A PSO-BASED TUNING ALGORITHM FOR QUADCOPTER CONTROLLERS

Sarah Pontes Madruga*, Gabriel Fernando Basso*, Augusto de Holanda B. M. Tavares*, Alisson Vasconcelos de Brito*, Tiago Pereira Nascimento*

*Universidade Federal da Paraíba
Rua dos Escoteiros, s/n, Mangabeira
João Pessoa, Paraíba, Brasil

Emails: madruga.sarah@gmail.com, gabriel.f.basso@gmail.com, augustotava@gmail.com, alisson@ci.ufpb.br, tiagopn@ci.ufpb.br

Abstract— In this paper, we present the trajectory and attitude control of a quadrotor UAV (Unmanned Aerial Vehicle), using a PD controller for the path control, and a PID for the attitude control. With our tuning method, we were able to achieve a performance similar to those of more complex controllers. We compared our results with those of traditional tuning techniques, through graphical analysis and MSE (Mean-Squared Error) calculation of two different generated trajectories. Although here we used classical PD and PID, the proposed algorithm can easily be applied to any other type of controller.

Keywords— Quadcopter, PID tuning, PSO.

Resumo— Neste trabalho, apresentamos o controle de trajetória e de postura de um VANT (Veículo Aéreo Não Tripulado) do tipo quadcóptero, usando um controlador PD para o controle de trajetória, e um PID para o controle de equilíbrio, ou postura. Com o método de sintonia proposto, foi possível obter um desempenho similar àqueles de controladores mais complexos. Os resultados foram comparados com técnicas de sintonia tradicionais, por meio de análise gráfica e cálculo do MSE (*Mean-Squared Error* - Erro Médio Quadrático) em duas trajetórias diferentes. Ainda que aqui tenham sido usados controladores PD e PID, o algoritmo proposto também pode ser utilizado para sintonizar outros tipos de controladores.

Palavras-chave— Quadcóptero, Sintonia PID, PSO.

1 Introduction

In the last decade, unmanned aerial vehicles (UAV) have been widely used to facilitate human life, either by doing dangerous or difficult tasks, or by making civilian chores more efficient. In this context, the quadrotor configuration is interesting, because it enables a vertical take-off and landing and hovering in small areas, which is impossible for a fixed-wing craft. The quadcopter is also more efficient than the traditional single propeller helicopter, since the small propellers reduce the torque on the system (Hussein and Nemah, 2015). Figure 1 shows a traditional quadcopter.

In spite of its simple mechanical design, its dynamics behavior makes the aircraft control substantially difficult, seeing that it is in fact an underactuated nonlinear system (Bouktir et al., 2008). That is, there are only four rotors that generate for input trusts to control six degrees of freedom.

It follows that quadcopters are generally known to be dynamically unstable systems (Hussein and Nemah, 2015), so a proper control method is necessary to achieve stability. Such methods are usually very complex (Liu et al., 2015), (Khan et al., 2011), (Cao et al., 2016) and may prove themselves a challenge when it comes to embedded systems, even though the technological advancement in the last few years has made UAVs development possible (Paiva et al., 2016).

As a result, we can find a lot of research on quadcopter modeling (Bouabdallah, 2007), (Bresciani, 2008), (Mahony et al., 2012), many of which try to simplify the system dynamics (Liu et al., 2013), (González-Sánchez et al., 2013), (Gan et al., 2013). Nonetheless, the community still lacks a standard model for this type of system (González-Sánchez et al., 2013). When it comes to controllers, there is a lot of research focusing only on attitude and altitude control, (Gan et al., 2013), (Khan and Kadri, 2015), even those which use classical controllers like PID (Paiva et al., 2016), (R.A. García, 2012), (Lukmana and Nurhadi, 2015).

