ACTIVE MODEL ROCKET STABILIZATION VIA COLD GAS THRUSTERS

Danylo Malyuta[1], Xavier Collaud[1], Mikael Gaspar[1], Gautier Rouaze[1], Raimondo Pictet[1], Anton Ivanov[1], and Nikolay Mullin[2]

[1]*Space Engineering Center (eSpace), EPFL, Station 11, 1015, Lausanne, Switzerland.*
[2]*Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow 105005, Russia.*

ABSTRACT

This paper describes the development and testing of a reaction control system (RCS) for a model rocket named FALCO-4. The rocket uses cold gas jets to keep itself perfectly vertical at low speeds. We first describe the mechanical layout of FALCO-4 and the characteristics of the cold gas propulsion system. We then propose a dynamical model of the rocket and a control scheme based on decoupled PID regulators for roll, pitch and yaw. The control scheme is then evaluated based on MATLAB® simulation and flight data collected over 4 test flights that showed the RCS to work well even though vertical orientation was only partially achieved due to insufficient thrust from the cold gas thrusters. Finally, the present approach is evaluated and improvements are suggested.

Key words: model rocket; active stabilization; PID control; vertical landing.

1. INTRODUCTION

Traditionally, rockets are built as expendable systems. Today, however, there is a growing trend towards designing partially or fully reusable rocket systems [1] [2] [3], spurred by the prospect of massive cost reductions (up to 100 times [1]) and the need for such systems in future manned planetary exploration. It is likely that all future rockets will include some reusability aspect. As such, the present project is an effort to accustom engineering students to the "reusable rocket" paradigm and the challenges that it entails.

Our team aims to design a cheap and accessible model rocket featuring all the necessary technology to be a plug-and-play platform for testing rocket landing algorithms. The present paper discusses the first step in this project – the design and validation of a cold gas reaction control system for maintaining a vertical attitude at low speeds. The mission for this paper is to maintain a vertical attitude around apogee.

This paper is organized as follows: Section 2 describes the mechanical layout of the rocket and the characteristics of the cold gas reaction control system. Section 3 develops the dynamical model of the rocket and the control scheme used. Section 4 presents the results from four test flights. Finally, Section 5 evaluates the effectiveness and suggests improvements over the present system.

2. MECHANICAL DESIGN

2.1. Rocket Structure

The final design, shown in Figure 1, contains the following major components:

1. LV-Haack (minimum drag) nose cone ($C = 1/3$).
2. Fiberglass tube (avionics compartment).
3. Fiberglass tube (RCS compartment).
4. Fiberglass tube (parachute and propulsion compartment).
5. Tapered, swept flat-plate fins (fiberglass-reinforced lemonwood).
6. Plywood (1 [mm]) and balsa (8 [mm]) sandwhich mounting bar.
7. Raspberry-Pi 1 Model B+ [4] (with USB ports removed).
8. 3D-printed cowling (ABS plastic) for Raspberry Pi Spy Camera [5].
9. DC/DC boost converter [6].
10. Tube coupler and RCS nozzle ring (pine).
11. RCS valves [7].
12. Pressure tubing [8].
13. Pressure regulator [9].
14. CO_2 cartridge [10].
15. CO_2 cartridge retainer (pine).
16. Horizontal CMOS camera [11].
17. Tube coupler (pine).
18. Parachute (drogue and main).
19. Engine mount tube (cardboard).

20. Centering ring (pine).
21. Pressure sensors [12].
22. Power electronics and MSP430G2553 slave microcontroller.
23. RCS activation alarm and debugging LEDs.
24. LiPo battery storage [13].
25. Engine nozzle.

Weighing 2560 [g], the rocket is 1.36 [m] long, 7.62 [cm] in diameter and is composed of two bodies that separate after apogee for parachute deployment. The fuselage length is determined by the size of the RCS and avionics compartments as well as a static stability requirement. The final static margin is 6.08 calibers between the center of mass and center of pressure, offering sufficient aerodynamic stability at lift-off. The diameter was dictated by the available tubing serving as a mold.

For propulsion, the Aerotech RMS H-238 reusable motor with 175 [Ns] specific impulse was chosen in order to obtain an off-launchrod velocity of 15-18 $\left[\frac{\text{m}}{\text{s}}\right]$ required for passive stabilization by the fins. The burn time is less than 1 [s] during which the rocket undergoes an acceleration of up to 78.9 $\left[\frac{\text{m}}{\text{s}^2}\right]$ and reaches a maximum speed of about 59 $\left[\frac{\text{m}}{\text{s}}\right]$. The maximum altitude is approximately 187 [m].

Finally, Figure 2 shows a top-level overview of the avionics that flew aboard the rocket. In addition, all software source code in MATLAB® and C is available at [14].

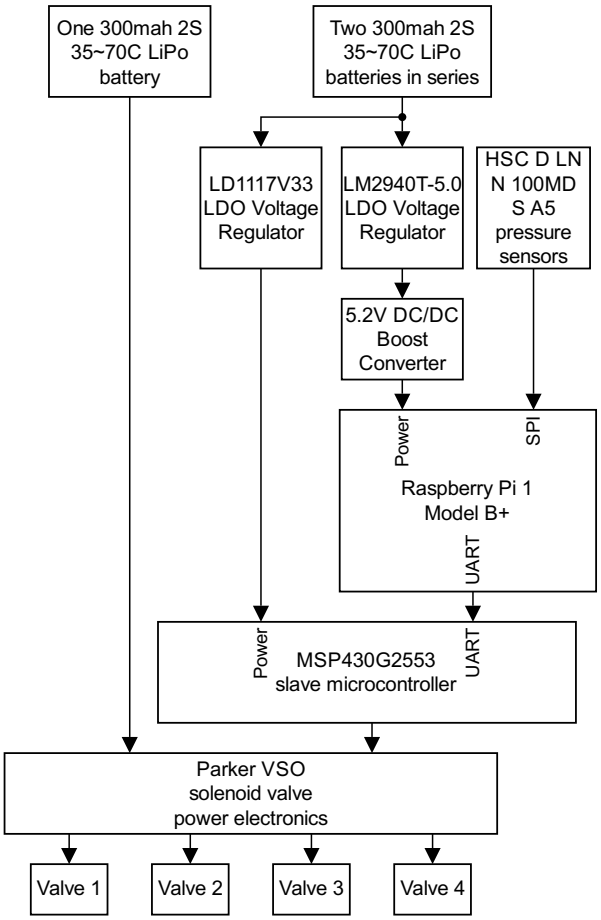

Figure 2. High-level layout of the rocket avionics.

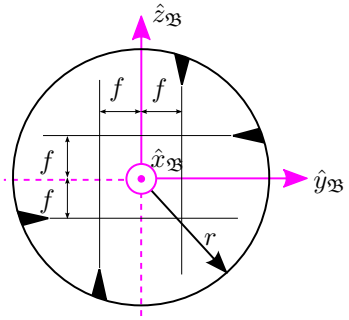

Figure 3. Layout of the RCS (top view). $f = 5$ [mm], $r = 3.81$ [cm].

2.2. Reaction Control System

The cold gas reaction control system is composed of a cartridge of 68 [g] of liquid CO_2 pressurized at $p_0 = 60$ [bar], a pressure regulator to $p_1 = 6$ [bar] and a set of pressure tubes leading to four proportional solenoid valves that expell gas in their respective directions. Figure 3 shows the geometric nozzle layout. The entire RCS assembly weighs 834 [g].

We now discuss the valve thrust calculations that drive the selection of these components in view of minimal weight and a 1 [N] thrust per valve requirement (this thrust was not achieved by the final RCS). We use numerical values relating to the aforementioned RCS components, however the following calculations have been done for several component candidates in order to select the present combination.

