A numerical approach for attitude control of a quadrotor

Huu-Phuc Nguyen, Jérôme De Miras*, Ali Charara and Stéphane Bonnet
Sorbonne universités, Université de Technologie de Compiègne,
CNRS, Heudiasyc UMR 7253, CS 60319, 60203 Compiègne CEDEX

ABSTRACT

This paper deals with a numerical approach to control aggressive maneuvers of a multi-rotor aerial vehicle. The proposed controller uses an approximate tabulated one-step time discretization of the state-space model to find out the outputs of controller. It objective is to minimize the distance between the plant output and a linear well chosen closed loop system used as reference, leading the system to adopt its dynamical behavior. The prediction horizon is only one step time that ensures the execution time is completely bounded. The results from simulation for quadrotor show the performance and robustness of the proposed controller.

1 INTRODUCTION

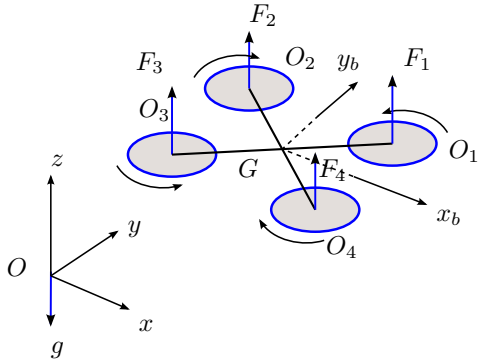

Figure 1: Quadrotor model

A quadrotor is a flying vehicle with four rotors. Figure 1 presents a scheme of the system in a 3D space. The rotor i rotates at speed ω_i and generates a lift force F_i and a drag torque. The quadrotor is an under-actuated system with four inputs and six degrees of freedom. The control problem for the quadrotor is usually divided into two stages: the attitude tracking control and the stabilization of the position. Usually, the position controller generates a desired attitude for the attitude controller. To provide solutions, a lot of control techniques were used: classic PID controllers, adaptive controllers, predictive controllers, controllers based on the Lyapunov criterion, etc.

Looking at the challenge of control of the quadrotor in the case of aggressive maneuvers in the literature, the following works contain many interesting points for studying and comparing. An open loop control was proposed in [1] using reachable sets where the complex aerobatic flights were decomposed into sequences of discrete maneuvers. A trajectory generation for multiple-flips has been proposed in the work of [2] using a simple learning approach. A survey of the methods for attitude control of a rigid body can be found in [3]. The minimal time problem has been considered in the work of [4]. In [5], a quadrotor with flip back behavior was used; the trajectory generation was solved in order to apply a state feedback controller based on a Lyapunov function. A full quaternion based attitude control for quadrotor was introduced in [6]. The authors in [7] designed a controller for flip control on Lie group SE(3). A learning control used in [8] allows the quadrotor doing aggressive maneuvers. The work in [9] have applied an optimal control LQR.

In this context, this paper proposes an attitude control for a quadrotor using a numerical approach. This controller uses a tabulated numerical model to represent the dynamics of the system as a prediction map over one sample time step. Based on this prediction map, the inputs of the system will be calculated upon the admissible input space as a solution of the minimization problem of the difference between the desired output and the predicted output. The rest of the paper will be organized as follows: firstly, the next section deals with the quadrotor dynamics model, then the scheme of the proposed control will be introduced as its application for attitude control of the quadrotor. Finally some results from MATLAB/SIMULINK will be presented.

