Pitch and Roll control of a Quadcopter using Cascade Iterative Feedback Tuning

Douglas A. Tesch * **Diego Eckhard** *
William Cechin Guarienti *

* *Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, (e-mail: douglasatesch@gmail.com, diegoeck@ufrgs.br, williamguarienti@gmail.com)*

Abstract: Quadcopter is a type of Unmanned Aerial Vehicle which is lifted and propelled by four rotors. The vehicle has a complex non-linear dynamic which makes the tuning of the roll and pitch controllers difficult. Usually the control design is based on a mathematical model which is strongly related to physical components of vehicle: mass, moment of inertia and aerodynamic. When a tool is attached to the vehicle, a new model must be computed to redesign the controllers. In this article we will adjust the controllers of a real experimental quadcopter using the Cascade Iterative Feedback Tuning method. The method is data-driven, so it does not uses a model for the vehicle; all it uses is input-output data collect from the closed-loop system. The method minimizes the H_2 error between the desired response and the actual response of the vehicle angle using the Newton-Raphson algorithm. The method achieves the desired performance without the need of the vehicle model, with low cost and low complexity.

Keywords: UAVs Applications

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) popularity has grow up in the last two decades. Increasing research in many areas like power-electronics, sensing, micro-controllers, navigation and power supply has reduced the cost of UAVs and improved the safety for commercial and military applications, see (Puri, 2005). In commercial context, there are applications as: supervision of cracks in buildings facades, supervision of high voltage towers, privileged aerial image, topography and plantation monitoring. In military context there are applications as in: fire monitoring, border patrol and aerial monitoring.

Multirotors are UAVs that have low cost and simple construction, what explain the large use in commercial applications and the choice of this research work. There are many multirotors structures with different number of rotors; the most common have 4, 6 and 8 rotors. In this work we choose the most used multirotor structure: it has 4 rotors and it is called *quadcopter*.

The quadcopter has 6 degrees of freedom (6DOF), it can move in the 3 directions of space: x, y an z with respective linear velocities U, V and W. It can also rotate around the three axis with angles: roll ϕ, pitch θ and yaw ψ with respective angular velocities P, Q an R.

This complex non-linear system has been the focus of several control researches as in (Bouabdallah et al., 2004), (Madani and Benallegue, 2006) and (Coza and Macnab, 2006). Although the literature describes all these complex non-linear control algorithms, the most common control structure used for pitch and roll control of quadcopters is the classical linear PID control in cascade format, where there are an inner control loop responsible for angular velocities $P(t)$ and $Q(t)$ and an outer control loop, responsible for the angles $\phi(t)$ and $\theta(t)$, as in Kada and Ghazzawi (2011); Inovan et al. (2014); Mahony et al. (2012); Patel and Barve (2014). The tuning of the PID parameters usually is done using a model of the process, obtained from physical quantities as: mass, size, inertial moment, etc. When the structure of the vehicle changes, for example when a camera is coupled, the system changes and also the model that represents the system. In this case, a new set of parameters for the PID controllers must be calculated, in order to ensure proper operation of the vehicle.

In this work, we propose the use of data-driven techniques to design the controllers of a real and commercial UAV in a practical way. Data-driven methods do not use a model to adjust the controller parameters. Instead, the methods use only sets of input-output data collected from the system. The most used methods are: Iterative Feedback Tuning (IFT) (Hjalmarsson et al., 1998) and Virtual Reference Feedback Tuning (VRFT) (Campi et al., 2002)(Bazanella et al., 2012). Since the controller are implemented in cascade, we propose the use of the Cascade Iterative Feedback Tuning (CIFT), an extension of IFT recently published in (Tesch et al., 2016).

The article is organized as follows. Section 2 describes the quadcopter used in this work. Section 3 describes the cascade structure of the controllers. The Cascade Iterative Feedback Tuning is presented in Section 4. Section 5 presents the methodology used to adjust the controller

⋆ This work was supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), CNPq/Brasil, UFRGS and FAPERGS.

parameters including how to perform the experiments and to validate the controllers. The results are shown in Section 6 and Section 7 concludes the work.