We assume CO_2 to be an ideal gas and the flow through the valve to undergo an isenthalpic throttling process. The flow conditions upstream are determined by the pressure regulator (see the layout schematic in Figure 4) and are set to the following values:

$$\begin{cases} T_1 = 273 \text{ [K]}, \\ p_1 = 6 \text{ [bar]}. \end{cases}$$

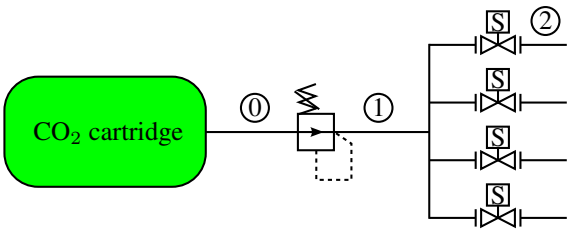

Figure 4. Pressure connections schematic for the RCS.

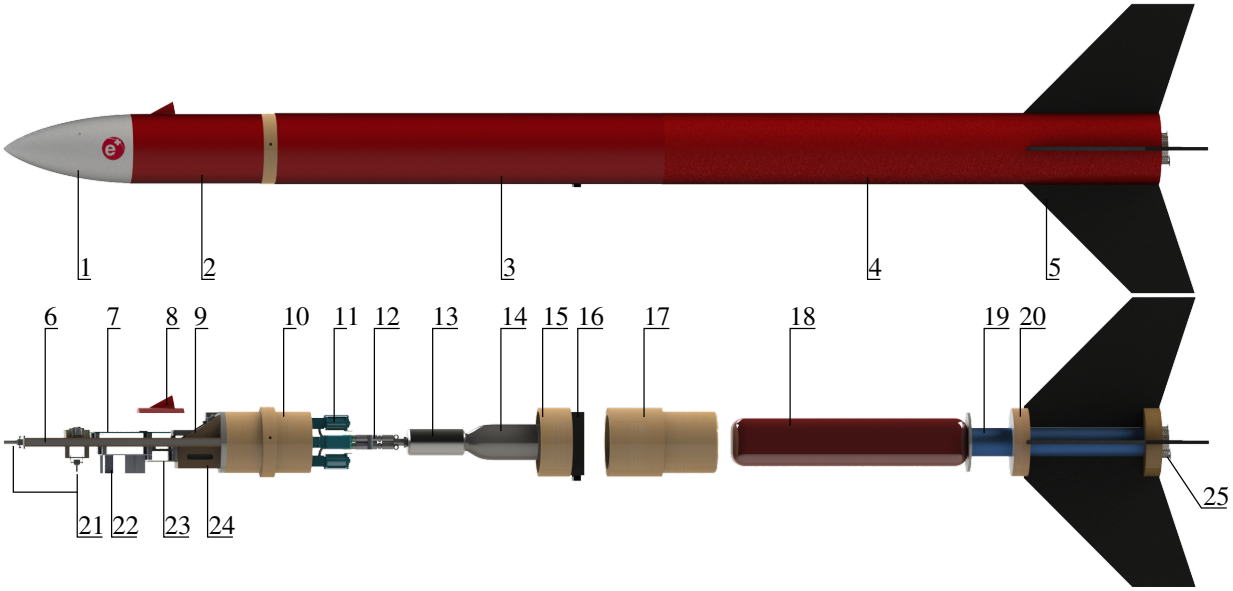

Figure 1. Structural layout of FALCO-4.

Where the indices refer to the locations of corresponding circled numbers in Figure 4. It is further assumed that $p_2 = p_{\text{atm}} = 101325$ [Pa]. It is then possible to compute the temperature of the flow after the valve by making the isentropic flow assumption:

$$T_2 = T_1 \left(\frac{p_2}{p_1}\right)^{\frac{\gamma-1}{\gamma}} \approx 184.48 \text{ [K]}, \tag{1}$$

where the CO_2 specific heat ratio γ is 1.28. We check for sonic conditions at the valve orifice, assumed to be the narrowest part of the flow:

$$p_{\text{cr}} = \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} p_1 \approx 329620 \text{ [Pa]} > p_2. \tag{2}$$

Therefore the flow downstream of the valve orifice is equal to the speed of sound. The nozzle exit velocity is therefore:

$$U_2 = \sqrt{\gamma R T_2} \approx 211.56 \left[\frac{\text{m}}{\text{s}}\right], \tag{3}$$

where the CO_2 gas constant R is 189 $\left[\frac{\text{J}}{\text{kg}\cdot\text{K}}\right]$. We now use the isenthalpic throttling process assumption to compute the mass flow rate through the valve:

$$\dot{m}_{\max} = c_d A \frac{p_1}{\sqrt{RT_1}} \sqrt{\gamma \left(\frac{p_{cr}}{p_1}\right)^{\frac{\gamma+1}{\gamma}}} \approx 1.7069 \left[\frac{\text{g}}{\text{s}}\right]. \tag{4}$$

Note that (4) gives the maximum mass flow rate since the maximum valve opening area A is used for the calculation. Finally, the maximum thrust is:

$$F_{\max} = U_2 \dot{m}_{\max} \approx 0.3611 \text{ [N]}. \tag{5}$$

Note that this value is very close to experimental measurements, for which maximum thrust varied between 0.32 and 0.37 [N]. In the future, maximum thrust could be increased by using de Laval nozzles instead of ejecting gas directly from the tube.

3. CONTROL SYSTEM DESIGN

3.1. Dynamical model

In this section, the dynamical attitude model of the rocket is developed using Euler-Lagrange formalism. Because only rocket attitude is controlled, translational dynamics are not modelled under the assumption that translational and rotational motions are decoupled about the center of mass. Furthermore, as the control system is turned on *after* engine cutoff, rocket mass is assumed to be constant.

The Lagrangian of a system is defined as:

$$\begin{aligned} \mathscr{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \mathscr{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathscr{U}(\boldsymbol{q}) \\ &= \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathscr{U}(\boldsymbol{q}), \end{aligned} \tag{6}$$

where $\mathscr{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}$ is the kinetic energy, $\mathscr{U}(\boldsymbol{q}) \in \mathbb{R}$ is the potential energy, $M(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix and $\boldsymbol{q} \in \mathbb{R}^n$ is the vector of generalized coordinates. In this case, $\boldsymbol{q} = [\varphi \quad \theta \quad \psi]^T \in \mathbb{R}^3$ and the Tait-Bryan Euler angle convention is used with (ψ, θ, φ) representing respectively (yaw, pitch, roll).

The Euler-Lagrange equation that describes the system dynamics is:

$$\frac{\text{d}}{\text{d}t}\left(\frac{\partial \mathscr{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial \mathscr{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \mathscr{F}(\boldsymbol{q}), \tag{7}$$

where $\mathscr{F}(\boldsymbol{q}) \in \mathbb{R}^n$ is the vector of generalized forces and torques acting on the rocket. [15] shows that substituting (6) into (7) gives:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \mathscr{F}(\boldsymbol{q}), \tag{8}$$

where $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal force matrix and $G(\boldsymbol{q}) \equiv (\partial \mathscr{U}(\boldsymbol{q})/\partial \boldsymbol{q}) \in \mathbb{R}^n$ is the gra-

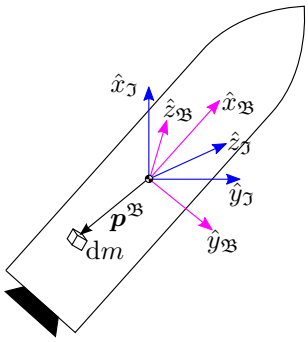

Figure 5. Infinitesimal element kinematics.

dient of potential energy with respect to the generalized coordinates.

The $C(\boldsymbol{q},\dot{\boldsymbol{q}})$ matrix is computed using the Christoffel symbols of the first kind:

$$c_{ij} = \sum_{k=1}^{n} \left\{ \frac{1}{2} \left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \dot{q}_k \right\}, \quad (9)$$

where c_{ij} and m_{ij} are the elements at row i and column j of the $C(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $M(\boldsymbol{q})$ matrices respectively. Given that $C(\boldsymbol{q},\dot{\boldsymbol{q}})$ depends on $M(\boldsymbol{q})$ via (9), we must only find the $M(\boldsymbol{q})$, $\boldsymbol{G}(\boldsymbol{q})$ and $\mathscr{F}(\boldsymbol{q})$ in order to obtain the full attitude dynamics.

