Effect of Particle Swarm Optimization Tuned P, PD, and PID Controllers on the Stability of a Quadrotor

Tolga Üstünkök and Murat Karakaya

Abstract—**Many popular quadrotor controllers are based on PID controllers. This study compares the behaviour of a quadrotor when its controller is the proportional (P) only, proportional (P) and derivative (D), and all terms of the PID controller which is tuned by a Particle Swarm Optimization (PSO) implementation. A P, PD, and PID controller integrated quadrotor model is used with realistic parameters while conducting experiments in simulation. Our goal is to find out if it is worth to use PID or some of its terms is enough to get a stable system. According to the preliminary results of the experiments, the statistical difference of results shows that PID is better than both P and PD for the given model.**

Index Terms—**PID, PD, particle swarm optimization, PSO, optimization, control systems, natural computing, evolutionary computing, quadrotor, flying robot.**

I. Introduction

A quadrotor is a four rotored flying machine. Because of their small and simple structures, they have very agile movement capabilities. The main body of a quadrotor mostly includes a power source, 4 motors, 4 motor controllers, 4 propellers, and 1 control card which includes other necessary modules like microcontroller, wireless module, GPS, etc. Thanks to this simple structure, they can be manufactured at a lower cost which makes them very popular lately. However, using quadrotors in relatively critical missions like transportation or surveillance requires them to be reliable. The first step to ensuring such reliability is applying a control system to correct the attitude (roll, pitch, yaw) of the quadrotor without continuous human interaction. However, the parameters of the control system is very important since they directly affect the stability of the quadrotor. To provide the necessary calculations to maintain the system as stable as possible, there are proposed control system algorithms such as Santos et al. (2010), Altug et al. (2002), Madani & Benallegue (2006), Xu & Ozguner (2006), Xiong & Zheng (2014), Rozi et al. (2017), Salih et al. (2010). We are focusing the control system algorithm proposed in Salih et al. (2010) which is called Proportional Integral Derivative (PID).

PID is a control system algorithm which consists of 3 terms. The first term is the proportional (P), the second is the integral (I) and the third is the derivative (D). Proportional term provides the most basic error correction feature which outputs some proportion of the calculated error of the system. Integral term sums up the error from the start time until now and outputs some proportion of that sum. Finally, derivative term returns a proportion of the difference between current and previous errors of the system. After getting the results of these terms, all of them sum up to a single number and this number is fed to the actuators of the system. The important thing is how much of the output of these terms should be used to correct the system without overwhelm the actuators and the system. The amount of the error to be used is denoted by K_P for P term, the amount of integral to be used is K_I for I term and the amount of derivative to be used is K_D for D term. It is needed to tune the K_P, K_I and K_D constants to fly the quadrotor stable. I and D terms of the algorithm mostly aims to stabilize the static steady state error of the system which is occurred because of the P term and environmental disturbances. You can create control systems by combining only some of these terms or by using all of them.

In our work, we will compare P, PD and PID in terms of time to tune and stability. While tuning these gains Particle Swarm Optimization (PSO) is used. PSO is used to solve various NP-hard problems. Examples include network planning problems Zhao et al. (2010), route planning problems Savuran & Karakaya (2016). PSO should be a very suitable candidate for such a mission because of its simplicity and relatively low computational costs.

In Section II-A, PID control algorithm is introduced. In the next section, the generalized form of PSO is summarized and a general purpose pseudocode of the algorithm is given. After that, in Section III, a dynamic model of quadrotor is introduced. In Section IV, how PSO is implemented to tune the gains of the terms by using the quadrotor model is explained in detail. Finally, the results of different experiments are discussed and the planned future works are given.

II. Background

A. Proportional Integral Derivative

One of the most popular control loop feedback mechanism is the PID controller Åström & Hägglund (1995). PID algorithm works with 2 essential inputs. One is the feedback from the overall system response and the other one is the set point (desired) value. These inputs are used later to calculate the system error. A generalized system diagram with a feedback control algorithm is given in Figure 1.

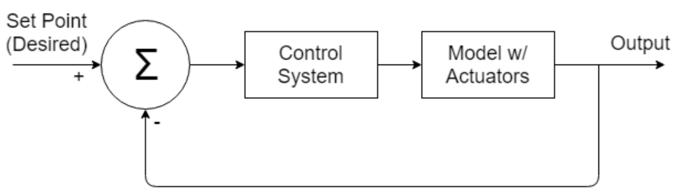

Fig. 1. Block diagram of a feedback controlled process.