In this paper, we propose a PSO (Particle Swarm Optimization)-based tuning technique for trajectory and attitude/altitude control using classical PD and PID controllers, for trajectory and attitude/altitude respectively. This approach allowed us to achieve an impressive performance without the need to rely on complex control algorithms that are not always suitable for embedded systems (R.A. García, 2012), (Paiva et al., 2016), (Zheng et al., 2016). Based on a graphical analysis and MSE (Mean-Squared Error) comparison, we concluded that our technique obtained better results than traditional tuning methods, and it can be an alternative for stable controllers on embedded systems.

This paper is organized as follows: in Section II, we will make a brief mathematical study of the quadcopter model and the PD and PID controllers employed; in Section III, the tuning method approach will be explained; in Section IV, we will make an analysis of the results; and Section V shows our conclusions and suggestions for future work.

Figure 1: A generic quadrotor UAV.

2 Modeling and Control of the Quadrotor

2.1 Quadrotor Model

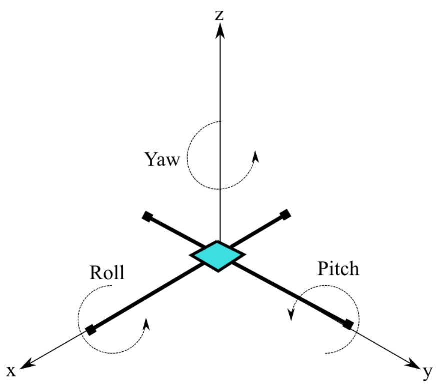

Figure 2: Axis labels and conventions. Yaw (ψ), pitch(θ), roll(ϕ).

In this Section, we present a synthesis of the quadrotor modeling used for the simulation. The reader can find more details in (Hartman et al., 2016) and (Mahony et al., 2012). Figure 2 shows our axis and angles convention.

We will start our modeling by presenting the inertia matrix for the quadrotor in Equation 1. This matrix describes the quadcopter mass moment of inertia across the axes and is very important to the flight dynamics of the system.

$$J_b = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{1}$$

J_b is the inertia of the quadcopter relative to the body frame, and J_{xx}, J_{yy} and J_{zz} are the inertia of the quadcopter across each body frame axis.

Next, come the thrust and torque coefficient relations. The thrust T of a single motor/propeller system can be calculated by Equation 2, where c_T is the lumped parameter thrust coefficient dependent on the rotor specifications, the air density, the radius of the rotor; and ω is the angular velocity of the rotor.

$$T = c_T \times \omega^2 \tag{2}$$

Similarly, it is possible to demonstrate Equation 3, for the torque coefficient. In this case, c_Q reffers to the lumped parameter torque coefficient, and and ω is still the angular velocity of the rotor.

$$Q = c_Q \times \omega^2 \tag{3}$$

With this information, Equation 4 shows a matrix that describes the thrusts and torques on the system, where d is the distance between the motors and their respective axes of rotation.

$$\begin{bmatrix} \Sigma T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & d.c_T & 0 & -d.c_T \\ -d.c_T & 0 & d.c_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{4}$$

Following, it is important to highight the Throttle Command Relation, presented in Equation 5. Here, ω_{ss} is the expected steady-state motor RPM, $(Throttle\%)$ is the throttle percentage command, c_R is the throttle % to RPM conversion coefficient, and b is the y-intercept of the linear regression relationship (Hartman et al., 2016).

$$\omega_{ss} = (Throttle\%)c_R + b \tag{5}$$

For the final matrix, we also need to take into account the gyroscopic forces resulting on the body, which are governed by the inertia of each motor rotating components (J_m), the rolling and pitching rates (P and Q) and the speed of each motor/prop system (ω_i). Hence, Equations 6 and 7 present the gyroscopic torques created by the motors for pitch and roll action.

$$\tau_{\phi_{gyro}} = J_m Q \left(\frac{\pi}{30}\right)(\omega_1 - \omega_2 + \omega_3 - \omega_4) \tag{6}$$

$$\tau_{\theta_{gyro}} = J_m P \left(\frac{\pi}{30}\right)(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \tag{7}$$