3.1.1. Finding the $M(\boldsymbol{q})$ matrix

Equation (6) implies that the $M(\boldsymbol{q})$ matrix may be found by rewriting the kinetic energy in quadratic form. Let us consider an infinitesimal element of mass $\mathrm{d}m$ on the rocket as shown in Figure 5. The element's position vector $\boldsymbol{p}^{\mathfrak{B}}$ (expressed in the $\mathfrak{B}$ frame) is fixed in the $\mathfrak{B}$ frame as the rocket is assumed to be a rigid body. The kinetic energy, however, must be written in the $\mathfrak{I}$ (inertial) frame. We can write:

$$\boldsymbol{p}^{\mathfrak{I}} = R^{\mathfrak{I}}_{\mathfrak{B}} \boldsymbol{p}^{\mathfrak{B}}, \quad (10)$$

where $R^{\mathfrak{I}}_{\mathfrak{B}}$ is the rotation matrix from the $\mathfrak{B}$ frame to the $\mathfrak{I}$ frame. From the Tait-Bryan convention we have:

$$R^{\mathfrak{B}}_{\mathfrak{I}} = R_\varphi R_\theta R_\psi, \quad (11)$$

$$R^{\mathfrak{I}}_{\mathfrak{B}} = \left(R_\varphi R_\theta R_\psi\right)^T = R_\psi^T R_\theta^T R_\varphi^T. \quad (12)$$

Now that we know an infinitesimal element's position in the $\mathfrak{I}$ frame, its velocity in the $\mathfrak{I}$ frame can be found by differentiating (10):

$$\dot{\boldsymbol{p}}^{\mathfrak{I}} = \dot{R}^{\mathfrak{I}}_{\mathfrak{B}} \boldsymbol{p}^{\mathfrak{B}} + R^{\mathfrak{I}}_{\mathfrak{B}} \dot{\boldsymbol{p}}^{\mathfrak{B}}. \quad (13)$$

As the position of the infinitesimal element is fixed, $\dot{\boldsymbol{p}}^{\mathfrak{B}} = \boldsymbol{0}$. To further develop (13), define the skew-symmetric matrix $S(\boldsymbol{\omega})$:

$$S(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \quad (14)$$

The rotation matrix has the property:

$$\dot{R}^{\mathfrak{I}}_{\mathfrak{B}} = R^{\mathfrak{I}}_{\mathfrak{B}} S(\boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}), \quad (15)$$

where $\boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}$ is the angular velocity of the $\mathfrak{B}$ frame with respect to the $\mathfrak{I}$ frame, written in the $\mathfrak{B}$ frame. Combining (13), (15) and the property $S(\boldsymbol{a})\boldsymbol{b} = -S(\boldsymbol{b})\boldsymbol{a}$ yields:

$$\dot{\boldsymbol{p}}^{\mathfrak{I}} = -R^{\mathfrak{I}}_{\mathfrak{B}} S(\boldsymbol{p}^{\mathfrak{B}}) \boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}. \quad (16)$$

Consider now the kinetic energy of the infinitesimal element in Figure 5:

$$\mathrm{d}\mathscr{K} = \frac{1}{2} \left(\left(\dot{\boldsymbol{p}}^{\mathfrak{I}}\right)^T \dot{\boldsymbol{p}}^{\mathfrak{I}} \right) \mathrm{d}m. \quad (17)$$

Substituting (16) into (17) and using the property $(R^{\mathfrak{I}}_{\mathfrak{B}})^T R^{\mathfrak{I}}_{\mathfrak{B}} = I$, we get:

$$\mathrm{d}\mathscr{K} = \frac{1}{2} \left[\left(\boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}\right)^T S^T(\boldsymbol{p}^{\mathfrak{B}}) S(\boldsymbol{p}^{\mathfrak{B}}) \boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}} \right] \mathrm{d}m. \quad (18)$$

To obtain the kinetic energy of the entire rocket, we simply integrate over the mass of the rocket. Since $\mathrm{d}m = \rho\,\mathrm{d}V$, we get:

$$\mathscr{K} = \frac{1}{2} \left(\boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}\right)^T \left[\int_V \rho S^T(\boldsymbol{p}^{\mathfrak{B}}) S(\boldsymbol{p}^{\mathfrak{B}})\,\mathrm{d}V \right] \boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}. \quad (19)$$

We identify the integral in (19) as the inertia tensor J of the rocket, expressed in the $\mathfrak{B}$ frame. This integral is not trivial given the multitude of elements inside a rocket – electronics, parachutes, coupling rings, engine casing, etc. However, J is directly obtained from any CAD package (in this case, Solidworks®) if each part of the CAD model is assigned the correct mass properties.

Note that a mapping exists from the Euler angle time derivatives to the angular velocity of the rocket [16]:

$$\omega^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}} = W_\eta(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad (20)$$

$$W_\eta(\boldsymbol{q}) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\varphi) & \sin(\varphi)\cos(\theta) \\ 0 & -\sin(\varphi) & \cos(\varphi)\cos(\theta) \end{bmatrix}. \quad (21)$$

Substituting (20) into (19) and replacing the integral with the inertia tensor J:

$$\mathscr{K} = \frac{1}{2} \dot{\boldsymbol{q}}^T W_\eta^T(\boldsymbol{q}) J W_\eta(\boldsymbol{q}) \dot{\boldsymbol{q}}. \quad (22)$$

Based on (6) and (22), the inertia matrix $M(\boldsymbol{q})$ can finally be identified:

$$M(\boldsymbol{q}) = W_\eta^T(\boldsymbol{q}) J W_\eta(\boldsymbol{q}). \quad (23)$$

3.1.2. Finding the $\boldsymbol{G}(\boldsymbol{q})$ vector

The discussion for (8) yields:

$$\boldsymbol{G}(\boldsymbol{q}) = \frac{\partial \mathscr{U}(\boldsymbol{q})}{\partial \boldsymbol{q}}. \quad (24)$$

For a rigid body rocket:

$$\mathscr{U}(\boldsymbol{q}) = Mg \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix}, \quad (25)$$

where M is the rocket mass, $g = 9.81\ \left[\frac{\mathrm{m}}{\mathrm{s}^2}\right]$ is the gravitational acceleration and z is the distance along the $\hat{z}_{\mathfrak{I}}$ axis *from* the $\mathfrak{I}$-frame origin *to* the $\mathfrak{B}$-frame origin, taken as

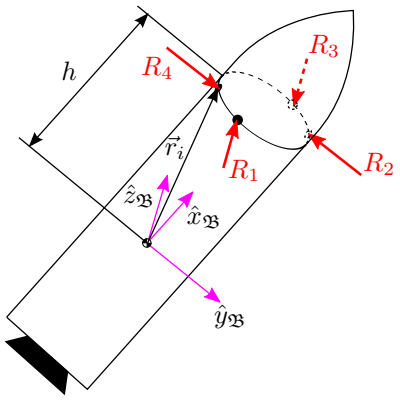

Figure 6. Free-body diagram for the RCS.

positive in that direction. However, we only consider *attitude*, so from (25) we conclude that $\mathscr{U}(\boldsymbol{q})$ is independent of the generalized coordinates $\boldsymbol{q}$. Therefore:

$$\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{0}. \tag{26}$$

3.1.3. Finding the generalized torque vector $\mathscr{F}(\boldsymbol{q})$

$\mathscr{F}(\boldsymbol{q})$ is composed of control torques, $\boldsymbol{T}^*(\boldsymbol{q})$, from the RCS and aerodynamic torques, $\boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}})$, from the aerodynamic moments acting about the rocket's center of pressure:

$$\mathscr{F}(\boldsymbol{q}) = \boldsymbol{T}^*(\boldsymbol{q}) + \boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{27}$$