The output of the control algorithm feeds the actuators of the quadrotor model. In our case, the actuators are the motors which provide the necessary angular velocity to the propellers to change the quadrotor's stance.

The overall PID controller consists of 3 terms as shown in (1).

$$u(t) = P(t) + I(t) + D(t) \tag{1}$$

$P(t)$ is the proportional (P) term, $I(t)$ is the integral (I) term and $D(t)$ is the derivative (D) term. There is an important reason that why (1) consists of 3 terms. That reason is a control system which is made up of only P cannot reach a static oscillation free steady state. To increase the success of whole controller and cancel out all or most of the static steady state oscillations I and D terms are introduced in addition to the P term.

In (1), all terms of PID are in time domain and depends on system error. The system error is defined as the difference between set-point and the measured process variable (actual sensor value) as shown in (2).

$$e(t) = y_{sp}(t) - y(t) \tag{2}$$

y_{sp} is set-point value and $y(t)$ is sensor value or current attitude of the quadrotor. Set-point is the attitude value which we desire how the quadrotor is positioned in 3-dimensional space.

The first term of PID is the P term which is shown in (3).

$$P(t) = K_p * e(t) \tag{3}$$

The proportion gain is called K_p. This constant needs to be tuned to get proper reactions from the system. $e(t)$ is the system error. Overall, the proportional term is the system error multiplied with a gain (proportion).

The next term is called I term. It is the integral of the system error as shown in (4).

$$I(t) = K_i * \int_{t_0}^{t_f} e(t)dt \tag{4}$$

The effect of the I term to a system is reducing the steady state error Åström & Hägglund (1995). The I term provides output even the system is currently stable at steady state because of its dependency on the past. The gain of the I term is held in a constant called K_i. This constant also needs to be tuned if the I term is used in the controller.

The final term of PID is the D term. It is shown in (5).

$$D(t) = K_d * \frac{de(t)}{dt} \tag{5}$$

The D term is the some proportion of rate of change of the system error. In other words derivative of the error, multiplied with a constant K_d. This constant is the gain of the D term. The effect of the D term to a system is observed as a damping factor. It delays the effect of the P term. As a result, the system is damped and always converges to a steady state when the K_p and K_d terms are positive numbers Åström & Hägglund (1995). However, there will be some side effects such as overdamp or underdamp, if the constants of the terms are not adjusted carefully. Both overdamp and underdamp increase the system error until the system reaches to steady state. Overdamp occurs if the gain of derivative term is relatively high with respect to the gain of the P term. In that state, the system will converge very slowly. On the other hand, if the constant of the D term is low and the constant of the proportional term is relatively high, then the system is under-damped. As a result, the system will again converges lately to an oscillation free steady state.

There are methods to tune the parameters of PID. Some of them are manual and some of them are automatic Åström & Hägglund (1995), *Manual Tune Procedures Products: PID Controllers* (n.d.). We only considered the automatic ones.

In this paper, we use some combination of terms of the PID controller such as P, PD, and PID as our controller. To tune the gains we proposed a natural computational method called Particle Swarm Optimization (PSO).

B. Particle Swarm Optimization

Eberhart and Kennedy proposed a simulation technique to simulate fish schools or bird flocks in 1995. Later, they formalized it as Particle Swarm Optimization (PSO) Kennedy & Eberhart (1995).

As the name suggests, there are particles which have positions and velocities. These are the basic computational units of PSO. The search process in PSO is made by moving the particles in the search space. Positions of particles are mapped to the solutions of the problem and modified by velocity of the particle. The velocities of particles are calculated as shown in (6) Eberhart & Shi (2001).

$$\begin{aligned} v_{id}(t+1) = w * v_{id}(t) + \\ c_1 * rand() * (pBest_{id} - x_{id}(t)) + \\ c_2 * rand() * (gBest_{id} - x_{id}(t)) \end{aligned} \tag{6}$$

$v_{id}(t+1)$ is the velocity of the ith particle in dimension d at time $t+1$. w is called inertial weight and it is the weight of the previous velocity to the current velocity. c_1 and c_2 are cognition and social weights of the particle's velocity. c_1 is the weight of personal best and c_2 is the weight of global best. $rand()$ function generates a random number between 0 and 1. x is the position of ith particle in dimension d. $pBest$ is the personal best position of that particles obtained so far. $gBest$ is the best position that is found in the swarm so far.

Positions of particles are are evaluated by a function called fitness function to decide whether or not the found solution is acceptable.

