Automatic PID Tuning via Differential Evolution for Quadrotor UAVs Trajectory Tracking

Wufan Wang, Xiaming Yuan, Jihong Zhu
Dept. of Computer Science and Technology, Tsinghua Univeresity, Beijing, China
{answerwwfthu,summersbright}@gmail.com, jhzhu@mail.tsinghua.edu.cn

***Abstract*—In this paper, a two-stage automatic proportional-integral-derivative (PID) tuning scheme based on the differential evolution (DE) algorithm is developed for the trajectory tracking control of quadrotor unmanned aerial vehicles (UAVs). Nonlinear dynamics model of the quadrotor is established, based on which the tuning scheme is conducted. In the first stage, the inner loop attitude PD controllers are tuned separately with respect to the roll, pitch and yaw channel to achieve a fast transient response. In the second stage, the outer loop position PID controllers are tuned upon the optimal attitude controllers to achieve a smooth and precise trajectory tracking performance. In order to deal with coupled dynamics between the *x/y* channel and the altitude channel, time domain performance indexes of *x/y* and altitude channels are both incorporated in the cost function to be minimized. An adaptive mutation operator is utilized in the DE algorithm to maintain the population diversity in the early phase of the algorithm and accelerate convergence in the later phase. Several maneuverings including take-off, smooth translation, circular and spiral climb motions are carried out to evaluate the effectiveness of the proposed scheme.**

I. Introduction

Quadrotor unmanned aerial vehicles (UAVs) have attracted great interest of researchers and developers in recent years. Owing to their distinct characteristics such as agility and small size, quadrotor UAVs can find various applications including inspection of power lines, atmospheric analysis for weather forecasts, traffic surveillance and military reconnaissance [1]. However, controlling a quadrotor is not an easy task because of its non-linearity, coupled dynamics and commonly under-actuated design configuration, which has led to several control algorithms proposed in the literature. The majority of those control algorithms focused on the stabilization problem which is the first step towards successfully autonomous flights. Some of them also tackle with position maintaining or velocity holding in order to fulfill certain manoeuvrings or trajectory tracking. For instance, a proportional-integral-derivative (PID) controller and dynamic surface control (DSC) are used respectively for the attitude control and altitude control of a quadrotor in [2]. A linear quadratic regulator (LQR) controller is developed in [3] to achieve trajectory tracking while a nonlinear control system based on state-dependent Riccati equations (SDRE) is presented in [4]. Both backstepping and sliding-mode techniques are utilized in the quadrotor controller design in [5] with their performance comparison presented. Online iterative learning control (ILC) methods, which are known to be powerful for tasks performed repeatedly, are developed in [6] for trajectory tracking control of UAVs. Several other techniques including adaptive control [7], robust control [8] and fuzzy logic control [9] have also been applied to quadrotor UAVs.

Among all the control methods mentioned above, PID controllers, which are simple in architecture and widely used in the process industry, have been shown to be robust, reliable, efficient and cost effective for most applications. However, sometimes their effectiveness can not be fully explored due to poor tuning. In fact, tuning PID parameters can be a very time-consuming and sometimes difficult task due to special properties of process models. To address this difficulty, much effort has been invested in developing effective PID tuning methods. Ziegler-Nichols (Z-N) [10] and Cohen-Coon (C-C) [11] are two classic approaches which have been used for years especially when little information about the process model under control is provided. Although they can guarantee the stability and robustness of a system, the gains are not always optimal since PID parameters are obtained for an operation point where the model can be considered linear. Iterative learning approach is used in [12] for the linear process to obtain optimal PID parameters whenever the same control task is repeated. Recently, intelligent optimization algorithms are extensively studied and have been applied to the automatic tuning of PID controllers. The genetic algorithm (GA) is used in [13] to tune an optimal PID controller for nonlinear process models while the artificial bee colony (ABC) algorithm is adopted in [14] to carry out online PID gains optimization for quadrotors.

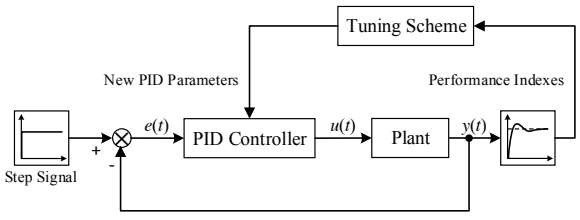


Fig. 1. Automatic PID tuning architecture

In this paper we will investigate the automatic PID tuning problem for the trajectory tracking control of quadrotor UAVs. A common architecture of automatic PID tuning based on the step-response performance of the closed-loop system is illustrated in Figure 1. Finding optimal gains for PID con-

trollers of the quadrotor is challenging due to its inherent complex characteristics and the nonlinear mapping between controller gains and performance indexes. In order to address the problem and explore the full potential of PID controllers and achieve high tracking performance, a two-stage optimal PID tuning scheme based on the differential evolution (DE) algorithm [15] is presented. The rest of this paper is organized as follows. Description of the differential evolution algorithm used in this problem and performance indexes of PID controllers widely used in the literature are outlined in Section II. Section III establishes a mathematical model for the quadrotor based on which a detailed description of the proposed tuning scheme as well as tuning results are presented in Section IV. Trajectory tracking simulations are carried out in Section V and this paper is concluded in Section VI.