Finally, we can construct the final matrix, which refers to the moments present in the body frame resulting from the aerodynamics, thrusts, and torques on the system. See Equation 8.

$$M = \begin{bmatrix} d.c_T.\omega_2^2 - d.c_T.\omega_4^2 + J_m Q\left(\frac{\pi}{30}\right)(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ -d.c_T.\omega_1^2 + d.c_T.\omega_3^2 + J_m P\left(\frac{\pi}{30}\right)(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \\ -c_Q.\omega_1^2 + c_Q.\omega_2^2 - c_Q.\omega_3^2 + c_Q.\omega_4^2 \end{bmatrix} \tag{8}$$

2.2 Quadrotor Control Strategy

As mentioned above, in this paper we use classical PD and PID approaches for trajectory, attitude and altitude control of the quadrotor. These controllers will be improved using a precise tuning algorithm. In Figure 3, a detailed block diagram explains the control strategy here employed.

First, a *Trajectory Generator* sends the (x, y) *Path Commands* to the *PD Trajectory Controller*. This controller then calculates the *Attitude Commands*, according to Equation 9, which can be easily recognized as the classic PD controller equation.

$$Att_{cmd} = Kp \times Error - Kd \times \frac{dVx}{dt} \quad (9)$$

Here, Att_{cmd} can be either the θ or ϕ commands, for the X position controller and Y position controller, respectively. The *Error* variable corresponds to the velocity error. Kp and Kd are the controller gains to be tuned (Hartman et al., 2016).

These *Attitude Commands* will then be sent to the *PID Attitude Controller*, where ϕ, θ, ψ and z corrections will be calculated by Equation 10 and received by the *Quadcopter Dynamics* block. The *State Input* is then sent to the controller blocks for feedback.

$$Corr = Kp \times Error + Ki \times \int Error \times dt - Kd \times \frac{dz}{dt} \quad (10)$$

Kp, Ki and Kd are the gains to be tuned; the variable $Error$ represents the error in altitude or attitude, that is, the errors in ϕ, θ, ψ or z. Also, the variable z in Equation 10 can be either ϕ, θ, ψ or z, depending on the one we want to control at the time (Hartman et al., 2016).

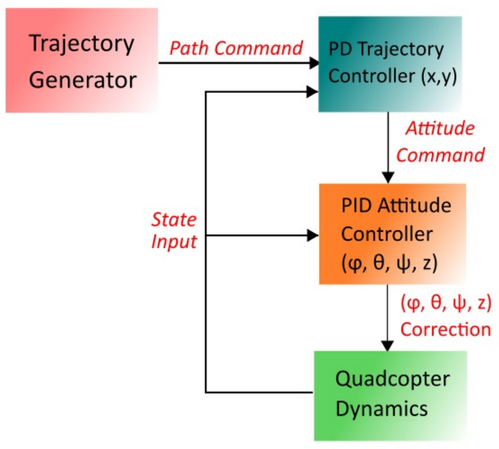

Figure 3: Block diagram of the control strategy.

3 PSO-Based Tuning Algorithm

Our tuning method consists of 3 steps. First, we execute a particle swarm search for the best values for the controller gains, by simulating the drone flight. We save each attempted combination of parameters and the resulting trajectory error, in order to map the parameter combinations into the error. Then, we use this saved data to model the Trajectory Error (TE) as a function of the parameters, that is, we model the Error as a high dimension function of the controller parameters. Finally, with this model, we search the parameter space for the smallest Error using again a PSO. With this, we avoid the excessive computational cost of the flight simulation, running the search in a mathematical construct of the simulation itself.

For clarity, the step-by-step goes as follows:

- *Particle Swarm Optimization (PSO):*

We used the PSO proposed in (Wang and Liu, 2016) and tested in (Basso et al., 2017). Unlike standard PSO algorithms, in which the particles are guided only by its best historical parameter set and swarm best parameters, it introduces the best neighbor information, an abandon mechanism and a chaotic search operator. This algorithm is thoroughly explained in the referred paper.