A group of particle which have same fitness function is called swarm. Swarms have a memory called global best ($gBest$). This is the index of a particle which gives the best solution obtained so far in the swarm.

After the velocities are calculated, positions of particles are updated as shown in (7).

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t) \tag{7}$$

$v_{id}(t)$ is the velocity of particle i in dimension d at time t. $x_{id}(t)$ is the position of particle i in dimension d at time t.

Algorithm 1 Pseudocode for Particle Swarm Optimization

```
$P \leftarrow InitializeParticles()$
for $i \leftarrow 0$ to $maxIterations$ do
    for all $p \in P$ do
        $fp \leftarrow fitness(p)$
        if $fp$ is better than $fitness(pBest)$ then
            $pBest \leftarrow p$
        end if
    end for
    $gBest \leftarrow$ best $p$ in $P$
    for all $p \in P$ do
        $v \leftarrow w*v + c_1*r_1*(pBest-p) + c_2*r_2*(gBest-p)$
        $p \leftarrow p + v$
    end for
end for
```

$x_{id}(t+1)$ is the position of particle i in dimension d at time $t+1$.

A pseudo-code is given in Algorithm 1 which intercepts the previously defined equations and methods as follows: The algorithm starts with initializing particles. The initialization process assigns random velocities and positions to all of the particles in that swarm. Then a loop is defined which repeats until max iteration count is reached. The iteration count of the loop specifies the stop condition of the algorithm. After that, positions of the all particles are evaluated to decide the personal bests and global best. Then, according to personal bests and the global best the new velocities and positions are recalculated with some randomness (r_1, r_2) in them. Then the whole process is repeated again until the specified iteration count is reached.

III. Quadrotor Model

To apply the previously mentioned control system and keep track of the behaviour of the quadrotor, it is necessary to derive a dynamic model. Quadrotors are typically hard to model because of their multi rotor nature compared to a helicopter. In this work, we developed our quadrotor model based on the model proposed in Xiong & Zheng (2014). To implement the model in MATLAB IDE, we modified the implementation presented in De Nardi (2013) to suit our objectives. The dynamic model in Xiong & Zheng (2014) is summarized as follows:

$$\ddot{x} = \frac{1}{m}(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)u_1 - \frac{K_1\dot{x}}{m} \tag{8}$$

$$\ddot{y} = \frac{1}{m}(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)u_1 - \frac{K_2\dot{y}}{m} \tag{9}$$

$$\ddot{z} = \frac{1}{m}(\cos\phi\cos\theta)u_1 - g - \frac{K_3\dot{z}}{m} \tag{10}$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} + \frac{J_r}{I_x}\dot{\theta}\Omega_r + \frac{l}{I_x}u_2 - \frac{K_4 l}{I_x}\dot{\phi} \tag{11}$$

$$\ddot{\theta} = \dot{\psi}\dot{\phi}\frac{I_z - I_x}{I_y} + \frac{J_r}{I_y}\dot{\phi}\Omega_r + \frac{l}{I_y}u_3 - \frac{K_5 l}{I_y}\dot{\theta} \tag{12}$$

$$\ddot{\psi} = \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{l}{I_z}u_4 - \frac{K_6}{I_z}\dot{\psi} \tag{13}$$

TABLE I
Quadrotor model parameters.

Variable	Value	Units
m	2.0	kg
$I_x = I_y$	1.25	Ns^2/rad
I_z	2.2	Ns^2/rad
$K_1 = K_2 = K_3$	0.01	Ns/m
$K_4 = K_5 = K_6$	0.012	Ns/m
l	0.20	m
J_r	1	Ns^2/rad
b	2	Ns^2
d	10	N ms^2
g	10	m/s^2

The Euler angles $[\phi, \theta, \psi]$ are the roll, pitch and yaw respectively. K_i are the drag coefficients. u_1 is the total force applied and calculated as in (14).

$$u_1 = F_1 + F_2 + F_3 + F_4 \tag{14}$$

Ω_r is the angular velocity of the whole quadrotor body and can be calculated from the individual angular velocities of the propellers as shown in (15).

$$\Omega_r = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \tag{15}$$

Ω_i is the angular velocity of the propeller i. The rolling, pitching, and yawing forces are defined as follows respectively:

$$u_2 = (-F_2 + F_4) \tag{16}$$

$$u_3 = (-F_1 + F_3) \tag{17}$$

$$u_4 = d(-F_1 + F_2 + F_3 + F_4)/b \tag{18}$$

The constants presented in Table I are gathered from De Nardi (2013), Boubertakh et al. (2013).