2. THE QUADCOPTER

The quadcopter used in this work has six degrees of freedom, with variables $x(t)$, $y(t)$, $z(t)$, $U(t)$, $V(t)$, $W(t)$, $\phi(t)$, $\theta(t)$, $\psi(t)$, $P(t)$, $Q(t)$ and $R(t)$ represented in Fig. 1. It is composed by several parts which are listed in Table 1 with their respective weight. The total weight of the UAV is $995g$.

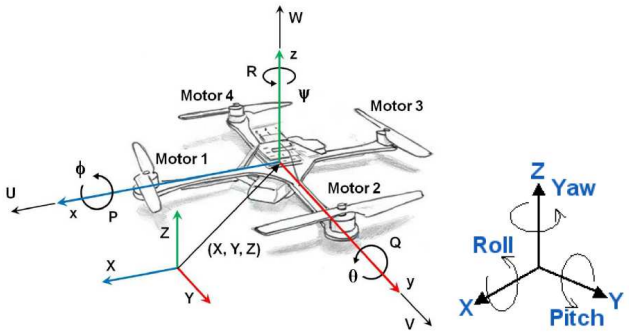

Fig. 1. Quadcopter states representation. Hartman et al. (2004).

Table 1. Quadcopter components and weight.

Component	Weight
Frame	272g
Motors	228g
Propellers	168g
Control Board	14g
Battery	187g
ESCs	76g
Others	50g
Sum	995g

The rotors (motors + propellers) were chosen such that the total thrust should be two times the weight. The motors are 2826 1100kv model from NTM Prop Drive and the propellers are 8×4.5, this propeller have 8" of size and 4.5" of step. Each rotor produces a maximal thrust force of $5.2214N$ totalising $20.8844N$, a little more than double of the weight.

The frame is the popular commercial model F450, largely used by aeromodel amateurs. The control board has a 8 bits micro-controller ATMega328p which runs at $16Mhz$. The inertial measurement unit (IMU) uses a gyroscope ITG3200 and accelerometer BMA180. The board uses a data fusion algorithm to estimate the angular velocities $P(t)$, $Q(t)$ and $R(t)$ and the angles $\phi(t)$, $\theta(t)$, $\psi(t)$. The Electronic Speed Control (ESC) are electronic devices that are responsibility for power supply the brushless motors, the model is SkyWalker-20A of the manufacture Hobbywing. The battery used is a lithium-ion polymer (LiPo) with capacity $2200mAh$.

The objective of this work is to adjust the roll and pitch PID controllers of the UAV using data-driven control methods. In order to run the experiments and collect the sets of data, a platform was used to restrict the movement of the UAV in some directions. This platform is presented in Fig. 2 where the only allowed movement is in the roll angle. In order to increase the angle, the quadcopter should increase the speed of motor 2 and reduce the speed of motor 4.

Fig. 2. The platform of tests.

3. PITCH AND ROLL CONTROL

The main objective of this work is to design PID controllers for *pitch* and *roll* angles. Since the UAV is symmetric, we will only present the design for the roll angle $\phi(t)$, but the same strategy may be used to design the pitch controllers. The roll control systems structure is composed by two controllers, as we can see in Fig. 3.

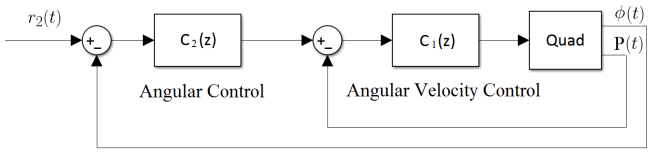

Fig. 3. Roll control system structure.

The IMU provides both angle $\phi(t)$ and angular velocity $P(t)$ for the control system. The angular velocity $P(t)$ is computed directly from the gyroscope while the angle $\phi(t)$ is estimated using a sensor-fusion technique that combines information from the gyroscope and accelerometer named complementary filter. To more detail about data-fusion technique employed in quadcopter, we indicate the following research (Salton et al., 2016). This two quantities ($\phi(t)$ and $P(t)$) are used in a cascade control structure composed by controllers $C_1(z)$ and $C_2(z)$. Controller $C_1(z)$ is responsible for controlling the angular velocity $P(t)$ while controller $C_2(z)$ is responsible for the angle $\phi(t)$. The reference $r_2(t)$ is the reference angle for the system, and usually is chosen by the pilot. The output of controller 2 ($u_2(t)$) is the reference for the inner control loop, while the output of controller 1 ($u_1(t)$) is the differential velocity applied to the motors. The relation between the outputs can be written as

$$u_2(t) = C_2(z)(r_2(t) - \phi(t)) \quad (1)$$

$$u_1(t) = C_1(z)(u_2(t) - P(t)), \quad (2)$$

where z is the forward-time operator $z(x(t)) = x(t+1)$.

