E \leq 0 \tag{32}$$

where $E = [e_1\ e_2]^T$ and $K = diag(k_1, k_2)$. Since $\dot{V}_2(e_1, e_2) \leq 0$, $\dot{V}_2(e_1, e_2)$ is a negative semi-definite.

Therefore, the control law in (31) will asymptotically stabilize the system.

4. Optimization of backstepping control parameters

4.1 Overview of particle swarm optimization

The PSO method is a member of a wide category of swarm intelligence methods for solving optimization problems. It is a population based search algorithm where each individual is referred to as a particle and represents a candidate solution. Each particle in PSO moves through the search space with an adaptable velocity that is dynamically modified according to its own moving experience and also to the moving experience of the other particles. In PSO each particle strives to improve itself by imitating the traits of its successful peers. Further, each particle has a memory and hence it is capable of remembering the best position in the search space it has ever visited. The position corresponding to the best fitness is known as *pbest* and the overall best out of all the particles in the population is called *gbest*.

The basic PSO algorithm consists of three steps: generating particles positions and velocities, velocity update, and finally, position update. Here, a particle refers to a point in the design space that changes its position from one move (iteration) to another based on velocity updates. First, the positions, x_i^k, and velocities, v_i^k, of the initial swarm of particles are randomly generated using the upper and lower bounds on the design variables values, x_{min} and x_{max}, as expressed in Eqs. (33) and (34) [21].

$$x_i^0 = x_{min} + rand(x_{max} - x_{min}) \tag{33}$$

$$v_i^0 = x_{min} + rand(x_{max} - x_{min}) \tag{34}$$

Here, the positions and velocities are given in a vector format with the subscript and superscript denoting the *i*th particle at iteration k, respectively. In Eqs. (33) and (34), *rand* is a uniformly distributed random variable that can take any value between 0 and 1. This initialization process allows the swarm particles to be randomly distributed across the design space.

The second step is to update the velocities of all particles at iteration $k+1$ using the objective or fitness values of particles, which are functions of the particle current positions in the design space at iteration k. The fitness function value of a particle determines which particle has the best global value in the current swarm, $gbest$, and also determines the best position of each particle over iteration, $pbest$, i.e. in the current and all previous moves. The velocity update formula uses these two pieces of information for each particle in the swarm along with the effect of current motion, v_i^k, to provide a search direction, v_i^{k+1}, for the next iteration. The velocity update formula includes some random parameters, represented by the uniformly distributed variables, *rand*, to ensure good coverage of the design space and avoid entrapment in local optima. The three values that effect the new search direction, namely, current motion, particle own memory, and swarm influence, are incorporated via a summation approach as shown in Eq. (35) with three weight factors, namely, inertia factor, w, self confidence factor, c_1, and swarm confidence factor, c_2.

$$v_i^{k+1} = w \cdot v_i^k + c_1 \cdot rand \cdot \left(pbest - x_i^k\right) + c_2 \cdot rand \cdot (gbest - x_i^k) \tag{35}$$

The appropriate value range for c_1 and c_2 is 1-2, but 2 is the most appropriate in many cases. The following inertia weight is used [22]:

$$w = w_{max} - (w_{max} - w_{min})\, k/k_{max} \tag{36}$$

where k_{max}, k is the maximum number of iterations and the current number of iterations, respectively. Where, w_{min} and w_{max} are the minimum and maximum weights, respectively. Appropriate values for w_{min} and w_{max} are 0.4 and 0.9, respectively [23].

Position update is the last step in each iteration. The Position of each particle is updated using its velocity vector as shown in Eq. (37) and depicted in Fig. 2.

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{37}$$

The three steps of velocity update, position update, and fitness calculations are repeated until a desired convergence criterion is met.

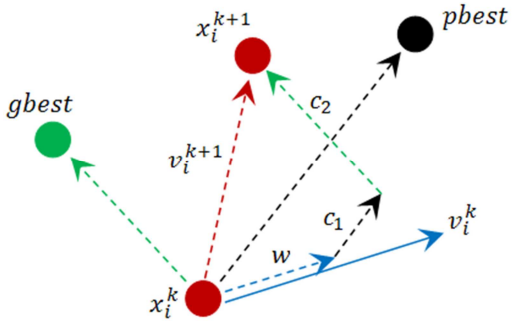

Fig. 2 Depiction of the velocity and position updates in PSO

4.2 Optimal backstepping control system

The dynamic model in (14) can be divided into two subsystems Π_1 and Π_2, listed as follows:

$$\Pi_1: \begin{cases} \ddot{\phi} = qr\left(\frac{I_{yy}-I_{zz}}{I_{xx}}\right) - \frac{J_r}{I_{xx}} q\Omega_d + \frac{l}{I_{xx}} u_2 \\ \ddot{\theta} = pr\left(\frac{I_{zz}-I_{xx}}{I_{yy}}\right) + \frac{J_r}{I_{yy}} p\Omega_d + \frac{l}{I_{yy}} u_3 \\ \ddot{\psi} = pq\left(\frac{I_{xx}-I_{yy}}{I_{zz}}\right) + \frac{1}{I_{zz}} u_4 \end{cases} \tag{38}$$

$$\Pi_2: \begin{cases} \ddot{x} = \left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)\frac{1}{m} u_1 \\ \ddot{y} = \left(c_\phi s_\theta s_\psi - s_\phi c_\psi\right)\frac{1}{m} u_1 \\ \ddot{z} = -g + \left(c_\phi c_\theta\right)\frac{1}{m} u_1 \end{cases} \tag{39}$$

Π_1 in (38) represents the rotation subsystem related to the dynamics of quadrotor roll motion ϕ, pitch motion θ and yaw motion ψ. Π_2 in (39) represents the position subsystem related to the dynamics of quadrotor longitude motion x, latitude motion y and altitude motion z. Hence, the control scheme advocated for the overall system is then logically divided into a rotation controller and a position controller.

In the previous section, a controller (31) has been designed to stabilize one DOF of the overall system. The coefficients k_1, k_2 are the control parameters and need to be positive to satisfy stability criteria. In a conventional backstepping method, these parameters are usually selected arbitrarily. It is also possible that the parameters are properly chosen, but it cannot be said that the optimal parameters are selected. To overcome this drawback, this paper adopts the PSO for determining the optimal value of the backstepping control parameters. The PSO is utilized off line to determine the backstepping controller parameters. The performance of the controller varies according to adjusted parameters. As aforementioned, each rotation and position subsystem is comprised of three DOFs. Thus, there are, in sum, six control parameters that need to be selected simultaneously for each subsystem.