2 QUADROTOR MODEL

2.1 *Mathematical model*

In the following, vectors and matrices are denoted in bold font. The quadrotor depicted on the Figure 1 is operated by changing the speeds of its four rotors. Using the equation of Newton-Euler, the system is given by the following equations in the body frame $Gx_by_bz_b$:

$$\begin{cases} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= f\mathbf{R}(q)\mathbf{e}_3 - mg\mathbf{e}_3 \\ \dot{q} &= \frac{1}{2}\Xi(q)\Omega \\ \mathbf{J}\dot{\Omega} &= (\mathbf{J}\Omega)\times\Omega + \Gamma \end{cases} \tag{1}$$

with $q = [q_0, q_1, q_2, q_3]^T$ a unit quaternion and

$$\Xi(q) = \begin{bmatrix} -q_1, q_0, -q_3, q_2 \\ -q_2, q_3, q_0, -q_1 \\ -q_3, -q_2, q_1, q_0 \end{bmatrix}^T$$

*Email address(es): (huu-phuc.nguyen,demiras,ali.charara,bonnetst)@hds.utc.fr

http://www.imavs.org/

http://www.imavs.org/

The variables and the parameters of the quadrotor are described in the following table:

G	Center of mass
$\mathbf{x} = [x, y, z]^T$	Position of quadrotor
$\mathbf{v} = [V_x, V_y, V_z]^T$	Velocity of the quadrotor
m	Mass of the quadrotor
$\mathbf{J}$	Inertia of the quadrotor
ω_i	i-rotor's speed
ω_{max}	Maximum speed of the rotors
Ω	Angular velocity
g	Gravity
$\mathbf{R}(q)$	Rotation matrix
$\mathbf{e}_3 = [0, 0, 1]^T$	Unit vector
k_t	Thrust coefficient
k_c	Drag torque coefficient
L	Half-distance between F_1F_2

Table 1: Notations

The thrust force and the moments applied on the main body are calculated with the speeds of the rotors as follows:

$$\begin{bmatrix} f \\ \Gamma_x \\ \Gamma_y \\ \Gamma_z \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ k_tL & k_tL & -k_tL & -k_tL \\ -k_tL & k_tL & k_tL & -k_tL \\ -k_c & k_c & -k_c & k_c \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2}$$

The skew-matrix operation of a vector $\mathbf{x} = [x_1, x_2, x_3]^T$ represents an easy way to compute the cross product of two vectors, $\mathbf{x} \times \mathbf{y} = [\mathbf{x}]^\times \mathbf{y}$.

$$[\mathbf{x}]^\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

The inverse operation is denoted as vee-operation such as if $[\mathbf{x}]^\times = \mathbf{X}$ then $\mathbf{X}^\vee = \mathbf{x}$. Considering only the attitude of the quadrotor and the vertical dynamic, the equation above can be written as (3).

$$\begin{cases} \dot{v_z} &= f\mathbf{R}(9) - mg \\ \dot{\mathbf{R}} &= \mathbf{R}[\Omega]^\times \\ \mathbf{J}\dot{\Omega} &= (\mathbf{J}\Omega) \times \Omega + \Gamma \end{cases} \tag{3}$$

where $\mathbf{R}(9)$ is the last item of the rotation matrix $\mathbf{R}$ and $\Gamma = [\Gamma_x, \Gamma_y, \Gamma_z]^T$. The system becomes a full actuated system with four inputs and four degrees of freedom.

2.2 *Full quadrotor model*

There are many toolkits and drag-drop environments to model the dynamics a vehicle using the multibody dynamics theory, for example SimMechanics from Mathworks, MapleSim from MapleSoft. In this paper, we used a quadrotor Parrot-AR2 model built with MapleSim depicted on Figure 2. This model contains a main rigid body, four arms with four brushless motors integrating four propellers to generate the aerodynamics forces. This model also includes some virtual sensors to obtain the quadrotor's state: the position, the velocity, the orientation and the angular velocity. This model is exported to SIMULINK's environment.

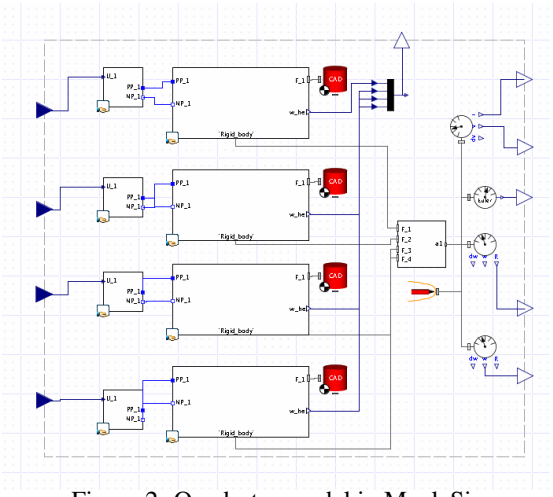

Figure 2: Quadrotor model in MapleSim

The parameters of the quadrotor are given in Table 2.

