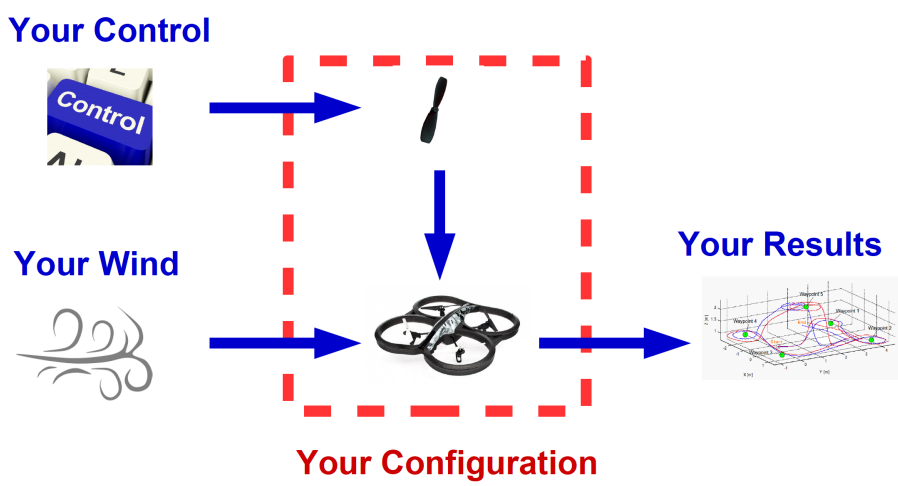
A toolbox for quadrotors: from aerodynamic science to control theory

Gabriele Perozzi

A toolbox for quadrotors: from aerodynamic science to control theory

Gabriele Perozzi

January 30, 2018



Abstract

In this paper the quadrotor unmanned aerial vehicle (UAV) is studied. Aerodynamic forces and moments are explained together with aerodynamic coefficients and a complete quadrotor model is illustrated. After that, feedback control, trajectory generation and proportional integrative derivative (PID) control algorithms are applied. Notions of controllability, observability and area of attraction are also introduced. It is always very hard to conciliate aerodynamic science with control theory because of different hypothesis. Thus, the main focus of this paper is to group them to create a basic guide that lets the reader to understand how quadrotors work and how to build a first linear control effectively, covering aerodynamics and control background together. Important remarks are done to the simplifications of aerodynamic coefficients and flight dynamic equations useful for control theory. Two quadrotor Simulink models are provided together with the article. One is built using the highly nonlinear aerodynamic coefficients, the other with some simplifying hypothesis. Both of them use only blocks provided with basic Simulink environment to ensure a better compatibility with most of Simulink versions, avoiding integrated matlab functions and any other auxiliary toolbox.

Contents

1 Introduction

The presented work is based on quadrotors having the configuration as in Fig. 1. Quadrotors are underactuated systems with 6 degrees of freedom and 4 controls.

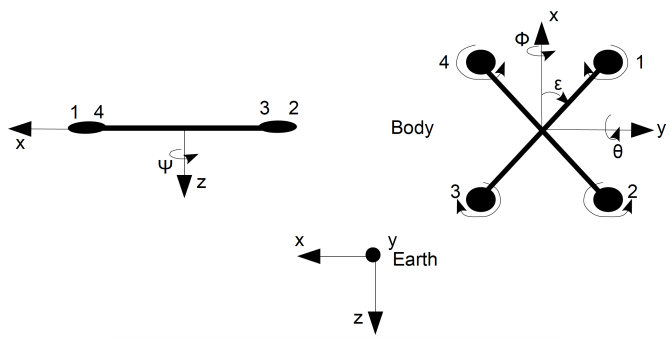


Figure 1: Quadrotor configuration and reference systems.

The paper outline is as follows. In Section 2 the inertial and body frames are introduced, the quadrotor flight dynamics and the aerodynamic coefficients are derived. In Section 3 some admissible simplifications are allowed to make the system easier to study, the control-oriented model is illustrated, notions of controllability, observability and area of attraction are introduced, then linear feedback state control, PID control and trajectory generation are explained for a quadrotor respecting its physical constraints. The Section 4 is dedicated to the explanation of the Simulink quadrotor models, provided together with this paper. Appendices A, B, C conclude the paper.

2 Quadrotor model

2.1 Body and inertial reference systems

A quadrotor is a vehicle that can fly in the airspace. Hence to be properly studied, two different frames must be considered. The body frame of the quadrotor, and an inertial frame which can be represented by the earth. Forces acting on the quadrotor are computed in body frame, and than transposed in inertial frame (see Fig. 2) thanks to the rotation matrix **R**.

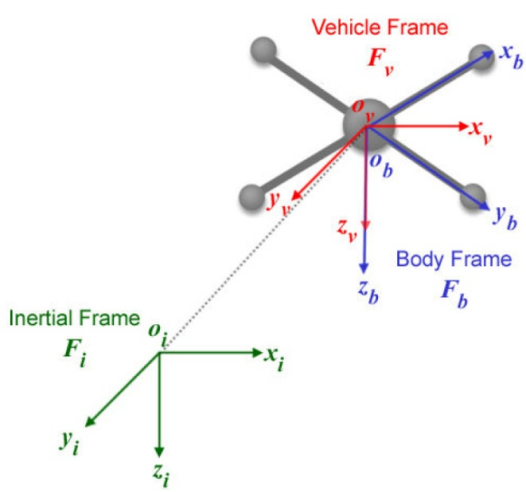


Figure 2: Quadrotor configuration and reference systems.

Then, 3 elemental rotations are introduced which rotate vectors by an angle ϕ around the x axes, by an angle θ around the y axes, by an angle ψ aroundthe z axes, using the *right hand* rule. The elementary rotation matrices are defined as

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad \mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$\mathbf{R}$ is the final rotational matrix defined as

$$\mathbf{R} = \mathbf{R}_z(\psi)\,\mathbf{R}_y(\theta)\,\mathbf{R}_x(\phi) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\phi s_\psi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix},$$

where $c_\psi = \cos(\psi)$, $s_\psi = \sin(\psi)$ and similarly. Then the passage from the earth inertial frame ($\mathcal{R}_i$) to the body frame ($\mathcal{R}_b$) is

$$[X^T]_{\mathcal{R}_b} = [X^T]_{\mathcal{R}_i}\mathbf{R}. \tag{1}$$

2.2 Flight dynamics

For the Parrot AR Drone 2.0 or similar shape, the rotors gyroscopic effects and inertial counter torques can be neglected since they are very small. Then the translation dynamics of the drone in the body frame yield

$$m\dot{\mathbf{u}} + m\varpi \times \mathbf{u} = \mathbf{F_{aero}} + m\mathbf{R}^T\mathbf{g}, \tag{2}$$

where m is the mass of the UAV, $\mathbf{u} = [u\ v\ w]^T$ is its linear velocity expressed in body frame, $\varpi = [p\ q\ r]^T$ is its angular velocity in body frame, $\mathbf{F_{aero}} = [F_{Xaero}\ F_{Yaero}\ F_{Zaero}]^T$ is the vector of the external aerodynamic forces in body frame, $\mathbf{g} = [0\ 0\ g]^T$ is the gravity acceleration in inertial frame, $\mathbf{R}$ is the rotational matrix. The rotational dynamics of the drone with respect to inertial earth frame are

$$\mathbf{I}\dot{\varpi} = -\varpi \times \mathbf{I}\varpi + \mathbf{L_{aero}}, \tag{3}$$

where $\mathbf{I}$ is the inertia matrix defined as:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix},$$

the UAV, and $\tau_{\mathbf{aero}} = [L_{aero}\ M_{aero}\ N_{aero}]^T$ is the external aerodynamic moments in the body frame. The relation between angular velocities and Euler angles

$$\begin{aligned} \dot{\phi} &= p + \tan\theta(q\sin\phi + r\cos\phi), \\ \dot{\theta} &= q\cos\phi - r\sin\phi, \\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}, \end{aligned} \tag{4}$$

out the singularity at $\theta \neq \frac{\pi}{2}$, which is never reached during flight (it is a reasonable assumption if we don't want to reach aggressive maneuvers). Hence the full model is given by the equations (2), (3), (4). The aerodynamic forces F_{Xaero}, F_{Yaero}, F_{Zaero}, moments L_{aero}, M_{aero}, N_{aero}, and related coefficients are derived below using a combination of momentum and blade element theory in helicopters.
Aerodynamic forces and moments for each rotor, where subscript j indicates the jth rotor, are derived as

$$\begin{aligned} F_{X\,j} &= -\rho AR^2 \frac{u_j - u_w}{\sqrt{(u_j-u_w)^2 + (v_j-v_w)^2}} C_{H\,j}\omega_j^2, \\ F_{Y\,j} &= -\rho AR^2 \frac{v_j - v_w}{\sqrt{(u_j-u_w)^2 + (v_j-v_w)^2}} C_{H\,j}\omega_j^2, \\ F_{Z\,j} &= -\rho AR^2 C_{T\,j}\omega_j^2, \\ L_j &= -\operatorname{sign}\omega_j \rho AR^3 \frac{u_j - u_w}{\sqrt{(u_j-u_w)^2 + (v_j-v_w)^2}} C_{Rm\,j}\omega_j^2, \\ M_j &= -\operatorname{sign}\omega_j \rho AR^3 \frac{v_j - v_w}{\sqrt{(u_j-u_w)^2 + (v_j-v_w)^2}} C_{Rm\,j}\omega_j^2, \\ N_j &= -\operatorname{sign}\omega_j \rho AR^3 C_{Q\,j}\omega_j^2, \end{aligned}$$

where ρ is the air density, A is the rotor area, R is the rotor radius, u_w, v_w, w_w are the wind velocities with respect to the earth in body frame, C_{Hj} is the hub force coefficient, C_{Tj} is the rotor thrust coefficient, ω_j is the rotor angular velocity, C_{Q_j} is the rotor drag moment coefficient, C_{Rmj} is the rotor rolling moment coefficient. Total aerodynamic forces are

$$F_{Xaero} = \sum_{j=1}^{4} F_{Xj}, \quad F_{Yaero} = \sum_{j=1}^{4} F_{Yj}, \quad F_{Zaero} = \sum_{j=1}^{4} F_{Zj}. \tag{5}$$

Total aerodynamic moments are

$$\begin{aligned} L_{aero} &= \sum_{j=1}^{4} (L_j + F_{Zj} l \mathrm{s}_j - h F_{Yj}), \\ M_{aero} &= \sum_{j=1}^{4} (M_j - F_{Zj} l \mathrm{c}_j + h F_{Xj}), \\ N_{aero} &= \sum_{j=1}^{4} (N_j + F_{Yj} l \mathrm{c}_j - F_{Xj} l \mathrm{s}_j), \end{aligned} \tag{6}$$

where h is the distance between rotors plane and the center of gravity of the UAV, l is the arm length, and with

$$\mathrm{c}_j = \cos\left(\frac{\pi}{2}(j-1) + \epsilon\right), \quad \mathrm{s}_j = \sin\left(\frac{\pi}{2}(j-1) + \epsilon\right),$$

where in our UAV configuration we have $\epsilon = \frac{\pi}{4}$. Thus, for vectors c_j and s_j we have cosines and sinus of the angles $[\frac{\pi}{4}, \frac{3}{4}\pi, \frac{5}{4}\pi, \frac{7}{4}\pi]$. Linear rotors speed in body frame are computed as a function of the state in body frame

$$\begin{bmatrix} u_j \\ v_j \\ w_j \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} lc_j \\ ls_j \\ h \end{bmatrix} + \begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

2.3 Aerodynamic coefficients

The blade flapping is modeled as a fictitious hinge spring as in Fig. 3.

