Quadcopter Control Approaches and Performance Analysis

Vasco Brito[1,2], Alexandre Brito[1], Luis Brito Palma[1,2] and Paulo Gil[1,2,3]
[1]*Universidade Nova de Lisboa-FCT-DEE, 2829-516 Caparica, Portugal*
[2]*Uninova-CTS, 2829-516 Caparica, Portugal*
[3]*Universidade de Coimbra-CISUC, 3030-290 Coimbra, Portugal*

Keywords: Quadcopter Drone, Kinematic and Dynamic Models, PID Controller, Sliding Mode Controller, Faults and Failures.

Abstract: This article presents the kinematic and dynamic model of a X8 quadcopter, as well as control methodologies based on the PID controller and Sliding Mode Controller. The main contributions are centered on the controllers tuning based on particle swarm optimization algorithm and on the controllers performance comparison for nominal operation and for faulty situations. In order to show the overall performance, simulation results for trajectory and orientation tracking control are presented.

1 INTRODUCTION

All aerial vehicles need to sustain a means to maintain its body aloft, for that purpose several kinds of vehicles were invented to tackle elevation in a completely different manner, for example the air balloon uses temperature to levitate, the air plane uses the pressure on the wings to glide and the helicopter uses the thrusting force of the propellers to hover.

From the various types of aerial vehicles mentioned before, this paper's content will fit in the Vertical Take-Off and Landing (VTOL) sort of aircraft. The methodology for this aerial technology is quite similar to the helicopter's principle, in which both of them hover by pushing the air downwards, (Luukkonen, 2011). The main difference relies on the number of propellers and their respective placement and angle allowing linear and angular movement, thereby granting them the same six degrees of freedom: North (x), East (y), Down (z), Roll (ϕ), Pitch (θ) and Yaw (ψ). In the world of Unmanned Aerial Vehicles (UAVs), the quadcopter's modelling and control are research fields that have been particularly growing. The commercial companies, military and even the tech community have been investing in these research fields, (Merz and Kendoul, 2013). This piece of technology is currently used to facilitate tasks that are simple enough for an unmanned drone to do, specially in the entertainment, filming, surveillance and logistic sectors (ie.:support on human rescue procedures, (TIME, 2015)). Although it looks simple to make a drone hover, move or even complete certain routes, it is not. It has proven to be an extremely hard task to develop a controller that could stabilize a quad-rotor during its complex movements, since this system yields a very fast dynamic behavior and is highly nonlinear.

A study of the most common controllers regarding quacopter flight was done here. In this work, the most used kinds of control approaches were implemented (PID (Mustapa et al., 2014) (Leong et al., 2012), Sliding Mode (Bouabdallah and Siegwart, 2005) (Yih, 2016)), and their behavior regarding faults/failures were evaluated.

A fault is an inconsistency happening in a running dynamical system that deviates its optimal functioning status but could compromise it as a whole (Blanke et al., 2006). A failure is a permanent interruption of a system's ability to perform a required function under specified operating conditions (Isermann and Ballé, 1997). In this research, a fault tolerant control approach was implemented, allowing the prevention of a global system failure.

These controllers were implemented and tested on a quadcopter model with two rotors on each arm antiparallel to each other and rotating in opposite directions. This eight rotor, four armed architecture is designated X8 quadcopter and is illustrated in Fig. 1 and Fig. 2. As it will be explained further in this paper, the main purpose of this architecture is to allow the drone's model, that happens to suffer a fault/failure in sensors or actuators, to keep flying and continue his task. The X8 architecture does not bring efficiency

Brito, V., Brito, A., Palma, L. and Gil, P.
Quadcopter Control Approaches and Performance Analysis.
DOI: 10.5220/0006902600860093
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 1*, pages 86-93
ISBN: 978-989-758-321-6

battery wise but it does bring a lot more safety to the aircraft's structure, as well as acuators redundancy (Brito et al., 2016).

2 MULTICOPTER'S MODEL

The quadcopter's model, based on the real quadcopter Fig. 1, and the techniques used to model its kinematics and dynamics will be briefly explained in this section.

Figure 1: Quadcopter X8-VB.