Parameters	Value	Unit
m	0.506	kg
I_x	2.38×10^{-3}	$kg.m^2$
I_y	3.85×10^{-3}	$kg.m^2$
I_z	5.9×10^{-3}	$kg.m^2$
L	0.15	m
k_t	2.3×10^{-5}	
k_c	2×10^{-6}	
ω_{max}	350	$rad.s^{-1}$

Table 2: Quadrotor parameters

3 PROPOSED CONTROL SCHEME

3.1 *Online control algorithm*

The time-invariant model of of the system has the following form:

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= h(\mathbf{x}(t)) \end{cases} \tag{4}$$

where $\mathbf{x} \in \mathcal{S} \subset \mathbb{R}^n$ is the system state vector, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^d$ its input control vector, and $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^m$ its output vector. The discretized form of the system is needed in order to construct a discrete-time control law. Generally, it is difficult to obtain an exact analytic expression for the discretization of the system. However, the numerical integration of f gives its approximate value for a given state and input vector as the form:

$$\begin{cases} \mathbf{x}_{k+1} &= p_x(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{y}_{k+1} &= p_y(\mathbf{x}_k, \mathbf{u}_k) \end{cases} \tag{5}$$

http://www.imavs.org/

using a sampling time Δt such as $t_k = k \cdot \Delta t$. Doing the prediction evaluations $(p_x,\ p_y)$ at each needed time by using a numerical integration of (4) would be computationally expensive which is problematic in a real-time implementation. A better approach is to build off-line this prediction in a map using a regular rectangular grid $\mathcal{G}$ constructed on the joint state and input spaces $\mathcal{S} \times \mathcal{U}$. For each point of $\mathcal{G}$, the equation of the system is solved over one sample time step using a simulation tool such as Simulink or using an ordinary differential equation solver such as Runge-Kutta. The solution vectors are then stored in a table (the map). The values of $p_x,\ p_y$ at unknown points can be interpolated from that table by using interpolation technique. The barycentric linear interpolation as described in [10] has been chosen for instance. The linear interpolation leads to errors proportional to the square of the grid size but it presents a low computing effort needed to get the interpolated values.

The aim of the controller is to make the error $\hat{\mathbf{y}} = \mathbf{y} - \mathbf{y}_c$ between the plant output $\mathbf{y}$ and a given set point $\mathbf{y}_c$ to be driven to zero. Defining the state $\dot{\hat{\mathbf{x}}}^g$ of a homogeneous stable linear system:

$$\dot{\hat{\mathbf{x}}}^g(t) = \mathbf{A} \cdot \hat{\mathbf{x}}^g(t), \hat{\mathbf{x}}^g \in \mathbb{R}^n, \mathbf{A} \in \mathcal{M}^{n \times n}. \tag{6}$$

and $\hat{\mathbf{x}}^g = \mathbf{x}^g - \mathbf{x}_c$ with $\mathbf{x}^g$ an internal state reference, the output $\mathbf{y}^g$ of the corresponding system will converge to the given set point $\mathbf{y}_c$. The Hurwitz matrix $\mathbf{A}$ is selected according to the known dynamics of the system to control. A discrete representation of the internal reference $\mathbf{x}^g_{k+1}$ at step $k+1$ can be computed from (6) at step k as:

$$\mathbf{x}^g_{k+1} = e^{\mathbf{A} \cdot \Delta t}(\mathbf{x}_k - \mathbf{x}_c) + \mathbf{x}_c. \tag{7}$$

Consider now that one of the objective of the control is to reduce as mush as possible the distance between $\mathbf{x}^g$ and $\mathbf{x}$ at each step time, as in a sliding mode, and $\mathbf{x}^g$ converges naturally to $\mathbf{x}_c$ then we can say that $\mathbf{y}$ will converge to $\mathbf{y}_c$.