II. Foundamentals

A. Differential Evolution

The differential evolution algorithm is an effective heuristic approach for minimizing nonlinear and non-differentiable continuous space functions [15]. It involves maintaining a population of candidate solutions subjected to iterations of recombination, evaluation and selection, which is very similar to the genetic algorithm [16]. The recombination process involves the creation of new candidate solution components by adding the weighted difference between two randomly selected population members to a third population member. As a consequence, population members are perturbed relative to the spread of the broader population. In conjunction with selection, the perturbation effect self-organizes the sampling of the problem space, gradually bounding it to known areas of interest as the population evolves. The main procedure of the differential evolution algorithm adopted in this paper is depicted in Algorithm 1 while the sampling procedure of a new candidate solution is described in Algorithm 2.

Note that in the pseudocode of differential algorithm, an adaptive mutation operator is adopted to keep the weight factor gradually changing from $2 \times Weight_{init}$ to $Weight_{init}$ as the population evolves. This trick helps to maintain the population diversity in the early phase of population evolution and accelerate algorithm convergence in the later phase.

B. Performance Index

Similar to any other controllers, the objective of PID controllers is to provide system stability as well as reference tracking. Several indexes have been proposed to evaluate the performance of a controller, among which the most common ones are the integrated absolute error (IAE), integrated squared error (ISE), integrated time squared error (ITSE), and integrated time absolute error (ITAE), which are normally calculated under step input in the time domain as:

$$\begin{aligned} IAE &= \int_0^t |e(t)|dt \quad ITAE = \int_0^t t|e(t)|dt \\ ISE &= \int_0^t e^2(t)dt \quad ITSE = \int_0^t te^2(t)dt \end{aligned} \tag{1}$$

Algorithm 1 Differential Evolution

Input: $Problem_{size}$, $Population_{size}$, $Weight_{init}$, P_{init}, G_{max}
Output: P_{best}
Initialize:
$Population \leftarrow$ InitPopulation($Problem_{size}$, $Population_{size}$)
EvaluatePopulation($Population$)
$P_{best} \leftarrow$ GetBestSolution($Population$)
$G \leftarrow 1$
while $G \leq G_{max}$ **do**
  $NewPopulation \leftarrow \emptyset$
  $operator \leftarrow e^{\frac{1-G}{G_{max}+1-G}}$
  $Weight_{factor} \leftarrow Weight_{init} \times 2^{operator}$
  for $i = 1$ to $Population_{size}$ **do**
    $P^{(i)} \leftarrow Population(i)$
    $S^{(i)} \leftarrow$ NewSample($P^{(i)}$, $Population$, $Problem_{size}$, $Population_{size}$, $Weight_{factor}$, $P_{crossover}$)
    if Cost($S^{(i)}$) $\leq$ Cost($P^{(i)}$) **then**
      $NewPopulation(i) \leftarrow S^{(i)}$
    else
      $NewPopulation(i) \leftarrow P^{(i)}$
    end if
  end for
  $Population \leftarrow NewPopulation$
  EvaluatePopulation($Population$)
  $P_{best} \leftarrow$ GetBestSolution($Population$)
  $G \leftarrow G + 1$
end while
Return P_{best}

Algorithm 2 NewSample

Input: $P^{(i)}$, $Population$, $Problem_{size}$, $Population_{size}$, $Weight_{factor}$, $P_{crossover}$
Output: $S^{(i)}$
Initialize:
$targetPop \leftarrow \{Population\} - \{P^{(i)}\}$
$perm \leftarrow$ randPerm($Population_{size} - 1$)
$tempSample \leftarrow targetPop(perm(1)) + Weight_{factor} \times (targetPop(perm(2)) - targetPop(perm(3)))$
$dimPerm \leftarrow$ randPerm($Problem_{size}$)
for $j = 1$ to $Problem_{size}$ **do**
  if exceedBound($tempSample(j)$) **then**
    $tempSample(j) \leftarrow$ randGenerate()
  end if
  if rand() $> P_{crossover}$ *and* $dimPerm(1) \neq j$ **then**
    $S^{(i)}(j) \leftarrow P^{(i)}(j)$
  else
    $S^{(i)}(j) \leftarrow tempSample(j)$
  end if
end for
Return $S^{(i)}$

Besides, other indexes such as overshoot, settling time and rise time in the time domain or bandwidth, damping ratio and undamped natural frequency in the frequency domain are also frequently used to evaluate the performance of a controller.

The goal of our automatic tuning scheme is to find PID gains with the best performance defined by one or a weighted combination of proper indexes via the differential evolution algorithm. While weights and number of criteria are diversely reported in the literature, it is generally accepted that time weighted indexes are more appropriate as the error arising early should be penalized less than the error occurring later in the transient response. In this paper, we use a weighted combination of ITAE, maximum overshoot, settling time and amplitudes of gains as the cost function to search for the optimal PID parameters. Detailed description with regard to the selection of performance indexes will be discussed later.

III. Mathematical Model

A non-linear model of the quadrotor is established in this section to carry out the proposed automatic PID tuning scheme. Figure 2 illustrates the coordinate system and configuration of rotors used in this paper.