- *Modeling the TE as a function of the controller parameters:*

During the PSO execution, many different parameter combinations are searched, each with a different effectiveness in the trajectory tracking. We save each trial parameters and Trajectory Error. This information is then used to model the error as a multidimensional function of the controller parameters.

We assume the TE to be a high order sparse polynomial function of the parameters, so all we have to do is take a truncated Taylor expansion and find the correct expansion weights. This is modeled as shown in Equation 11:

$$TE = P \times w \quad (11)$$

Where TE stands for the Trajectory Error; P is the vector containing each polynomial combination of parameters, up to the desired order; and w is the vector with the weight of the Taylor expansion.

This minimization problem is solved by employing an $L0$ minimization algorithm, as shown in (Mohimani et al., 2009).

- *Second Particle Swarm Optimization (PSO):*

With the model obtained, we perform another PSO for the optimal parameters.

4 Simulation and Results

In this section, we discuss the results of the application of the referred tuning method to a quadcopter Simulink model, which was adapted from (Hartman et al., 2016) and (Mahony et al., 2012). The PSO simulations were also conducted in this same platform.

We simulated two different trajectories, a circle and a triangle. To each of them, we compared the 3D Graphic, the 2D Trajectory and the Altitude of classical and proposed tuning methods, as well as the MSE.

In the first trajectory, a 2m radius circle, the quadcopter starts on the ground, then it goes up till a 3m height, and it goes back to the ground after the circle is completed in the air. In the next figures, the reference is shown in blue, while the output is red. Figures 4, 5 and 6 show the results of the quadcopter behavior for PD and PID gains tuned using classical methods.

Even though we can see a considerable deviation from the reference path on the output in Figure 5, the most unstable part is the altitude, which has a considerable amount of oscillation around the 3m reference, as we can see in Figures 4 and 6. The goal here is to

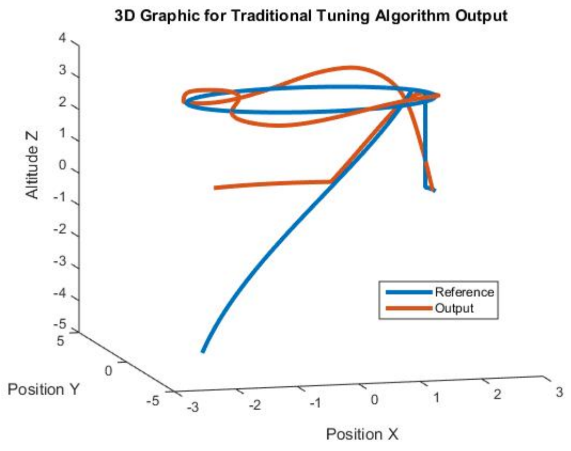

Figure 4: Comparison between reference and 3D behavior of the quadcopter for traditional tuning.

improve this with the PSO-based tuning. The results for the circle trajectory using the new proposed tuning method can be seen in Figures 7, 8 and 9.

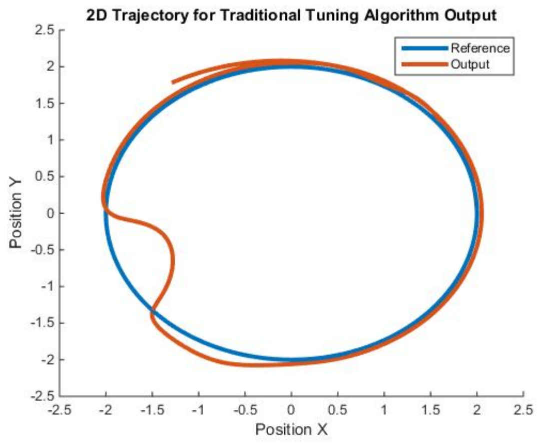

Figure 5: Comparison between reference and 2D trajectory with traditional tuning.