Finding $\boldsymbol{T}^*(\boldsymbol{q})$. The most general control that can be applied on the rocket is a torque vector $\boldsymbol{\tau}^{\mathfrak{B}} = [\tau_x^{\mathfrak{B}} \ \tau_y^{\mathfrak{B}} \ \tau_z^{\mathfrak{B}}]^T$, expressed in the $\mathfrak{B}$-frame and positive according to the right-hand rule. To find $\boldsymbol{T}^*(\boldsymbol{q})$, we apply the principle of virtual work. Because we consider the rocket as a *single* rigid body, the virtual work principle degenerates to:

$$\delta W = \left(\boldsymbol{\tau}^{\mathfrak{B}}\right)^T \delta \boldsymbol{r} = \left(\boldsymbol{T}^*\right)^T \delta \boldsymbol{q}, \tag{28}$$

where $\delta \boldsymbol{r}$ is an infinitesimal change in the position vector of the body (expressed in the same frame as $\boldsymbol{\tau}^{\mathfrak{B}}$, i.e. the $\mathfrak{B}$ frame) and $\delta \boldsymbol{q}$ is an infinitesimal change in the generalized coordinates. To find $\delta \boldsymbol{r}$ we "integrate" (20). Note that because $\boldsymbol{\omega}^{\mathfrak{B}}_{\mathfrak{B}\mathfrak{I}}$ is not integrable [17], this approach only works thanks to the infinitesimal change assumption. Therefore:

$$\delta \boldsymbol{r} = W_\eta(\boldsymbol{q}) \delta \boldsymbol{q}. \tag{29}$$

By substituting (29) into (28), the generalized control torques are expressed:

$$\boldsymbol{T}^*(\boldsymbol{q}) = W_\eta^T(\boldsymbol{q}) \boldsymbol{\tau}^{\mathfrak{B}}. \tag{30}$$

However, $\boldsymbol{\tau}^{\mathfrak{B}}$ can be related to the thrust forces generated by the four RCS nozzles of the rocket. For this, consider Figures 3 and 6. Noting that $f \ll r$, calling $\vec{r}_i$ the position vector of the i-th nozzle and keeping in mind that thrusts (R_1,R_3) and (R_2,R_4) are parallel to the $\pm\hat{z}_{\mathfrak{B}}$ and $\mp\hat{y}_{\mathfrak{B}}$ axes respectively, we find the relation between RCS thrusts and the control torque vector:

$$\boldsymbol{\tau}^{\mathfrak{B}} = \begin{bmatrix} -f & f & -f & f \\ -h & 0 & h & 0 \\ 0 & -h & 0 & h \end{bmatrix} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix} := B\boldsymbol{R}, \tag{31}$$

where we call B the "mapping" from RCS thrusts to control torques. In summary, substituting (31) into (30):

$$\boldsymbol{T}^*(\boldsymbol{q}) = W_\eta^T(\boldsymbol{q}) B \boldsymbol{R}. \tag{32}$$

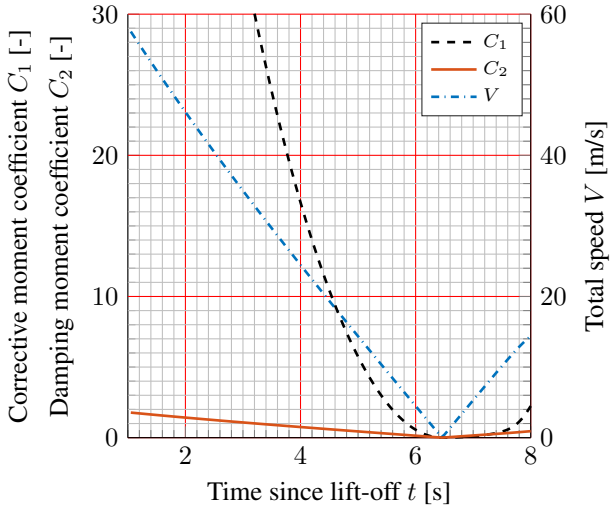

Figure 7. Evolution of C_1 and C_2 based on OpenRocket® simulation data.

Finding $\boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}})$. We only consider the corrective and damping moment coefficients for the aerodynamic torques acting on the rocket, denoted as C_1 and C_2 respectively. These are given by [18]:

$$C_1 = \frac{1}{2}\rho A_{ref} V^2 C_{N\alpha}(Z - W), \tag{33}$$

$$C_2 = \frac{1}{2}\rho A_{ref} V \sum_{i=1}^{n_c} \left\{C_{N\alpha,i}(Z_i - W)^2\right\}, \tag{34}$$

where $A_{ref} = \pi r^2$ is the nose cone base area, V is the speed, $C_{N\alpha}$ is the total normal force coefficient derivative, $C_{N\alpha,i}$ is the i-th structural component's normal force coefficient derivative, Z is the distance from the nose to the rocket center of pressure, Z_i is the distance from the nose to the i-th structural component's center of pressure, W is the distance from the nose to the center of mass, α is the angle of attack (in radians) and n_c is the total number of components of the rocket (FALCO-4 has one nose cone, four body tubes and one set of four fins).

The rocket was modeled in OpenRocket®, whose technical documentation [19] [20] provides a method to evaluate (33) and (34). Thus, a plot of C_1 and C_2 for FALCO-4 is produced in Figure 7.

For test flights, the RCS control is activated from the 4th second of flight onwards in order to conserve CO_2

pressure for the low speed flight near apogee. Therefore, based on Figure 7, the control must be able to deal with the following range of aerodynamic coefficients:

$$\begin{cases} C_1 \in [0, 17], \\ C_2 \in [0, 0.75]. \end{cases} \tag{35}$$

Since C_1 and C_2 depend on speed, the attitude-only dynamical model of the rocket has no state available to compute these coefficients. Therefore, the corrective and damping moments are considered as disturbances that the RCS must be able to counteract in order to keep the rocket vertical. We assume that the corrective and damping moments act through the yaw (ψ) and pitch (θ) Euler angles. In reality, these moments would act about the axis $((R^{\mathfrak{I}}_{\mathfrak{B}}\hat{x}_{\mathfrak{B}}) \times \hat{x}_{\mathfrak{I}})$, but unfortunately the orientation of this axis degenerates when the rocket is perfectly vertical, i.e. when $R^{\mathfrak{I}}_{\mathfrak{B}}\hat{x}_{\mathfrak{B}}$ is parallel to $\hat{x}_{\mathfrak{I}}$. Note that zero aerodynamic roll moment is assumed. Therefore, the aerodynamic moments are expressed as:

$$\begin{cases} A^*_{\varphi} = 0, \\ A^*_{\theta} = -C_1(\theta - \theta_o) - C_2(\dot{\theta} - \dot{\theta}_o), \\ A^*_{\psi} = -C_1(\psi - \psi_o) - C_2(\dot{\psi} - \dot{\psi}_o), \end{cases} \tag{36}$$

where $\{\theta_o, \psi_o\}$ accounts for the "pitch" and "yaw" of the local wind vector in case of side wind. Note that the rocket passively stabilizes about the local wind vector, not $\hat{x}_{\mathfrak{I}}$. The local wind vector has no "roll", so $\varphi_o = 0$. In matrix notation, (36) is written as:

$$\boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}) = A_1(\boldsymbol{q} - \boldsymbol{q}_o) + A_2(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_o), \tag{37}$$

with:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -C_1 & 0 \\ 0 & 0 & -C_1 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -C_2 & 0 \\ 0 & 0 & -C_2 \end{bmatrix}. \tag{38}$$

3.1.4. The dynamical equations

In summary, the following dynamical equations for the rocket attitude were obtained:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{T}^*(\boldsymbol{q}) + \boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}),$$

with

$$\begin{cases} M(\boldsymbol{q}) = W_{\eta}^T(\boldsymbol{q}) J W_{\eta}(\boldsymbol{q}), \\ \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{0}, \\ \boldsymbol{T}^*(\boldsymbol{q}) = W_{\eta}^T(\boldsymbol{q}) B \boldsymbol{R}, \\ \boldsymbol{A}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}) = A_1(\boldsymbol{q} - \boldsymbol{q}_o) + A_2(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_o), \end{cases} \tag{39}$$

where $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is given by (9), $W_{\eta}(\boldsymbol{q})$ is given by (21), $J \in \mathbb{R}^3$ is the rocket inertia tensor in the $\mathfrak{B}$-frame, $\boldsymbol{0} \in \mathbb{R}^3$ is a zero vector, B and $\boldsymbol{R}$ are given by (31) and finally A_1 and A_2 are given by (38).