Each rotors' torque and orientation has to be taken into account to allow the multicopter to hover, move in three dimensions and rotate in the other three degrees of freedom.

$$U_1 = \sum_{n=1}^{8} F_n \tag{1}$$

$$U_2 = f_4 + f_8 - f_2 - f_6 \tag{2}$$

$$U_3 = f_3 + f_7 - f_1 - f_5 \tag{3}$$

$$U_4 = f_1 + f_3 + f_6 + f_8 - f_2 - f_4 - f_5 - f_7 \tag{4}$$

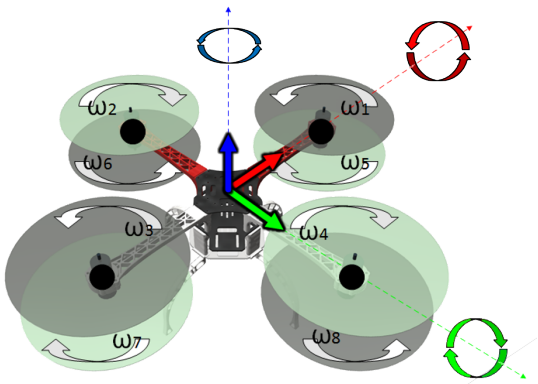

Figure 2: Representation of the X8 quadcopter. The XYZ axis and its rotations are represented with RGB colors, respectively.

Equation (1) represents the sum of forces in all rotors, attaining the variation of altitude (Z axis). The equation (2) refers to the roll (ϕ) rotation and equation (3) the pitch (θ) rotation which enable the multicopter's movement in the y and x axis, represented in Fig. 2 as green and red, respectively. The last equation (4) refers to the yaw (ψ) responsible for the rotation around the Z axis, also represented in Fig. 2 as blue.

To obtain the dynamic model of this aircraft, one must take into consideration its kinematics model, in other words, its linear and angular position (5), as well as its linear and angular velocity (6).

$$\Gamma = \begin{bmatrix} P_E & \Theta_E \end{bmatrix}^T = [x\ y\ z\ \phi\ \theta\ \psi]^T \tag{5}$$

$$v = \begin{bmatrix} V_A & \Omega_A \end{bmatrix}^T = [u\ v\ w\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T \tag{6}$$

These equations represent the twelve states of the system. Using the Newton-Euler method with the kinematic model to describe the combined translation and rotational dynamics of the quadcopter, therefore obtaining its dynamic model represented by the system of equations (7).

$$\begin{cases} \ddot{x} = \Big(sin(\phi)\sin(\psi) - cos(\phi)cos(\psi)sin(\theta)\Big)\frac{U_1 - F_{Dx}}{m} \\ \ddot{y} = \Big(cos(\psi)\sin(\phi) + cos(\phi)sin(\psi)sin(\theta)\Big)\frac{U_1 - F_{Dy}}{m} \\ \ddot{z} = -g + \Big(cos(\phi)\cos(\theta)\Big)\frac{U_1 - F_{Dz}}{m} \\ \ddot{\phi} = \frac{1}{I_{xx}}\Big((I_{yy} - I_{zz})\dot{\theta}\dot{\psi} - J_t\dot{\theta}\omega + U_2\Big) \\ \ddot{\theta} = \frac{1}{I_{yy}}\Big((I_{zz} - I_{xx})\dot{\phi}\dot{\psi} - J_t\dot{\phi}\omega + U_3\Big) \\ \ddot{\psi} = \frac{1}{I_{zz}}\Big((I_{xx} - I_{yy})\dot{\phi}\dot{\theta} + U_4\Big) \end{cases} \tag{7}$$

In the set of equations (7), g is the gravity acceleration, m is its mass, I_{xx}, I_{yy} and I_{zz} represent the body inertia in three dimensions, J_t is the rotor inertia, ω is the rotor rotation and, F_{Dx}, F_{Dy} and F_{Dz} (8) are the drag forces imposed, contrary to its movement.

$$F_D = C\frac{\rho v^2}{2}A \tag{8}$$

In equation (8), C refers to the drag coefficient, ρ the air density, A the impact area with air and v is the speed.