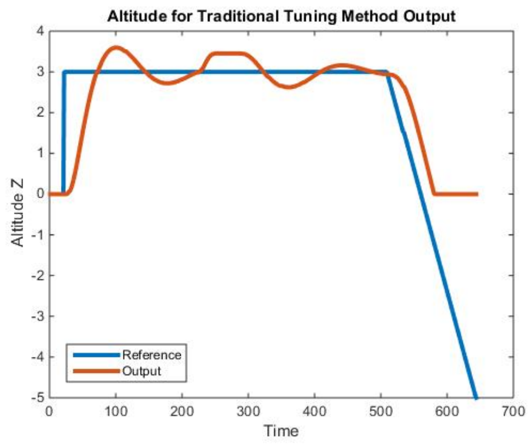

Figure 6: Altitude behavior with traditional tuning.

With the new tuning method, we can see that the altitude is much more stable, showing only a small peak at the beginning, but achieving stable regimen very quickly, as ilustrated in Figure 9. The 2D trajectory is also better, following the reference completely. The deviation seen at the end of the red line in Figure 8 is due to the landing of the quadcopter, which can be verified in the 3D representation in Figure 7.

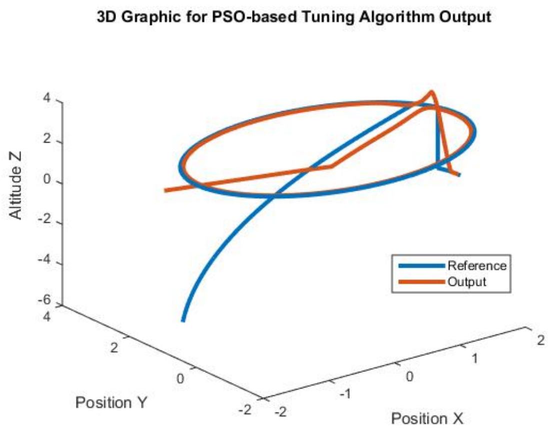

Figure 7: Comparison between reference and 3D behavior of the quadcopter for the PSO-based tuning.

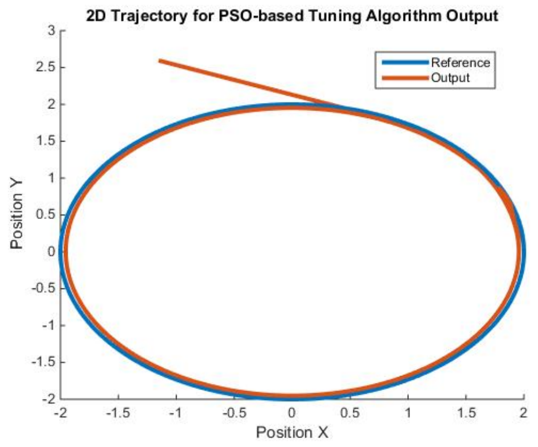

Figure 8: Comparison between reference and 2D trajectory with the PSO-based tuning.

Now for the triangle trajectory, a righ isosceles triangle with 1m catheti. For the traditional tuning, we have Figures 10, 11 and 12. Although the oscillation here is also evident in Figures 10 and 12, if we check the graphics scale, we can see that it is in the order of centimeters, which is much smaller than in the previous trajectory, the circle. This is expected, since the triangle is a much simpler path to follow, because it consists of straight lines.

Using the PSO-based tuning, however, the oscillations are even smaller, in the order of millimeters, and the convergence of altitude is still faster than before. This can be verified in Figures 13, 14 and 15.