To construct the algorithm, a vector $\mathbf{v} = \left[\mathbf{x}^T, \mathbf{y}^T\right]^T$ is used. The state vector $\mathbf{x}$ and the output vector $\mathbf{y}$ are then extracted from $\mathbf{v}$:

$$\mathbf{x} = \mathbf{P}_x \cdot \mathbf{v}, \mathbf{y} = \mathbf{P}_y \cdot \mathbf{v}$$

At step $k+1$ this vector is

$$\mathbf{v}_{k+1} = \left[p_x(\mathbf{x}_k, \mathbf{u}_k)^T, p_y(\mathbf{x}_k, \mathbf{u}_k)^T\right]^T = p(\mathbf{x}_k, \mathbf{u}_k). \tag{8}$$

The objective of the controller is to find at each time-step k a control vector $\mathbf{u}_k$ that will make the next-step plant output $\mathbf{y}_{k+1}$ be as close of $\mathbf{y}^g_{k+1}$ as possible:

$$\mathbf{u}_k = \arg\min_{u \in \mathcal{U}} \left\| \mathbf{y}^g_{k+1} - \mathbf{P}_y \cdot p(\mathbf{x}_k, \mathbf{u}) \right\|_2. \tag{9}$$

Due to the interpolation error, the modeling error in (4) and the unknown disturbance, there exists a prediction error between the predicted plant state and the outputs at step k obtained from data at step $k-1$ and their estimations obtained from the measurements at step k. The error signal for the output predictions is handled similarly and the error vector for the combined vector $\mathbf{v}_k$ is defined by:

$$\epsilon_k = p(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) - \mathbf{v}_k.$$

The use of the prediction map to calculate the input value needs compensate this error. The error dynamics is supposed slow relative to the controller. But in the case of noisy conditions, the error signal contains high frequency. Compensating for high frequency errors can indeed lead to oscillations and controller instability. To overcome this phenomena, a low-pass filter is used to remove the high frequency. The error signal for the state prediction used at step k can be written as:

$$\varepsilon_k = \varepsilon_{k-1} + \alpha \epsilon_k \tag{10}$$

with α a damping coefficient in $(0, 1]$, forming a numerical low-pass filter.

The complete algorithm for closed-loop system from [11] is shown in Figure 3.

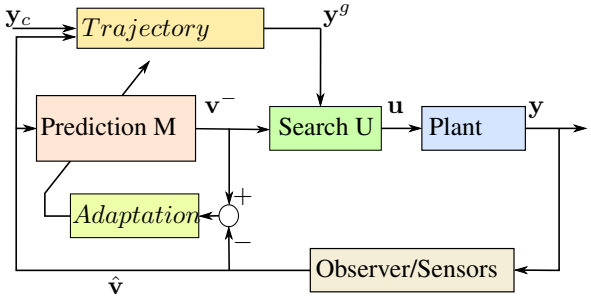

Figure 3: The closed - loop system

At step $k+1$ the output $\mathbf{y}_k$ is measured. Then the block "$Observer/Sensors$" is used to estimate state and output vector and returns $\hat{\mathbf{v}}_k$ an estimation of $\mathbf{v}_k$. After that the error ε_k and the prediction $\mathbf{v}^-_k$ of $\mathbf{v}$ is calculated by the block "$Adaptation$" using (10) and the block "$Prediction\ M$" using (8) respectively, allowing computation of the internal reference $\mathbf{x}^g_{k+2}$ following Eq.(7) by the block "$Trajectory$". Finally, the input vector is calculated using the block "$Search\ U$" that contains the algorithm to find the approximate solution of (9). In a perfect world, the control input $\mathbf{u}_k$ is supposed to be readily computed and immediately applied to the plant from the measurements at step k without any delay of the state estimation. In a real time system, some methods were proposed to compensate for a delay between the output control and the measurement, for example [12].