The system of equations (7) obtained represents the linear and angular accelerations of the quadcopter and its position and velocity equations can be attained by means of integration (7). For a better understanding, the $\ddot{x}$ and $\ddot{y}$ accelerations are always dependent on the roll (ϕ) and pitch (θ) but the yaw (ψ) makes its appearance if the quadcopter's referential in relation to the earth axis is changed rotation wise.

In the $\ddot{z}$ acceleration the yaw (ψ) does not influence in any way and the constant acceleration of gravity is introduced. The last part of $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$ equations refers to the mass m of the drone, control input U_1 and their respective drag forces. The angular accelerations are influenced by the drone's inertial momentum (I_{xx}, I_{yy}, I_{zz} and J_t) as well as its control inputs (U_2, U_3 and U_4).

3 CONTROL APPROACHES

This section will further explain the controllers' approaches and algorithms.

3.1 PID Controller

The Proportional-Integral-derivative controller (PID Controller) is a very common solution in feedback control of industrial processes that is meant to force the process variable to follow the reference taking in consideration its present (P), past (I) and future (D) of the error, (Astrom, 1995). The implemented controller's architecture is represented in Fig. 3, where it was added an anti-windup term in order to nullify the integral element's error accumulation.

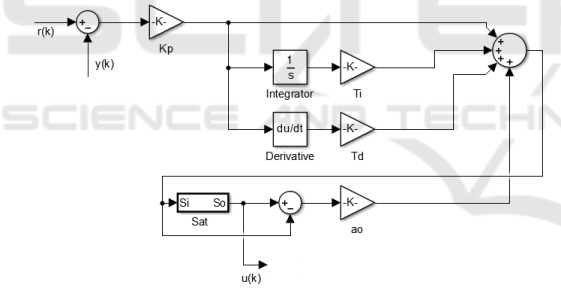

Figure 3: PID with anti-windup architecture (continuous-time).

The implemented PID discrete-time algorithm follows the equations (9) to (18), where $e(k)$ represents the control error between the reference $r(k)$ and the sensor data $y(k)$, bi is the integral gain, ad and bd are derivative gains, ao is the anti-windup gain, K_p is the proportional gain, T_s is the sampling time, T_i is the integral coefficient, T_d is the derivative coefficient, T_t is the anti-windup coefficient, $P(k)$, $I(k)$ and $D(k)$ are the PID controller gains, $v(k)$ is the non-saturated control actuation and $u(k)$ is the control actuation with saturation.

$$e(k) = r(k) - y(k) \tag{9}$$

$$bi = \frac{K_p T_s}{T_i} \tag{10}$$

$$ad = \frac{2T_d - N\,T_s}{2T_d + N\,T_s} \tag{11}$$

$$bd = \frac{2K_p N\,T_s}{2T_d + N\,T_s} \tag{12}$$

$$ao = \frac{T_s}{T_t} \tag{13}$$

$$P(k) = K_p\,e(k) \tag{14}$$

$$D(k) = ad\,D(k-1) - bd\Big(y(k) - y(k-1)\Big) \tag{15}$$

$$v(k) = P(k) + I(k-1) + D(k) \tag{16}$$

$$u(k) = sat\Big(v(k), ulow, uhigh\Big) \tag{17}$$

$$I(k) = I(k-1) + bi\,e(k) + ao\Big(u(k) - v(k)\Big) \tag{18}$$

3.2 Sliding Mode Controller

A Sliding Mode Controller (SMC) is a type of control that consists in the change of the system state using a switching function. This technique is very efficient when dealing with nonlinear systems that can be affected by external disturbances, like aircrafts, (Shtessel et al., 2014), (Spurgeon, 2014), (Utkin et al., 1999). The controller approach implemented is further described by the architecture in Fig. 4 and the implemented discrete-time algorithm is defined by the set of equations (19) to (25), (Palma et al., 2015).