Despite the fact that the 2D trajectory output in Figure 14 was slightly smaller than the reference, the overall MSE was still better than the traditional tuning

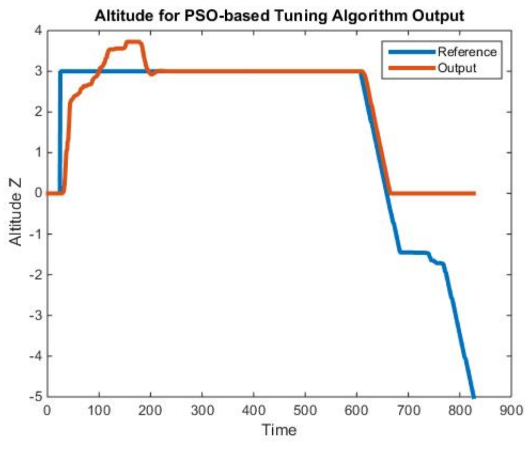

Figure 9: Altitude behavior with PSO-based tuning.

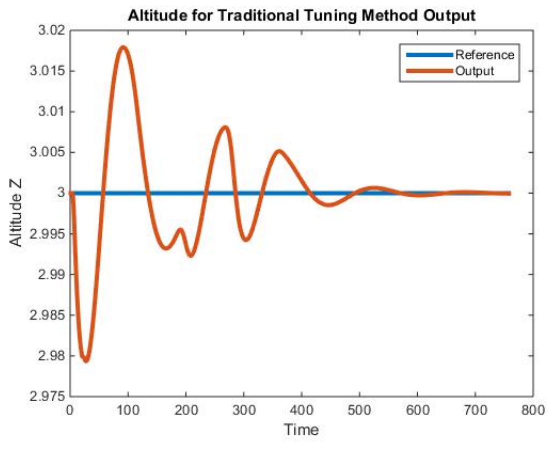

Figure 12: Altitude behavior with traditional tuning.

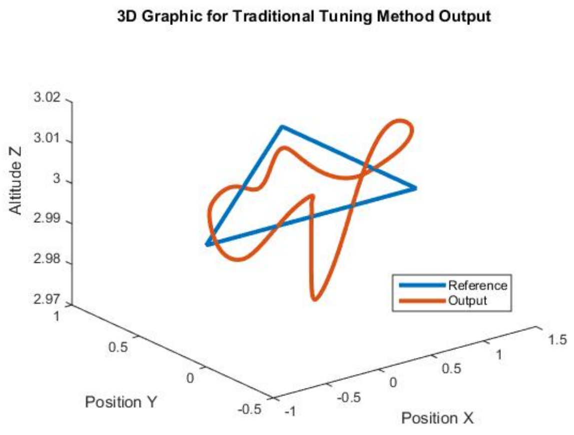

Figure 10: Comparison between reference and 3D behavior of the quadcopter for traditional tuning.

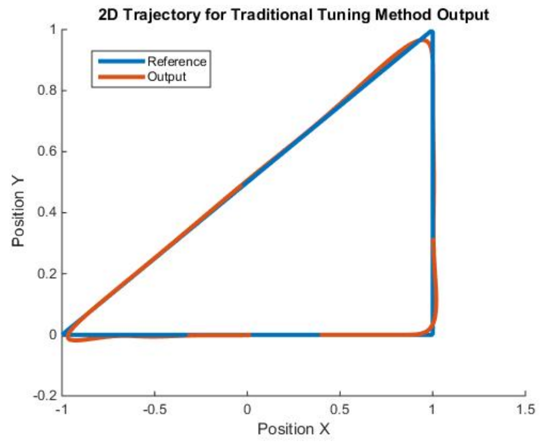

Figure 11: Comparison between reference and 2D trajectory with traditional tuning.

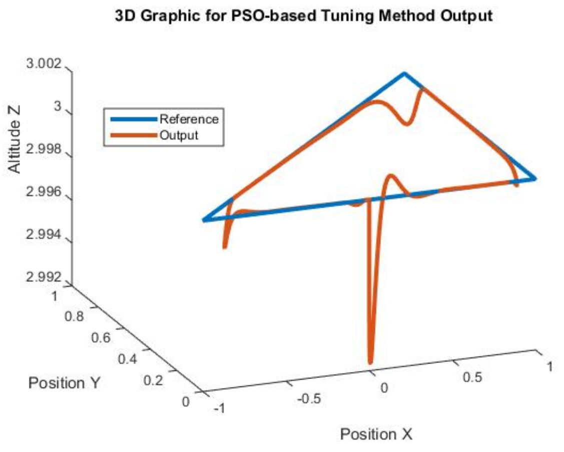

Figure 13: Comparison between reference and 3D behavior of the quadcopter for the PSO-based tuning.