In the present study, an integral absolute error (*IAE*) is utilized to judge the performance of the controller. *IAE* criterion is widely adopted to evaluate the dynamic performance of the control system [24]. The index *IAE* is expressed as follows:

$$IAE = \int_0^t |e(t)| \mathrm{d}t \tag{40}$$

Since the system is comprised of two subsystems, hence the vector integral absolute error for the rotation subsystem is taken as $IAE_R = [\, IAE_\phi \;\; IAE_\theta \;\; IAE_\psi]$, where the subscripts denote roll, pitch and yaw, respectively. On the other hand, the vector integral

absolute error for the position subsystem is taken as $IAE_P = [\,IAE_x\ IAE_y\ IAE_z]$, where the subscripts denote longitude, latitude and altitude, respectively.

For the rotation controller, the PSO algorithm is utilized to minimize the fitness function J_R, expressed as:

$$J_R = IAE_R \cdot W \tag{41}$$

and for the position controller, the PSO algorithm is utilized to minimize the fitness function J_P, expressed as:

$$J_P = IAE_P \cdot W \tag{42}$$

where $W = [W_1\ W_2\ W_3]^T$ is the weighting vector used to set the priority of the multiple objective performance index (MOPI) parameters and the value of "W" varies from 0 to 1. In this case, equal weights for the three objectives to be met by each controller are considered. For fitness function calculation, the time-domain simulation of the quadrotor system model is carried out for the simulation period, t. The aim is to minimize this fitness function in order to improve the system response in terms of the steady-state errors. The PSO based approach to find out the optimal set of backstepping controller parameters is shown in Fig. 3.

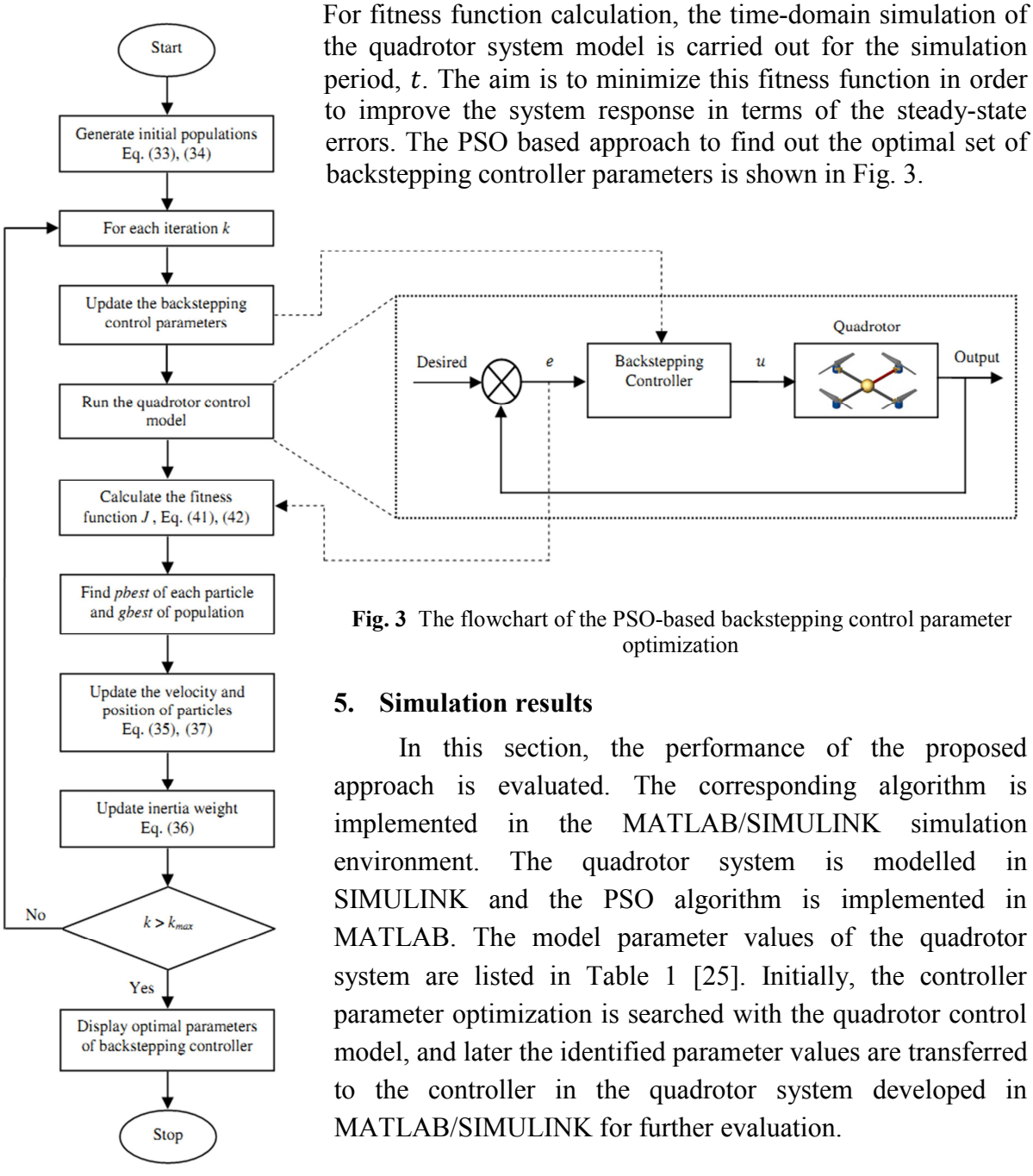

Fig. 3 The flowchart of the PSO-based backstepping control parameter optimization

5. Simulation results

In this section, the performance of the proposed approach is evaluated. The corresponding algorithm is implemented in the MATLAB/SIMULINK simulation environment. The quadrotor system is modelled in SIMULINK and the PSO algorithm is implemented in MATLAB. The model parameter values of the quadrotor system are listed in Table 1 [25]. Initially, the controller parameter optimization is searched with the quadrotor control model, and later the identified parameter values are transferred to the controller in the quadrotor system developed in MATLAB/SIMULINK for further evaluation.