$$e(k) = r(k) - y(k) \tag{19}$$

$$d_e(k) = g_e\frac{e(k) - e(k-1)}{T_s} + p_c\,d_e(k-1) \tag{20}$$

$$g_{aw} = \frac{T_s\,\lambda}{c} \tag{21}$$

$$i_e(k) = i_e(k-1) + T_s\,e(k) + g_{aw}\Big(u(k-1) - v(k-1)\Big) \tag{22}$$

$$\sigma_c(k) = c_e(k) + \lambda\,i_e(k) + d_e(k) \tag{23}$$

$$v(k) = \rho_c\frac{\sigma_c(k)}{|\sigma_c(k)| + \varepsilon_c} \tag{24}$$

$$u(k) = sat\Big(v(k), [0;1]\Big) \tag{25}$$

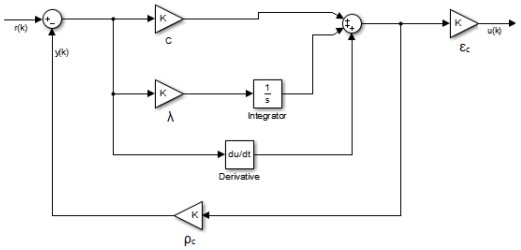

Figure 4: Sliding Mode Controller architecture (continuous-time).

3.3 Cascade Control

In Fig. 5 the cascade control methodology to move the quadcopter along the XY plane is illustrated. The quadcopter will follow the trajectory y_{sp} while comparing with the current y in the outer loop, by giving the setpoint to the attitude controller which will follow the desired orientation in the inner loop. The roll and pitch orientations (inner loop) will affect the movement in the Y and X axis (outer loop), respectively.

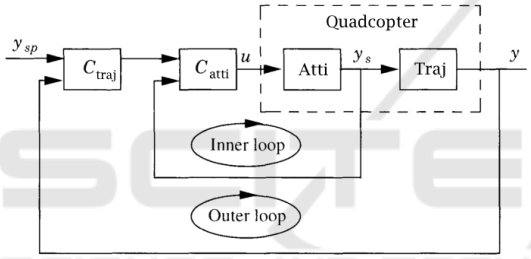

Figure 5: Cascade control for XY plane trajectory.

3.4 Particle Swarm Optimization

The controllers mentioned above were optimized using the Particle Swarm Optimization (PSO) algorithm, (Palma et al., 2015). This method minimizes the cost function by optimizing the control parameters, (Kennedy and Eberhart, 1995). The advantage being that it is not largely affected by the size and non-linearity of the system, while converging to optimal or sub-optimal solutions where most analythical methods fail to converge, (Del Valle et al., 2008).

The particles' respective position $p_i(k)$ and speed $s_i(k)$ on a controller's parameter space are updated through equations (26) and (27), (Kennedy and Eberhart, 1995), where w, c_1 and c_2 are real value parameters, $o_i(k)$ and o_g are the location of the best position of the particle i (history) and the best position found by any particle, respectively. k is the current sample and $rand(.)$ represents a random value in the range [0;1].

$$p_i(k) = p_i(k-1) + s_i(k) \tag{26}$$

$$\begin{aligned} s_i(k) = w\, s_i(k-1) + \\ +c_1\, rand(.)\Big(o_i(k-1) - p_i(k-1)\Big) + \\ +c_2\, rand(.)\Big(o_g(k-1) - p_i(k-1)\Big) \end{aligned} \tag{27}$$

In this work, the cost function $J_c(k)$ to be minimized is defined by equation (28), where $(k-n+1:k)$ represents the set of samples of the active control in a sliding window, *mse* and *var* are the control mean-squared error and the control variance, respectively.

$$\begin{aligned} J_c(k) = \frac{1}{2} mse\big(e(k-n+1:k)\big) + \\ +\frac{1}{2} var\big(u(k-n+1:k)\big) \end{aligned} \tag{28}$$

4 FAULTS AND FAILURES

In this paper, an additive sensor fault was implemented as well as a failure in motor 1, and thereafter the controllers behavior was tested in order to evaluate their robustness.

The actuators failures and its countermeasures could be determined using equations (1, 2, 3 and 4). It is important to note that the system is able to tolerate these failures because of the X8 architecture's fault/failure tolerant proprieties.

Taking into account the studies in (Brito et al., 2016) one can obtain the Table 1 regarding the actuators faults and the action to take in each case, (Brito, 2016).