3.2. Open-loop simulation

Using (39) we can develop a non-linear Simulink® model (Figure 8) that is used for simulation. For control system design, this plant is linearized about the rocket's vertical orientation (i.e. $\boldsymbol{q} = \boldsymbol{0}$). Using this plant we can simulate the attitude of a rocket when it is flying quickly (Figure 9) or slowly (Figure 10) with zero side wind in both cases. A side wind can also be simulated.

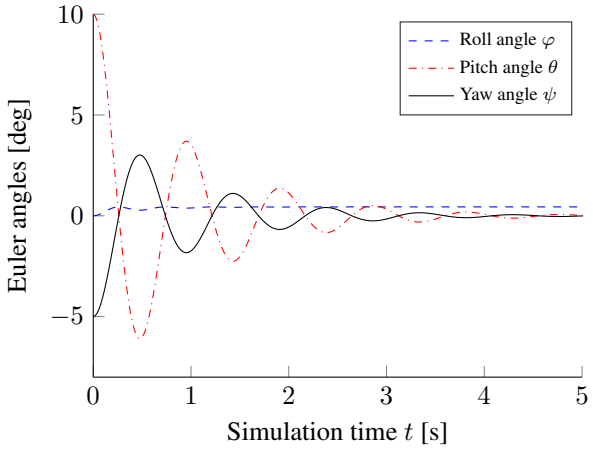

Figure 9. Euler angles evolution for a fast-moving rocket with $C_1 = 16$ and $C_2 = 0.75$ (i.e. 4 seconds into the flight moving at ≈ 24 [m/s]).

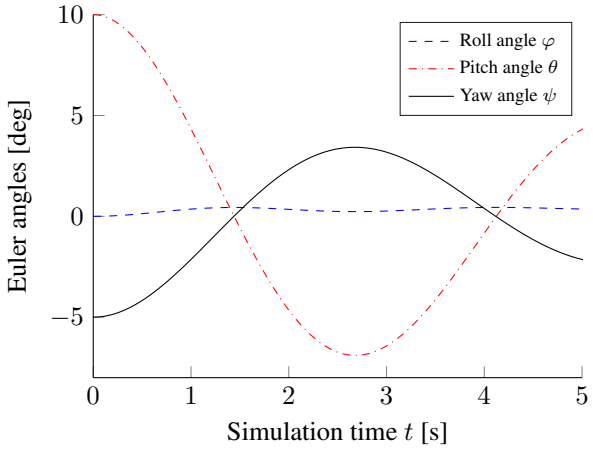

Figure 10. Euler angles evolution for a fast-moving rocket with $C_1 = 0.5$ and $C_2 = 0.1$ (i.e. 6 seconds into the flight moving at ≈ 4 [m/s]).

In our application a restoring aerodynamic moment is only beneficial in the absence of side wind. When side wind is present, the local velocity vector would be off the geometric vertical (axis $\hat{x}_{\mathfrak{I}}$) and the control would have to work against the aerodynamic moments. As the absence of side wind cannot be guaranteed, a neutrally stable rocket ($C_1 = C_2 = 0$) would be optimal for our application. However, for safety and due to the presence of body lift and other aerodynamic effects, a certain passive stability was built into the FALCO-4 by the four tail fins discussed in Section 2.1. However, at low speeds (e.g around apogee) aerodynamic effects become smaller (see Figure 7), for which reason the control was designed for the case of small aerodynamic coefficients $C_1 = 0.5$ and $C_2 = 0.1$ (see Figure 10).

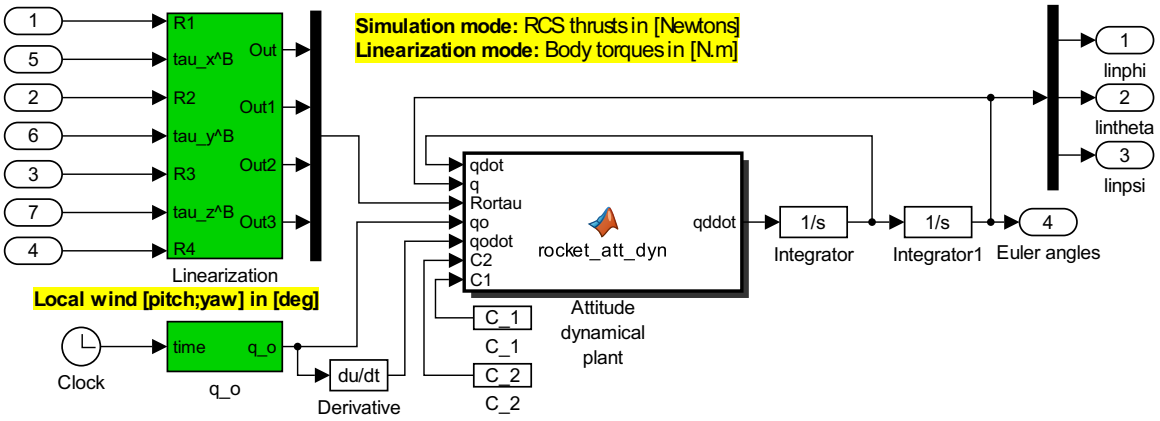

Figure 8. Simulink® block diagram of the rocket plant.

Now that the non-linear plant has been created, it is possible to develop and test a control scheme for vertically stabilizing the rocket.

3.3. Control scheme

3.3.1. Plant linearization

Physically, the system inputs are the four RCS thrusts. However, it is easier to control in three torques $[\tau_x^{\mathfrak{B}}\ \tau_y^{\mathfrak{B}}\ \tau_z^{\mathfrak{B}}]^T$ than in the four thrusts $[R_1\ R_2\ R_3\ R_4]^T$. This is because $W_\eta^T(\boldsymbol{q})$ in (30) is simpler than $W_\eta^T(\boldsymbol{q})B$ in (32). Figure 8 handles this via the green `Linearization` block which, using switches, selects four RCS thrusts $\{R_1, R_2, R_3, R_4\}$ for simulation or three torques $\{\tau_x^{\mathfrak{B}}, \tau_y^{\mathfrak{B}}, \tau_z^{\mathfrak{B}}\}$ for linearization.

Using MATLAB®'s `linmod` to linearize the Simulink® plant model for Figure 10's conditions, we obtain:

$$\begin{cases} \ddot{\varphi} = 337.8378\tau_x^{\mathfrak{B}}, \\ \ddot{\theta} = -1.3901\theta - 0.2780\dot{\theta} + 2.7802\tau_y^{\mathfrak{B}}, \\ \ddot{\psi} = -1.3901\psi - 0.2780\dot{\psi} + 2.7802\tau_z^{\mathfrak{B}}. \end{cases} \tag{40}$$

The coefficients above result from FALCO-4's geometric and dynamic properties, such as its inertia tensor values. Converting (40) into state space form, it can be shown that each axis (roll, pitch and yaw) is controllable using RCS torques $[\tau_x^{\mathfrak{B}}\ \tau_y^{\mathfrak{B}}\ \tau_z^{\mathfrak{B}}]^T$ as inputs.

Notice that each equation in (40) is *decoupled*. Therefore, decoupled PID controllers for roll, pitch and yaw can be designed (at least in the linear case). Since the pitch and yaw dynamics of (40) are identical, they are assigned identical PID controllers.

3.3.2. Pitch control design

Figure 11 shows the implemented feedback control scheme.

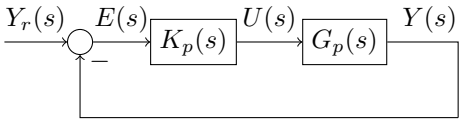

Figure 11. Yaw control block diagram.