There are four degrees of freedom (DoF) in our case. They are:

1) Roll (ϕ)
2) Pitch (θ)
3) Yaw (ψ)
4) Altitude (z)

As a result, since each control algorithm has its own gain, there are 4 different control systems.

IV. An Implementation of PSO for Tuning Gain of the Terms of PID

Since the tuning parameters of PID is an optimization problem, PSO should be a good fit. The process of tuning starts with defining the decision variables. In the tuning problem, decision variables are the gains of the controller terms. In PSO terminology, the array of gains is called the position vector of a particle. The position vectors of a particle can be shown as (19), (20), and (21) for a P, PD, and PID controller respectively.

$$P = [K_P^1 \quad K_P^2 \quad K_P^3 \quad K_P^4] \tag{19}$$

$$P = [K_P^1 K_D^1 K_P^2 K_D^2 K_P^3 K_D^3 K_P^4 K_D^4] \tag{20}$$

$$P = [K_P^1 K_I^1 K_D^1 K_P^2 K_I^2 K_D^2 K_P^3 K_I^3 K_D^3 K_P^4 K_I^4 K_D^4] \tag{21}$$

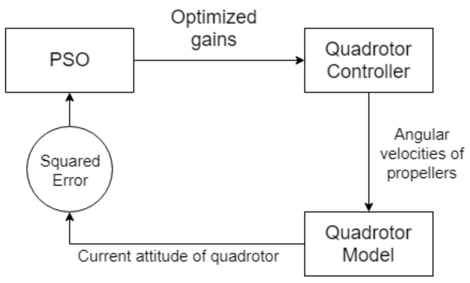

Fig. 2. Tuning flow diagram of the quadrotor controller.

TABLE II
PSO HYPER-PARAMETERS.

Variable	Default Value	Ranges
Number of particles	10	[5, 10, 15, 20]
c_1	2	-
c_2	2	-
w	0.9	-
Iteration count	100	-
V_{max}	2	-

Superscripts denote the controller index. For example, K_P^1 means the proportional gain of the roll controller.

To evaluate these positions a fitness function is needed. The fitness function is sum of mean absolute errors of each DoF for some interval of time. For this problem, fitness function (in other words cost function) is defined as in (22) Boubertakh et al. (2013).

$$SE = \sum_{i=1}^{4} \int_{t_0}^{t_f} e_i^2(t)dt \tag{22}$$

$e_i(t)$ is the system error as defined previously in (2). Index i is the controller index from 1 to 4 since we have 4 degrees of freedom (DoF).

The purpose of the PSO in this context is to minimizing the fitness function.

The flow of the algorithm is shown in Figure 2. Flow starts from the PSO by generating random positions and velocities for particles. The position of the best particle is given to the controller as tuned gains. Then, the quadrotor model is run for 15 seconds for each gain. Then, the cost is calculated by using (22). Finally, the whole process is repeated until a stop condition is satisfied. In our case, the stop condition is number of iterations.

To compute the results, the hyper-parameter values of the PSO is chosen by trial and error as Table II.

V. Results & Discussions

In order to compare the success of P, PD, and PID terms as a controller, three separate controller implementations are made. Each controller implementation is tuned 10 times with the same hyper-parameter values to see if it is consistent. After finding the best performing combination of PID terms, the number of particles are changed within the range that is given in Table II to see the effect of particle count to the solution. All tests run in a computer with an Intel i7-7500U 2.90GHz processor and 16GB of RAM.

TABLE III
K_P VALUES THAT ARE OBTAINED FROM THE DEFAULT HYPER-PARAMETER VALUES AT TABLE II.

Sln. #	K_P^1	K_P^2	K_P^3	K_P^4
1	3.88	2.82	0.32	4.57
2	3.86	3.42	2.19	3.80
3	3.15	2.78	0.39	5.24
4	6.98	4.10	0.50	5.63
5	4.20	1.68	0.28	5.07
6	3.71	2.60	2.30	3.79
7	3.78	2.82	2.81	3.78
8	3.83	0.77	1.29	5.11
9	3.36	3.50	0.06	5.34
10	3.34	3.80	0.41	4.51

TABLE IV
K_P AND K_D VALUES THAT ARE OBTAINED FROM THE DEFAULT HYPER-PARAMETER VALUES AT TABLE II.