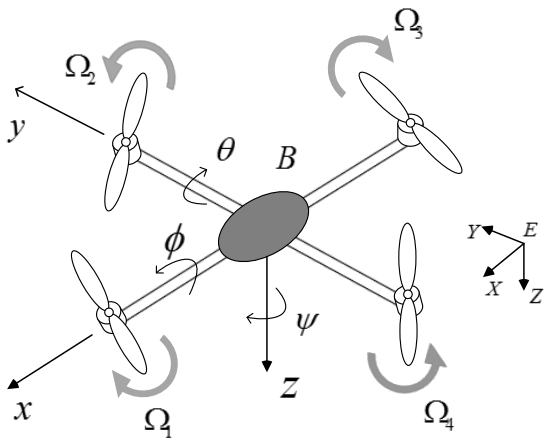


Fig. 2. Coordinate system of a quadrotor

By applying the Newton-Euler formalism [5], the dynamics model of the quadrotor can be expressed as follows

$$\begin{aligned}
\dot{p} &= rq(\frac{I_y - I_z}{I_x}) + \frac{J_r}{I_x}\Omega q + \frac{l}{I_x}U_2 \\
\dot{q} &= pr(\frac{I_z - I_x}{I_y}) - \frac{J_r}{I_y}\Omega p + \frac{l}{I_y}U_3 \\
\dot{r} &= pq(\frac{I_x - I_y}{I_z}) + \frac{l}{I_z}U_4 \\
\dot{u} &= rv - qw - gsin\theta \\
\dot{v} &= pw - ru + gsin\phi cos\theta \\
\dot{w} &= \frac{qu - pv + gcos\theta\cos\phi - U_1}{m}
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
\Omega &= \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3 \\
U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U_2 &= b(-\Omega_2^2 + \Omega_4^2) \\
U_3 &= b(\Omega_1^2 - \Omega_3^2) \\
U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)
\end{aligned} \tag{3}$$

The kinematics model is formulated as

$$\begin{aligned}
\dot{\phi} &= p + (rcos\phi + qsin\phi)tan\theta \\
\dot{\theta} &= qcos\phi - rsin\phi) \\
\dot{\psi} &= \frac{rcos\phi + qsin\phi}{cos\theta} \\
\dot{x} &= w(sin\phi sin\psi + cos\phi cos\psi sin\theta) \\
&\quad - v(cos\phi sin\psi - cos\psi sin\phi sin\theta) + ucos\phi cos\theta \\
\dot{y} &= v(cos\phi cos\psi + sin\phi sin\psi sin\theta) \\
&\quad - w(cos\psi sin\phi - cos\phi sin\psi sin\theta) + usin\psi cos\theta \\
\dot{z} &= w(sin\phi sin\psi + cos\phi cos\psi sin\theta) \\
&\quad - v(cos\phi sin\psi - cos\psi sin\phi sin\theta) + ucos\phi cos\theta
\end{aligned} \tag{4}$$

In above equations, ϕ, θ and ψ are roll, pitch and yaw angles respectively. p, q and r are angular velocities. x, y and z are positions with respect to the x, y and z axes of the earth-fixed coordinate while u, v and w are velocity components in the body-fixed coordinate. I_x, I_y, I_z represent the mass moments of inertia. J_r is the rotor inertia while l is the length of the rotor arm from the origin of the coordinate system. In Eq. (2), b and d are thrust and drag coefficients respectively. $\Omega_1 \sim \Omega_4$ are the angular velocities of four rotors. A simulation environment, which serves as the basis of the proposed tuning scheme, is established upon the nonlinear dynamics model Eq. (2-4). It should be noted that since the quadrotor structure is axisymmetric, attitude controllers in θ and ϕ channels as well as position controllers in x and y channels share the same controller gains.

IV. Two-stage Tuning Scheme

In this section, the two-stage automatic PID tuning scheme based on the differential evolution algorithm is discussed in detail. The first part of this section concerns with the inner loop attitude PD controller tuning while the second part focuses on the outer loop position PID controller tuning, both of which are of great importance to achieve high performance trajectory tracking. Tuning results with respect to both inner and outer loop controllers are illustrated in the third part. The control architecture for the quadrotor is illustrated in Figure 3 where the inner and outer loop are involved with attitude and position controllers respectively.

A. Inner Loop PD Tuning

The goal of the inner loop controller tuning stage is to find a set of optimal PD gains so that the quadrotor can achieve fast transient response with regard to angular commands. In our tuning scheme, PD controllers of the roll, pitch and yaw channels are tuned separately using the step angular command.