From (40), the pitch transfer function is:

$$G_p(s) = \frac{2.7802}{s^2 + 0.2780s + 1.3901}. \tag{41}$$

A PID controller of the following form is sought:

$$K(s) = K\big(1 + \frac{1}{T_i s} + T_d s\big). \tag{42}$$

In order to obtain the coefficients in (42), Ziegler-Nichols tuning rules are applied to $G_p(s)$:

$$\begin{cases} K &= 1.4197, \\ T_i &= 0.8565, \\ T_d &= 0.2141. \end{cases} \tag{43}$$

However, subsequent hardware-in-the-loop testing revealed that the integral term caused persistent saturation due to sensor noise, fast nature of rocket dynamics and the limited thrust of the final RCS. As a result, the integrator term was removed.

Manual tuning of the remaining proportional and derivative terms gives the final pitch controller architecture:

$$\begin{cases} K &= 5, \\ T_d &= 3, \\ K_p(s) &= K + T_d s. \end{cases} \tag{44}$$

A Bode plot of the loop transfer function $K_p(s) \cdot G_p(s)$ is shown in Figure 12. K was tuned by hardware-in-the-loop testing to avoid excessive saturation in the control signal. T_d was chosen to place a zero after the resonance frequency in Figure 12 so as to prevent the -180 degree drop in phase, therefore stabilizing the system.

The lack of an integrator means that a steady-state offset is present in the response. However, for a 10 degree disturbance, this offset is only 2.6 degrees in simulation. This is within our tolerated precision. For this reason,

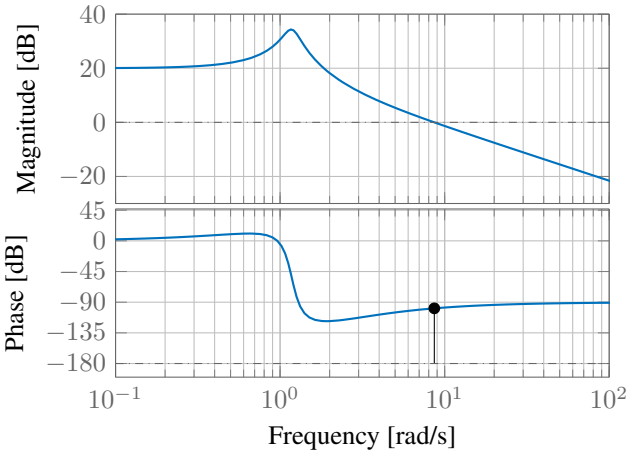

Figure 12. Bode plot of $K_p(s) \cdot G_p(s)$.

the PD controller was chosen as a satisfactory alternative to the aforementioned PID. It was also chosen over a loop-shaping approach for implementation simplicity. Finally, as can be seen from Figure 12, the controller offers infinite gain margin and 81 degrees of phase margin. Such robustness is important as we cannot measure the exact final parameters of the rocket to a great precision. Also, the coefficients C_1 and C_2, and thus the dynamical model, are bound to change during flight.

For final implementation, (44) is discretized using backwards Euler difference (i.e. substituting $s = \frac{z-1}{zT_s}$):

$$K_p(z) = K + T_d \frac{1 - z^{-1}}{T_s}, \tag{45}$$

where T_s is the sampling time. The coefficients K and T_d therefore remain the same. Note, that θ and $\dot{\theta}$ were Kalman filtered prior to being passed into the controller.

3.3.3. Roll control design

The scheme from Figure 11 is also used for roll control. However, unlike pitch and yaw where we desire to control *angles*, we only wish for the rocket to have zero roll *rate* – we do not care what "roll angle" the symmetric rocket has. Therefore the roll equation of (40) can be written as a first-order relation in roll rate:

$$\dot{\omega}_\varphi = 337.8378 \tau_x^{\mathfrak{B}}. \tag{46}$$

Taking $\tau_x^{\mathfrak{B}}$ as input, the transfer function for roll is:

$$G_r(s) = \frac{337.8378}{s}. \tag{47}$$

Since (47) contains an integrator, an interator in the controller is not needed for zero steady-state offset. As such, a proportional controller is deemed to be sufficient for roll control. The proportionality constant is chosen according to the equation:

$$K = \frac{\max(\tau_x^{\mathfrak{B}})}{\max(\omega_\varphi)}, \tag{48}$$

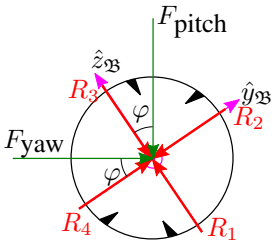

Figure 13. Roll-mapping of the RCS thrusts.

where $\max(\tau_x^{\mathfrak{B}}) = 2fR_{\max}$ and $\max(\omega_\varphi)$ is the roll rate deemed to be so large as to make us want to saturate the RCS in trying to reduce it. This situation may arise after, for example, an accident during launch that induces a severe spin in flight. We chose $\max(\omega_\varphi) = 100$ [deg/s]. Lastly, $R_{\max} = 0.36$ [N] is the maximum thrust that an individual RCS nozzle can produce. Therefore the final (both continuous and discrete time) roll controller is:

$$K_r(z) = 0.002063. \tag{49}$$

This gives an infinite gain margin and a 90 degree phase margin – which, again, offers good stability.

3.3.4. Optimal thrust allocator

To develop the controllers, we replaced the RCS thrusts with net torques about the rocket's center of mass. Now, a combination of $\{R_1, R_2, R_3, R_4\}$ to produce these net torques must be found. As an infinite number of such combinations exist – and as this allocation problem is linear – linear programming is used to find the *optimal* $\{R_1, R_2, R_3, R_4\}$ combination. On-board, this is implemented as the Simplex algorithm [21].

The idea is to use a "mapping" which eliminates roll from the pitch and yaw stabilization: as the rocket rolls, we want to re-distribute the thrusts amongst the four RCS nozzles in order to continue yaw and pitch stabilization in the same direction. Using Figure 13, we can write:

$$\begin{cases} F_{\text{yaw}} = -R_1 S_\varphi - R_2 C_\varphi + R_3 S_\varphi + R_4 C_\varphi, \\ F_{\text{pitch}} = -R_1 C_\varphi + R_2 S_\varphi + R_2 C_\varphi - R_4 S_\varphi, \\ \tau_x^{\mathfrak{B}} = -R_1 f + R_2 f - R_3 f + R_4 f, \end{cases} \tag{50}$$

where:

$$\begin{cases} F_{\text{yaw}} = \tau_z^{\mathfrak{B}}/h \\ F_{\text{pitch}} = \tau_y^{\mathfrak{B}}/h \\ S_\varphi = \sin(\varphi) \quad \text{and} \quad C_\varphi = \cos(\varphi) \end{cases} \tag{51}$$

The optimal thrust allocation problem is then:

$$\begin{aligned} &\min\left(R_1 + R_2 + R_2 + R_4\right) \\ &\text{subject to} \\ &\begin{cases} -R_1 S_\varphi - R_2 C_\varphi + R_3 S_\varphi + R_4 C_\varphi = \tau_z^{\mathfrak{B}}/h, \\ -R_1 C_\varphi + R_2 S_\varphi + R_2 C_\varphi - R_4 S_\varphi = \tau_y^{\mathfrak{B}}/h, \\ -R_1 f + R_2 f - R_3 f + R_4 f = \tau_x^{\mathfrak{B}}, \\ R_1, R_2, R_3, R_4 \geq 0. \end{cases} \end{aligned} \tag{52}$$

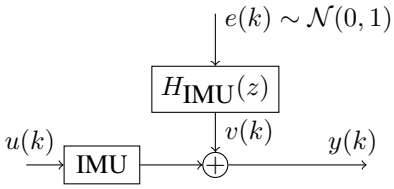

Figure 14. Additive noise model for the IMU.

On-board, (52) is solved using the Simplex algorithm [21] at a frequency of 50 [Hz].