3.2 Optimization algorithm

As mentioned before, the expression for p is unknown then the minimization problem (9) has to be solved using a

http://www.imavs.org/

derivative free optimization solver. In this subsection an iterative algorithm will be used to approximate the solution of (9). This algorithms starts by using the input interval $\mathcal{U} = [\underline{\mathbf{u}}, \overline{\mathbf{u}}]$ that is used to build the prediction map.

$$\mathcal{U}_{i\in 1,\ldots,d} = \left\{ u_j | u_j = \underline{u}_i + j\frac{\overline{u}_i - \underline{u}_i}{N_i}, j \in 0,\ldots,N_i \right\}$$

The input space is divided using triangularization technique that is based on the simplex and the affine-envelope of a set of vector. These simplexes are also used at the interpolation step to calculate the barycentric coordinates of a point that is not belong to the rectangular grid $\mathcal{G}$. In this algorithm, the simplexes and the affine-envelope are defined as:

$$\textbf{conv}\,(\mathcal{X}) = \left\{ \sum_{i=1}^{k} \omega_i \cdot \mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, \omega_i \in \mathbb{R}^+, \sum_{i=1}^{k} \omega_i = 1 \right\}$$

$$\textbf{aff}\,(\mathcal{X}) = \left\{ \sum_{i=1}^{k} \omega_i \cdot \mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}, \omega_i \in \mathbb{R}, \sum_{i=1}^{k} \omega_i = 1 \right\}$$

The algorithm based upon the work presented in [11], [13] is illustrated on Figure 4. The idea of this algorithm is to find out a simplex σ_f in the inputs space containing the solution of the problem (9) that minimizes the distance of the corresponding simplex Γ_f in the output space to the reference point $\mathbf{y}^g$. Firstly, one input simplex σ_0 is chosen permitting to calculate its output simplex Γ_0. A quasi-gradient is then built based on the projection of $\mathbf{y}^g$ onto Γ_0. The simplex σ_0 is reflected about its edge satisfying the output simplex moves toward the reference point $\mathbf{y}^g$ while keeping its beside the input space. This procedure is repeated until the final simplex σ_f is found out. Finally, the control input is calculated as an orthogonal projection of this point onto the solution simplex σ_f. To compensate the error of the prediction map ε_k, the output simplex Γ is calculated by $\Gamma = \mathbf{P}_y.p(\sigma, \mathbf{u}) - \varepsilon_k$.

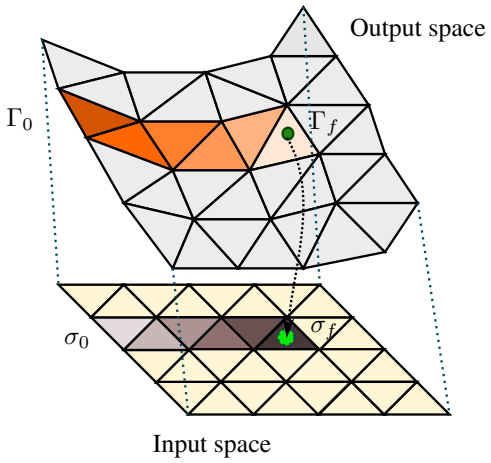

Figure 4: Optimal algorithm visualization

3.3 Implementation of the algorithm for the quadrotor

For the implementation of the algorithm, the prediction map should be built from the equation dynamics of the system. The reference equation is (3). The difficulty is that the orientation represented by the unit quaternion contains a constraint. The norm of the quaternion is equal to one $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$, so that, it is not possible to use directly q as four independent elements. In the other hand, the use of the three Euler angles contains a singularity. Avoiding these problems, a new variable will be introduced in order to represent the orientation dynamics. Define now the rotation error between two steps as

$$\Delta\mathbf{R}_{k+1} = \frac{1}{2}\left(\mathbf{R}_{k+1}^T\mathbf{R}_k - \mathbf{R}_k^T\mathbf{R}_{k+1}\right) \quad (11)$$