Sln. #	K_P^1	K_D^1	K_P^2	K_D^2	K_P^3	K_D^3	K_P^4	K_D^4
1	2.98	5.91	5.46	1.65	0.76	3.97	8.09	1.71
2	2.69	1.01	1.25	0.50	1.12	0.97	8.28	1.74
3	3.42	2.76	4.85	6.13	3.98	5.80	8.17	1.63
4	3.46	0.72	0.55	0.72	1.16	4.13	0.04	2.59
5	1.70	0.75	0.93	0.98	0.60	0.61	0.92	2.68
6	5.36	2.03	1.69	7.20	3.42	8.12	7.87	1.48
7	3.16	2.09	1.87	1.52	1.79	0.86	8.03	1.73
8	1.20	1.78	3.84	0.73	2.21	0.19	7.39	1.94
9	3.43	0.79	2.40	2.42	1.11	2.27	8.11	1.74
10	1.85	1.56	3.16	2.71	2.41	0.40	7.68	1.92

The PSO algorithm outputs 4, 8, and 12 values which are the gains of the P, PD, and PID terms respectively. These values are formed as the vectors in (19), (20), and (21). Then, we input these vectors into the respected quadrotor control algorithm. After completing the computations, the quadrotor model controller outputs the total errors of each DoF according to (22) and generates a flight trajectory in 4 axes.

The initial conditions are given as follows:

- The initial angles and height: $\phi = 0°$, $\theta = 0°$, $\psi = 0°$, $Z = 0m$
- The desired angles and height: $\phi = -0.2°$, $\theta = -0.2°$, $\psi = 0°$, $Z = 5m$
- The cost function takes integral from $t_0 = 0s$ to $t_f = 15s$ with $dt = 0.01s$.

A. Comparison of the P, PD, and PID Controllers

Table III shows the gains found by PSO for a P only controller with default hyper-parameter values. Figure 3 shows the square error (SE) versus the number of iterations for each individual run. Each value in Table III is given to the quadrotor model separately as a candidate solution. Using each candidate solution, the quadrotor model calculates the respected errors of 4 DoF. This process is repeated for all of the remaining controllers.

According to the generated flight trajectories, none of the tuned P only controllers converge. Figure 6 shows the flight trajectory of the P only controller with the minimum square error for a 30 seconds flight. The desired conditions are same as with the used in tuning process.

Similarly, Table IV shows the gains found by PSO for PD controller. Again this one is also tuned with default hyper-parameter values. The desired conditions are same with the P

TABLE V
K_P, K_I, AND K_D VALUES THAT ARE OBTAINED FROM THE DEFAULT HYPER-PARAMETER VALUES AT TABLE II.

Sln. #	K_P^1	K_I^1	K_D^1	K_P^2	K_I^2	K_D^2	K_P^3	K_I^3	K_D^3	K_P^4	K_I^4	K_D^4
1	2.72	0.05	3.21	0.44	3.55	3.16	2.57	1.20	0.77	3.85	3.51	1.71
2	1.44	0.81	1.24	3.66	1.86	2.33	3.21	3.52	2.03	3.90	3.46	1.79
3	4.01	1.76	3.22	2.24	1.52	1.14	1.93	0.42	1.58	3.21	3.10	1.68
4	0.65	3.52	3.66	0.64	1.33	1.64	2.71	3.78	2.46	4.03	3.84	1.82
5	1.39	3.59	3.33	1.12	1.67	1.77	2.33	0.78	1.94	3.51	3.37	1.78
6	1.74	1.17	1.86	3.59	4.26	1.52	1.43	0.73	4.87	3.26	3.37	1.83
7	5.07	2.47	2.14	2.57	3.62	1.32	3.55	1.50	2.40	3.85	3.91	1.88
8	0.75	0.59	4.24	1.96	4.05	3.37	3.37	0.05	3.29	4.53	3.58	1.76
9	3.83	3.75	3.18	0.27	0.47	1.17	2.02	0.24	1.47	3.96	2.79	1.60
10	0.54	0.34	2.45	2.50	1.19	2.78	0.67	4.42	2.35	4.51	3.91	1.86

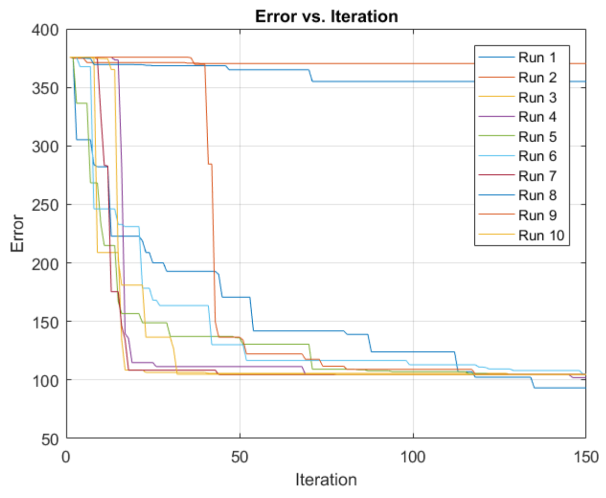

Fig. 3. Square error curves of each individual run of P only controller.