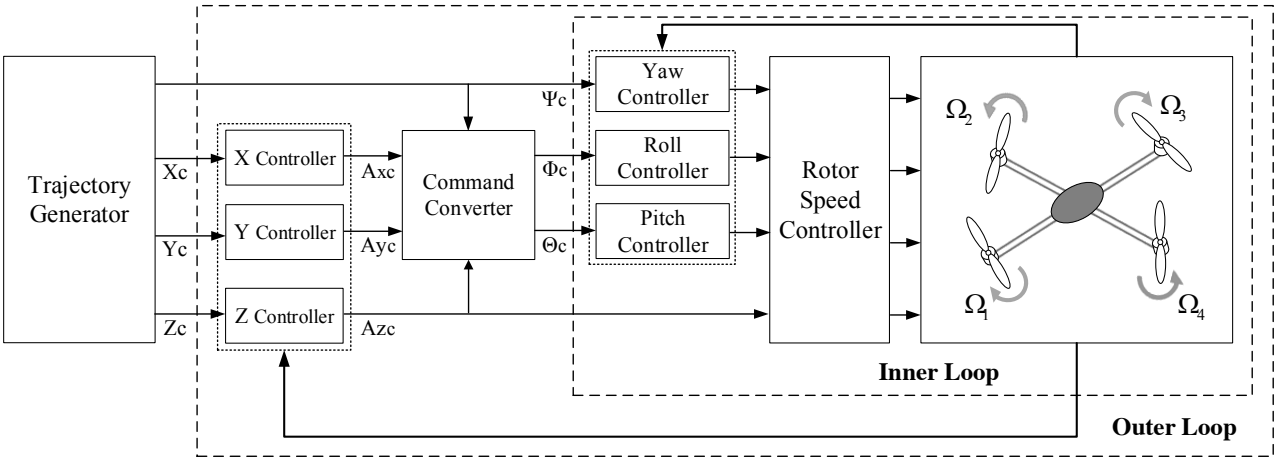


Fig. 3. Control architecture of the quadrotor

Performance indexes applied in the inner loop tuning involves IAE, overshoot OS, settling time t_s and amplitudes of gains K_p, K_d. The cost functions to be optimized regarding pitch/roll and yaw channels are denoted separately as

$$Cost_{\theta/\phi} = 10 \times IAE + \frac{t_s}{2} + OS + \frac{(K_p + K_d)}{200} \tag{5}$$

$$Cost_{\psi} = IAE + \frac{t_s}{2} + 10 \times OS + \frac{(K_p + K_d)}{10} \tag{6}$$

The fitness function is calculated as follows

$$Fitness = \frac{1}{(1 + Cost)} \tag{7}$$

During the automatic tuning process, a step angular command is first fed into a specific inner loop channel. The angular response of the quadrotor is then measured and used to calculate performance indexes, based on which the cost function and fitness value are calculated. Afterwards, the differential algorithm searches in the solution space and drives the population towards optimal PD gains based on their fitness values. The automatic tuning procedure with respect to the pitch channel is shown in Figure 4.

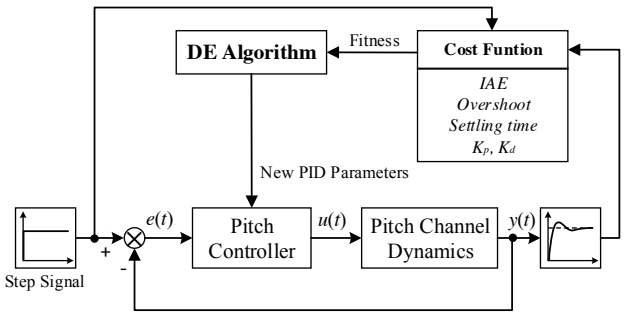


Fig. 4. Tuning procedure of the pitch channel

B. Outer Loop PID Tuning

Procedure of the outer loop PID tuning is very alike to that of the inner loop. In this stage, the PID controller of altitude channel is first tuned with the following cost function

$$Cost_z = IAE + \frac{t_s}{5} + 5 \times OS \tag{8}$$

Once having obtained optimal gains of attitude PD controllers and the altitude controller, x and y position PID controllers are tuned using the step position command to achieve a smooth and precise trajectory tracking performance. The cost

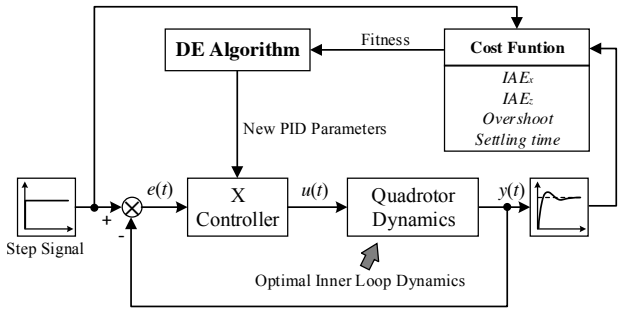
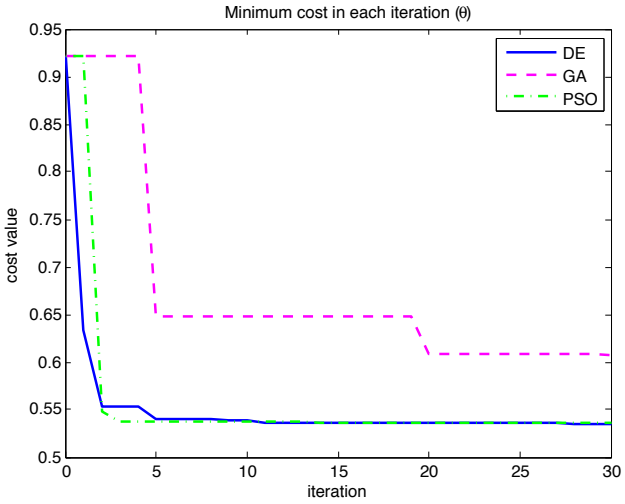