3.3.5. Sensor noise model

An important part of control system design is modeling the feedback dynamics – i.e. the sensors. Here the IMU dynamics are ignored (assume it is a unit gain), however we do model sensor noise according to Figure 14.

To identify $H_{\text{IMU}}(z)$, the IMU was placed on a steady surface and the output angle signals were measured for one minute. A Discrete Fourier Transform of the output signal was then taken in MATLAB®. Assuming white Gaussian noise $e(k) \sim \mathcal{N}(0,1)$ as the input, the following transfer function was obtained by fitting and then discretizing a second-order model to fit the blue noisy empirical transfer function in the top plot of Figure 15:

$$H_{\text{IMU}}(s) = K_{IMU} \cdot \frac{\omega_1^2}{s^2 + 2\omega_1\zeta s + \omega_1^2} \cdot \left(1 + \frac{s}{\xi}\right). \quad (53)$$

Figure 15 shows the visual match between our measurements and the identified noise filter $H_{\text{IMU}}(z)$. Finally, a first-order extended Kalman filter was used to filter $y(k)$ before passing it to the pitch, yaw and roll controllers (and can be consulted in the GNC source code [14]).

3.4. Closed-loop simulation

Using (45) as the pitch and yaw controller, (49) as the roll controller and (39) as the dynamical plant, Figure 18 shows the master view of the Simulink® diagram used for simulation. Figures 16 and 17 show results for a simulation where the rocket is nearing apogee ($C_1 = 0.5$, $C_2 = 0.1$) with the initial attitude $\{\varphi_0, \dot{\varphi}_0, \theta_0, \dot{\theta}_0, \psi, \dot{\psi}_0\} = \{0, 30, -7, 0, 7, 0\} \cdot \pi/180$.

Figure 16 shows the evolution of rocket attitude after RCS activation at $t = 1$ [s]. Note that prior to $t = 1$ [s], the Kalman filter converges to the true rocket attitude. Noise is effectively being filtered and the rocket is maintained reasonably vertical by the RCS, however wind disturbances after about 4 [s] have little hope of being corrected. This is because real tests of the RCS show the compressed CO_2 container to lose pressure rather quickly. This phenomenon was taken into account as an exponential decrease in RCS thrust saturation level after 1 second of allowed full-thrust operation – see top of Figure 17. This simple model allows to simulate the effect

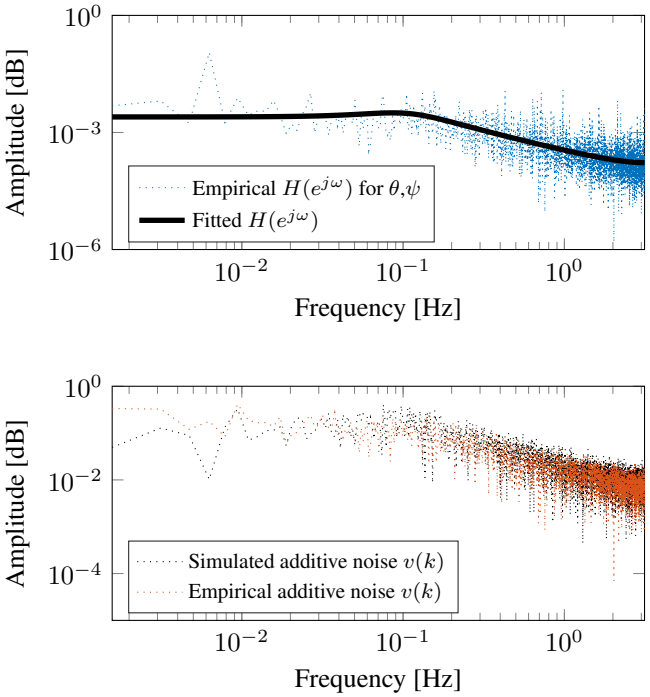

Figure 15. Identification of the IMU noise filter.

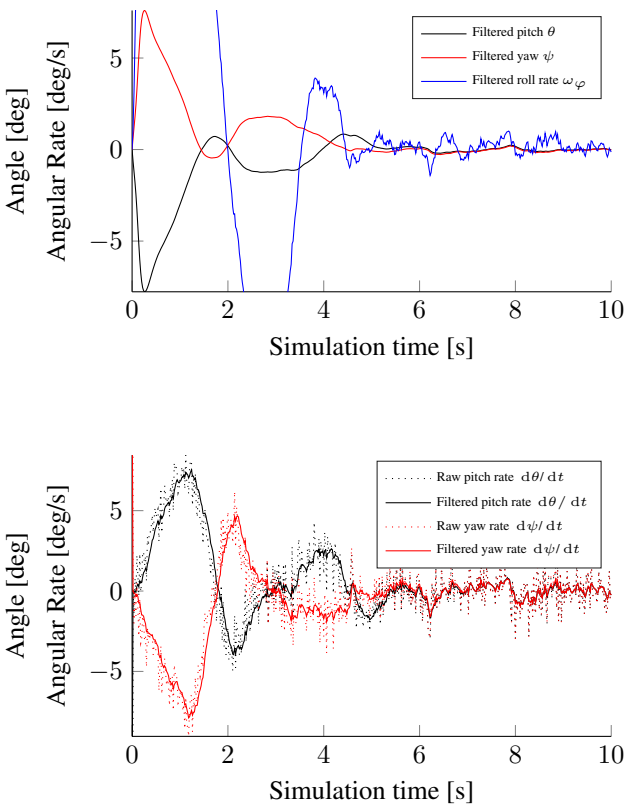

Figure 16. Closed-loop simulation angle and angular rate evolution.

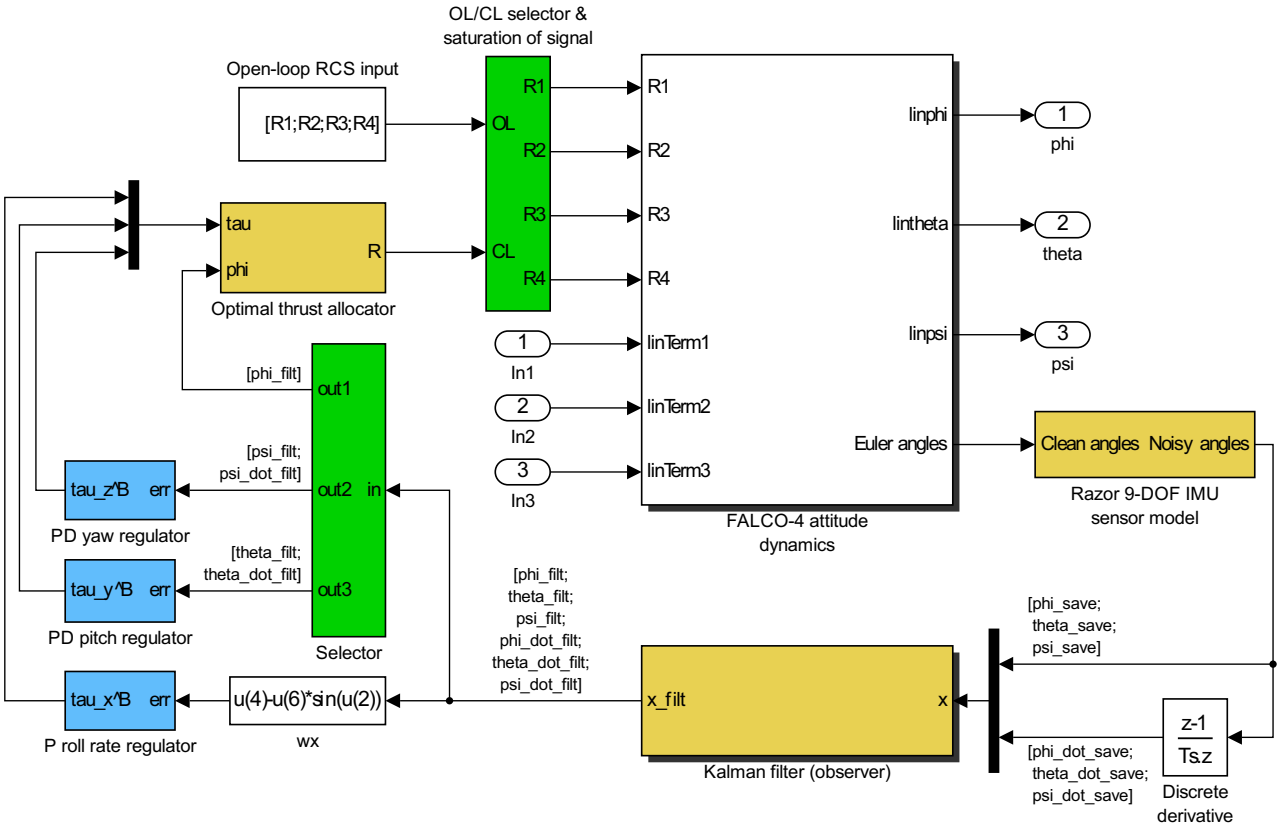

Figure 18. Attitude control system overview. Green – selectors; orange – computations; blue – controllers.