Table 1 Parameters of the quadrotor

Parameter	Description	Value	Units
g	Gravity	9.81	m/s^2
m	Mass	0.5	kg
l	Distance	0.2	m
I_{xx}	Roll inertia	4.85×10^{-3}	$kg \cdot m^2$
I_{yy}	Pitch inertia	4.85×10^{-3}	$kg \cdot m^2$
I_{zz}	Yaw inertia	8.81×10^{-3}	$kg \cdot m^2$
b	Thrust factor	2.92×10^{-6}	
d	Drag factor	1.12×10^{-7}	

In this study, the following values are assigned for controller parameter optimization:

i. Dimension of the search space = 6 (i.e., $k_{i=1\ldots6}$ or $k_{i=7\ldots12}$);
ii. Population/swarm size = 15;
iii. The number of maximum iteration = 20;
iv. The self and swarm confident factor, c_1 and c_2 = 2;
v. The inertia weight factor w is set by (36), where $w_{max} = 0.9$ and $w_{min} = 0.4$;
vi. The searching ranges for the backstepping parameters are limited to $[0, 20]$;
vii. The simulation time, t is equal to 10s;
viii. Optimization process is repeated 20 times;

The finest set of values among the simulation runs is selected as the best optimized controller value. The parameter and fitness values of each particle during the simulation for the rotation and position controller are summarized in Table 2 and 3 respectively. For the rotation controller, the best fitness value is $7.312e-008$, which appeared in the iteration number 7, and the optimal parameters are $k_1 = 14.64$, $k_2 = 14.14$, $k_3 = 14.38$, $k_4 = 14.21$, $k_5 = 14.61$ and $k_6 = 14.11$. The variation of the fitness function with the number of iterations is shown in Fig. 4.

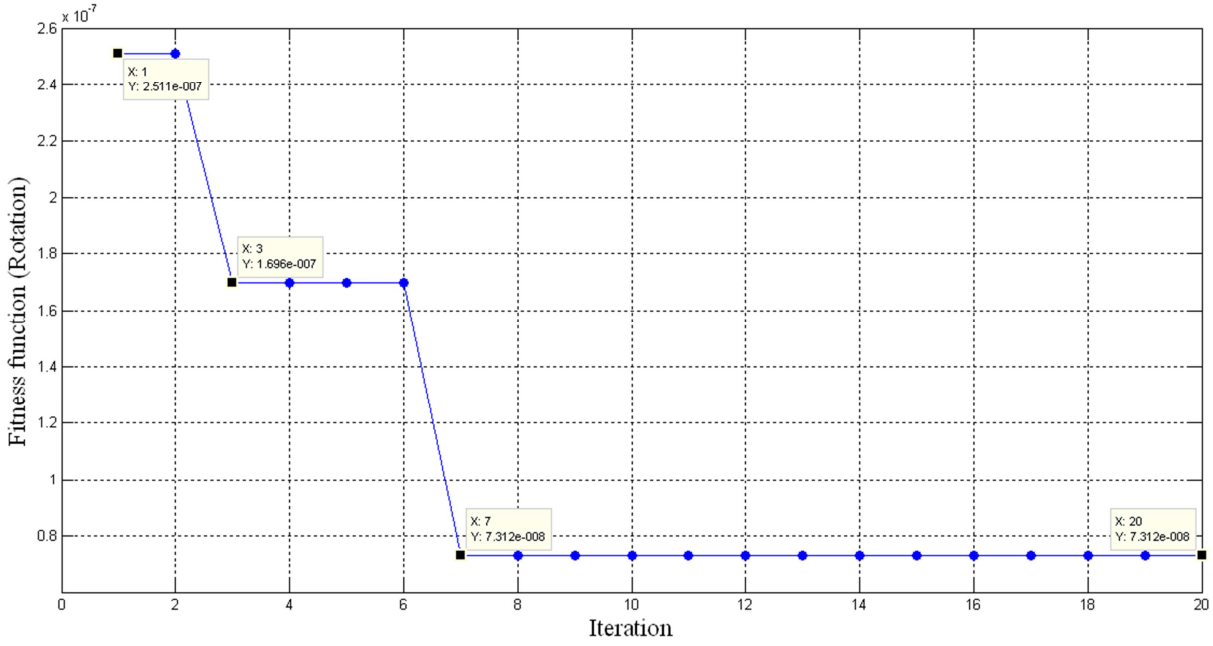

Fig. 4 The convergence of fitness function for the rotation controller with the number of iterations

On the other hand, the variations of backstepping control parameters with respect to the number of iterations are shown in Fig. 5. For the position controller, the best fitness value is 0.1543, which appeared in the iteration number 9, and the optimal parameters are $k_7 = 15.21$, $k_8 = 14.29$, $k_9 = 15.01$, $k_{10} = 14.75$, $k_{11} = 15.42$ and $k_{12} = 14.95$. The variation of the fitness function with the number of iterations is shown in Fig. 6, while the variations of backstepping control parameters with respect to the number of iterations are shown in Fig. 7. As can be seen for both control parameter optimization, throughout about 20 iterations, the PSO method can prompt convergence and obtain a good fitness value. These results show that the PSO approach can search optimal backstepping controller parameters quickly and efficiently.