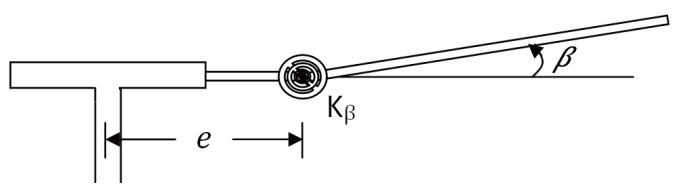


Figure 3: Fictious spring modeling the blade flapping

With the hypothesis that β is small, the equation is

$$I_x \ddot{\beta} + \omega^2 \left(I_x + M_{blade} x_g e R^2\right) \beta = M_a - k_\beta \beta,$$

where, ω is the rotor angular velocity, I_x is the inertia of the blade, $x_g R$ is the position of the CoG of the blade, M_{blade} is the mass of the blade, and M_a is the aerodynamic momentum of each element of the blade. Then the blade flapping equation can be rewritten as

$$\begin{aligned} &\frac{d^2\beta}{d\psi^2} + \frac{1}{8}\gamma(1 + \frac{4}{3}\mu \sin\psi)\frac{d\beta}{d\psi} + \left((1+\delta) + \frac{1}{8}\gamma(\frac{4}{3}\mu\cos\psi + \mu^2 \sin 2\psi)\right)\beta \\ &= -\frac{1}{8}\gamma\left(\theta_0(1 + \frac{8}{3}\mu\sin\psi + 2\mu^2\sin^2\psi) - \theta_{tw}(\frac{4}{5} + 2\mu\sin\psi + \frac{4}{3}\mu^2\sin^2\psi) - \frac{4}{3}\lambda - 2\mu\lambda\sin\psi\right), \end{aligned}$$

where ψ is the azimuth angle of the blade. If the transition phase is stable and the blade movement is periodic in ψ the equation becomes

$$\beta = a_0 + a_1 \sin\psi + b_1 \cos\psi, \tag{7}$$

with

$$a_0 = -\frac{\gamma}{8(1+\delta)}\left(\theta_0(1+\mu^2) - \theta_{tw}\left(\frac{4}{5}+\frac{2}{3}\mu^2\right) - \frac{4}{3}\lambda\right),$$
$$\delta a_1 = -\frac{\gamma}{4}\mu\left(\frac{4}{3}\theta_0 - \theta_{tw} - \lambda\right) + \frac{\gamma}{8}b_1\left(1-\frac{\mu^2}{2}\right),$$
$$\delta b_1 = -\frac{\gamma}{6}\mu a_0 - \frac{\gamma}{8}a_1\left(1+\frac{\mu^2}{2}\right),$$
$$\delta = \frac{M_{blade}x_g e R^2}{I_x},$$
$$\gamma = \frac{\rho a \bar{c} R^4}{I_x},$$

where $\bar{c}$ is the average length of the blade chord.
The advance ratio indicates the component of the UAV velocity parallel to the rotor disk with respect to the blade tip speed:

$$\mu_j = \frac{\sqrt{(u_j-u_w)^2+(v_j-v_w)^2}}{R|\omega_j|}.$$

The inflow ratio is the ratio between the component of UAV velocity perpendicular to the rotor disk with respect to the blade tip speed:

$$\lambda_j = \sigma a \frac{(1+\frac{3}{2}\mu_j^2)\frac{\theta_0}{6} - (1+\mu_j^2)\frac{\theta_{tw}}{8} - \frac{\lambda_j}{4}}{2\sqrt{\mu_j^2+\lambda_j^2}} + \frac{w_w - w_j + qlc_j - pls_j}{R|\omega_j|},$$

where σ is the rotor solidity ratio, a is the lift curve slope of the blade section, θ_0 is the angle of attack of the root profile, θ_{tw} is the twist angle of the blades.
The rolling moment of a propeller exists in forward flight when the advancing blade is producing more lift than the retreating one and it is the integration over the entire rotor of the lift of each section acting at a given radius. Its coefficient is:

$$C_{Rm_j} = \sigma a\big(\frac{\mu_j}{8}(\lambda_j - \frac{4}{3}\theta_0 + \theta_{tw}) + \frac{b_1}{16}(1-\frac{\mu_j^2}{2})\big).$$

The thrust is the resultant of the vertical forces acting on all the blade elements. Its coefficient is:

$$C_{Tj} = \sigma a\big((1+\frac{3}{2}\mu_j^2)\frac{\theta_0}{6} - (1+\mu_j^2)\frac{\theta_{tw}}{8} - \frac{\lambda_j}{4}\big).$$

The hub forces is the resultant of the horizontal forces acting on all the blade elements. Its coefficient is:

$$\begin{aligned}
C_{Hj} &= C_{HPj} + C_{Hij},\\
\frac{C_{HPj}}{\sigma} &= \frac{\mu_j}{4}(C_{D0}+C_{Di}\theta_0^2) + C_{Di}\big(\frac{\mu_j}{24}(3\theta_{tw}^2 - 8\theta_0\theta_{tw}) + \frac{\theta_0}{24}(3\mu_j^2 b_1 - 12\lambda_j\mu_j - 4b_1)\\
&\quad - \frac{\theta_{tw}}{16}(\mu_j^2 b_1 - 4\lambda_j\mu_j - 2b_1)\big) + C_{Di}\big(\frac{\mu_j^2}{8}a_0a_1 + \frac{\mu_j}{16}(a_1^2-b_1^2) + \frac{1}{4}\lambda_j b_1\big),\\
\frac{C_{Hij}}{\sigma a} &= \frac{\theta_0}{4}(\lambda_j\mu_j + \frac{2}{3}b_1) - \frac{\theta_{tw}}{8}(\lambda_j\mu_j + b_1) + \frac{\mu_j}{8}(a_0^2+b_1^2) - \frac{3}{8}\lambda_j b_1 + \frac{1}{12}a_0a_1,
\end{aligned}$$

where C_{D0} is the drag coefficient of the blade section, C_{Di} is the induced drag coefficient of the blade section.
The drag moment about the rotor shaft is caused by the aerodynamic forces acting on the blade elements, the horizontal forces acting on the rotor are multiplied by the moment arm and integrated over the rotor. Its coefficient is:

$$\begin{aligned}
C_{Qj} &= C_{QPj} + C_{Qij},\\
\frac{C_{QPj}}{\sigma} &= \frac{1}{8}(C_{D0}+C_{Di}\theta_0^2)(1+\mu_j^2) - C_{Di}\theta_0\theta_{tw}(\frac{1}{5}+\frac{\mu_j^2}{6}) + C_{Di}\theta_{tw}^2(\frac{1}{12}+\frac{\mu_j^2}{16}) - C_{Di}\lambda_j\\
&\quad\times(\frac{\theta_0}{3} - \frac{\theta_{tw}}{4}) + C_{Di}\Big(\frac{\mu_j^2}{8}(a_0^2+\frac{a_1^2}{4}+\frac{3b_1^2}{4}) + \frac{1}{16}(a_1^2+b_1^2) + \frac{\lambda_j^2}{4} + \frac{\mu_j}{6}a_0a_1 - \frac{\lambda_j\mu_j}{4}b_1\Big)\\
\frac{C_{Qij}}{\sigma a} &= \lambda_j(\frac{\theta_0}{6} - \frac{\theta_{tw}}{8} - \frac{\lambda_j}{4}) - \frac{\mu_j^2}{8}(a_0^2+\frac{a_1^2}{4}+\frac{3b_1^2}{4}) - \frac{1}{16}(a_1^2+b_1^2) - \frac{\mu_j}{6}a_0a_1 + \frac{\lambda_j\mu_j}{4}b_1,
\end{aligned}$$

2.4 Rotors dynamics

The rotors are driven by DC-motors, which binds electrical and mechanical quantities. Rotors can be represented as in Fig. 4, and described by the dynamic system

$$\begin{cases} L\frac{di}{dt} & = u - Ri - K_e\omega_m \\ J\frac{d\omega_m}{dt} & = \tau_m - \tau_d \end{cases},$$

where u is the motor input, R is the motor inertial resistance, L is the motor inductance, K_e is the motor constant corresponding to the back EMF constant, τ_m is the motor torque, ω_m is the motor angular speed, τ_d is the motor load, J is the motor moment of inertia, i is the current. Considering a small motor with a very low inductance

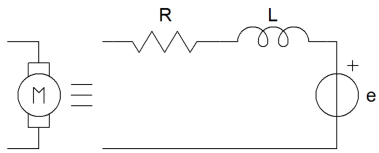


Figure 4: Rotor model

the given system can be simplified and approximated, then linearized around an operating point ω_0, and further identified as:

$$G(s) = \frac{1}{bs+1},$$

where the time constant $b = 0.1$ can be found considering Parrot Drones. Time constant slightly varies based on the considered motor and thus on the considered UAV mass. Precisely, in the Parrot Drones, the time constant is less than 0.1 for the increasing rotor angular velocities, and it is almost equal to 0.1 for decreasing rotor angular velocities.

3 Trajectory and Control

3.1 Model simplification

The presented quadrotor model is highly nonlinear and thus it is not compatible to be studied in control theory. A trade-off should be found between the aerodynamic science (which allows highly nonlinearities to better represent the real quadrotor behaviour) and the control theory (which allows the system to be as much simple as possible). Identification results thanks to indoor experiments on Parrot drone at low/moderate velocity, validates the previous UAV model and allows some simplifications to be accepted:

- $\theta_{tw} = 0$;
- $C_{Di} = 0$;
- $\lambda_j = \lambda_{stat} - \frac{4}{\sigma a} K_z \frac{w_j - w_w}{R|\omega_j|}, \quad K_z, \lambda_{stat} \geq 0$;
- $C_{Tj} = C_{Tstat} + K_z \frac{w_j - w_w}{R|\omega_j|}, \quad C_{Tstat} \geq 0$,
  such models of λ_j and C_{Tj} are rather precise in the climbing phase, but less accurate in descent phase, since the model tends to slightly overestimate the propulsion in the descent phase;
- $C_{Hj} = K_D \mu_j, \quad K_D \geq 0$;
  the UAV drag is modeled as $\rho A R^2 \sum C_{Hj}\omega_j^2$, corresponding to the rotors at low UAV speed, otherwise at higher speed we need to add the body drag effect since it depends on square of velocity. However the constant K_D is identified with tests considering the whole UAV body and rotors;
- Blade flapping is neglected.