Table 1: Motor failure and reconfiguration.

Failure	*Reconf.*	*Control Retuning*
Motor 1	Motor 7	Pitch, Yaw
Motor 2	Motor 8	Roll, Yaw
Motor 3	Motor 5	Pitch, Yaw
Motor 4	Motor 6	Roll, Yaw
Motors 1, 3	Motors 2, 4	Pitch, Roll, Yaw
Motors 2, 4	Motors 1, 3	Pitch, Roll, Yaw
Motors 5, 7	Motors 6, 8	Pitch, Roll, Yaw
Motors 6, 8	Motors 5, 7	Pitch, Roll, Yaw

In Table (1), the reconfiguration column shows the rotor(s) to disable in case of the failure in the first column rotor. The third column regards the controllers that need to be adjusted according to each new actuators configuration.

5 SIMULATION RESULTS

The simulation results showing the controllers' behavior will be presented in this section, and so will be

their respective gains. The sampling period used in the simulations was 0.1 *s*. Part of the position and orientation variables are normalized between [-1, 1] and others are between [0, 1].

5.1 Controller's Optimization based on PSO

Fig. 6 illustrates the PSO algorithm's iterations to optimize the gains of a controller. From top to bottom can be observed the altitude($y3$) and its reference($r3$), U1 is the control action, Ha3 is the Harris index (Brito Palma et al., 2013), G1, G2, G3 and G4 are the gains associated with the controller, Co is the value of the cost function, Act is the active controller and Bct is the best controller found at the moment. Fig. 6 presents the PSO algorithm for the sliding mode controller, in which, from 50 controllers the 36th was the best, presenting a cost function value of 2.5393e+15.

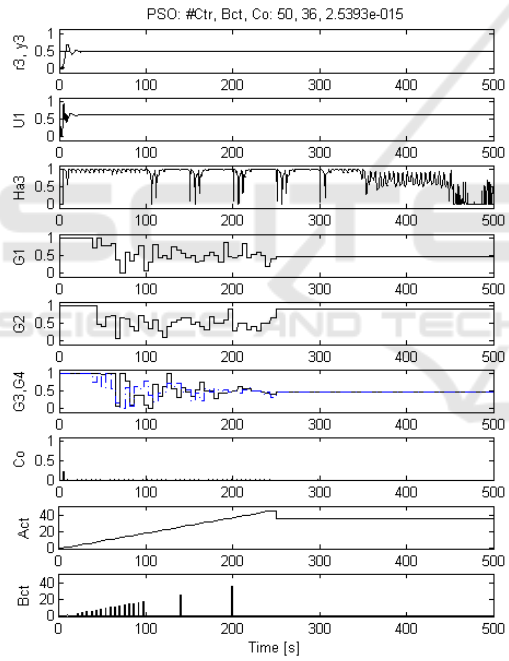

Figure 6: PSO algorithm for the sliding mode controller.

5.2 PID Controller

The PID gains obtained from the PSO optimization process around a setpoint of 0.5 were: $K_p = 2.7272, T_i = 4.224$ and $T_d = 1.0356$.

In Fig. 8 can be observed the movement of the quadcopter along a square trajectory for a constant altitude of 0.5. The movement in the XY plane has a direct correlation with the pitch and roll orientation of the drone. This means that leaning the quadcopter forward and backward (pitch) enables the movement

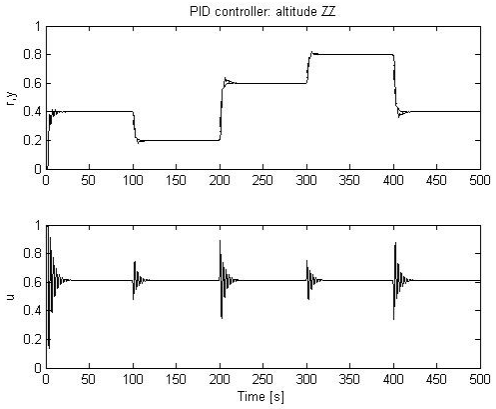

Figure 7: PID simulation result.

in the X axis. Likewise, leaning sideways enables the movement in the Y axis.