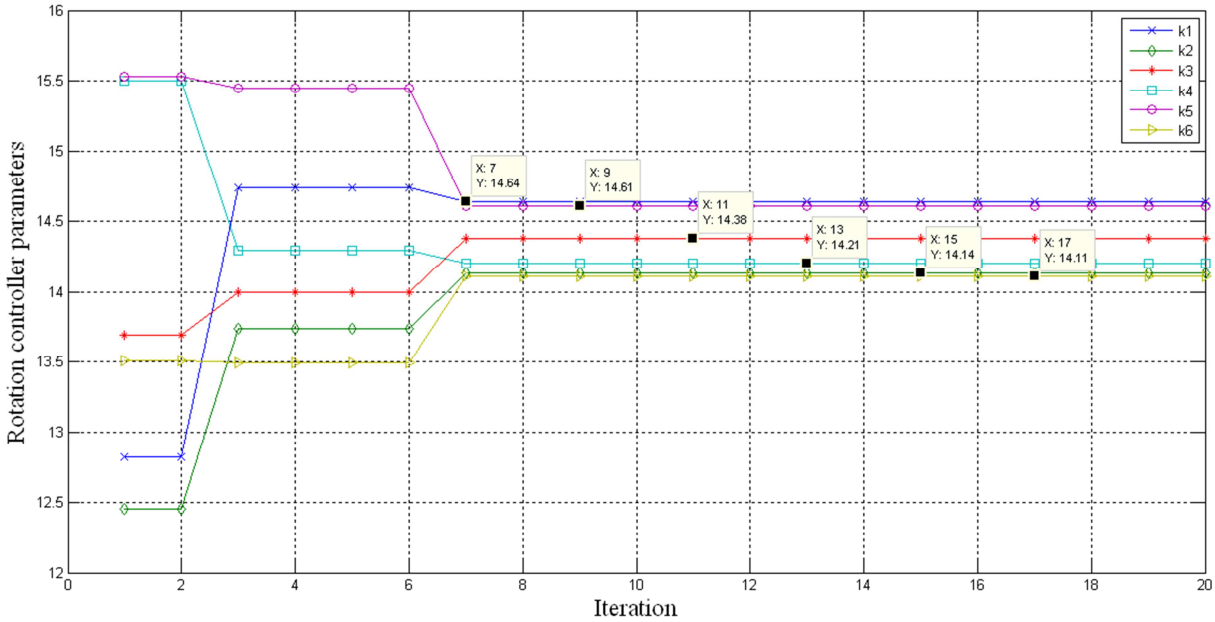

Fig. 5 The variations of rotation controller parameters versus the number of iterations

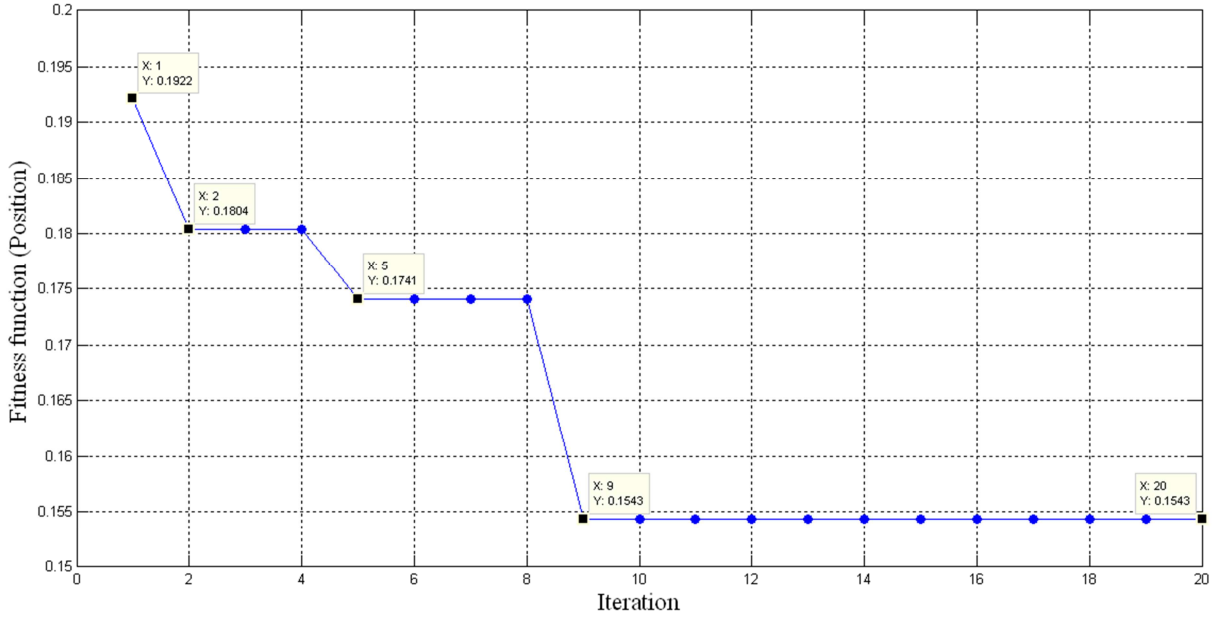

Fig. 6 The convergence of fitness function for the position controller with the number of iterations

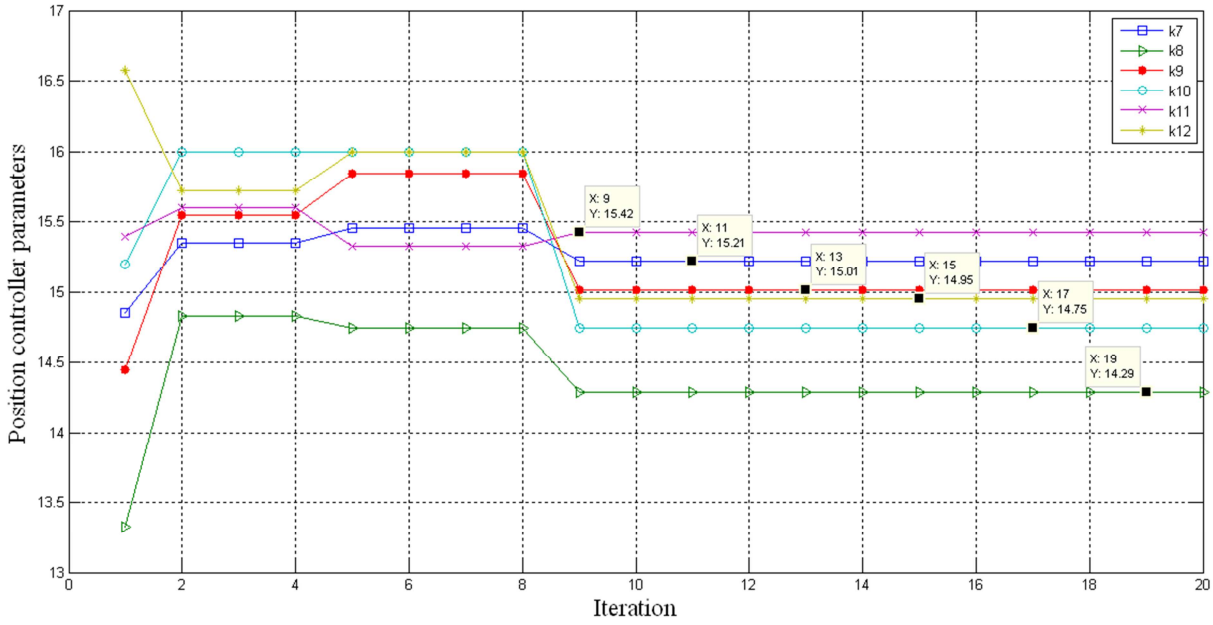

Fig. 7 The variations of position controller parameters versus the number of iterations