Usually both controllers $C_1(z)$ and $C_2(z)$ are discrete-time single-input-output PID controller, which can be written as

$$C_1(z, \rho_1) = \rho_1^T \overline{C}_1(z) \tag{3}$$
$$C_2(z, \rho_2) = \rho_2^T \overline{C}_2(z), \tag{4}$$

where the parameter ρ_1 and ρ_2 are a vectors containing the gains k_p, k_i and k_d and $\overline{C}_1(z)$ and $\overline{C}_2(z)$ are vector of transfer functions. Using this structure the PID controllers can be written as:

$$C_1(z, \rho_1 = [\, k_{p,1} \;\; k_{i,1} \;\; k_{d,1} \,] \begin{bmatrix} 1 \\ \frac{z}{z-1} \\ \frac{z-1}{z} \end{bmatrix}, \tag{5}$$

$$C_2(z, \rho_2) = [\, k_{p,2} \;\; k_{i,2} \;\; k_{d,2} \,] \begin{bmatrix} 1 \\ \frac{z}{z-1} \\ \frac{z-1}{z} \end{bmatrix}, \tag{6}$$

where

$$\rho_1 = \begin{bmatrix} k_{p,1} \\ k_{i,1} \\ k_{d,1} \end{bmatrix}, \; \rho_2 = \begin{bmatrix} k_{p,2} \\ k_{i,2} \\ k_{d,2} \end{bmatrix}, \; \overline{C}_1(z) = \overline{C}_2(z) = \begin{bmatrix} 1 \\ \frac{z}{z-1} \\ \frac{z-1}{z} \end{bmatrix}.$$

4. CASCADE ITERATIVE FEEDBACK TUNING

The data-driven method used in this work is an extension of the classical IFT (Hjalmarsson et al., 1994) for cascade systems, named Cascade Iterative Feedback Tuning (CIFT) (Tesch et al., 2016). The CIFT is an iterative method for tuning the controllers of a cascade system with two loops without the use of any previous knowledge of the system model. The method uses only input and output measured signals in an iterative way for finding the optimal controllers in a H_2 sense. The cost function minimised by the method is given by

$$\begin{aligned} J(\rho_1, \rho_2) =& \|y_d(t) - y_2(t, \rho_1, \rho_2)\|^2 \\ =& \frac{1}{N} \sum_{t=1}^{N} (y_d(t) - y_2(t, \rho_1, \rho_2))^2. \end{aligned} \tag{7}$$

where ρ_1 and ρ_2 are the parameter vectors of the controllers, $y_2(t, \rho_1, \rho_2)$ is the output of the system and $y_d(t)$ is the *desired* output response of the system for some reference signal and N is the number of samples used in the experiment. Therefore, the algorithm CIFT performs a minimization in the cost function $J(\rho_1, \rho_2)$ for finding the parameter vectors ρ_1 and ρ_2 that makes the output of the system $y_2(t, \rho_1, \rho_2)$ the closest possible to the desired response $y_d(t)$.

The CIFT method was design specifically to deal with two problems of the classical IFT algorithm when used in cascade structures. First, the CIFT adjust both controllers $C_1(z)$ and $C_2(z)$ at once, avoiding the need of tuning them separately which would use the double of experiments. Second, the method uses the same cost function to tune both controllers ($J(\rho_1, \rho_2)$) and it therefore achieve an optimum controller, while if each controller was sequentially tuned only a sub-optimal solution would be found. In the sequence we describe how the method uses input-output data from the process to optimise the cost function $J(\rho_1, \rho_2)$.

4.1 The Method

The CIFT was developed for a system in a cascade format conform demonstrated in Fig. 4 where $G_1(z)$ and $G_2(z)$ are the unknown transfer functions of the inner and outer loops, $C_1(z)$ and $C_2(z)$ are the discrete linear and time invariant controllers, $y_1(t)$ and $y_2(t)$ are the inner and outer output responses, $v_1(t)$ and $v_2(t)$ are the zero mean white noises that corrupt the inner and outer loops, $u_1(t)$ and $u_2(t)$ are the control signals of both loops and $r_1(t)$ and $r_2(t)$ are the reference signal for the respective control loops.