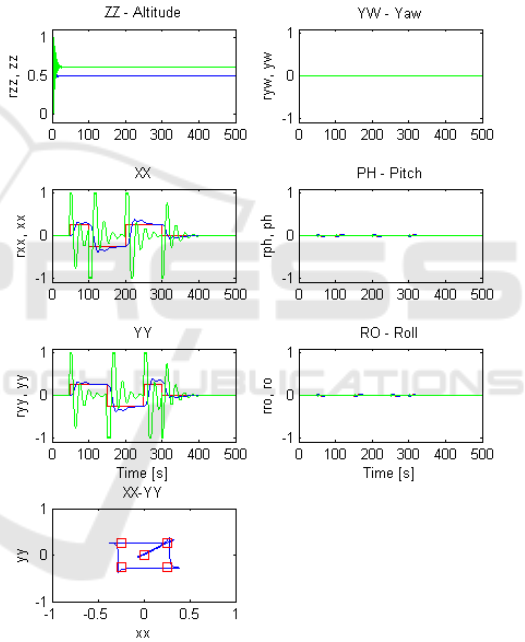

Figure 8: PID simulation results for trajectory and orientation tracking (red is the setpoint, blue is the output and green is the control action.

In Fig. 7 the set of references was [0.4, 0.2, 0.6, 0.8, 0.4], where r (reference) and y (sensor data) are represented in the graphic above and the u (control actuation) below, which represents a nominal hovering situation.

5.3 Sliding Mode Controller

The gains obtained for the SMC controller after optimization around the setpoint of 0.5 were: $c = 7.1163$, $\lambda = 7.7377$, $g_e = 16.4048$, $\rho = 5.3944$, $\varepsilon_c = 18.7194$ and $p_c = 0.2553$.

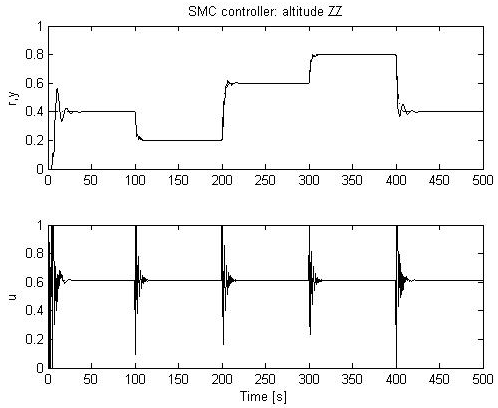

Figure 9: Sliding Mode Controller simulation result.

In Fig. 10 the same trajectory in the XY plane was simulated. The sliding mode controller presents a faster dynamic than the PID due to its nonlinear behavior, as depicted in pitch and roll figures.

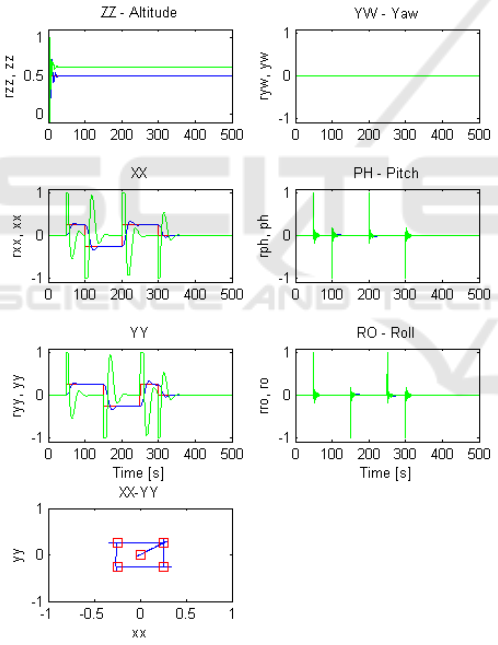

Figure 10: Sliding Mode Controller simulation results for trajectory and orientation tracking.

In the Fig. 9 the r (reference) and y (sensor data) are represented in the graphic above and the u (control actuation) below. The same reference [0.4, 0.2, 0.6, 0.8, 0.4] was defined for this simulation and it also demonstrates a nominal hovering situation.