Note that the rotation error matrix $\Delta\mathbf{R}_{k+1}$ can be verified being a skew-matrix. So that, this matrix is represented by using three independent variables $\mathbf{e}_R^{k+1} = \Delta\mathbf{R}_{k+1}^{\vee}$. Then from (3), the sub-dynamics of the quadrotor can be written in a discrete time domain as (12).

$$[v_z^{k+1}, \mathbf{e}_R^{k+1}, \Omega_{k+1}]^T = F(f_k, \Gamma_k, v_z^k, \mathbf{e}_R^k, \Omega_k, \mathbf{R}_k(9)) \quad (12)$$

Recall that the rotation matrix with the angular velocity Ω_k and the sampling time T_s is calculated by

$$\mathbf{R}_{k+1} = \mathbf{R}_k \exp\left(T_s\Omega_k\right) \quad (13)$$

Thanks to the Rodrigues' rotation formula, $\Delta\mathbf{R}_{k+1}$ has the approximation form:

$$\Delta\mathbf{R}_{k+1} \approx -T_s[\Omega_k]^{\times} \quad (14)$$

That gives the rotation error approximated by

$$\mathbf{e}_R^{k+1} \approx -T_s\Omega_k \quad (15)$$

Combination of (15), (12) and (3) allows a method to build an approximation map of the attitude dynamics of the quadrotor. Once the prediction map is built, the control algorithm above will be used for the stabilization of the attitude of the quadrotor.

4 SIMULATION RESULTS

In this section, we will introduce some simulation results from MATLAB/SIMULINK. The quadrotor model used in these simulations has been exported from MAPLESIM. The sampling time $T_s = 0.01s$ was chosen for the prediction map. A 100 Hz frequency for the control of a quadrotor is rather a low value. The internal reference linear state-space system for each axis $\mathbf{e}_R$, Ω and v_z has an equivalent canonical first-order transfer function $W(s) = \frac{\tau_s}{\tau_s.s+1}$ with a time constant of $\tau_s = 30^{-1}\ s$, $\tau_s = 50^{-1}\ s$ and $\tau_s = 2^{-1}\ s$ respectively. In the first simulation, the attitude trajectory is constructed from three RPY angles (roll, pitch, yaw) $y_{di} = 0.5\sin(t - i)$

http://www.imavs.org/

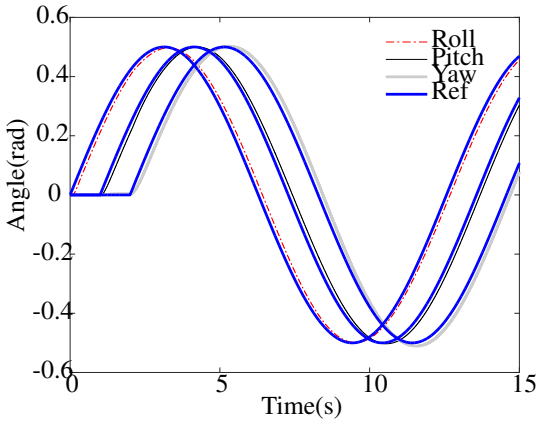

Figure 5: RPY angles and its reference

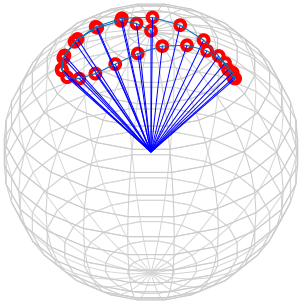

Figure 6: Attitude tracking

with $i = 1, 2, 3$ for roll-pitch-yaw angle respectively. The response angles from the quadrotor and its reference are shown in Figure 5.

This figure shows that the quadrotor tracked well the desired attitude. On the illustration shown in Figure 6, the trajectory generated by the vector $\mathbf{e}_3 = [0, 0, 1]^T$ converges to its reference.