Table 2 The rotation controller parameters and the fitness value of each optimal particle

Iteration No.	Optimal parameters	Fitness value
1	$k_1 = 12.82, k_2 = 12.46$ $k_3 = 13.69, k_4 = 15.50$ $k_5 = 15.53, k_6 = 13.51$	$2.511e-007$
3	$k_1 = 14.74, k_2 = 13.74$ $k_3 = 14.00, k_4 = 14.29$ $k_5 = 15.44, k_6 = 13.49$	$1.696e-007$
7	$k_1 = 14.64, k_2 = 14.14$ $k_3 = 14.38, k_4 = 14.21$ $k_5 = 14.61, k_6 = 14.11$	$7.312e-008$
20	$k_1 = 14.64, k_2 = 14.14$ $k_3 = 14.38, k_4 = 14.21$ $k_5 = 14.61, k_6 = 14.11$	$7.312e-008$

Table 3 The position controller parameters and the fitness value of each optimal particle

Iteration No.	Optimal parameters	Fitness value
1	$k_7 = 14.85, \quad k_8 = 13.32$ $k_9 = 14.45, \quad k_{10} = 15.20$ $k_{11} = 15.39, k_{12} = 16.68$	$2.511e-007$
5	$k_7 = 15.45, \quad k_8 = 14.74$ $k_9 = 15.84, \quad k_{10} = 16.00$ $k_{11} = 15.33, k_{12} = 16.00$	$1.696e-007$
9	$k_7 = 15.21, \quad k_8 = 14.29$ $k_9 = 15.01, \quad k_{10} = 14.75$ $k_{11} = 15.42, k_{12} = 14.95$	$7.312e-008$
20	$k_7 = 15.21, \quad k_8 = 14.29$ $k_9 = 15.01, \quad k_{10} = 14.75$ $k_{11} = 15.42, k_{12} = 14.95$	$7.312e-008$

To explore the effectiveness of the proposed optimal backstepping controller, three simulation experiments have been performed on the quadrotor. In the first experiment, the simulation results of the proposed controller in a stabilizing problem are given. In the second, the performance of the scheme is investigated in the attitude tracking problem. Finally, the *x*-*y* position tracking problem is carried out in order to demonstrate the effectiveness of the designed controller.

5.1 Simulation experiment 1: stabilizing problem

In this simulation experiment, the control objectives are to reach and maintain the quadrotor at a certain desired altitude/attitude, such that the helicopter can hover at a fixed point. The desired altitude/attitude is given by $x_{id} = [z_d, \phi_d, \theta_d, \psi_d] = [15, 0, 0, 0]^T$. The initial states are given by $z = 0$, $\phi = 0.2$, $\theta = 0.2$ and $\psi = 0.2$. Simulation results show the control design is able to stabilize the helicopter in a hover mode. Under the proposed OBC, it can be observed that the altitude/attitude of the quadrotor can be maintained at the desired altitude/attitude, that is, the hovering flight is stable, as shown in Fig. 8. One can also note from this figure that the attitude states converge rapidly to the zero set-point for a given initial condition as the system starts, and hence the stabilization of the quadrotor system is achieved.

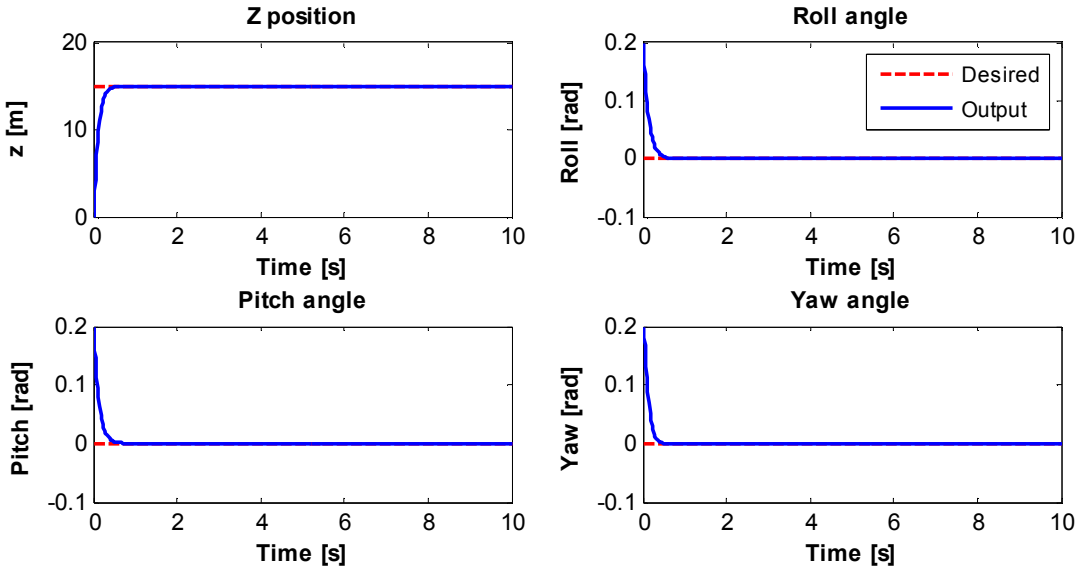

Fig. 8 Altitude/attitude of the hovering quadrotor using OBC

5.2 Simulation experiment 2: attitude tracking problem

In this simulation experiment, the performance of the proposed control approach is investigated in the attitude tracking problem of the quadrotor. The periodic sinusoidal functions are used as a reference to the attitude angles and the response is shown in Fig. 9. As it can be seen, the attitude angles track the desired reference trajectories smoothly. The results also show that the OBC gives a very small tracking error, which indicate a satisfactory tracking performance.