The Fig. 11 shows the behavior of the SMC when in the presence of a 10% additive abrupt fault above and its respective control actuation below.

In Fig. 12 a fault in motor 1 can be observed as well as its reconfiguration by disabling the motor 7 at

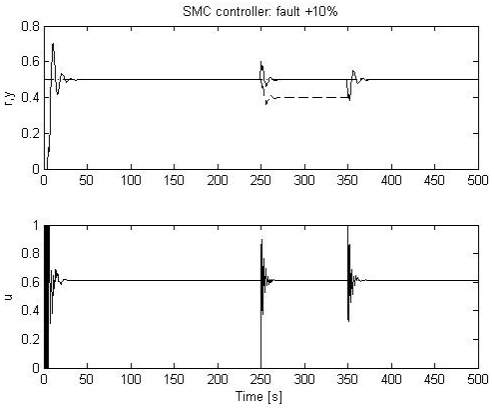

Figure 11: Sliding Mode Controller with fault simulation.

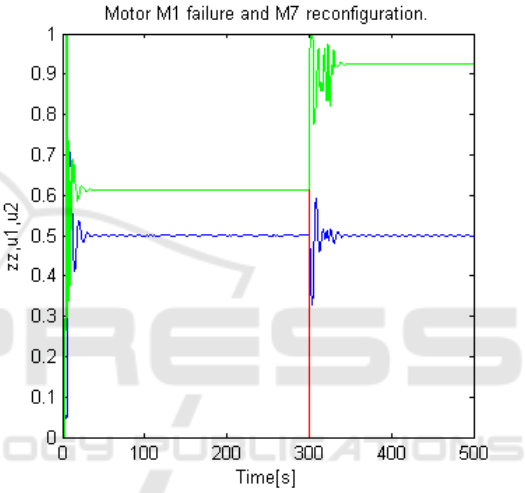

Figure 12: Fault in motor 1 and reconfiguration by disabling motor 7.

300 seconds when the drone loses altitude. After the reconfiguration the thrust lost from the disabled motors has to be compensated by the other motors (their control action increases from 0.6 to 0.9).

5.4 Controllers Performance Comparison

Table 2 presents the performance indexes mean-squared error (*MSE*) and variance of control action (Var_u) in altitude (Fig. 7 and Fig. 9).

Table 3 presents the performance indexes mean-squared error (*MSE*) and the variance of control

Table 2: Controller performance comparison with SP = [0.4 0.2 0.6 0.8 0.4] in altitude.

Controller	*MSE*	*Var_u*
PID	1.0903e-003	2.7262e-003
SMC	2.2237e-003	5.9492e-003

Table 3: Controller performance for sensor fault recovery in altitude.

Controller	*MSE*	*Var_u*
PID	1.5060e-003	2.8384e-003
SMC	3.3218e-003	3.9561e-003

action (Var_u), for an additive faulty sensor situation (+10%).

From the graphics one must observe that both controllers serve their intent maintaining the altitude and responding to the different amplitude steps.

It is important to note that the non-linear SMC controller has more control variance than the linear PID controller because it has a faster response time and has some chattering. The linear PID controller demonstrates a slight better overall performance since it is working around a linear operating point.

Although the controllers performance indexes observed in Table 2 and 3 are similar, one can conclude that, in case of an additive sensor fault, the PID controller shows a better response in relation to the nominal behavior than the SMC. It is also noticeable that, the control variance (*Var_u*) of the SMC is not affected by the sensor fault, it even decreases its value.

Table 4: Controller performance comparison with square setpoint in the XY plane.

Controller	$MSE(X)+MSE(Y)$
PID	0.0297
SMC	0.0217

Without faults, the XY trajectories related with the PID and SM controllers were compared using a performance index given by the sum of the mean square error (MSE) according to the X axis plus the MSE according to the Y axis.

The trajectory tracking of the PID and Sliding Mode controllers present some overshoot, due to their pure feedback architectures, which is not desired for quadcopter path following. Differential Flatness control allows a null overshoot and near perfect tracking of the desired orientations and trajectories, thanks to its feedforward aspect that preemptively calculates the best actuations for each trajectory (Brito et al., 2018).