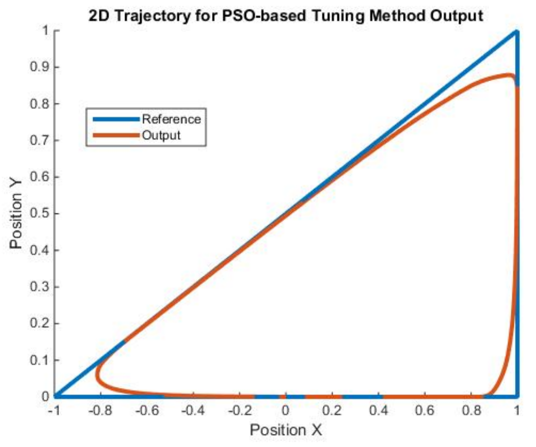

Figure 14: Comparison between reference and 2D trajectory with the PSO-based tuning.

Table 1: Mean-Squared Error analysis

	MSE - Circle Path	MSE - Triangle Path
Traditional tuning method	2.6469	0.0517
PSO-based tuning method	2.3244	0.0397
Improvement	12.18%	23.21%

one, as shown in Table 1. The gains adopted in each simulation are displayed in Table 2.

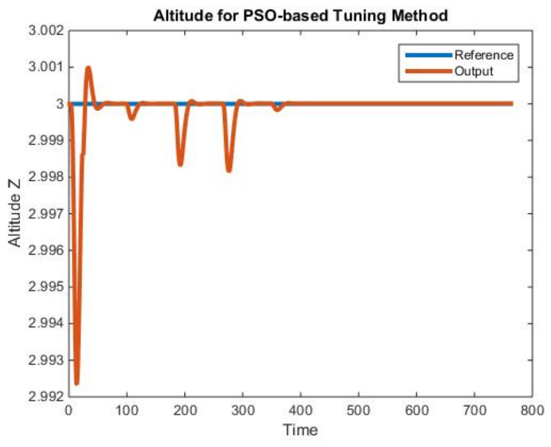

Figure 15: Altitude behavior with PSO-based tuning.

Table 2: Gains calculated and used in the simulations

	Traditional Tuning	**PSO-based Tuning**
Kp_θ **(PD)**	0.32	0.7368
Kd_θ **(PD)**	0.1	0.0030
Kp_ϕ **(PD)**	0.32	1.0815
Kd_ϕ **(PD)**	0.1	0.0099
Kp_ϕ	2	29.3560
Ki_ϕ	1.1	1.7548
Kd_ϕ	1.2	6.1140
Kp_θ	2	43.9193
Ki_θ	1.1	0.2127
Kd_θ	1.2	6.0170
Kp_ψ	4	0.3468
Ki_ψ	0.5	0.4036
Kd_ψ	3.5	11.4075
Kp_z	2	23.1958
Ki_z	1.1	0.0366
Kd_z	3.3	9.4616

5 Conclusions

In this paper, we developed a PSO-based tuning technique that improved the quality of our classical PD and PID controllers when compared to the traditional tuning methods, with no added complexity to the control system. From the results shown in the previous Sections, we can see that the algorithm is capable of achieving the system convergence. This was verified in two different paths with different complexities. This method can also be applied to other types of controllers in order to upgrade their performances.

For future work related to this subject, we suggest using the technique in an embedded system for real time flying experiments, in order to verify how the controller tuned with this algorithm will behave in comparison with the gains obtained by traditional tuning. Also, implementing this algorithm with a more robust controller that is very sensitive to tuning, like the Nonlinear Model Predictive Controller (NMPC) (Khan et al., 2011), may have very interesting applications.