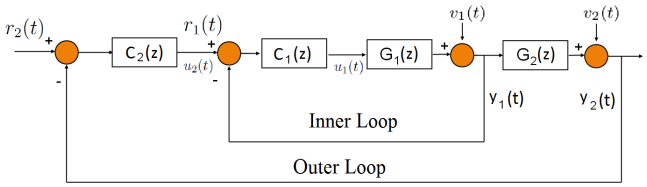

Fig. 4. Example of a system in cascade format.

The CIFT uses the same performance criterion from the classical IFT (the norm H_2), but the cost function for the CIFT now depends on two parameters ρ_1 and ρ_2 (controllers $C_1(z)$ and $C_2(z)$). The cost function for the CIFT algorithm is given by equation (7). Usually the desired response for the system $y_d(t)$ is computed using a desired transfer $T_d(z)$:

$$y_d(t) = T_d(z) r_2(t). \tag{8}$$

such that this transfer function describes the desired relation between the reference signal $r_2(t)$ and the desired output $y_d(t)$. It is common to choose a first or second order transfer function for $T_d(z)$ where $T_d(1) = 1$ such that the closed-loop system presents null steady-state error for constant references.

The CIFT method minimizes the cost function $J(\rho_1, \rho_2)$ using iterative algorithms based on the gradient of the cost function. The most used algorithm is the Newton-Raphson described as

$$\rho^{i+1} = \rho^i - \widehat{H}(\rho^i)^{-1} \widehat{\nabla J}(\rho^i). \tag{9}$$

where

$$\rho = \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix},$$

i is the iteration number, $\widehat{\nabla J}(\rho^i)$ is an estimate of the cost function gradient and $\widehat{H}(\rho^i)$ is an estimate of the cost function Hessian matrix. This algorithm is known for the fast convergence to the minimum of the criterion, when initialized close to the minimum. Since the cost function $J(\rho_1, \rho_2)$ is not convex, it may occur that the algorithm do not find the minimum, but this topic will not be covered in this article. A profound convergence analysis of the Newton-Raphson algoritm in data-driven control can be found in the book (Bazanella et al., 2012) which describes the algorithm and several tools to improve its convergence.

In order to use the Newton-Raphson algorithm, the user must estimate the cost function gradient and Hessian matrix, and the CIFT method describes a methodology to estimate these quantities which only uses input-output closed-loop data from the systems, without the need of a process model. In the sequence we describe the three experiments that need to be run in order to collect the input-output data, and how to estimate the cost function gradient and Hessian.

4.2 CIFT algorithm

The CIFT is an iterative method that can use the Newton-Raphson algorithm to minimize the cost function $J(\rho_1, \rho_2)$. The cost function gradient and Hessian matrix are estimated only using collect data from 3 closed-loop experiments. At each iteration the following steps must be executed:

- Excite the system in closed loop with $r_2(t) = r^1(t)$ and collect both outputs $y_1(t)$ and $y_2(t)$ named $y_1^1(t)$ and $y_2^1(t)$ respectively.
- Calculate the error of this experiment as: $r_2^2(t) = r^1(t) - y_2^1(t)$.
- In a second experiment, excite the process in closed-loop with the computed reference $r_2^2(t)$. Collect again the both outputs and name them y_1^2 e y_2^2 respectively.
- Run a third experiment with the same reference of the first, in other words, $r_2(t) = r^1(t)$. Collect the output sinal of the outer loop and name it $y_2^3(t)$.
- Use the data collected to compute the following estimates

$$\widehat{\frac{\partial y_2(t)}{\partial \rho_1}} = \frac{1}{C_1}\frac{\partial C_1}{\partial \rho_1}\left[y_2^2(t) - \frac{1}{C_2}\left(y_1^1(t) - y_1^2(t)\right)\right] \quad (10)$$

$$\widehat{\frac{\partial y_2(t)}{\partial \rho_2}} = \frac{1}{C_2}\frac{\partial C_2}{\partial \rho_2}\left[y_2^2(t)\right] \quad (11)$$

$$\widehat{\frac{\partial y_2(t)}{\partial \rho}} = \begin{bmatrix} \widehat{\frac{\partial y_2(t)}{\partial \rho_1}}^T & \widehat{\frac{\partial y_2(t)}{\partial \rho_2}}^T \end{bmatrix}^T. \quad (12)$$