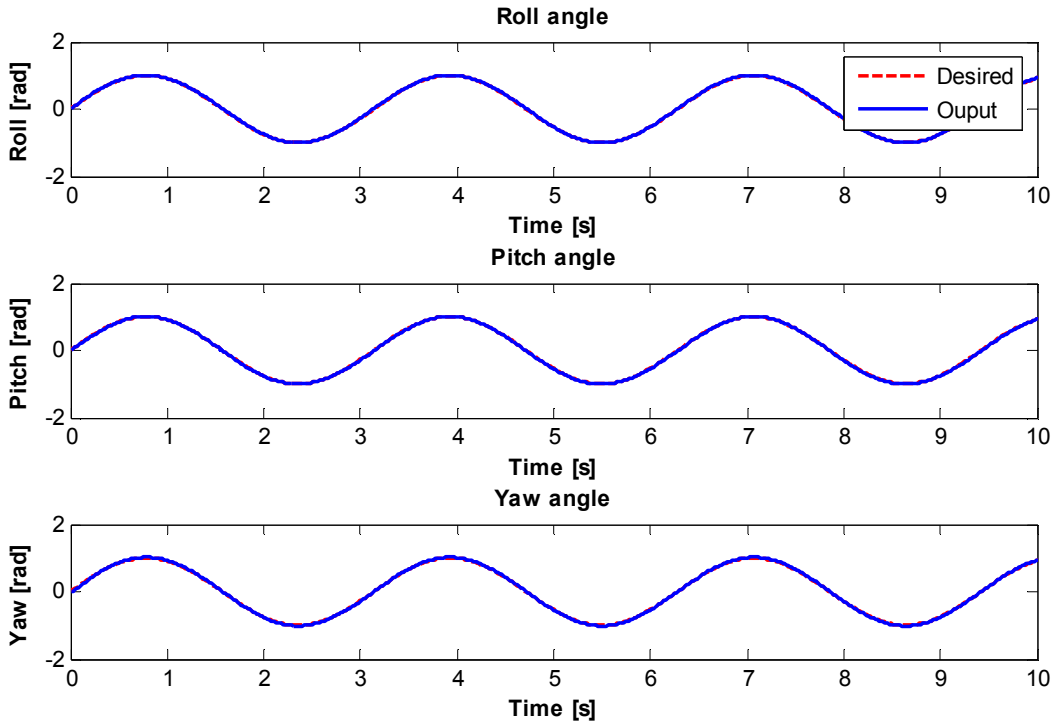

Fig. 9 Attitude tracking of the quadrotor using OBC

5.3 Simulation experiment 3: position tracking problem

To further highlight the advantage of the proposed control structure, the OBC for *x*-*y* position tracking is simulated. Considering that the turning curve manoeuver is an important practical trajectory manoeuver that the quadrotor needs to perform, the control performance of ∞-shape trajectory tracking is examined. The desired trajectory is generated using the following command:

$$\begin{cases} x_d = 5\cos\left(\frac{2\pi}{5}t\right), \\ y_d = 5\left[1 - \sin\left(\frac{\pi}{5}t\right)\right]. \end{cases} \tag{43}$$

The initial state of the quadrotor is set to be $[x_0, y_0] = [5,5]$m. The simulation results of *x*-*y* position tracking for the OBC approach are shown in Fig. 10. As it can be seen, the quadrotor can track the desired reference trajectory accurately by using the proposed control scheme.

5.4 Comparison with the improperly selected BC parameters

As mentioned before, the improper selection of the backstepping control parameters leads to inappropriate responses of the system. Results from Figs. 11-13 demonstrate that the poorly defined backstepping control parameters ($k_{i=1\ldots12} = 5$) will degrade the performance response of the system. From Fig. 11, some oscillation in the transient response can be observed. The settling time is also significantly longer than that achieved by using OBC. At the same time, the ability of the system to track the reference trajectory is also affected as shown in Figs. 12 and 13.