References

Basso, G. F., Amorim, T. G. S. D., Brito, A. V. and Nascimento, T. P. (2017). Kalman filter with dynamical setting of optimal process noise covariance, *IEEE Access* **5**: 8385–8393.

Bouabdallah, S. (2007). *Design and control of quadrotors with application to autonomous flying*, PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

Bouktir, Y., Haddad, M. and Chettibi, T. (2008). Trajectory planning for a quadrotor helicopter, *2008 16th Mediterranean Conference on Control and Automation*, pp. 1258–1263.

Bresciani, T. (2008). *Modelling, Identification and Control of a Quadrotor Helicopter*, PhD thesis, Lund University.

Cao, G., Lai, E. M. K. and Alam, F. (2016). Gaussian process model predictive control of unmanned quadrotors, *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 200–206.

Gan, D., Cai, G., Dias, J., Seneviratne, L., Dias, J. and Seneviratne, L. (2013). Attitude control of quad-rotor uavs using an intuitive kinematics model, *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, pp. 597–600.

González-Sánchez, M., Amézquita-Brooks, L., Liceaga-Castro, E. and d. C. Zambrano-Robledo, P. (2013). Simplifying quadrotor controllers by using simplified design models, *52nd IEEE Conference on Decision and Control*, pp. 4236–4241.

Hartman, D., Landis, K., Mehrer, M., Moreno, S. and Kim, J. (2016). Parameter driven simulink model for quadcopter simulation and control system design. [http://www.mathworks.com/matlabcentral/fileexchange/48053-quad-sim; accessed 6-March-2018].

Hussein, M. T. and Nemah, M. N. (2015). Modeling and control of quadrotor systems, *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pp. 725–730.

Khan, H. S. and Kadri, M. B. (2015). Attitude and altitude control of quadrotor by discrete pid control and non-linear model predictive control, *2015 International Conference on Information and Communication Technologies (ICICT)*, pp. 1–11.

Khan, R., Williams, P., Hill, R. and Bil, C. (2011). Fault tolerant flight control system design for uav's using nonlinear model predictive control, *2011 Australian Control Conference*, pp. 297–302.

Liu, Y., Longo, S. and Kerrigan, E. C. (2013). Nonlinear predictive control of autonomous soaring uavs using 3dof models, *2013 European Control Conference (ECC)*, pp. 1365–1370.

Liu, Y., Montenbruck, J. M., Stegagno, P., Allgöwer, F. and Zell, A. (2015). A robust nonlinear controller for nontrivial quadrotor maneuvers: Approach and verification, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5410–5416.

Lukmana, M. A. and Nurhadi, H. (2015). Preliminary study on unmanned aerial vehicle (uav) quadcopter using pid controller, *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, pp. 34–37.

Mahony, R., Kumar, V. and Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor, *IEEE Robotics Automation Magazine* **19**(3): 20–32.

Mohimani, H., Babaie-Zadeh, M. and Jutten, C. (2009). A fast approach for overcomplete sparse decomposition based on smoothed ell^0 norm, *IEEE Transactions on Signal Processing* **57**(1): 289–301.

Paiva, E., Soto, J., Salinas, J. and Ipanaqué, W. (2016). Modeling, simulation and implementation of a modified pid controller for stabilizing a quadcopter, *2016 IEEE International Conference on Automatica (ICA-ACCA)*, pp. 1–6.

R.A. García, F.R. Rubio, M. O. (2012). Robust pid control of the quadrotor helicopter, *IFAC Proceedings Volumes*, pp. 229–234.

Wang, C.-F. and Liu, K. (2016). A novel particle swarm optimization algorithm for global optimization, *Intell. Neuroscience* **2016**.

Zheng, H., Zeng, Q., Chen, W., Zhu, H. and Chen, C. (2016). Improved pid control algorithm for quadrotor based on mcs, *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 1780–1783.