Fig. 5. Tuning procedure of the x channel

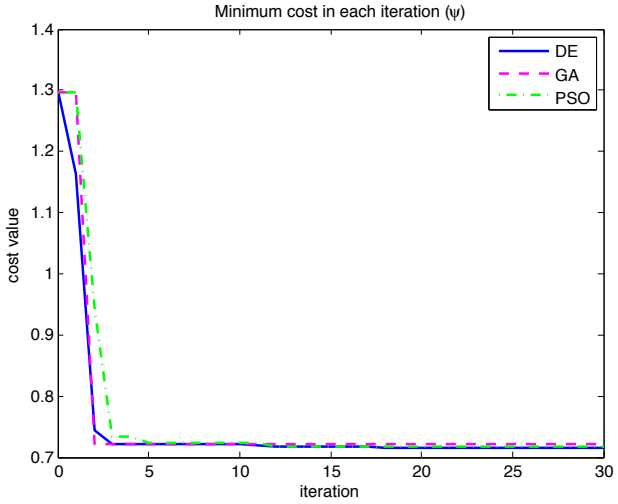
function for tuning x and y PID controllers is calculated as

$$Cost_{x/y} = 5 \times IAE + 10 \times IAE_z + \frac{t_s}{5} + \frac{OS}{5} \tag{9}$$

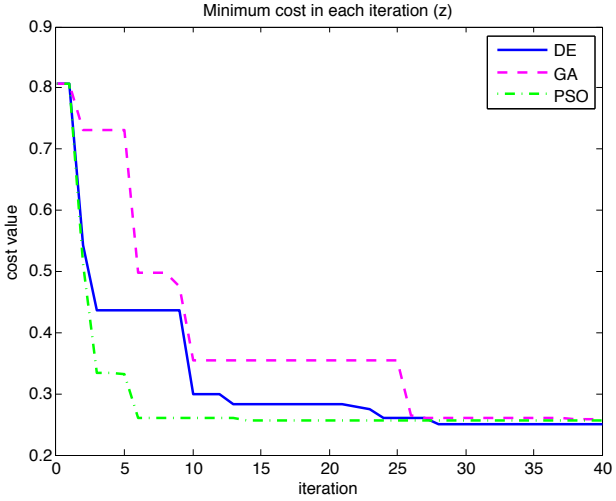
where IAE_z is the integrated absolute error of the z channel. This term is added to deal with the coupled dynamics of the quadrotor between the x and z channel. The fitness function in the outer loop PID tuning stage remains the same as that in the first stage. The tuning process of the x channel is presented in Figure 5. Note that the inner loop dynamics in the second stage is optimal since the attitude PD controllers have been assigned with the optimal gains.



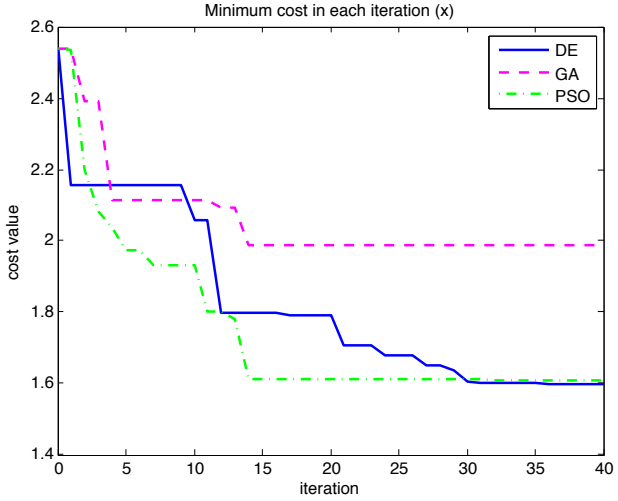
(a) Cost values for tuning θ



(b) Cost values for tuning ψ



(c) Cost values for tuning z



(d) Cost values for tuning x

Fig. 6. Tuning results for inner loop and outer loop controllers

C. Tuning Results

Amplitudes of the step commands in the first and second tuning stage are $\frac{\pi}{4} rad$ and $1m$, respectively. A population size of 20 and a maximum generation of 30 are used for the inner loop PD controllers tuning while a larger population size of 30 and a maximum generation of 40 are used for the outer loop PID controllers tuning. For comparison, two other algorithms including the genetic algorithm and the particle swarm optimization are also implemented. The searching space of three algorithms is assigned as $[0.01, 20]$ for the inner loop and $[0.01, 30]$ for the outer loop. All three algorithms are given with the same initial population in each tuning process. The best fitness values of three algorithms in each iteration are illustrated in Figure 6 while detailed tuning results are presented in Table I-IV. From the figure, we can observe that optimal controller gains obtained by the differential evolution algorithm can achieve smaller cost values than that obtained by the genetic algorithm and particle swarm optimization for all channels. The results clearly show the superiority of the differential evolution algorithm over the other two algorithms with respect to this specific problem. We can also see from the tables that even though optimal gains obtained by the differential evolution algorithm can achieve the minimum cost, performance indexes corresponding to these gains are not necessarily the best. That is, we should always make a trade-off among different performance indexes based on our goals.