- Estimate the cost function gradient:

$$\widehat{\nabla J}(\rho^i) = \frac{2}{N}\sum_{t=1}^{N}[y_d(t) - y_2^3(t)]\widehat{\frac{\partial y_2(t)}{\partial \rho}}$$

- Estimate the cost function Hessian:

$$\widehat{H}(\rho^i) = \frac{2}{N}\sum_{t=1}^{N}\widehat{\frac{\partial y_2(t)}{\partial \rho}}\widehat{\frac{\partial y_2(t)}{\partial \rho}}^T \quad (13)$$

- Compute the new controller parameters using the Newton-Raphson algorithm:

$$\rho^{i+1} = \rho^i - \widehat{H}(\rho^i)^{-1}\widehat{\nabla J}(\rho^i). \quad (14)$$

After each iteration a new set of controller parameters (ρ_1 and ρ_2) is obtained. If the closed-loop response is not satisfactory, the user must run another iteration of the method to compute a better solution to the optimisation problem. The procedure can be repeated until a satisfactory solution is obtained.

5. QUADCOPTER CONTROLLER TUNING METHODOLOGY

This section has as objective to describe the methodology used to tuning the controllers of a real quadcopter using CIFT. The proposed methodology guides the designer to improve the performance of the quadcopter and reduce the time spent tuning the quadcopter controller.

The methodology is simple and practical to be used. It is simple because it only needs input-output data from experiments and do not need advanced knowledge of control for tuning the algorithm. The user just defines the desired response that he desires. It is practical because achieves good performance with little time spent. The proposed quadcopter controller tuning methodology is the following:

(1) **Fly**: In a normal operational condition, do a free fly with a stable controller and be aware to any undesirable performance.
(2) **Bad Performance or New Device**: If it is observed that the performance is not good enough for the application, it is recommended to stop the flight and to proceed the controller tuning. Also, when a new device is attached to the quadcopter (a camera for example), the quadcopter dynamics will change, and we also recommended to proceed the controller tuning.
(3) **CIFT Controller Tuning**: To realize the controller tuning the user must hold the quadcopter conform the platform in Fig 2. The platform used in the experiments is simple and low cost. Also, the user can improvise one, for example, hanging the vehicle between two trees. The user must choose a desired response and run the three experiments to improve the performance of the control. If the achieved performance is not satisfactory, another iteration should be performed using more three experiments.
(4) **Test the New Controllers**: Again, in a normal operational condition, execute a free fly with the quadcopter and verify if the performance is satisfactory. In case the of unacceptable performance, stop the fly and go back to **CIFT Controller Tuning**. At this point, it may be necessary to use a more complex controller structure. This procedure can be repeated until the desired performance is reached.

6. RESULTS

In this section, the application of the proposed methodology to a real quadcopter is described. The quadcopter has a cascade structure of control, therefore the algorithm will tune both inner and outer controllers concomitantly. Initially two proportional controllers were used for the inner and outer loops:

$$\begin{aligned} C_1(z, \rho_1) &= \rho_1 = k_{p,1} \\ C_2(z, \rho_2) &= \rho_2 = k_{p,2} \end{aligned} \quad (15)$$

such that

$$\frac{\partial C_1}{\partial \rho_1} = 1 \quad (16)$$

$$\frac{\partial C_2}{\partial \rho_2} = 1 \quad (17)$$

since the controllers are proportional and $\rho_1 = k_{p,1}$, $\rho_2 = k_{p,2}$. The sample time of the controller is $10ms$ such that the signals are sampled 100 times each second.

The reference signal is a sine with period of 5 seconds (500 samples). This reference signal is smooth and can represent the kind of references a pilot choose to guide the vehicle.

$$r_2(t) = 20sin\left(\frac{2\pi t}{5}\right) \tag{18}$$

This signal will be applied to the vehicle during 30 seconds (3000 samples).