3.2 Control-oriented model

The system can be rewritten in the state-space form

$$\dot{X} = f(X,\, U,\, d) \tag{8}$$

where f is expressed in (2), (3), (4) with the simplifying hypothesys. The state vector X is chosen as

$$X = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r]^{\,T},$$

the control input is

$$U = [U_z\; U_\theta\; U_\phi\; U_\psi]^T,$$

where its relation with the rotor velocities is defined by an invertible matrix

$$\begin{bmatrix} U_z \\ U_\theta \\ U_\phi \\ U_\psi \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ K_f l \mathrm{c}_j & K_f l \mathrm{c}_j & K_f l \mathrm{c}_j & K_f l \mathrm{c}_j \\ -K_f l \mathrm{s}_j & -K_f l \mathrm{s}_j & -K_f l \mathrm{s}_j & -K_f l \mathrm{s}_j \\ K_m & -K_m & K_m & -K_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{9}$$

with $\omega_{min} \leq \omega_j \leq \omega_{max}$, where $K_f = \rho A R^2 C_{T\,stat}$ and $K_m = \rho A R^3 \big(\frac{\sigma C_{D0}}{8} + \lambda_{stat}\sigma a(\frac{\theta_0}{6} - \frac{\lambda_{stat}}{4})\big)$. In this work, the control inputs are selected to be proportional to the terms with ω_j^2. Thus, expanding (2) and (3), the other terms dependent linearly on ω_j and wind velocities are considered as disturbances. Since we do not know in advance the wind perturbations, then we cannot use these terms in controls. Such a decomposition of thrust (which is proportional to ω_j and ω_j^2) and selection of disturbances are almost exact in the hover flight, where we have $(p,\, q,\, r) \approx (\dot{\phi},\, \dot{\theta},\, \dot{\psi})$.
Linear rotors velocities in body frame are computed as a function of the state in body frame

$$\begin{bmatrix} u_j \\ v_j \\ w_j \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} lc_j \\ ls_j \\ h \end{bmatrix} + \begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

We will call signals, whose influence is compensated by robust abilities of the control, as "disturbances", which include external inputs (the wind velocities), state components and controls. We have the disturbance $d = [d_x, d_y, d_z, d_\phi, d_\theta, d_\psi]$, where d_x, d_y, d_z are expressed in body frame, explicited as

$$\begin{aligned}
d_x =& \sum_{j=1}^{4} -\rho A R K_D\,(u_j - u_w)\,|\omega_j|, \\
d_y =& \sum_{j=1}^{4} -\rho A R K_D\,(v_j - v_w)\,|\omega_j|, \\
d_z =& \sum_{j=1}^{4} -\rho A R K_z\,(w_j - w_w)\,|\omega_j|, \\
d_\phi =& \sum_{j=1}^{4} \Big(\omega_j \rho A R^2 (u_j - u_w)\frac{\sigma a}{2}(\frac{\theta_0}{3} - \frac{\lambda_{stat}}{4}) + |\omega_j|\rho A R\big(hK_D\,(v_j - v_w) - lK_z\,(w_j - w_w)\,\mathrm{s}_j\big) \\
&+ \mathrm{sign}\,(\omega_j)\,\frac{1}{2}\rho A R K_z\,(u_j - u_w)\,(w_j - w_w)\Big), \\
d_\theta =& \sum_{j=1}^{4} \Big(\omega_j \rho A R^2\,(v_j - v_w)\,\frac{\sigma a}{2}(\frac{\theta_0}{3} - \frac{\lambda_{stat}}{4}) + |\omega_j|\rho A R\big(-hK_D\,(u_j - u_w) + lK_z\,(w_j - w_w)\,\mathrm{c}_j\big) \\
&+ \mathrm{sign}\,(\omega_j)\,\frac{1}{2}\rho A R K_z\,(v_j - v_w)\,(w_j - w_w)\Big), \\
d_\psi =& \sum_{j=1}^{4} \Big(\omega_j \rho A R^2 K_z\,(w_j - w_w)\,(\frac{2\theta_0}{3} - 2\lambda_{stat}) - |\omega_j|\rho l A R K_D\big((v_j - v_w)\,\mathrm{c}_j - (u_j - u_w)\,\mathrm{s}_j\big) \\
&- \mathrm{sign}\,(\omega_j)\,\rho A R\Big(\frac{\sigma C_{D0}}{8}\big((u_j - u_w)^2 + (v_j - v_w)^2\big) - \frac{4}{\sigma a}K_z^2\,(w_j - w_w)^2\Big)\Big).
\end{aligned}$$

In earth frame, using (1), the linear dynamics disturbances become

$$\begin{aligned}
d_{xe} =& c_\psi c_\theta d_x + (c_\psi s_\theta s_\phi - s_\psi c_\phi) d_y + (s_\phi s_\psi + c_\psi s_\theta c_\phi) d_z \\
d_{ye} =& s_\psi c_\theta d_x + (s_\psi s_\theta s_\phi + c_\psi c_\phi) d_y + (s_\psi s_\theta c_\phi - c_\psi s_\phi) d_z \\
d_{ze} =& - s_\theta d_x + c_\theta s_\phi d_y + c_\theta c_\phi d_z
\end{aligned}$$

Under the hypothesis of quasi-hover the system is explicited in compact form as

$$\begin{cases}
\dot{x}_1 = x_4 \\
\dot{x}_4 = (s_\phi s_\psi + c_\psi s_\theta c_\phi) \frac{1}{m} (U_z - d_{xe}) \\
\dot{x}_2 = x_5 \\
\dot{x}_5 = (s_\psi s_\theta c_\phi - c_\psi s_\phi) \frac{1}{m} (U_z - d_{ye}) \\
\dot{x}_3 = x_6 \\
\dot{x}_6 = g - (c_\phi c_\theta) \frac{1}{m} (U_z + d_{ze}) \\
\dot{x}_7 = x_{10} \\
\dot{x}_{10} = x_{11} x_{12} \frac{I_{xx} - I_{zz}}{I_{xx}} + \frac{1}{I_{xx}} (U_\phi + d_\phi) \\
\dot{x}_8 = x_{11} \\
\dot{x}_{11} = x_{10} x_{12} \frac{I_{zz} - I_{xx}}{I_{yy}} + \frac{1}{I_{yy}} (U_\theta + d_\theta) \\
\dot{x}_9 = x_{12} \\
\dot{x}_{12} = x_{10} x_{11} \frac{I_{xx} - I_{yy}}{I_{zz}} + \frac{1}{I_{zz}} (U_\psi + d_\psi)
\end{cases} \tag{10}$$

3.3 State controllability

The state controllability property of a generic system describes the ability of an external input, such as the control signal, to move the state of the system from an initial point to a given final point in a finite time.
Let's consider the LTI system:

$$\begin{cases}
\dot{X} = A\,X + B\,U \\
Y = C\,X + D\,U
\end{cases}$$

Then the system is state controllable if the controllability matrix

$$R = [B \quad AB \quad A^2 B \dots A^{n-1} B]$$

has rank n, where n is the dimension of the matrix A. In matlab, the function *rank(ctrb(A,B))*, gives the rank of the controllability matrix, which should be equal to the dimension of the matrix A, in our case $n = 12$.
This concept is useful to be sure that the system can be fully controllable with the given control input. It means for example that, to control properly the quadrotor system the full control vector must be considered, to control the attitude sub-system only 3 elements of the control vector can be considered (U_ϕ, U_θ, U_ψ), and so on.

3.4 Observability

A system is observable if the behavior of the entire system can be determined from the output. It means that the values of the state can be reconstructed using the output values generally in the sensors.
Let's consider again the same system for the controllability, then if the row rank of the following observability matrix

$$O = [C \quad CA \quad CA^2 \dots CA^{n-1}]^T$$

is equal to n, then the system is observable.
This is a key concept in estimation algorithms. For example, if we want to estimate the wind velocity using the quadrotor as wind sensor, the quadrotor must be observable in earth frame using cameras, since the wind estimation algorithms use information related to the drone velocity with respect to the wind in earth frame. In matlab, the function *rank(obsv(A,C))*, gives the rank of the observability matrix, which should be equal to the dimension of the matrix A, in our case $n = 12$.

3.5 Area of Attraction

Another key concept in control algorithms, is the Area of Attraction (AoA). The AoA is the area of the state-space where the state of the system, from any initial point, can reach a final point, called equilibrium point. Any point initially placed on the unstable equilibrium point, inside this area, will reach the equillibrium point in finite-time (stability notion) or in infinite time (asymptotical stability notion). The AoA concept is strictly correlated with the control algorithms, because different controls have different AoAs. For example the linear control, such as PID, have small AoA around the chosen equilibrium point. Many equilibrium points can be chosen in the state-space and each of them has a different small local AoA. While the nonlinear controllers, has a larger domain of stability.
For complex controls and with high dimension systems, AoA can be found only empirically in simulations because of the high computational complexity. However, the AoA of linear controls, such as feedback control in Section 3.7 for the system (8), can be found analytically following the steps (see code in Appendix A):

1. Given the wind $d_w = (u_w,\, v_w,\, w_w)^T$, find the equilibrium point $(X_0,\, U_0)$ such that $f(X_0, U_0, d_w) = 0$.
2. Find the matrices A, B.
3. Given the matrices A, B, find the gain K such that $A - B\,K$ is Hurwitz.
4. Find $P = P^T > 0$ such that $(A - BK)^T P + P(A - BK) < 0$.
5. Find δx such that $dV = 2\delta x^T P f(X_0 + \delta x,\, U_0 - K\delta x,\, d_w) \leq 0$.
6. δx is the area in 12 dimensions of the state-space, inside this area the system is stable. We finally found the Area of Attraction.
7. Repeat the entire process for another chosen wind d_w.

3.6 Physical constraints and trajectory generation

Since UAVs are physical objects, then they are subjected to physical constraints, which correspond to their maximal linear and angular velocities and accelerations. Consequently, a smooth position trajectory is desired. In literature many articles are presented to generate desired trajectories admissible to the system dynamics. One method is to use a third order filter to obtain realistic velocity and acceleration trajectories

$$\frac{\xi_f}{\xi_{wp}} = \frac{1}{(1+\tau s)^3},$$

where ξ_{wp} is the way-point step reference, ξ_f is the filtered signal, τ is the arbitrary weight to create the desired smooth trajectory. The reference signal is third order filtered as in Fig. 5, with $G_1 = 1/(\tau_{pos}\tau_{vel}\tau_{acc})$, $G_2 = \tau_{pos}\tau_{vel} + \tau_{acc}\tau_{vel} + \tau_{pos}\tau_{acc}$, $G_3 = \tau_{pos} + \tau_{vel} + \tau_{acc}$ where τ_{pos}, τ_{vel}, τ_{acc} are the three weighted parameters for position, velocity and acceleration respectively. This method allows to consider the limitations, inside the integrals. Results are illustrated in Figures 6, 7, 8. Different gains allow to obtain different trajectories.