6 CONCLUSIONS

In this paper, fault tolerant control approaches of a quadcopter model with X8 configuration were presented.

The particle swarm optimization (PSO) algorithm reveals a good approach to obtain sub-optimal controllers, as an optimal one would require a lot of computational time and effort.

According to the simulation results, both controllers (PID and SMC) were able to fulfill their control and tracking tasks.

Regarding the altitude trajectory tracking, both controllers presented similar performance.

The Sliding Mode Controller presents a better performance in the XY trajectory tracking.

The fault tolerant architecture presented an efficient reconfiguration of the system when in faulty situations, in which the quadcopter continues to fly when faults/failures happened in the sensors/actuators.

Future work will take into account the overshoot in the performance index of the PSO optimization in order to improve the trajectory tracking.

ACKNOWLEDGEMENTS

This work has been supported by Departamento de Engenharia Eletrotecnica of Faculdade de Ciências e Tecnologia of Universidade Nova de Lisboa, by Uninova-CTS research unit, by CISUC research unit and by national funds through FCT - Fundacão para a Ciência e a Tecnologia within the research unit CTS - Centro de Tecnologia e Sistemas (project UID/EEA/00066/2013). The authors would like to thank all the institutions.

REFERENCES

Astrom, K. J. (1995). PID controllers: theory, design and tuning.

Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2006). *Diagnosis and Fault-Tolerant Control*.

Bouabdallah, S. and Siegwart, R. (2005). Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2247–2252. IEEE.

Brito, A., Brito, V., Palma, L. B., Coito, F., and Gil, P. (2018). Trajectory control approaches for a fault tolerant quadcopter. In *2018 International Young Engineers Forum (YEF-ECE)*, pages 13–18. IEEE.

Brito, V. (2016). *Fault Tolerant Control of a X8-VB Quadcopter*. Msc, FCT-UNL.

Brito, V., Brito Palma, L. F. F., Vieira Coito, F., and Valtchev, S. (2016). Modeling and Supervisory Control of a Virtual X8-VB Quadcopter. In *17th International Conference on Power Electronics and Motion Control*, page 8, Varna, Bulgária.

Brito Palma, L., Moreira, J., Gil, P., and Coito, F. (2013). Hybrid Approach for Control Loop Performance Assessment. *KES-IDT - 5th International Conference on Intelligent Decision Technologies, 2013*.

Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J.-C., and Harley, R. G. (2008). Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. *Evolutionary Computation, IEEE Transactions on*, 12(2):171–195.

Isermann, R. and Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4:1942—-1948 vol.4.

Leong, B. T. M., Low, S. M., and Ooi, M. P. L. (2012). Low-cost microcontroller-based hover control design of a quadcopter. In *Procedia Engineering*, volume 41, pages 458–464.

Luukkonen, T. (2011). Modelling and Control of Quadcopter. *Journal of the American Society for Mass Spectrometry*, 22(7):1134–1145.

Merz, T. and Kendoul, F. (2013). Dependable low-altitude obstacle avoidance for robotic helicopters operating in rural areas. *Journal of Field Robotics*, 30(3):439–471.

Mustapa, Z., Saat, S., Husin, S. H., and Abas, N. (2014). Altitude controller design for multi-copter UAV. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 382–387. IEEE.

Palma, L. B., Coito, F. V., Ferreira, B. G., and Gil, P. S. (2015). PSO based on-line optimization for DC motor speed control. In *9th International Conference on Compatibility and Power Electronics, CPE 2015*, pages 301–306.

Shtessel, Y., Edwards, C., Fridman, L., and Levant, A. (2014). *Sliding Mode Control and Observation*. Control Engineering. Springer New York, New York, NY.

Spurgeon, S. (2014). Sliding mode control : a tutorial. *Proc. of the European Control Conference (ECC), 2014*, (1):2272–2277.

TIME (2015). How Drones Are Already Being Used to Help Save People.

Utkin, V., Guldner, J., and Shi, J. (1999). Sliding mode control in electro-mechanical systems.

Yih, C.-C. (2016). Flight control of a tilt-rotor quadcopter via sliding mode. In *2016 International Automatic Control Conference (CACS)*, pages 65–70. IEEE.