We choose a second order reference model with unit steady-state gain:

$$T_d(z) = \frac{(1-0.94)^2 z}{(z-0.94)^2}.$$

This reference model is used to compute the desired reference signal:

$$y_d(t) = \frac{(1-0.94)^2 z}{(z-0.94)^2} r_2(t).$$

The reference signal and the desired output can be viewed in Fig 5.

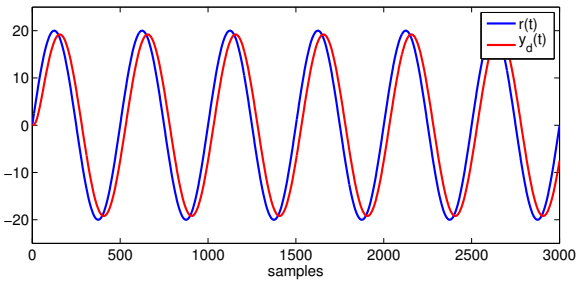

Fig. 5. Reference signal (blue) and desired output response (red).

Initially the system was using the following controllers:

$$C_1(z, \rho_1) = 0.1000$$
$$C_2(z, \rho_2) = 1.0000 \tag{19}$$

which render a bad performance as we can see in Fig. 6.

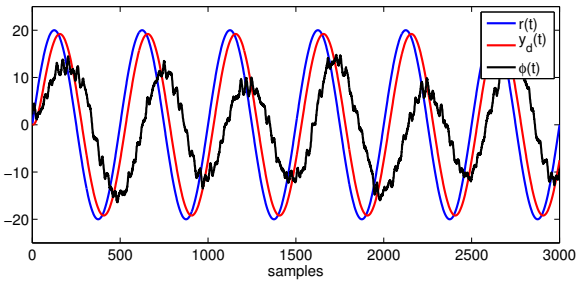

Fig. 6. Reference signal (blue), desired output response (red) and actual roll angle (black) at first experiment of first iteration.

The quality of these controllers can be measured with the root mean square error:

$$J_{RMS} = \sqrt{\frac{1}{3000}\sum_{i=1}^{3000}\left(y_d(t) - \phi(t)\right)^2}$$

Table 2. Cost function and parameters at each iteration

Iteration	J_{RMS}	$C_1(z)$	$C_2(z)$
1	11.054	0.1000	1.0000
2	7.834	0.1005	1.5039
3	5.298	0.1011	2.2472
4	3.021	0.1014	3.1623
5	1.855	0.1017	3.9936
6	1.162	0.1018	4.6963
7	0.975	0.1020	5.0329
8	0.972	0.1019	4.9806

Using the initial controller the error was computed as $J_{RMS} = 11.054$ degrees which is pretty large.

The Cascate Iterative Feedback Tuning method was then applied to the vehicle to improve the performance of the controllers. At each iteration, three experiments are used. The output of the second experiments for the first iteration can be seen in Fig 7. We can see that the second experiment is similar to the first and third experiments. The third experiment is pretty close to the first experiment.

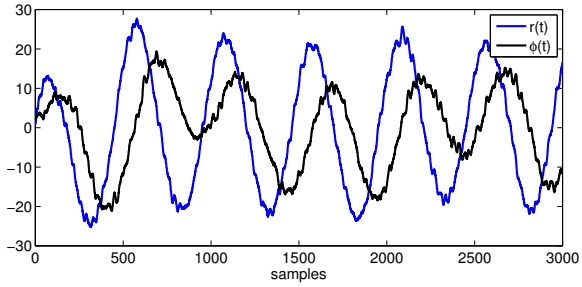

Fig. 7. Reference signal (blue) and actual roll angle (black) at second experiment of first iteration.

Eight iterations of the method were performed, where each iteration used three experiments, totalising 24 experiments of 30 seconds each. The total time to tune the controllers is about 15 minutes. The obtained controllers at each iteration can be seen in Table 2 together with the obtained cost value. After 8 iterations the cost was reduced from 11.054 degrees to 0.972 degrees.

The controllers at the eight iteration are

$$C_1(z, \rho_1) = 0.1019$$
$$C_2(z, \rho_2) = 4.9806 \tag{20}$$

which render a really good performance, as we can also see in Fig 8. In this figure, the desired response is hidden behind the obtained response for the roll angle.