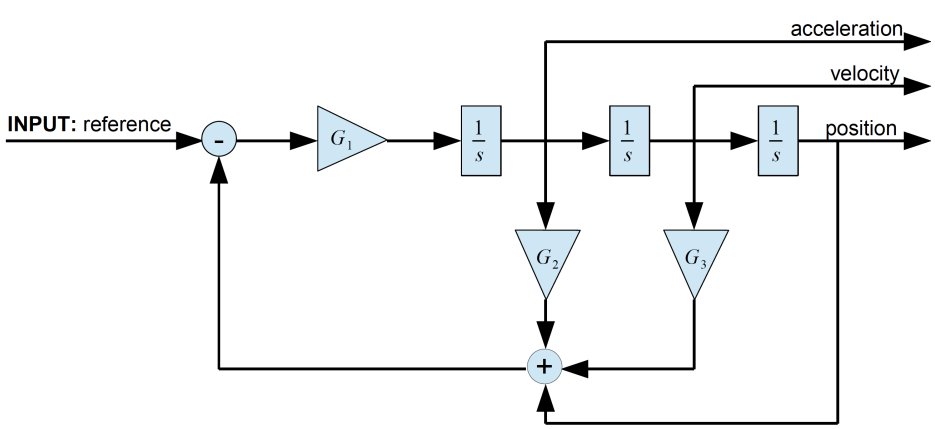


Figure 5: Third order filter scheme for trajectory

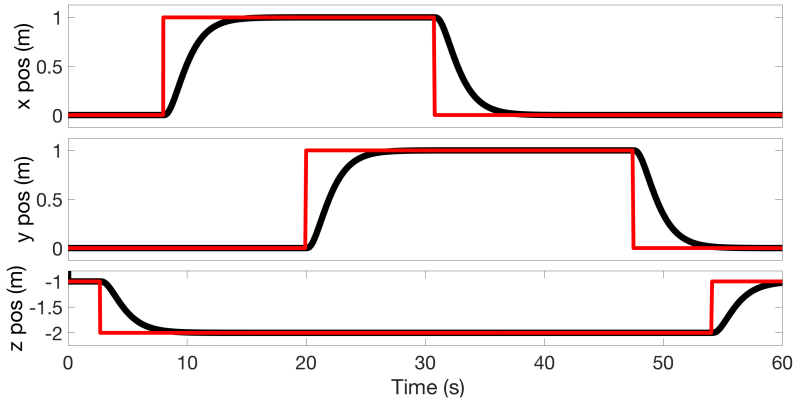


Figure 6: Quadrotor filtered position. (—: filtered trajectory; —: reference trajectory)

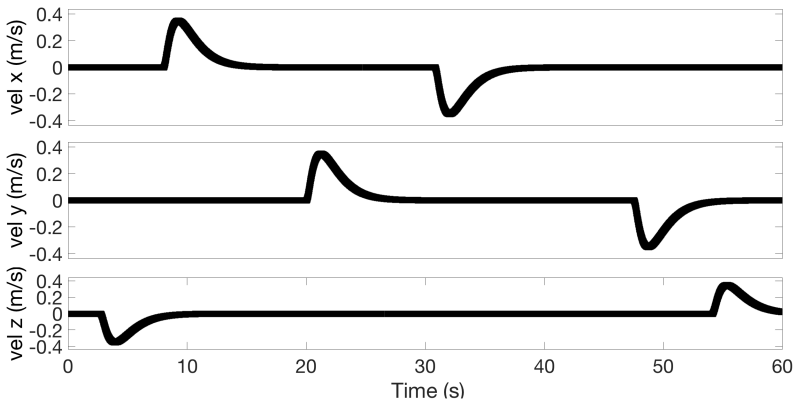


Figure 7: Quadrotor filtered velocity

The physical constraints are then chosen as

$$v_{min} = \begin{bmatrix} \dot{x}_{min} \\ \dot{y}_{min} \\ \dot{z}_{min} \end{bmatrix}, \quad v_{max} = \begin{bmatrix} \dot{x}_{max} \\ \dot{y}_{max} \\ \dot{z}_{max} \end{bmatrix}; \quad a_{min} = \begin{bmatrix} \ddot{x}_{min} = \frac{T_{min}}{m} \sin\theta_{min} \\ \ddot{y}_{min} = \frac{T_{min}}{m} \sin\phi_{min} \\ \ddot{z}_{min} = \frac{T_{min}}{m} - g \end{bmatrix}, \quad a_{max} = \begin{bmatrix} \ddot{x}_{max} = \frac{T_{max}}{m} \sin\theta_{max} \\ \ddot{y}_{max} = \frac{T_{max}}{m} \sin\phi_{max} \\ \ddot{z}_{max} = \frac{T_{max}}{m} - g \end{bmatrix};$$

respectively for linear velocity and acceleration, where T_{max} is 90% of the maximal thrust achievable by the UAV to ensure that we still have some available thrust to proper control the drone in case of safe maneuvers; T_{min} is the minimal thrust. The reference angles ϕ_{des}, θ_{des} are saturated in an admissible range of ϕ_{min}, θ_{min} minimal values and ϕ_{max}, θ_{max} maximal values, to avoid the singularities. Physical constraints for angles and torque controls are imposed as

$$\Delta_{min} = \begin{bmatrix} \phi_{min} \\ \theta_{min} \\ \psi_{min} \end{bmatrix}, \quad \Delta_{max} = \begin{bmatrix} \phi_{max} \\ \theta_{max} \\ \psi_{max} \end{bmatrix}; \quad \Omega_{min} = \begin{bmatrix} \dot{\phi}_{min} \\ \dot{\theta}_{min} \\ \dot{\psi}_{min} \end{bmatrix}, \quad \Omega_{max} = \begin{bmatrix} \dot{\phi}_{max} \\ \dot{\theta}_{max} \\ \dot{\psi}_{max} \end{bmatrix};$$

$$U_{\phi,\theta,\psi min} = \begin{bmatrix} -l\rho AR^2 C_{T\,stat} \sin(\epsilon)(\omega_{max}^2 - \omega_{min}^2) \\ -l\rho AR^2 C_{T\,stat} \cos(\epsilon)(\omega_{max}^2 - \omega_{min}^2) \\ -2\rho AR^3 C_Q(\omega_{max}^2 - \omega_{min}^2) \end{bmatrix}, \quad U_{\phi,\theta,\psi max} = \begin{bmatrix} l\rho AR^2 C_{T\,stat} \sin(\epsilon)(\omega_{max}^2 - \omega_{min}^2) \\ l\rho AR^2 C_{T\,stat} \cos(\epsilon)(\omega_{max}^2 - \omega_{min}^2) \\ 2\rho AR^3 C_Q(\omega_{max}^2 - \omega_{min}^2) \end{bmatrix}.$$

3.7 Feedback state control

The first simplest control is the stabilization of the drone in different operating points (see Appendix B). To this end, let's consider the system:

$$\dot{X} = f(X, U, d),$$

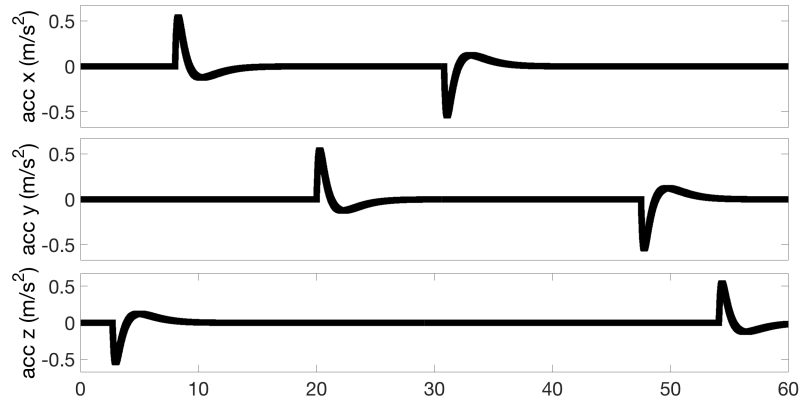


Figure 8: Quadrotor filtered acceleration

Supposing the trim point is (X_0, U_0, d_0) (important remark is that the angles depend by the linear quadrotor velocity and the wind, thus the effect of the wind should be considered while computing the trim point), let denote by δX, δU and δd the deviations from that point of X, U and d, respectively, then:

$$X = X_0 + \delta X, \quad U = U_0 + \delta U, \quad d = d_0 + \delta d.$$

Expanding the model equations in a truncated Taylor series of first order the following is obtained

$$\delta X = f(X_0, U_0, d_0) + \frac{\partial f}{\partial X}(X_0, U_0, d_0)\delta X + \frac{\partial f}{\partial U}(X_0, U_0, d_0)\delta U + \frac{\partial f}{\partial d}(X_0, U_0, d_0)\delta d$$

Denote constant matrices:

$$A = \frac{\partial f}{\partial X}(X_0, U_0, d_0)\delta X, \quad B = \frac{\partial f}{\partial U}(X_0, U_0, d_0)\delta U, \quad D = \frac{\partial f}{\partial d}(X_0, U_0, d_0)\delta d, \quad G = F(X_0, U_0, d_0),$$

then the linearized equations of the system around the point (X_0, U_0, d_0) can be rewritten as follows

$$\delta \dot{X} = A\delta X + B\delta U + D\delta d + G.$$

To stabilize the quadrotor around the chosen trim condition, a simple feedback state control can be applied

$$U = K_0\, X + K_1,$$

where K_1 stabilizes the UAV against the gravity force. Thus, if the system is chosen to be stabilized around stationary case (but any desired operating point can be chosen), the control gain can be computed as

$$K_1 = \begin{bmatrix} -\frac{g}{b_1} & 0 & 0 & 0 \end{bmatrix}^T,$$

where $b_1 = \rho A R^2 C_{Tstat}$ is the term proportional to the control $U_z = -mg$ in $\dot{z}$ dynamics for the stationary flight. To find K_0 the closed-loop system must be analyzed

$$\begin{cases} \dot{X} = & (A + B\,K_0)X + BK_1 + G \\ Y = & X \end{cases}$$

It means to find K_0 such that the eigenvalues are at the desired location in the roots graph. Hence, this linear system is stable if and only if the eigenvalues of $A + B\,K_0$ are either real and negative or else complex with non-positive real part, and it is asymptotically stable if the complex eigenvalues of $A + B\,K_0$ have strictly negative real part. Since the system is fully controllable the gain can be tuned using the Matlab function *place*.