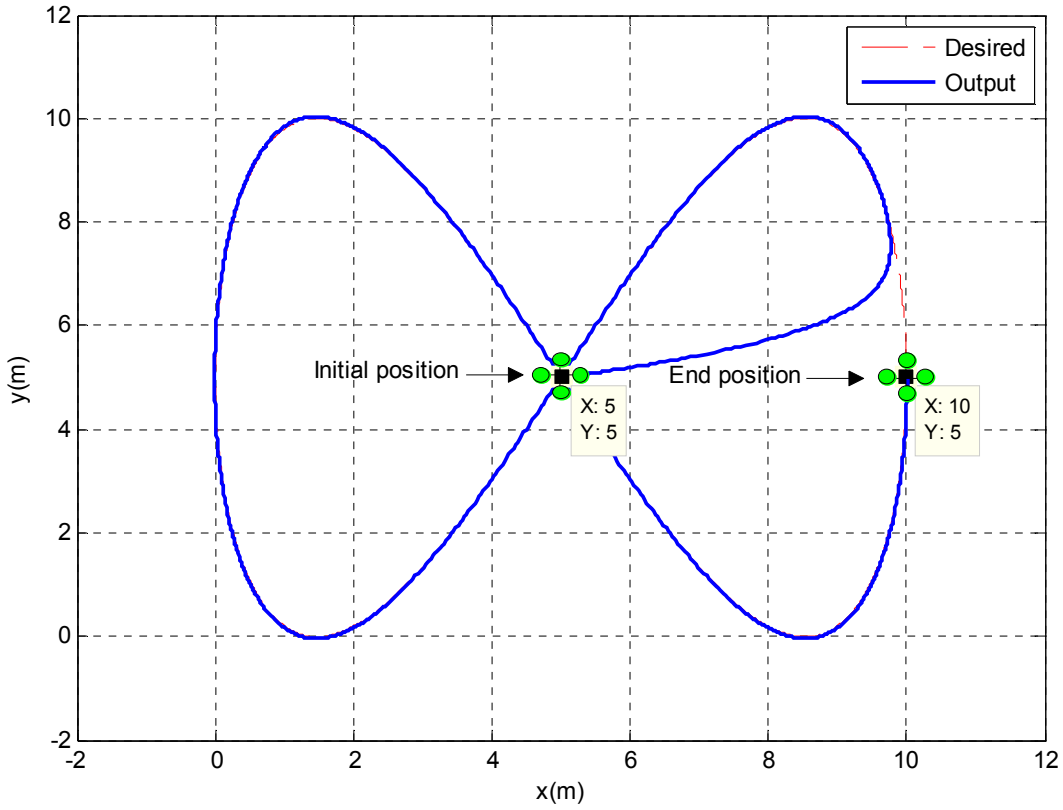

Fig. 10 *x-y* position ∞-shape trajectory tracking response using OBC

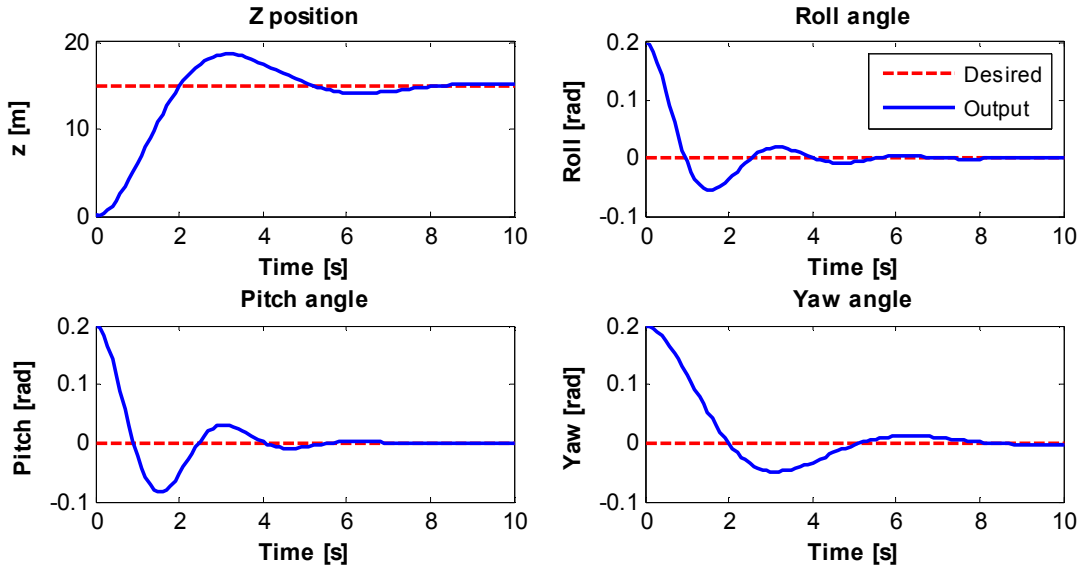

Fig. 11 Altitude/attitude of the hovering quadrotor using BC with improper parameters

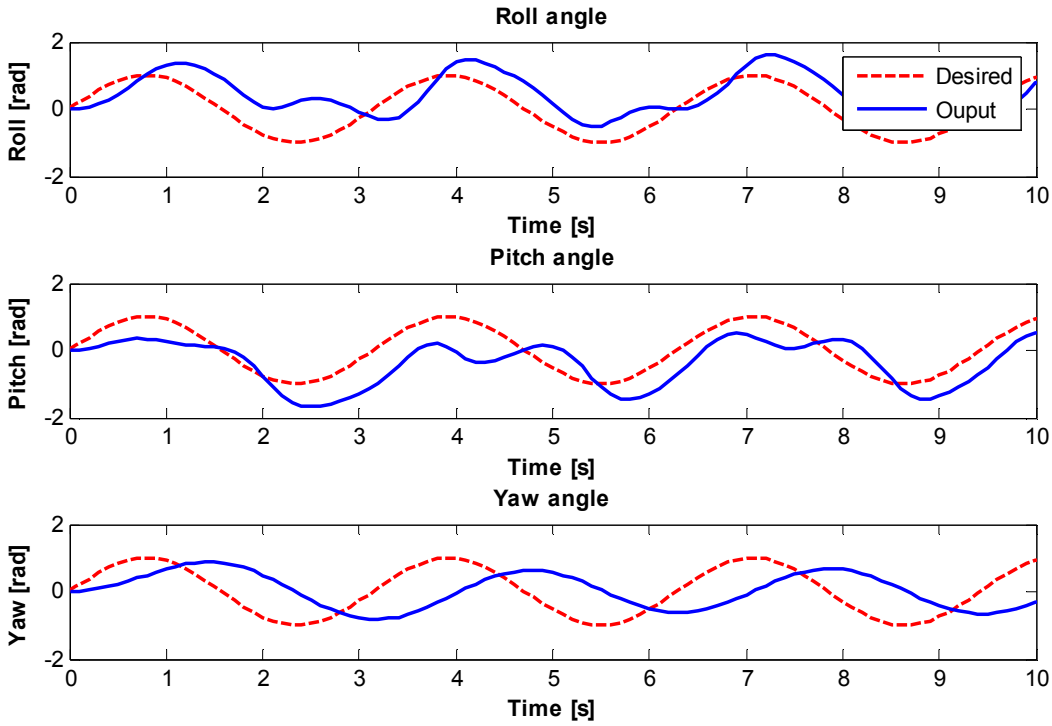

Fig. 12 Attitude tracking response due to periodic sinusoidal function using BC with improper parameters

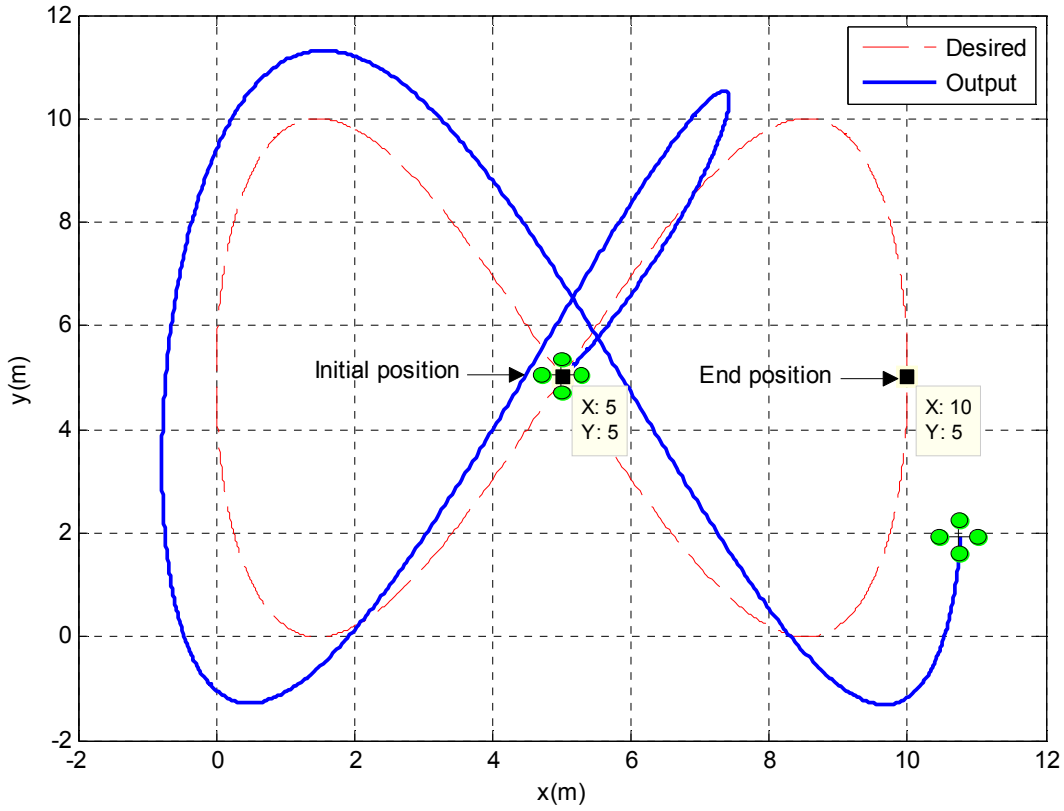

Fig. 13 *x-y* position ∞-shape trajectory tracking response using BC with improper parameters