V. Simulation

In this section, several maneuverings including take-off, smooth translation, circular and spiral climb motions in three

TABLE I
TUNING RESULTS FOR THE PITCH/ROLL CONTROLLER

Method	K_p	K_d	IAE	$OS(\%)$	$t_s(s)$
DE	18.016	11.78	0.015	0.03	0.477
GA	18.315	10.894	0.024	1.93	0.399
PSO	19.254	12.09	0.014	0.05	0.47

TABLE II
TUNING RESULTS FOR THE YAW CONTROLLER

Method	K_p	K_d	IAE	$OS(\%)$	$t_s(s)$
DE	16.964	7.738	0.049	0.26	0.788
GA	16.904	7.845	0.05	0.09	0.829
PSO	18.272	8.105	0.047	0.15	0.784

TABLE III
TUNING RESULTS FOR THE ALTITUDE CONTROLLER

Method	K_p	K_i	K_d	IAE	$OS(\%)$	$t_s(s)$
DE	29.932	0.002	9.19	0.073	0.4	0.779
GA	29.356	0.014	9.07	0.08	0.45	0.78
PSO	28.48	0.002	8.928	0.075	0.46	0.789

TABLE IV
TUNING RESULT FOR THE X/Y POSITION CONTROLLER

Method	K_p	K_i	K_d	IAE	$IAEz$	$OS(\%)$	$t_s(s)$
DE	11.39	0.0008	4.7889	0.21	0.034	0.67	1.054
GA	29.118	0.016	9.963	0.205	0.069	0.05	1.396
PSO	10.515	0.0007	4.4941	0.216	0.031	1.33	1.058

dimensions are carried out to evaluate the performance of automatically tuned controllers. To ensure a smooth trajectory that is continuous in positions, velocities and accelerations [17], the following polynomial is used in each maneuvering case

$$r(t) = 10 \times (\frac{t}{T})^3 - 15 \times (\frac{t}{T})^4 + 6 \times (\frac{t}{T})^5, t \in [0, T] \quad (10)$$

with the boundary conditions given as

$$\begin{aligned} r(0) &= 0, r(T) = 1 \\ \dot{r}(0) &= \dot{r}(T) = \ddot{r}(0) = \ddot{r}(T) = 0 \end{aligned} \quad (11)$$

A. Take-Off Control

In the take-off simulation, the quadrotor is commanded to take off smoothly and finally hover at a certain desired height with certain time constraint. As a consequence, performance of the altitude PID controller can be assessed during the process. The desired take-off trajectory is designed as follows.

$$\begin{aligned} Z_c(t) &= h_d \times r(t), \\ X_c(t) &= 0, \\ Y_c(t) &= 0, t \in [0, T] \end{aligned} \quad (12)$$

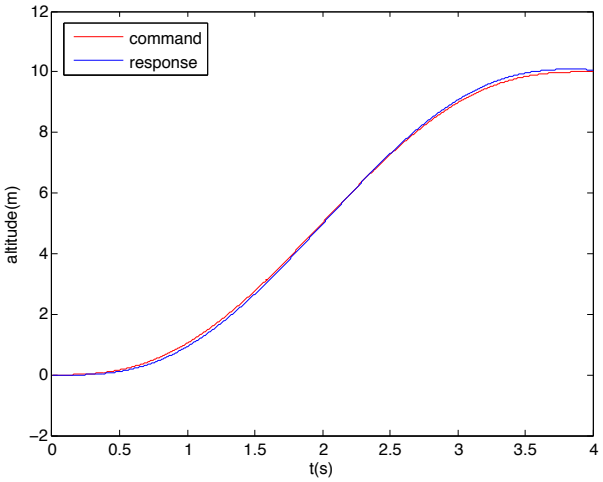


Fig. 7. Altitude tracking in the take-off process

where the hover height h_d is set to $10m$ and the time interval is chosen as $T = 4s$. The simulation result is illustrated in Figure 7, from which we can notice the tracking performance is rather satisfying despite of the small error occurred at the end of the take-off process.

B. Smooth Translation Control

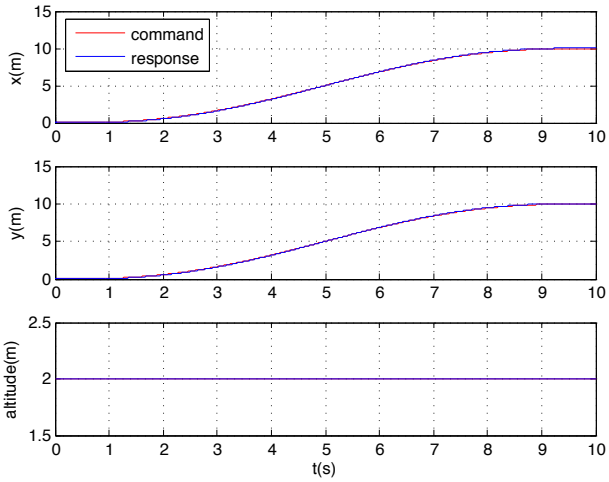


Fig. 8. Position tracking in three channels (slow translation motion)

In order to evaluate the maneuverability of the quadrotor in the x-y plane, a smooth translation trajectory is designed as follows

$$\begin{aligned} X_c(t) &= C \times r(t), \\ Y_c(t) &= C \times r(t), \\ Z_c(t) &= 2, t \in [0, T] \end{aligned} \quad (13)$$

Two simulations including a slow translation motion with $C = 10m$, $T = 10s$ and a fast translation motion with $C = 50m$, $T = 10s$ are conducted separately. The quadrotor is assumed to hover at a constant altitude of $2m$ when the simulation begins and maintain this altitude until the end of simulation.