3.8 PID control

An enhanced version of the previous feedback state control is the PID. While the feedback state allows only the proportional term, the PID can provide better stability using the integrative and the derivative terms too. The

proposed PID is valid for linearized system around hover flights, so under the following hypothesis:

$$
\begin{gathered}
U_z = -mg, \\
\cos\phi = 1, \quad \cos\theta = 1, \quad \sin\phi = \phi, \quad \sin\theta = \theta, \\
\dot{\phi} = p, \quad \dot{\theta} = q, \quad \dot{\psi} = r, \\
p_{des} = q_{des} = r_{des} = 0.
\end{gathered}
$$

Since the x, y cannot be directly controlled, then auxiliary controls U_x, U_y are added, which will be used to find the controlled angles. Position controller is structured as

$$
\begin{aligned}
U_x =& K_{Px}(x_{des} - x) + K_{Ix}\int(x_{des} - x) + K_{Dx}(\dot{x}_{des} - \dot{x}) + \ddot{x}_{des}, \\
U_y =& K_{Py}(y_{des} - y) + K_{Iy}\int(y_{des} - y) + K_{Dy}(\dot{y}_{des} - \dot{y}) + \ddot{y}_{des}, \\
U_z =& g - K_{Pz}(z_{des} - z) + K_{Iz}\int(z_{des} - z) + K_{Dz}(\dot{z}_{des} - \dot{z}) + \ddot{z}_{des},
\end{aligned}
$$

where K_{Pi}, K_{Ii}, K_{Di} for $i = x$, y, z are the tuning parameters for proportional, integrative and derivative terms. Attitude controller is structured as

$$
\begin{aligned}
U_\phi =& I_{xx}\left(K_{P\phi}(\phi_c - \phi) + K_{I\phi}\int(\phi_c - \phi) - K_{D\phi}p\right), \\
U_\theta =& I_{yy}\left(K_{P\theta}(\theta_c - \theta) + K_{I\theta}\int(\theta_c - \theta) - K_{D\theta}q\right), \\
U_\psi =& I_{zz}\left(K_{P\psi}(\psi_{des} - \psi) + K_{I\psi}\int(\psi_{des} - \psi) - K_{D\psi}r\right),
\end{aligned}
$$

where K_{Pi}, K_{Ii}, K_{Di} for $i = \phi$, θ, ψ are the tuning parameters. The controlled angles ϕ_c, θ_c are derived from U_x, U_y and using the desired value of ψ_{des}

$$
\begin{aligned}
\phi_c &= -\frac{1}{g}\left(U_x \sin\psi_{des} - U_y \cos\psi_{des}\right), \\
\theta_c &= -\frac{1}{g}\left(U_x \cos\psi_{des} + U_y \sin\psi_{des}\right).
\end{aligned}
$$

The generic scheme of hierarchical control algorithm presented above is given in Fig. 9. The attitude control loop usually works 10 times faster than the position control loop.

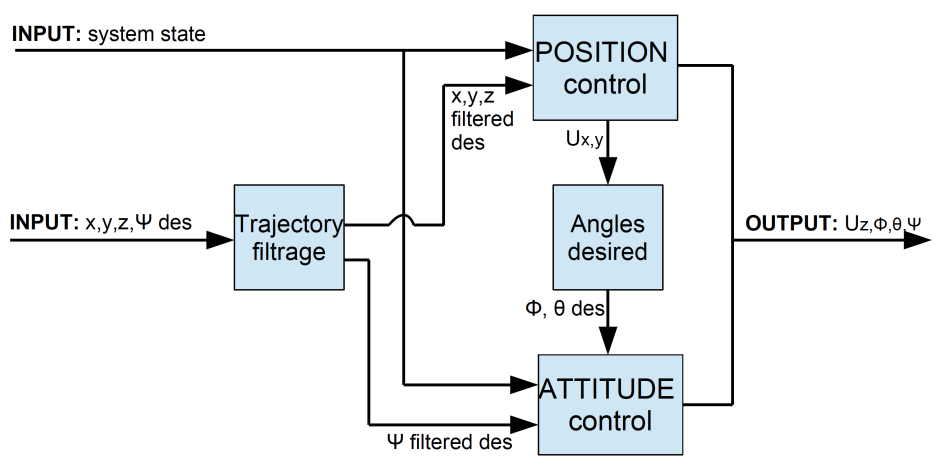


Figure 9: Hierachical control

4 Simulink models

Two quadrotor models are implemented:

- A nonlinear model which includes all the nonlinear aerodynamic coefficients (called complex);
- A nonlinear model which includes the identification in Section 3.1 (called simple).

The best strategy to work with these models is to use at first the simple one, which is more suitable to be studied for control and estimation techniques, then validate the algorithms with the complex model, checking the differences due to the neglected nonlinear terms. Their peculiarities are that:

- Only basic simulink blocks are used, avoid the necessity of interpreted or external matlab functions, allowing better compatibility with most of simulink versions. In particular the used blocks are: Constant, Mux, Demux, Gain, Integrator, In, Out, Product, Terminator, Switch, Compare To Constant, Relational Operator, Logical Operator, Abs, Add, Divide, Math Function, Product of Elements, Sqrt, Subtract, Sum of Elements, Unary Minus, Unit Delay (it is only used in the complex model), Trigonometric Function, Subsystem, While Iterator Subsystem (it is only used in the complex model), From, Goto, Stop Simulation, To Workspace, Cross vector.
- Auxiliary matlab/simulink toolbox, such as the purchasable Aerospace toolbox provided by Simulink, is not needed except of the basic simulink package;
- They are based on a detailed model ensuring a good quality simulator close to the reality;
- They are fully configurable changing the parameter values, allowing to study any kind of quadrotor having the shape similar to the commercial Parrot AR 2.0 Drone;
- The complex model can recognize the vortex ring state and stop the simulation accordingly. When the variable "test", in matlab workspace, is equal to 1.

Input and Output of the model are defined as (see Figures 12, 13):

- Input
  - Rotors velocity: One vector of angular velocities expressed in rad/s, where the 4 rotors are indicated with their signs according to the velocity direction, ex: $+$, $-$, $+-$;
  - Wind: One vector of the wind velocity expressed in earth frame of the 3 components in x, y, z axes in m/s.
- Output
  - State: One vector of 12 components of the state of the system. In order: 3 linear velocities in earth frame expressed in m/s, 3 angular velocities expressed in rad/s, 3 angles expressed in rad, 3 position expressed in m.

Aerodynamic coefficients, forces, moments and flight dynamics are illustrated in the schemes in Figures 14, 15.
The following parameter values can be changed to simulate quadrotors with different shapes and in different environment conditions using the higly nonlinear coefficients (see Fig. 16):

- Environment:
  - Gravity acceleration;
  - Air density.
- Quadrotor body:
  - Mass of the drone;
  - Moment of Inertia on x axes;
  - Moment of Inertia on y axes;
  - Moment of Inertia on x-z axes;
  - Angle ϵ (see Fig. 1);

  - Arm length;
  - Distance between the center of gravity of the drone and the rotors plane.

- Rotors:
  - Angle of attack of the blade at the root profile;
  - Twist of the blade;
  - Rotor radius;
  - Lift curve slope of the blade section;
  - Blades number;
  - main chord length of the blades;
  - Drag coefficient of the blade section;
  - Induced drag coefficient of the blade section.

- Blade flapping equation (see (7)):
  - Position of the spring (e);
  - Center of gravity of the blade;
  - Mass of the blade;
  - Inertia of the blade.

- Auxiliary parameter to solve the λ equation:
  - Maximum number of allowed iterations to solve the nonlinear λ equation, after that the system is in unstable position and automatically blocks the simulation: It means we are in vortex ring state.

- UAV initial conditions:
  - Initial linear velocity in earth frame;
  - Initial position in earth frame;
  - Initial angular velocity;
  - Initial angles.

The following parameter values can be changed to simulate quadrotors with different shapes and in different environment conditions using the simplifying hypothesis based on identification in Section 3.1 (see Fig. 17):

- Environment:
  - Gravity acceleration;
  - Air density.

- Quadrotor body:
  - Mass of the drone;
  - Moment of Inertia on x axes;
  - Moment of Inertia on y axes;
  - Angle ϵ (see Fig. 1);
  - Arm length;
  - Distance between the center of gravity of the drone and the rotors plane.

- Rotors:
  - Angle of attack of the blade at the root profile;
  - Rotor radius;
  - Lift curve slope of the blade section;

  - Blades number;
  - Main chord length of the blades;
  - Rotors drag coefficient (K_D);
  - Rotors thrust coefficient (K_z);
  - Rotors thrust coefficient in stationary phase (C_{Tstat}).

- UAV initial conditions:
  - Initial linear velocity in earth frame;
  - Initial position in earth frame;
  - Initial angular velocity;
  - Initial angles.

An auxiliary rotor model is also provided with a configurable mask, which allows to transform the control input to the rotors velocity taking into account the rotor transfer function. Simulink blocks used are: Constant, Sqrt, Product, Divide, Abs, Gain, Discrete-Time Integrator, In, Out, Sum. Input and Output of the model are defined as (see Fig. 18):

- Input:
  - control: vector of 4 components (U_z, U_ϕ, U_θ, U_ψ)
- Output:
  - omega: vector of 4 components. Rotors angular velocity in rad/s.

The following parameter values can be changed to simulate rotors with different shapes and in different environment conditions using the mixer matrix (9) (see Fig. 19):

- Rotor transfer function:
  - Denominator;
  - Numerator;
  - Rotor angular velocity in hover phase.
- Mixer Matrix:
  - Air density;
  - Rotor radius;
  - Rotor thrust coefficient in hover flight (C_{Tstat});
  - Drag coefficient of the blade section;
  - Angle of attack at the root profile of the blade section;
  - Rotors position (ϵ, see Fig. 1);
  - Blades number;
  - Main chord length of the blade section;
  - Drone arm length.