6. Conclusions

In this paper, the application of an optimal backstepping controller for manoeuvring a quadrotor UAV is successfully demonstrated. First, a mathematical model of the quadrotor is introduced. Then, the proposed optimal backstepping controller which can automatically select the controller parameters by using the PSO algorithm is developed. The backstepping control design is derived based on the Lyapunov function so that the stability of the system can be guaranteed. Finally, the proposed control scheme is applied for the quadrotor UAV stabilization and trajectory tracking missions. Simulation results show that high-precision transient and tracking responses can be achieved by using the proposed control system.

REFERENCES

[1] Kaloust J, Ham C, Siehling J, Jongekryg E & Han Q, Nonlinear robust control design for levitation and propulsion of a maglev system, *IEE Proceedings Control Theory and Applications*, 151 (2004) 460-464.

[2] De Queiroz M S & Dawson D M, Nonlinear control of active magnetic bearings: a backstepping approach, *IEEE Transactions on Control Systems Technology*, 4 (1996) 545-552.

[3] Kim K S & Kim Y, Robust backstepping control for slew maneuver using nonlinear tracking function, *IEEE Transactions on Control Systems Technology*, 11 (2003) 822-829.

[4] Karimi A & Feliachi A, Decentralized adaptive backstepping control of electric power systems, *Electric Power Systems Research*, 78 (2008) 484-493.

[5] Wu Z J, Xie X J, Shi P & Xia Y, Backstepping controller design for a class of stochastic nonlinear systems with Markovian switching, *Automatica*, 45 (2009) 997-1004.

[6] Krstic M, Kanellakopoulos I & Kokotovic P, *Nonlinear and adaptive control design*, New York 1995.

[7] Madani T & Benallegue A, Backstepping Control for a Quadrotor Helicopter, in *Proc of IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2006) 3255-3260.

[8] Bouadi H, Bouchoucha M & Tadjine M, Modelling and Stabilizing Control Laws Design Based on Backstepping for an UAV Type-Quadrotor, in *Proc of the IFAC Symposium on IAV*, Toulouse, France, 2007.

[9] De Vries E & Subbarao K, Backstepping based nested multi-loop control laws for a quadrotor, in *Proc of 11th International Conference on Control Automation Robotics & Vision (ICARCV)*, (2010) 1911-1916.

[10] Raffo G V, Ortega M G & Rubio F R, Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle, in *Proc of American Control Conference*, (2008) 3356-3361.

[11] Regula G & Lantos B, Backstepping based control design with state estimation and path tracking to an indoor quadrotor helicopter, *Electrical Engineering and Computer Science*, 53 (2011) 151-161.

[12] Franci C, Župerl U, Control strategy for assuring constant surface finish by controlling cutting forces, *Transactions of Famena*, 37 (2013) 41-52.

[13] Rini D P, Shamsuddin S M & Yuhaniz S S, Particle swarm optimization: Technique, system and challenges. *International Journal of Computer Applications*, 14 (2011) 19–27.

[14] Gjelaj A, Balič J, Fičko M, Intelligent optimal tool selections for CNC programming of machine tools, *Transactions of Famena*, 37 (2013) 31-40.

[15] Sinha S, Patel R & Prasad R, Application of GA and PSO tuned fuzzy controller for AGC of three area thermal-thermal-hydro power system. *International Journal of Computer Theory and Engineering*, 2 (2010) 1793-8201.

[16] Lozina Z, Sedlar D, Vučina D, Model update with the observer/Kalman filter and genetic algorithm approach, *Transactions of Famena*, 36 (2012) 9-22.

[17] Sedlar D, Lozina Z, Vucina D. Comparison of genetic and bees algorithms in the finite element model update, *Transactions of Famena*, 35 (2011) 1-12.

[18] Kennedy J & Eberhart R, Particle swarm optimization, In *Proc of IEEE International Conference on Neural Networks*, (1995) 1942-1948.

[19] Zuo Z, Trajectory tracking control design with command-filtered compensation for a quadrotor, *IET Control Theory and Applications*, 4 (2010) 2343-2355.

[20] Olfati-Saber R, Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles, PhD thesis, Massachusetts Institute of Technology, 2000.

[21] Hassan R, Cohanim B, De Weck O & Venter G. A comparison of particle swarm optimization and the genetic algorithm, In *Proc of the 1st AIAA multidisciplinary design optimization specialist conference*, 2005.

[22] Lalitha M P, Reddy V V & Usha V, Optimal DG placement for minimum real power loss in radial distribution systems using PSO, *Journal of Theoretical and Applied Information Technology*, 13 (2010) 107-116.

[23] Eberhart R C & Shi Y, Comparing inertia weights and constriction factors in particle swarm optimization, In *Proc of the IEEE Congress on Evolutionary Computation*, (2000) 84-88.

[24] Allaoua B, Gasbaoui B & Mebarki B, Setting up PID DC motor speed control alteration parameters using particle swarm optimization strategy, *Leonardo Electronic Journal of Practices and Technologies*, 14 (2009) 19-32.

[25] Voos H, Nonlinear control of a quadrotor Micro-UAV using Feedback-Linearization, In *Proc of IEEE International Conference on Mechatronics (ICM)*, (2009) 1-6.

Submitted: 29.4.2014

Accepted: 17.9.2014

Mohd Ariffanan Mohd Basri
ariffanan@fke.utm.my
Kumeresan A. Danapalasingam
kumeresan@fke.utm.my
Abdul Rashid Husain
rashid@fke.utm.my
Dept. of Control and Mechatronics Eng.,
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
UTM Skudai, 81310 Johor, Malaysia.