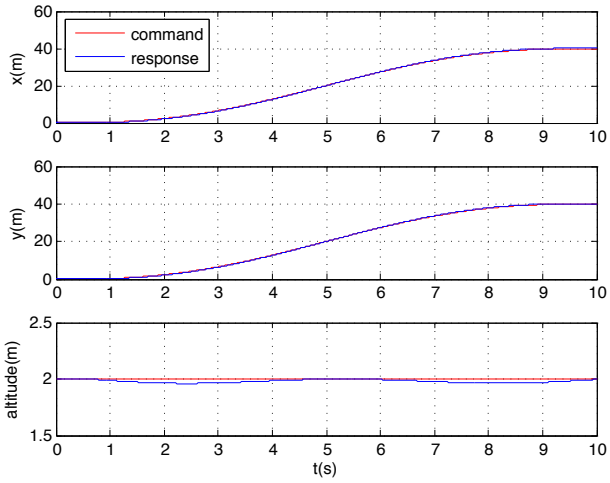


Fig. 9. Position tracking in three channels (fast translation motion)

Simulation results can be seen in Figure 8 and Figure 9. Through comparison of two figures, we can observe that tracking performance with regard to x and y channels in two cases are satisfying. However, tracking performance of the altitude channel in the fast translation is a little bit worse than that in the slow translation, which is caused by highly coupled dynamics of the quadrotor between the x/y channel and the altitude channel in drastic maneuverings. It should be noted that coupled dynamics between the x/y channel and the altitude channel is the inherit characteristic of the quadrotor that can only be impaired but can not be eliminated. The tracking error of the altitude channel in the fast translation is relatively small and thus acceptable.

C. Circular Motion Control

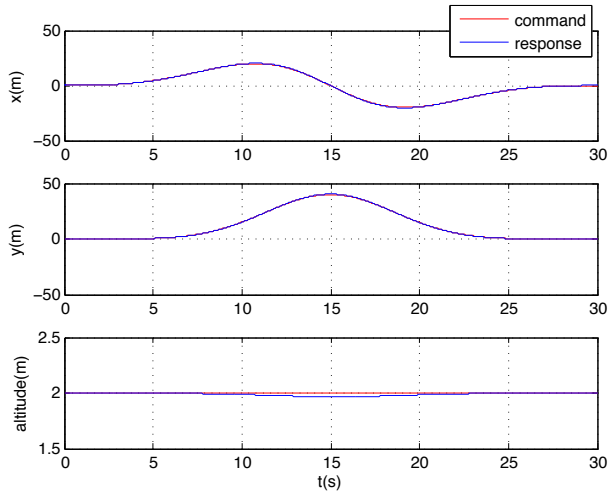


Fig. 10. Position tracking in three channels (circular motion)

In the circular motion, the quadrotor rotates around a central point by adjusting its attitude and the yaw angle remains zero during the whole process. Similar to the smooth translation mentioned above, the quadrotor is assumed to hover at a constant altitude of $2m$ when the simulation begins and maintain this altitude all along the simulation. The circular trajectory is calculated as

$$\begin{aligned} X_c(t) &= R \times sin(2\pi \times r(t)), \\ Y_c(t) &= R \times cos(2\pi \times r(t) + \pi) + R, \\ Z_c(t) &= 2, t \in [0, T] \end{aligned} \tag{14}$$

where the circular radius $R = 20m$ and the simulation duration $T = 30s$. Result of this simulation is presented in Figure 10, from which we can see that high tracking performance has been achieved for all three position channels.

D. Spiral Climb Control

Different from that in the circular motion, the yaw command in the spiral climb process changes dynamically to ensure the nose of the quadrotor pointing to the rotation center at anytime. As a result, gains of both the inner and outer loop controllers are evaluated. The spiral climb trajectory is designed as

$$\begin{aligned} X_c(t) &= R \times sin(\frac{\pi}{10} \times t), \\ Y_c(t) &= R \times cos(\frac{\pi}{10} \times t), \\ Z_c(t) &= 10 \times r(t), \\ \psi(t) &= \frac{\pi}{10} \times t, t \in [0, T] \end{aligned} \tag{15}$$

where the turning radius R is set as $10m$ and a simulation duration of $T = 40s$ is chosen. Quadrotor trajectory in the spiral climb motion and tracking results with regard to three position channels and the yaw channel are illustrated in Figure 11 and Figure 12, respectively. Through the simulation results we can see that the quadrotor spirals up around a specific axis for two rounds with regard to the reference inputs with almost no error. Since the range of the yaw angle is $[0, 2\pi]$, a discontinuous point appears in the yaw channel when the quadrotor finishes one circle as displayed in Figure 12.