A Area of Attraction in matlab

```
%Area of Attraction
syms omega1_2 omega2_2 omega3_2 omega4_2 U v w p q r phi theta psi t1 t2 t3 t4 t5 t6 ∆x

P = sdpvar(9,9);
matrix=-double(vpa(subs(de2bi(0:511),0,-1)))';
∆ = sdpvar(1);

%choose initial conditions
x0=[0;0;0;0;0;0;0;0;0]; %change only the first three components if we are in different operating point
wind=[1;2;3]; %change wind value to check the AoA of a wind range: 3 cycles for

uu=[t3; t4; t5; t6];
xx=[x0(1); x0(2); x0(3); x0(4); x0(5); x0(6); t1; t2; x0(9)];

%System equations-----------------------------------------------------------------------------------
u=uu;
x=xx;
mu=sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)./(R.*abs(sqrt(u)));
%coeff drag
CH=Kd.*mu;
%coeff trust
CT=CTstat+K_CT*(x(3)-wind(3))./(R.*abs(sqrt(u)));
%inflow ratio
lambda=4.*(-CT./(sigma*a)+theta_0/6);
%coeff rolling moment
CRm=sigma*a.*((mu./8).*(lambda-(4/3)*theta_0));
%coeff torque
CQi=sigma.*a.*lambda.*(theta_0/6-lambda./4);
CQp=sigma./8.*(CD_0+CD_i*theta_0^2).*(1+mu.*mu)-CD_i.*(lambda.*theta_0/3-lambda.*lambda./4);
CQ=CQi+CQp;
%sin and cos
seno=[sin(epsilon); sin(pi/2+epsilon); sin(pi+epsilon); sin(3*pi/2+epsilon)];
coseno=[cos(epsilon); cos(pi/2+epsilon); cos(pi+epsilon); cos(3*pi/2+epsilon)];
%aerodynamic forces and momentum
 %avoid singularity
if (x(1)-wind(1))^2+(x(2)-wind(2))^2==0
    Fxj=0.*[1;1;1;1];
    Fyj=0.*[1;1;1;1];
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=0.*[1;1;1;1];
    Mj=0.*[1;1;1;1];
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
else
    Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
end
%aeroynamics
F=[sum(Fxj);sum(Fyj);sum(Fzj)];
J=[sum(Lj)-h*sum(Fyj)+l*sum(Fzj.*seno);...
   sum(Mj)+h*sum(Fxj)-l*sum(Fzj.*coseno);...
   sum(Nj)+l*sum(Fyj.*coseno)-l*sum(Fxj.*seno)];
%dynamic equations
f1=F(1)/m-g*sin(x(8))+(x(3)*x(5)-x(6)*x(2));
f2=F(2)/m+cos(x(8))*sin(x(7))*g-(x(6)*x(1)-x(3)*x(4));
f3=F(3)/m+cos(x(8))*cos(x(7))*g-(x(3)*x(2)-x(5)*x(2));
f4=J(1)/Ixx-x(5)*x(6)*(Izz-Iyy)/Ixx;
f5=J(2)/Iyy-x(4)*x(6)*(Ixx-Izz)/Iyy;
f6=J(3)/Izz-x(4)*x(5)*(Iyy-Ixx)/Izz;
f7=x(4)+tan(x(8))*(x(5)*sin(x(7))+x(6)*cos(x(7)));
f8=x(5)*cos(x(7))-x(6)*sin(x(7));
f9=(x(5)*sin(x(7))+x(6)*cos(x(7)))/cos(x(8));
FU=[f1;f2;f3;f4;f5;f6;f7;f8;f9];
%-----------------------------------------------------------------------------------------
```

```
%wind effect
t0=[0 0 363.666^2 363.666^2 363.666^2 363.666^2];
t = fsolve(@(tt) fcn_change(tt,FU,t1,t2,t3,t4,t5,t6),t0);



%System equations-------------------------------------------------------------------------------------
u=[omega1_2;omega2_2;omega3_2;omega4_2];
x=[U,v,w,p,q,r,phi,theta,psi];
mu=sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)./(R.*abs(sqrt(u)));
%coeff drag
CH=Kd.*mu;
%coeff trust
CT=CTstat+K_CT*(x(3)-wind(3))./(R.*abs(sqrt(u)));
%inflow ratio
lambda=4.*(-CT./(sigma*a)+theta_0/6);
%coeff rolling moment
CRm=sigma*a.*((mu./8).*(lambda-(4/3)*theta_0));
%coeff torque
CQi=sigma.*a.*lambda.*(theta_0/6-lambda./4);
CQp=sigma./8.*(CD_0+CD_i*theta_0^2).*(1+mu.*mu)-CD_i.*(lambda.*theta_0/3-lambda.*lambda./4);
CQ=CQi+CQp;
%sin and cos
seno=[sin(epsilon); sin(pi/2+epsilon); sin(pi+epsilon); sin(3*pi/2+epsilon)];
coseno=[cos(epsilon); cos(pi/2+epsilon); cos(pi+epsilon); cos(3*pi/2+epsilon)];
%aerodynamic forces and momentum
 %avoid singularity
    Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
%aeroynamics
F=[sum(Fxj);sum(Fyj);sum(Fzj)];
J=[sum(Lj)-h*sum(Fyj)+l*sum(Fzj.*seno);...
   sum(Mj)+h*sum(Fxj)-l*sum(Fzj.*coseno);...
   sum(Nj)+l*sum(Fyj.*coseno)-l*sum(Fxj.*seno)];
%dynamic equations
f1=F(1)/m-g*sin(x(8))+(x(3)*x(5)-x(6)*x(2));
f2=F(2)/m+cos(x(8))*sin(x(7))*g-(x(6)*x(1)-x(3)*x(4));
f3=F(3)/m+cos(x(8))*cos(x(7))*g-(x(3)*x(2)-x(5)*x(2));
f4=J(1)/Ixx-x(5)*x(6)*(Izz-Iyy)/Ixx;
f5=J(2)/Iyy-x(4)*x(6)*(Ixx-Izz)/Iyy;
f6=J(3)/Izz-x(4)*x(5)*(Iyy-Ixx)/Izz;
f7=x(4)+tan(x(8))*(x(5)*sin(x(7))+x(6)*cos(x(7)));
f8=x(5)*cos(x(7))-x(6)*sin(x(7));
f9=(x(5)*sin(x(7))+x(6)*cos(x(7)))/cos(x(8));
%-----------------------------------------------------------------------------------------

%linear system matrices
AM = jacobian([f1, f2, f3, f4, f5, f6, f7, f8, f9], [x(1), x(2), x(3), x(4), x(5), x(6), x(7), ...
    x(8), x(9)]);
BM = jacobian([f1, f2, f3, f4, f5, f6, f7, f8, f9], [u(1), u(2), u(3), u(4)]);

%to calculate jacobian
U=x0(1);v=x0(2);w=x0(3);p=x0(4);q=x0(5);r=x0(6);phi=t(1);theta=t(2);psi=x0(9);
omega1_2=t(3);omega2_2=t(4);omega3_2=t(5);omega4_2=t(6);

BM=eval(BM);
AM=eval(AM);

%gain control
eigenvalues=[-1, -2, -3, -4, -5, -6, -7, -8, -9];
K=place(AM,BM,eigenvalues);

%find P and optimize Linear system
epsilon = 1e-3;
G = [P ≥ epsilon*eye(9) , (AM-BM*K)'*P+P*(AM-BM*K)≤ -epsilon*eye(9)];
optimize(G,[],sdpsettings('solver','sdpt3'));
```

```
%find Δ and optimize NonLinear system
for ii=1:512
    Δx=matrix(:,ii).*Δ;
    %NonLinear System F(x0+Δx,u0-KΔx,d)
%System equations-----------------------------------------------------------------------------------
u=[t(3);t(4);t(5);t(6)]-K*Δx;
x=[x0(1);x0(2);x0(3);x0(4);x0(5);x0(6);t(1);t(2);x0(9)]+Δx;
mu=sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)./(R.*abs(sqrt(u)));
%coeff drag
CH=Kd.*mu;
%coeff trust
CT=CTstat+K_CT*(x(3)-wind(3))./(R.*abs(sqrt(u)));
%inflow ratio
lambda=4.*(-CT./(sigma*a)+theta_0/6);
%coeff rolling moment
CRm=sigma*a.*((mu./8).*(lambda-(4/3)*theta_0));
%coeff torque
CQi=sigma.*a.*lambda.*(theta_0/6-lambda./4);
CQp=sigma./8.*(CD_0+CD_i*theta_0^2).*(1+mu.*mu)-CD_i.*(lambda.*theta_0/3-lambda.*lambda./4);
CQ=CQi+CQp;
%sin and cos
seno=[sin(epsilon); sin(pi/2+epsilon); sin(pi+epsilon); sin(3*pi/2+epsilon)];
coseno=[cos(epsilon); cos(pi/2+epsilon); cos(pi+epsilon); cos(3*pi/2+epsilon)];
%aerodynamic forces and momentum
 %avoid singularity
    Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
%aeroynamics
F=[sum(Fxj);sum(Fyj);sum(Fzj)];
J=[sum(Lj)-h*sum(Fyj)+l*sum(Fzj.*seno);...
   sum(Mj)+h*sum(Fxj)-l*sum(Fzj.*coseno);...
   sum(Nj)+l*sum(Fyj.*coseno)-l*sum(Fxj.*seno)];
%dynamic equations
f1=F(1)/m-g*sin(x(8))+(x(3)*x(5)-x(6)*x(2));
f2=F(2)/m+cos(x(8))*sin(x(7))*g-(x(6)*x(1)-x(3)*x(4));
f3=F(3)/m+cos(x(8))*cos(x(7))*g-(x(3)*x(2)-x(5)*x(2));
f4=J(1)/Ixx-x(5)*x(6)*(Izz-Iyy)/Ixx;
f5=J(2)/Iyy-x(4)*x(6)*(Ixx-Izz)/Iyy;
f6=J(3)/Izz-x(4)*x(5)*(Iyy-Ixx)/Izz;
f7=x(4)+tan(x(8))*(x(5)*sin(x(7))+x(6)*cos(x(7)));
f8=x(5)*cos(x(7))-x(6)*sin(x(7));
f9=(x(5)*sin(x(7))+x(6)*cos(x(7)))/cos(x(8));
F=[f1;f2;f3;f4;f5;f6;f7;f8;f9];
%-------------------------------------------------------------------------------------
    L = [Δx≥epsilon*ones(9,1), 2.*Δx'*P*F≤-epsilon*ones(9,1)];
    optimize(L,[],sdpsettings('solver','sdpt3'));

    %Δ value for each iteration: first element of each column
    Δ_value(ii)=abs(Δx(1));
end

%radius of the AoA: minimum of a vector which has 512 elements
AoA_radius=min(Δ_value);
```

B Feedback control in simulink/matlab

```
function [K_u_x] = fun_wind_effect(vector)
syms omega1_2 omega2_2 omega3_2 omega4_2 U0 v0 w0 p0 q0 r0 phi0 theta0 psi0 t1 t2 t3 t4 t5 t6

%aerodynamic coefficient values
chord=0.0175;
bladesnumber=2;
CTstat=0.0223;
h=-0.0250;
```