As the response was really satisfactory, there is no need to implement more complex controllers as PD or PID. The proportional controllers are enough to achieve the desired response.

7. CONCLUSION

This article described the use of Cascade Iterative Feedback Tuning to adjust the pitch and roll controllers of a quadcopter. The CIFT is a data-driven method that does not use a model of the process to tune the controllers. All the needed information comes from closed-loop data collected from the vehicle. The method uses

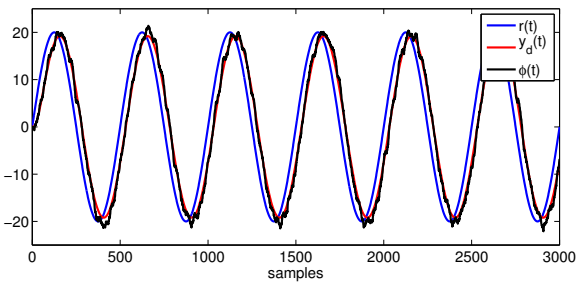

Fig. 8. Reference signal (blue), desired output response (red) and actual roll angle (black) at first experiment of 8th iteration.

the Newton-Raphson algorithm to minimise an H_2 cost function. The only parameter the user need to specify is the desired output response for the closed-loop system. The method was applied to a real quadcopter attaining really good performance. The mean error was reduced from 11.054 degrees to 0.972 degrees using only proportional controllers. The method is suitable for tuning controllers of quadcopters, achieving the desired performance with low cost and without the need of complex models.

REFERENCES

Bazanella, A.S., Campestrini, L., and Eckhard, D. (2012). *Data-driven Controller Design: The H_2 Approach*. Springer, Netherlands. doi:10.1007/978-94-007-2300-9.

Bouabdallah, S., Noth, A., and Siegwart, R. (2004). Pid vs lq control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, volume 3, 2451–2456. IEEE, New York.

Campi, M.C., Lecchini, A., and Savaresi, S.M. (2002). Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, 38(8), 1337–1346.

Coza, C. and Macnab, C. (2006). A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization. In *Fuzzy Information Processing Society (NAFIPS), Annual meeting of the North American*, 454–458. IEEE, New York.

Hartman, D., Landis, K., Mehrer, M., Moreno, S., and Kim, J. (2004). Quadcopter dynamic modeling and simulation. *In: freeware project presented at "2014 MATLAB and Simulink Student Design Challenge"*.

Hjalmarsson, H., Gevers, M., Gunnarsson, S., and Lequin, O. (1998). Iterative feedback tuning: theory and applications. *IEEE control systems magazine*, 18(4), 26–41.

Hjalmarsson, H., Gunnarsson, S., and Gevers, M. (1994). A convergent iterative restricted complexity control design scheme. In *Proc. 33rd Conference on Decision and Control*, 1735–1740. New York: IEEE, Lake Buena Vista, Fl, USA.

Inovan, R., Ataka, A., Tnunay, H., Abdurrahman, M.Q., Cahyadi, A., and Yamamoto, Y. (2014). A cascade controller for linearized quadrotor model. In *Advanced Robotics and Intelligent Systems (ARIS), 2014 International Conference on*, 161–164. IEEE.

Kada, B. and Ghazzawi, Y. (2011). Robust pid controller design for an uav flight control system. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 2, 19–21.

Madani, T. and Benallegue, A. (2006). Backstepping control for a quadrotor helicopter. In *Intelligent Robots and Systems (RSJ), International Conference on*, 3255–3260. IEEE, New York.

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.

Patel, K. and Barve, J. (2014). Modeling, simulation and control study for the quad-copter uav. In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, 1–6. IEEE.

Puri, A. (2005). A survey of unmanned aerial vehicles (uav) for traffic surveillance. *Department of computer science and engineering, University of South Florida*.

Salton, A.T., Eckhard, D., Flores, J.V., Fernandes, G., and Azevedo, G. (2016). Disturbance observer and nonlinear damping control for fast tracking quadrotor vehicles. In *IEEE Multi-Conference on Systems and Control*. IEEE, Buenos Aires.

Tesch, D., Eckhard, D., and Bazanella, A.S. (2016). Iterative feedback tuning for cascade systems. In *European Control Conference*. Denmark.