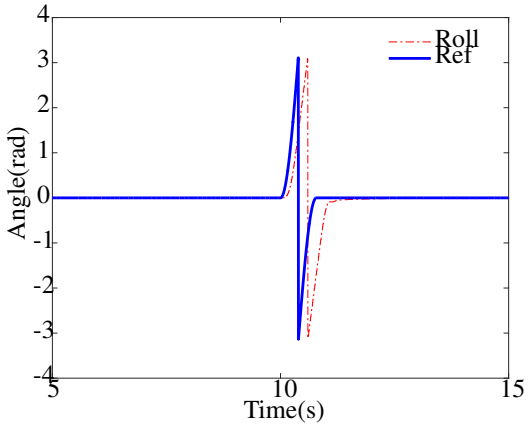

Figure 7: Roll angle and its reference

The second test is performed by doing a single flip about the x-axis. The pitch and yaw angle are driven to zeros while the roll angle was tracking a rotating trajectory. In this test, the desired vertical velocity v_z is controlled so as to keep the position of the quadrotor at 2.5 m. Figure8 shows that the vertical unit vector z_b rotates well one perfect round about x-axis. This significant is also justified in Figure 7.

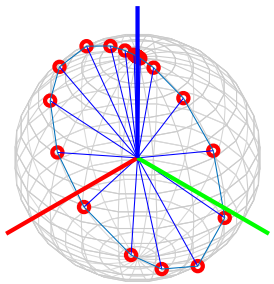

Figure 8: Rotation about x-axis

Figure 9 shows the vertical position z, its reference and the vertical velocity. The position z (dash-dotted curve) tends to its reference 2.5 m. The descent while flipping explains why the user usually accelerate the quadrotor before and during doing flips.

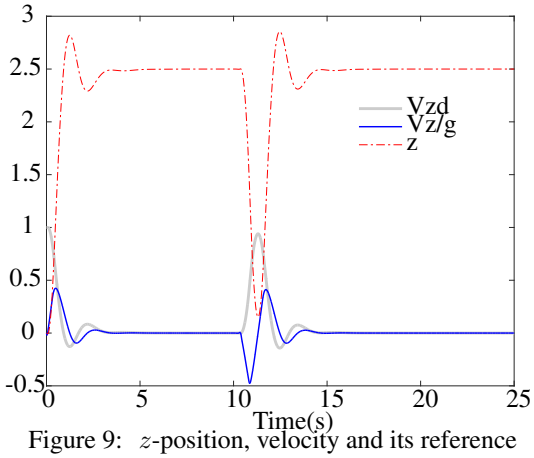

Figure 9: z-position, velocity and its reference

Figure 10 shows the rotors' nominal speed ω_i/ω_{max}. In hover flight, the rotors' speed are kept at a constant value. To do a single flip, Figure 10 shows the left rotors' speed ω_1 and ω_2 are increased while the right rotors' speed ω_3 and ω_4 are decreased in order to generate a torque that twists the quadrotor about x-axis. After the loop is perfectly done, the controller has gotten the smallest speed for the rotors to stabilize faster the roll angle. Finally, the vertical position is compensated.

Figure 11 shows the nominal inputs $T = f/(k_t * \omega_{max}^2)$, $G_1 = \Gamma_x/(k_t * L * \omega_{max}^2)$, $G_2 = \Gamma_y/(k_t * L * \omega_{max}^2)$ et $G_3 = \Gamma_z/(k_c * \omega_{max}^2)$. The nominal thrust T also shows that the acceleration was increased to accelerate the quadrotor in order to perform the looping while the roll torque $G1$ rotates the system about x-axis.