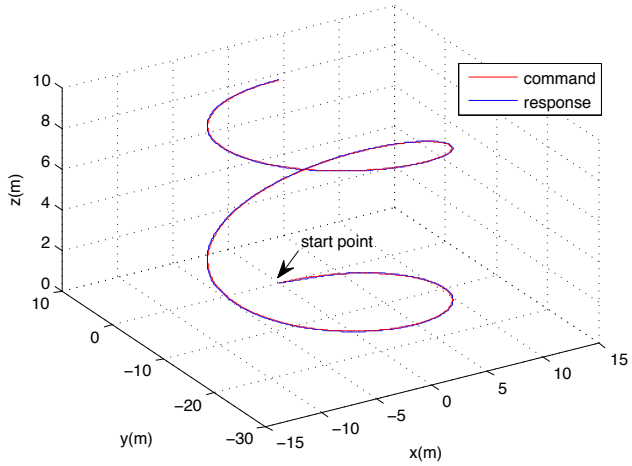


Fig. 11. Quadrotor trajectory in spiral climb

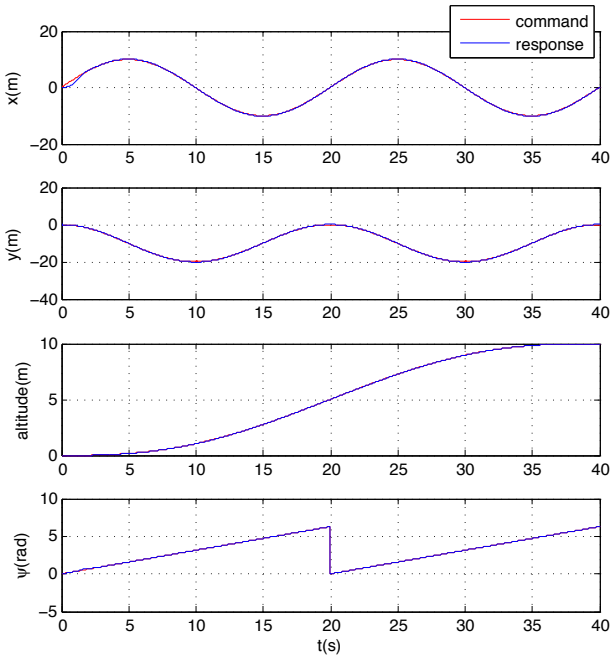


Fig. 12. Position and yaw tracking (spiral climb motion)

VI. Conclusion

In this paper, we have investigated the problem of automatic PID controller tuning for the trajectory tracking control of quadrotors. Our contributions are twofold. First, a two-stage automatic tuning scheme which optimizes PID parameters as design variables in a multi-objective manner via the differential evolution algorithm is proposed. Second, in order to deal with coupled dynamics between the x/y channel and the altitude channel, IAEs of both x/y and altitude channels are incorporated in the cost function to be minimized. Nonlinear dynamics model of the quadrotor is also established, based on which tuning performance of the differential evolution algorithm is compared with that of the genetic algorithm and the particle swarming optimization. Comparison results show the superiority of differential evolution over the other two algorithms. The effectiveness of the proposed tuning scheme is verified through different trajectory tracking simulations.

References

[1] Zulu. A, John. S. A Review of Control Algorithms for Autonomous Quadrotors. *Open Journal of Applied Sciences*, vol. 4, pp. 547-556, 2014.

[2] K.U. Lee, H.S. Kim, J.B. Park, et al. Hovering Control of a Quadrotor. *International conference on Control, Automation and Systems (ICCAS)*, pp. 162-167, 2012.

[3] Cowling I D, Yakimenko O A, Whidborne J F, et al. A Prototype of an Autonomous Controller for a Quadrotor UAV Control Conference. *European Control Conference (ECC)*, pp. 4001-4008, 2007

[4] Voos H. Nonlinear State-dependent Riccati Equation Control of a Quadrotor UAV. *IEEE International Conference on Control Applications*, pp. 2547-2552, 2006.

[5] Bouabdallah S, Siegwart R. Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2247-2252, 2005.

[6] Pipatpaibul P, Ouyang P R. Application of Online Iterative Learning Tracking Control for Quadrotor UAVs. *Isrn Robotics*, vol. 2013, pp. 162-167, 2013.

[7] Diao C, Xian B, Yin Q, et al. A Nonlinear Adaptive Control Approach for Quadrotor UAVs. *Asian Control Conference (ASCC)*, pp. 223 - 228, 2011.

[8] Bai Y, Liu H, Shi Z, et al. Robust Control of Quadrotor Unmanned Air Vehicles. *Chinese Control Conference (CCC)*, pp. 4462-4467, 2012.

[9] Santos M, Lpez V, Morata F. Intelligent Fuzzy Controller of a Quadrotor. *International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 141 - 146, 2010.

[10] J. G. Ziegler and N. B. Nichols. Optimum setting for automatic controllers. *ASME Trans.*, vol. 64, pp. 759-768, 1942.

[11] G. H. Cohen and G. A. Coon, Theoretical investigation of retarded control *ASME Trans.*, vol. 75, pp. 827-834, 1953.

[12] Xu J X, Huang D. Optimal Tuning of PID Parameters Using Iterative Learning Approach. *IEEE International Symposium on Intelligent Control*, pp. 226-231, 2007.

[13] Herrero J M, Blasco X, Martłnez M, et al. Optimal PID Tuning with Genetic Algorithms for Non-linear Process Models. *World Congress of the International Federation of Automatic Control (IFAC)*, pp. 31-36, 2002.

[14] Ghiglino P, Forshaw J L, Lappas V J. Online Evolutionary Swarm Algorithm for Self-Tuning Unmanned Flight Control Laws. *Journal of Guidance, Control, and Dynamics*, vol.38, no. 4, pp. 772-782. 2015.

[15] Storn B R, Price K. Differential Evolution C a Simple and Efficient Heuristic for Global Optimization over Coninuous Spaces. *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997.

[16] Brownlee. J. *Clever Algorithms: Nature-Inspired Programming Recipes*. LuLu Enterprises, 2011.

[17] Dombre E, Khalil W. *Modeling, performance analysis and control of robot manipulators*. ISTE Ltd, 2007.