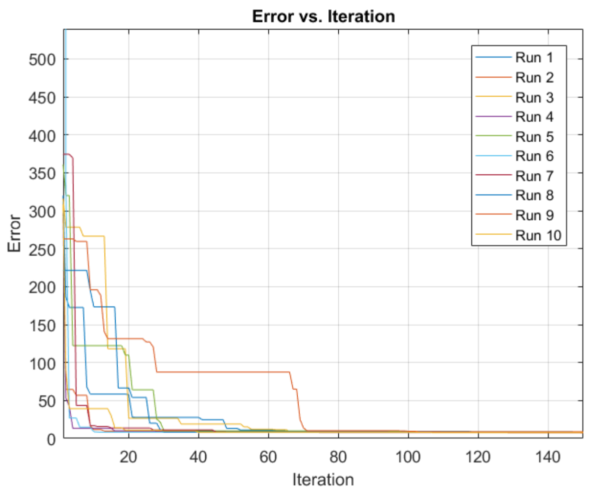

Fig. 5. Square error curves of each individual run of PID controller.

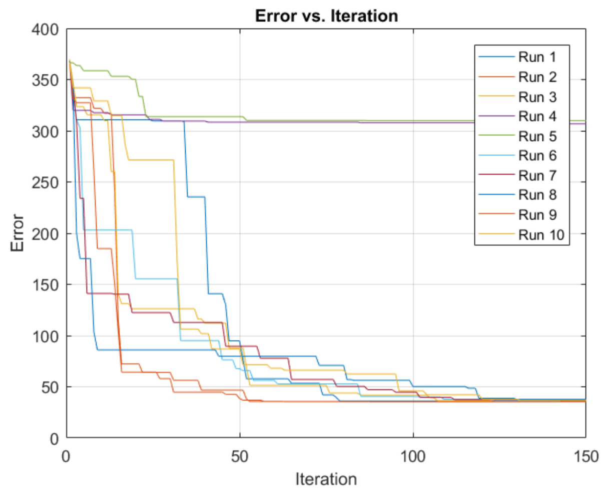

Fig. 4. Square error curves of each individual run of PD controller.

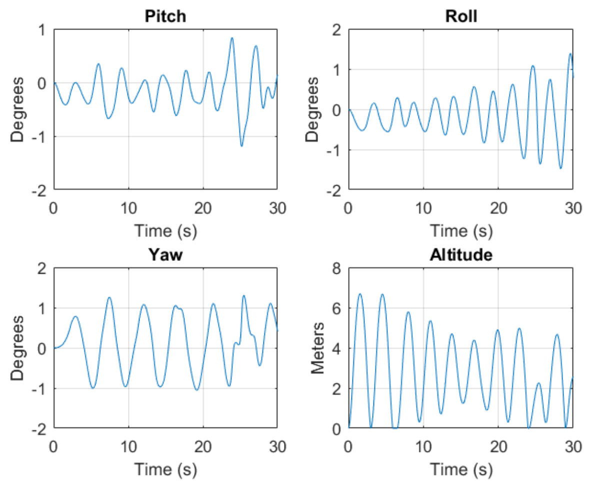

Fig. 6. Flight trajectory of the best found gains with P only controller for 30 seconds.

only experiment. Figure 4 shows the SE curves and Figure 7 shows the flight trajectory of the best one from the table.

All of the above processes for P and PD controllers are repeated for the PID controller. SE curves can be seen in Figure 5 and flight trajectory for a 30 seconds flight is presented in Figure 8.

To see which controller setup performs better, mean and standard deviation (SD) of the errors are calculated at the bottom of tables. One can be accurately deducted the PID controller outperforms both P only and PD controller for this specific model. With this information in mind, PID controller is used in following tests.