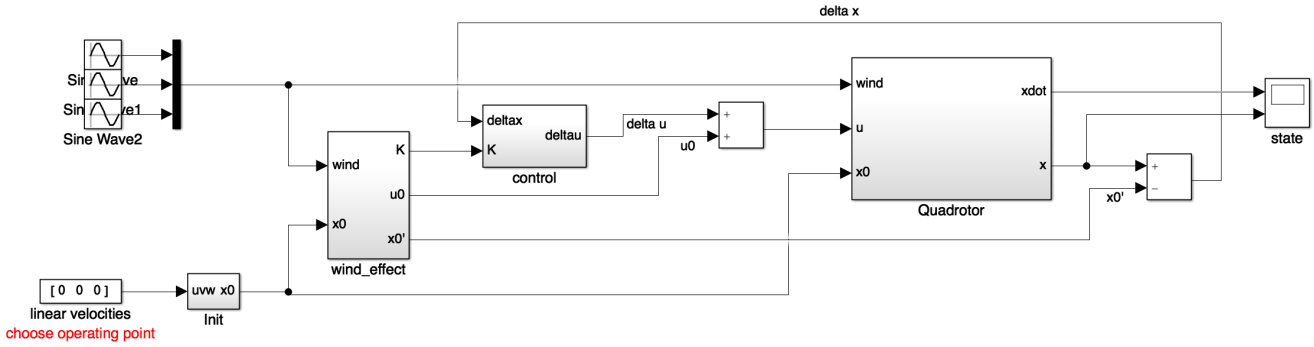


Figure 10: Overall simulink scheme of the feedback control

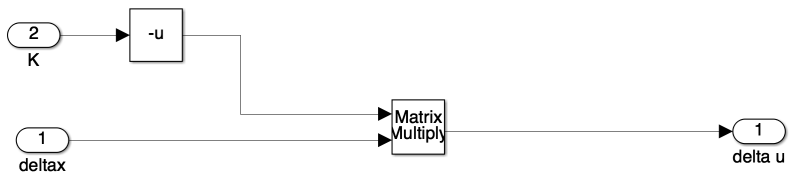


Figure 11: Simulink block of the feedback control

```
Ixx=3.56e-3;
Iyy=4.02e-3;
Izz=7.12e-3;
m=0.4720;
rho=1.2500;
R=0.1;
A=3.1416*R^2;
sigma=(bladesnumber*chord)/(3.1416*R);
theta_0=23.9;
l=0.185;
a=4.6542;
epsilon=-3.1416/2;
Kd=0.06;
CD_0=2.15;
CD_i=0;
g=9.81;
K_CT=0.09;

wind=vector(1:3);
x0=vector(4:12);

uu=[t3; t4; t5; t6];
xx=[x0(1); x0(2); x0(3); x0(4); x0(5); x0(6); t1; t2; x0(9)];

%System equations-----------------------------------------------------------------------------
u=uu;
x=xx;
if not(isempty(find(¬u, 1))) || (x(1)-wind(1))^2+(x(2)-wind(2))^2==0
    mu = 0.*[1;1;1;1];
else
    mu=sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)./(R.*abs(sqrt(u)));
end
%coeff drag
CH=Kd.*mu;
%coeff trust
if not(isempty(find(¬u, 1))) || x(3)-wind(3)==0
    CT=0.*[1;1;1;1];
else
    CT=CTstat+K_CT*(x(3)-wind(3))./(R.*abs(sqrt(u)));
end
```

```
%inflow ratio
lambda=4.*(-CT./(sigma*a)+theta_0/6);
%coeff rolling moment
CRm=sigma*a.*((mu./8).*(lambda-(4/3)*theta_0));
%coeff torque
CQi=sigma.*a.*lambda.*(theta_0/6-lambda./4);
CQp=sigma./8.*(CD_0+CD_i*theta_0^2).*(1+mu.*mu)-CD_i.*(lambda.*theta_0/3-lambda.*lambda./4);
CQ=CQi+CQp;
%sin and cos
seno=[sin(epsilon); sin(pi/2+epsilon); sin(pi+epsilon); sin(3*pi/2+epsilon)];
coseno=[cos(epsilon); cos(pi/2+epsilon); cos(pi+epsilon); cos(3*pi/2+epsilon)];
%aerodynamic forces and momentum
 %avoid singularity
if (x(1)-wind(1))^2+(x(2)-wind(2))^2==0
    Fxj=0.*[1;1;1;1];
    Fyj=0.*[1;1;1;1];
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=0.*[1;1;1;1];
    Mj=0.*[1;1;1;1];
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
else if x(1)-wind(1)==0
   Fxj=0.*[1;1;1;1];
   Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
   Fzj=-rho.*A.*R^2.*u.*CT;
   Lj=0.*[1;1;1;1];
   Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
   Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
    else if x(2)-wind(2)==0
        Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
        Fyj=0.*[1;1;1;1];
        Fzj=-rho.*A.*R^2.*u.*CT;
        Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
        Mj=0.*[1;1;1;1];
        Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
        else
        Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
        Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
        Fzj=-rho.*A.*R^2.*u.*CT;
        Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
        Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
        Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
        end
    end
end
%aeroynamics
F=[sum(Fxj);sum(Fyj);sum(Fzj)];
J=[sum(Lj)-h*sum(Fyj)+l*sum(Fzj.*seno);...
   sum(Mj)+h*sum(Fxj)-l*sum(Fzj.*coseno);...
   sum(Nj)+l*sum(Fyj.*coseno)-l*sum(Fxj.*seno)];
%dynamic equations
f1=F(1)/m-g*sin(x(8))+(x(3)*x(5)-x(6)*x(2));
f2=F(2)/m+cos(x(8))*sin(x(7))*g-(x(6)*x(1)-x(3)*x(4));
f3=F(3)/m+cos(x(8))*cos(x(7))*g-(x(3)*x(2)-x(5)*x(2));
f4=J(1)/Ixx-x(5)*x(6)*(Izz-Iyy)/Ixx;
f5=J(2)/Iyy-x(4)*x(6)*(Ixx-Izz)/Iyy;
f6=J(3)/Izz-x(4)*x(5)*(Iyy-Ixx)/Izz;
f7=x(4)+tan(x(8))*(x(5)*sin(x(7))+x(6)*cos(x(7)));
f8=x(5)*cos(x(7))-x(6)*sin(x(7));
f9=(x(5)*sin(x(7))+x(6)*cos(x(7)))/cos(x(8));
FU=[f1;f2;f3;f4;f5;f6;f7;f8;f9];
%-------------------------------------------------------------------------------------

%wind effect
t0=[0 0 363.666^2 363.666^2 363.666^2 363.666^2];
t = fsolve(@(tt) fcn_change(tt,FU,t1,t2,t3,t4,t5,t6),t0);


%System equations-------------------------------------------------------------------------------------------
u=[omega1_2;omega2_2;omega3_2;omega4_2];
x=[U0,v0,w0,p0,q0,r0,phi0,theta0,psi0];
mu=sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)./(R.*abs(sqrt(u)));
```

```
%coeff drag
CH=Kd.*mu;
%coeff trust
%avoid singularity
CT=CTstat+K_CT*(x(3)-wind(3))./(R.*abs(sqrt(u)));
%inflow ratio
lambda=4.*(-CT./(sigma*a)+theta_0/6);
%coeff rolling moment
CRm=sigma*a.*((mu./8).*(lambda-(4/3)*theta_0));
%coeff torque
CQi=sigma.*a.*lambda.*(theta_0/6-lambda./4);
CQp=sigma./8.*(CD_0+CD_i*theta_0^2).*(1+mu.*mu)-CD_i.*(lambda.*theta_0/3-lambda.*lambda./4);
CQ=CQi+CQp;
%sin and cos
seno=[sin(epsilon); sin(pi/2+epsilon); sin(pi+epsilon); sin(3*pi/2+epsilon)];
coseno=[cos(epsilon); cos(pi/2+epsilon); cos(pi+epsilon); cos(3*pi/2+epsilon)];
%aerodynamic forces and momentum
 %avoid singularity

    Fxj=-rho.*A.*R^2.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fyj=-rho.*A.*R^2.*((x(2)-wind(2))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CH.*u;
    Fzj=-rho.*A.*R^2.*u.*CT;
    Lj=[1;-1;1;-1].*rho.*A.*R^3.*((x(1)-wind(1))/sqrt((x(1)-wind(1))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Mj=[1;-1;1;-1].*rho.*A.*R^3.*((x(2)-wind(2))/sqrt((x(2)-wind(2))^2+(x(2)-wind(2))^2)).*CRm.*u;
    Nj=[1;-1;1;-1].*rho.*A.*R^3.*CQ.*u;
%aeroynamics
F=[sum(Fxj);sum(Fyj);sum(Fzj)];
J=[sum(Lj)-h*sum(Fyj)+l*sum(Fzj.*seno);...
   sum(Mj)+h*sum(Fxj)-l*sum(Fzj.*coseno);...
   sum(Nj)+l*sum(Fyj.*coseno)-l*sum(Fxj.*seno)];
%dynamic equations
f1=F(1)/m-g*sin(x(8))+(x(3)*x(5)-x(6)*x(2));
f2=F(2)/m+cos(x(8))*sin(x(7))*g-(x(6)*x(1)-x(3)*x(4));
f3=F(3)/m+cos(x(8))*cos(x(7))*g-(x(3)*x(2)-x(5)*x(2));
f4=J(1)/Ixx-x(5)*x(6)*(Izz-Iyy)/Ixx;
f5=J(2)/Iyy-x(4)*x(6)*(Ixx-Izz)/Iyy;
f6=J(3)/Izz-x(4)*x(5)*(Iyy-Ixx)/Izz;
f7=x(4)+tan(x(8))*(x(5)*sin(x(7))+x(6)*cos(x(7)));
f8=x(5)*cos(x(7))-x(6)*sin(x(7));
f9=(x(5)*sin(x(7))+x(6)*cos(x(7)))/cos(x(8));
%-----------------------------------------------------------------------------------

%linear system matrices
AM = jacobian([f1, f2, f3, f4, f5, f6, f7, f8, f9], [x(1), x(2), x(3), x(4), x(5), x(6), x(7), ...
    x(8), x(9)]);
BM = jacobian([f1, f2, f3, f4, f5, f6, f7, f8, f9], [u(1), u(2), u(3), u(4)]);

%variables to evaluate jacobian
U0=x0(1);v0=x0(2);w0=x0(3);p0=x0(4);q0=x0(5);r0=x0(6);phi0=t(1);theta0=t(2);psi0=x0(9);
omega1_2=t(3);omega2_2=t(4);omega3_2=t(5);omega4_2=t(6);

BM=eval(BM);
AM=eval(AM);

%gain control
eigenvalues=[-1, -2, -3, -4, -5, -6, -7, -8, -9]; %TODO: optimize the choise of the eigenvalues
K=place(AM,BM,eigenvalues);

K_u_x=[ K(:) ; t(3);t(4);t(5);t(6);x0(1);x0(2);x0(3);x0(4);x0(5);x0(6);t(1);t(2);x0(9)];
end
```

```
function F = fcn_change(tt,F1,t1,t2,t3,t4,t5,t6)
    F=double(vpa(subs(F1,[t1,t2,t3,t4,t5,t6],[tt(1),tt(2),tt(3),tt(4),tt(5),tt(6)])));
end
```