5 CONCLUSION

This paper presents a numerical tabulated approach to the attitude control problem. The numerical behavior of the sys-

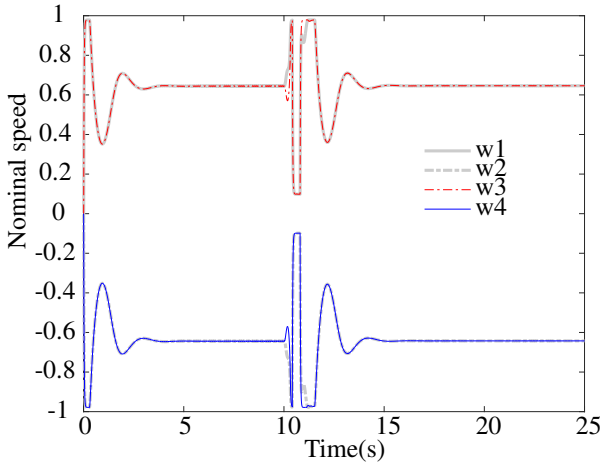

Figure 10: Rotors' nominal speed

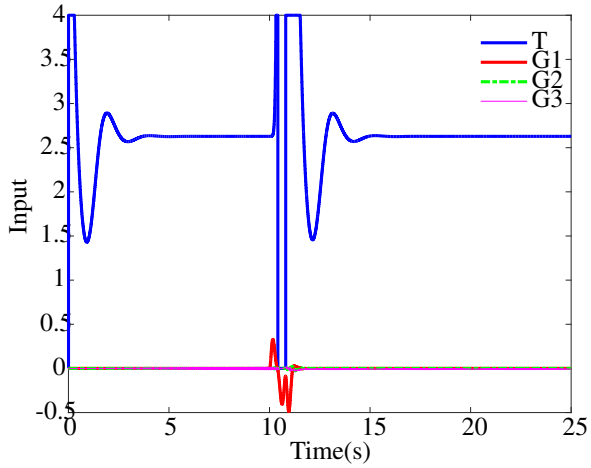

Figure 11: Looping: input

tem is a hybrid model that uses the incremental error between two time steps, avoiding the singularity and ambiguity of the attitude representation. This control could be combined with a position control in order to track complex position trajectory. The validation of the proposed controller on a embedded system requires a good observer for the attitude. Moreover, the prediction map could be improved online using learning-based method.

REFERENCES

[1] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1649–1654, 2010.

[2] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1642–1648, 2010.

[3] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch. Rigid-body attitude control. *IEEE Control Systems*, 31(3):30–51, 2011.

[4] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525, 2011.

[5] Cutler Mark and How Jonathan. *Actuator Constrained Trajectory Generation and Control for Variable-Pitch Quadrotors*. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2012.

[6] E. Fresk and G. Nikolakopoulos. Full quaternion based attitude control for a quadrotor. In *Control Conference (ECC), 2013 European*, pages 3864–3869, 2013.

[7] F. Goodarzi, D. Lee, and T. Lee. Geometric nonlinear pid control of a quadrotor uav on se(3). In *Control Conference (ECC), 2013 European*, pages 3845–3850, 2013.

[8] T. Tomix, M. Maier, and S. Haddadin. Learning quadrotor maneuvers from optimal control and generalizing in real-time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1747–1754, 2014.

[9] Sara Spedicato, Giuseppe Notarstefano, Heinrich H. Bulthoff, and Antonio Franchi. *Aggressive Maneuver Regulation of a Quadrotor UAV*, pages 95–112. Springer International Publishing, Cham, 2016.

[10] R. Munos and A. Moore. Barycentric interpolators for continuous space & time reinforcement learning. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 1024–1030, Cambridge, MA, USA, 1999. MIT Press.

[11] Stéphane Bonnet. *Appoches numérique pour la commande des systèmes dynamiques*. PhD thesis, Université de Technologie de Compiègne, France, 2008.

[12] J. De Miras and S. Bonnet. Nonlinear control of a magnetic levitation shaft by numerical inversion of its behavioral model. In *13th IEEE European Control Conference*, Strasbourg, France, June 2014.

[13] H-P. Nguyen, J. De Miras, S. Bonnet, and A. Charara. Nonlinear control of the pvtol aircraft by numerical inversion of its behavioral model. In *23th IEEE International Conference on Control Application*, Antibe, France, October 2014.

http://www.imavs.org/