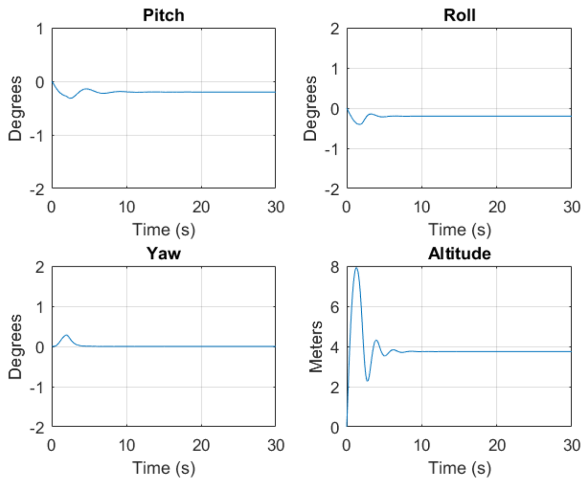

Fig. 7. Flight trajectory of the best found gains with PD controller for 30 seconds.

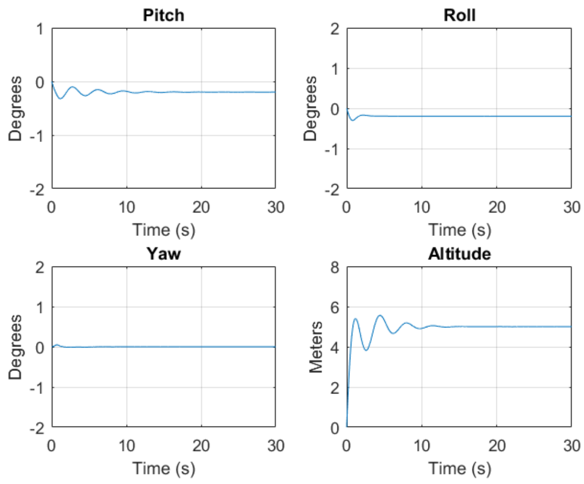

Fig. 8. Flight trajectory of the best found gains with PID controller for 30 seconds.

B. Impact of Number of Particles to the Error

We also want to see if the quality of results are changed or not when we change particle count in the swarm. To understand the effects of number of particles to the errors got from the quadrotor model and calculation time, we fixed all the other hyper-parameter values to default values except number of particles.

It can be seen that average error of the system is slightly decreases as the number of particles increases. The increase in number of particles also increases the calculation time. However, the decrease in average error is much slower than the increase in the calculation time.

1) Discussions: The impact of number of particles on the observed total error is calculated by difference in error divided by difference in time. The difference in average error is 4.8564 between 5 particles and 20 particles. On the other hand, difference in time is 78.6589 seconds. We have an error to time ratio of 0.0279. The decrease in average error is not worth to the increase in time. However, by looking the decrease in standard deviation we can say that as the particle count increases the accuracy of the results are increasing.

TABLE VI
SQUARED ERRORS OF THE MODEL WITH A P ONLY CONTROLLER.

Sln. #	ϕ **Error**	θ **Error**	ψ **Error**	**Z Error**	Σ **Error**
1	0.58	0.49	6.32	118.27	125.66
2	0.76	0.61	0.50	136.13	138.00
3	1.17	0.90	6.85	113.24	122.15
4	0.71	1.60	12.59	126.02	140.92
5	1.37	2.80	12.38	116.63	133.18
6	0.32	0.50	0.06	145.60	146.49
7	0.37	0.52	0.06	151.43	152.39
8	1.38	1.87	5.09	127.37	135.72
9	1.35	1.30	5.35	104.71	112.71
10	0.72	0.30	3.16	121.57	125.76
Mean	0.88	1.09	5.24	126.10	133.30
SD	0.39	0.76	4.35	13.85	11.35

TABLE VII
SQUARED ERRORS OF THE MODEL WITH PD CONTROLLER.

Sln. #	ϕ **Error**	θ **Error**	ψ **Error**	**Z Error**	Σ **Error**
1	0.00	0.03	0.10	35.92	36.05
2	0.06	0.03	0.09	35.24	35.41
3	0.02	0.00	0.00	35.40	35.43
4	0.03	0.02	0.01	306.70	306.77
5	0.03	0.04	0.01	309.90	309.98
6	0.02	0.00	0.01	36.44	36.47
7	0.02	0.01	0.03	35.68	35.75
8	0.03	0.10	0.25	37.37	37.75
9	0.04	0.01	0.01	35.50	35.56
10	0.01	0.02	0.05	36.70	36.79
Mean	0.03	0.03	0.06	90.49	90.59
SD	0.01	0.03	0.07	108.91	108.89

TABLE VIII
SQUARED ERRORS OF THE MODEL WITH PID CONTROLLER.