C Quadrotor models in simulink

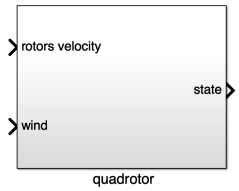


Figure 12: Simulink quadrotor model

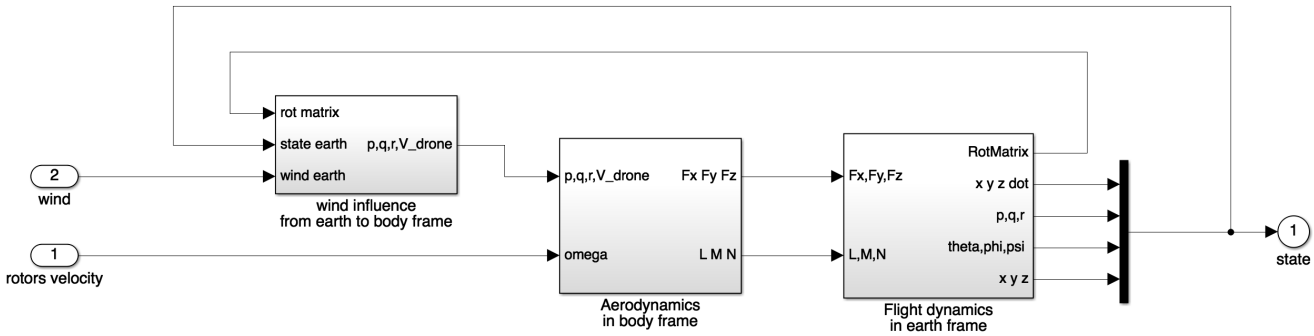


Figure 13: Simulink submodels inside the quadrotor model

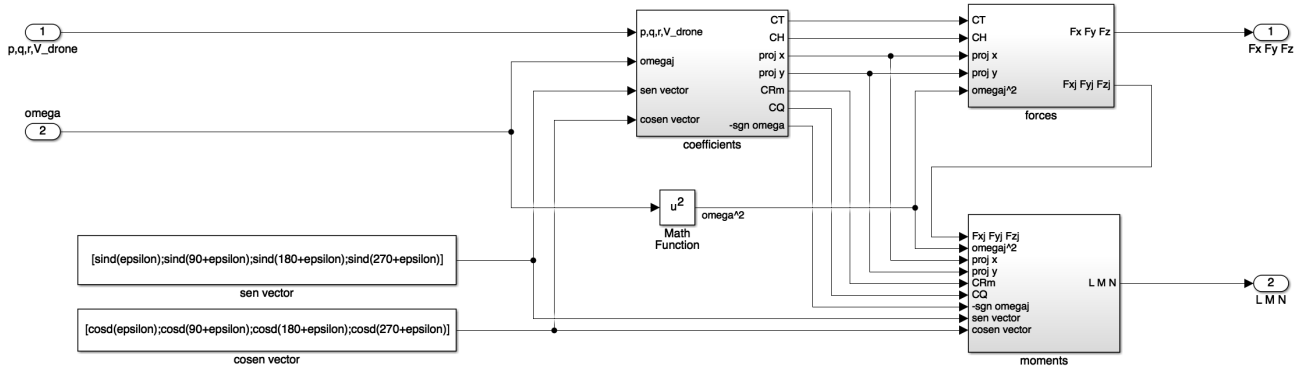


Figure 14: Aerodynamic coefficients, forces and moments. The "coefficient" submodel is different in complex and simple models.

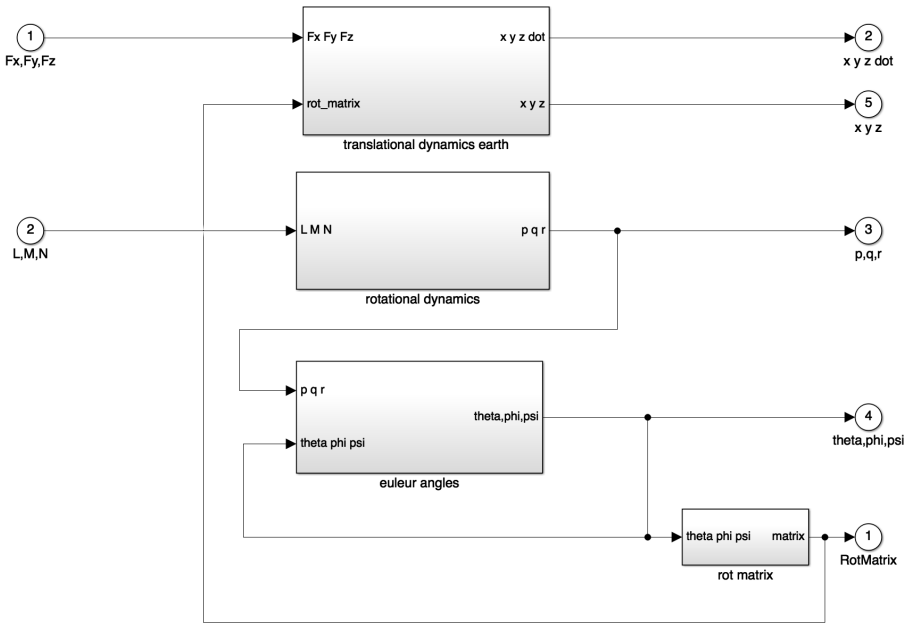


Figure 15: "Flight dynamics" simulink submodel

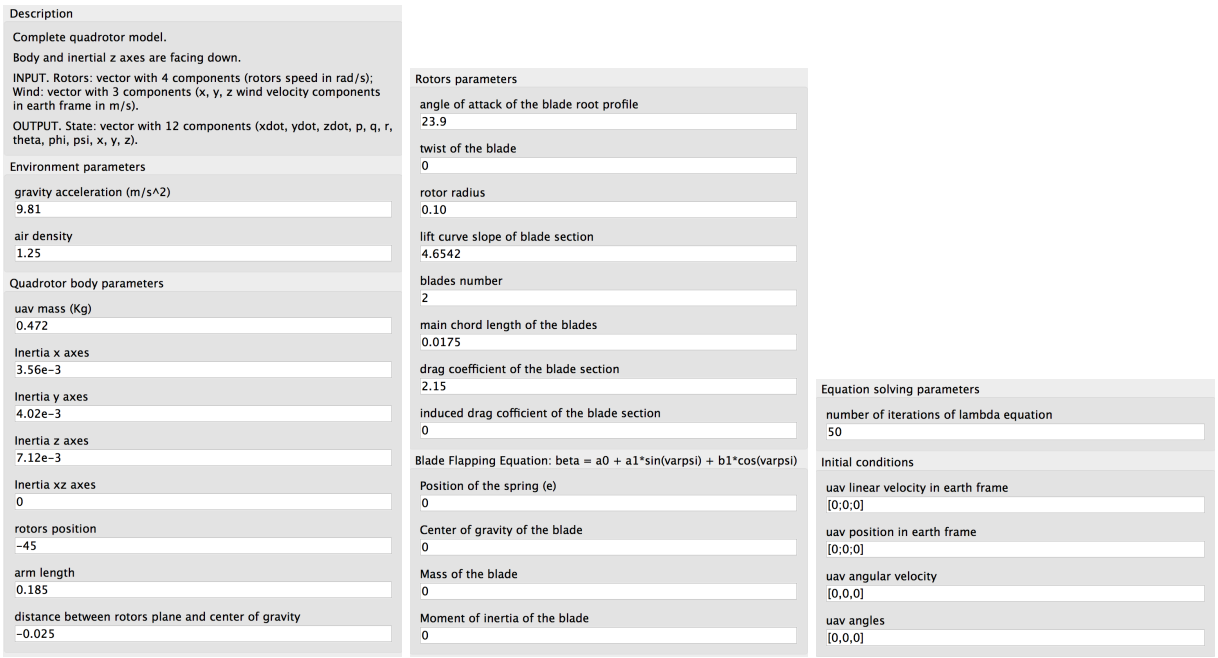


Figure 16: Fully configurable mask of the implemented complex simulink model. (Values are an example).

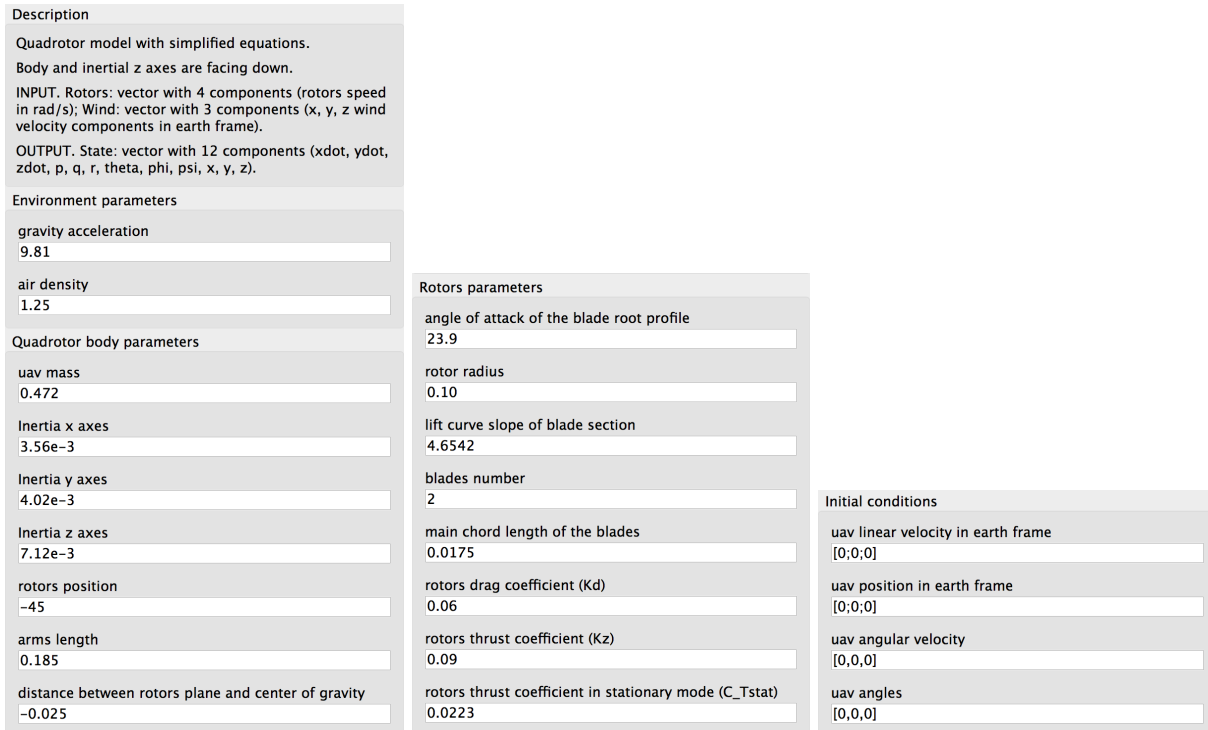


Figure 17: Fully configurable mask of the implemented simple simulink model. (Values are an example).

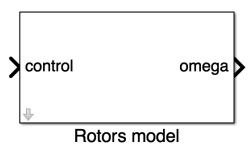


Figure 18: From control to rotors.

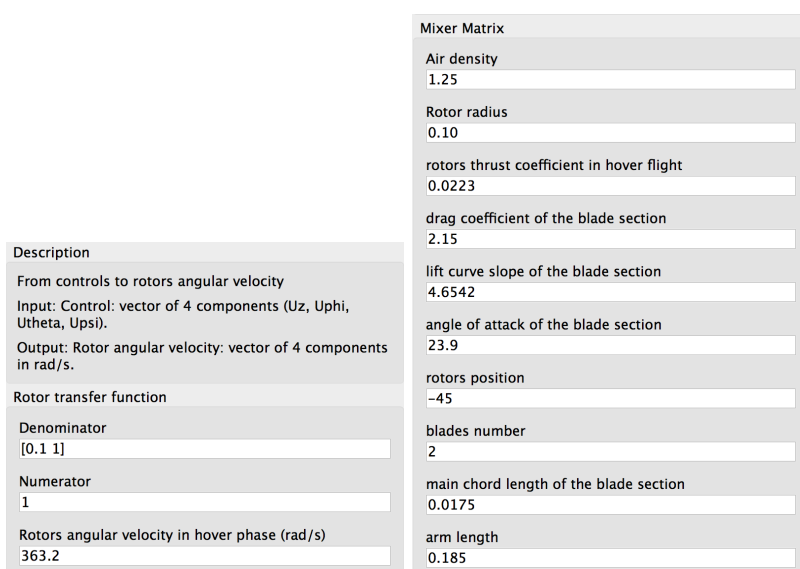


Figure 19: Fully configurable mask of the model from control to rotors angular velocity. (Values are an example).