Sln. #	ϕ **Error**	θ **Error**	ψ **Error**	**Z Error**	Σ **Error**
1	0.00	0.03	0.00	8.24	8.27
2	0.01	0.01	0.00	7.85	7.87
3	0.01	0.03	0.01	8.40	8.44
4	0.15	0.03	0.02	8.42	8.62
5	0.06	0.02	0.03	8.44	8.55
6	0.02	0.02	0.00	7.67	7.71
7	0.01	0.03	0.00	7.30	7.34
8	0.01	0.01	0.00	8.26	8.29
9	0.01	0.04	0.00	8.86	8.90
10	0.01	0.01	0.01	7.73	7.75
Mean	0.03	0.02	0.01	8.12	8.18
SD	0.04	0.01	0.01	0.44	0.46

TABLE IX
AVERAGES, STANDARD DEVIATIONS OF ERRORS FOR DIFFERENT PARTICLE COUNTS. THE COMPUTATION TIMES ARE ALSO GIVEN.

	5 Part.	**10 Part.**	**15 Part.**	**20 Part.**
Mean	11.55	9.90	7.68	7.64
SD	8.45	4.15	0.62	0.51
Time(s)	144.67	280.99	428.51	571.57

VI. CONCLUSIONS & FUTURE WORKS

In this preliminary work, we analyzed the impact of the P, PD, and PID controllers on the quadrotor stability. We used a PSO algorithm to optimize the gains of the P, PD, and PID controllers. We found that PID controller gives better results compared to the results provided by the P and PD controllers. In fact, P only controller did not converge at any point in time at all. The most affected DoF was the altitude and the least affected one was the yaw.

In future, we are planning shortening the computation time so that it will be an acceptable solution to an online tuning application.

References

Altug, E., Ostrowski, J. P. & Mahony, R. (2002), Control of a quadrotor helicopter using visual feedback, *in* ‘Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on’, Vol. 1, IEEE, pp. 72–77.

Åström, K. J. & Hägglund, T. (1995), *PID controllers: theory, design, and tuning*, Vol. 2, Instrument society of America Research Triangle Park, NC.

Boubertakh, H., Bencharef, S. & Labiod, S. (2013), Pso-based pid control design for the stabilization of a quadrotor, *in* ‘3rd International Conference on Systems and Control’, pp. 514–517.

De Nardi, R. (2013), ‘The qrsim quadrotors simulator’, *RN* **13**(08), 08.

Eberhart & Shi, Y. (2001), Particle swarm optimization: developments, applications and resources, *in* ‘Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)’, Vol. 1, pp. 81–86 vol. 1.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* ‘Neural Networks, 1995. Proceedings., IEEE International Conference on’, Vol. 4, pp. 1942–1948 vol.4.

Madani, T. & Benallegue, A. (2006), Backstepping control for a quadrotor helicopter, *in* ‘Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on’, IEEE, pp. 3255–3260.

Manual Tune Procedures Products: PID Controllers (n.d.), Technical report, Red Lion Controls.

Rozi, H. A., Susanto, E. & Dwibawa, I. P. (2017), Quadrotor model with proportional derivative controller, *in* ‘2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)’, pp. 241–246.

Salih, A. L., Moghavvemi, M., Mohamed, H. A. F. & Gaeid, K. S. (2010), Modelling and pid controller design for a quadrotor unmanned air vehicle, *in* ‘2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)’, Vol. 1, pp. 1–5.

Santos, M., López, V. & Morata, F. (2010), Intelligent fuzzy controller of a quadrotor, *in* ‘2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering’, pp. 141–146.

Savuran, H. & Karakaya, M. (2016), ‘Efficient route planning for an unmanned air vehicle deployed on a moving carrier’, *Soft Computing* **20**(7), 2905–2920.

Xiong, J.-J. & Zheng, E.-H. (2014), ‘Position and attitude tracking control for a quadrotor uav’, *ISA transactions* **53**(3), 725–731.

Xu, R. & Ozguner, U. (2006), Sliding mode control of a quadrotor helicopter, *in* ‘Decision and Control, 2006 45th IEEE Conference on’, IEEE, pp. 4957–4962.

Zhao, R., Zhang, Y. & Lehnert, R. (2010), ‘Survivable topology design of hierarchical hybrid fibre-vdsl access networks using enhanced discrete binary pso’, *International Journal of Autonomous and Adaptive Communications Systems* **3**(4), 419–438.