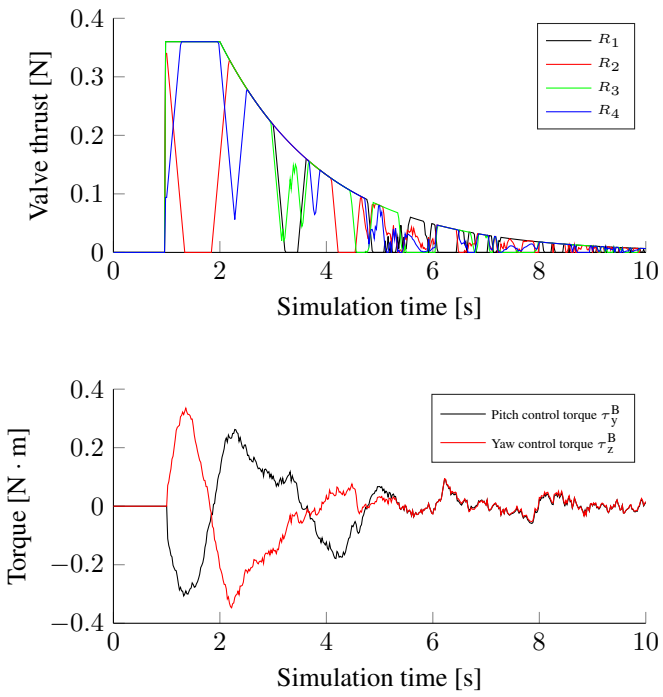

Figure 17. Closed-loop simulation of control torques and RCS thrusts mapped by (52).

of a decreasing thrust level and predicts that after a short time the RCS is only be able to resist (rather than completely correct) rocket deviation from the vertical. The next section describes how test flight data confirms these simulation results.

4. EXPERIMENTAL RESULTS

On June 6, 2015, five test flights were conducted with the objective of recording the difference in the rocket's deviation from the vertical between the controlled and uncontrolled configurations. Initially, three controlled and three uncontrolled flights were planned. However, a limited CO_2 cartridge supply made only one controlled launch possible due to an extra ground test using one cartridge and a valve malfunction forcing the first flight to be uncontrolled, during which a hard landing ruptured the RCS pressure tubing and thus emptied a second cartridge. Nevertheless, four uncontrolled flights were made – of which the last could not be used for data analysis due to corrupted accelerometer data at lift-off.

Figure 19 summarizes the main result of flight data analysis. It shows the angles between the $\hat{x}_{\mathfrak{B}}$ and $\hat{x}_{\mathfrak{I}}$ axes – i.e. the angle by which the rocket attitude deviates from the vertical. For the uncontrolled flights, the graphs are very similar with about 90 degree deviation from the ver-

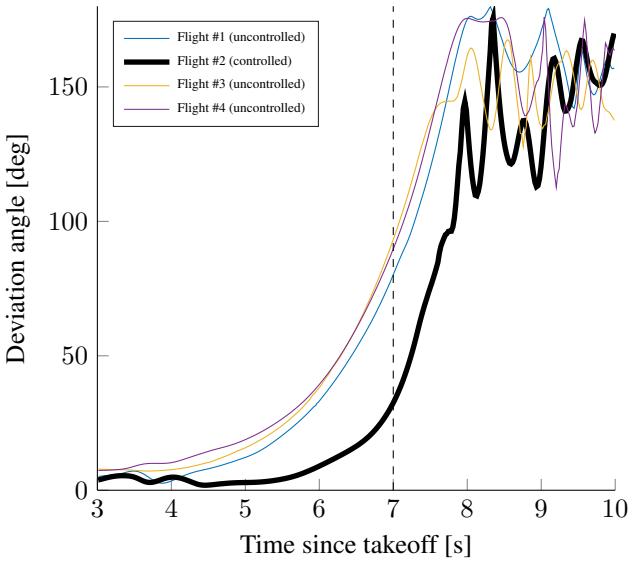

Figure 19. Deviation angle between rocket longitudinal axis and the vertical (test flight data).

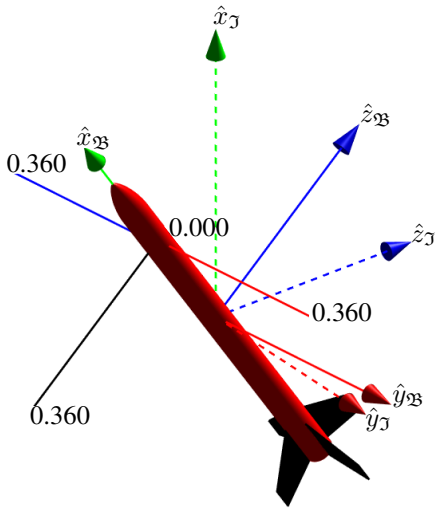

Figure 21. Attitude and RCS thrusts at $t = 7.035$ [s] (flight #2 data).

tical at the approximate moment of apogee (shown by the dashed vertical line in Figure 19). Meanwhile, the controlled flight's deviation angle is almost 3 times smaller – i.e. ≈ 30 degrees at apogee. Note that parachutes open at $t \approx 8$ [s], so the deviation angle thereafter is meaningless. These results are visually confirmed by on-board video recordings of each flight.

It is worth noting that the RCS was turned on 4 seconds into the flight. Correspondingly, in Figure 19 there is a drop in deviation angle at 4 seconds with almost no deviation up to about 5 seconds, while the deviation angle for all three uncontrolled flights steadily increases. This suggests that the RCS was doing its job very well while the CO_2 tank pressure was high and thus valve thrust was sufficient. However, after the first second of operation the dropping pressure in the CO_2 cartridge rendered the RCS unable to fully cancel the flight moments. As a result, the RCS could only *resist* rather than *remove* vertical deviation for the remainder of the flight up to apogee – which is what Figure 19 shows.

Figure 20 shows the commanded RCS thrusts by the onboard computer for the controlled flight. Note that Figure 20 shows the *expected* RCS thrusts. These were not actually measured in flight. The valves were calibrated in open-loop on the ground, so a value in the graph simply indicates the value that the controller assigned to that valve and not the one that was actually output. As previously noted, after ≈ 5 [s] CO_2 pressure began droppping and the open-loop PWM-to-thrust mappings no longer held.

Figure 21 allows to visualize Figure 20 at $t = 7.035$ [s] (the same color scheme for RCS thrusts is used). As the rocket tipped at apogee, the RCS saturated trying to push it back to the vertical orientation. This is the "resistance to deviation" effect discussed for Figure 19.

5. DISCUSSION AND CONCLUSION

The experimental results closely match what was predicted through simulation. The initial goal of maintaining a vertical alignment via cold gas thrusters was partially achieved. The control works as expected, the only shortcoming being a drop in RCS gas pressure. Nevertheless, the results show that vertical attitude can be maintained throughout the controlled flight if sufficient thrust is achieved.

The following is a non-exhaustive list of improvements to make in future vehicles of this type:

1. Close the loop on valve thrusts (i.e. measure thrust levels). This will allow to implement cascade control and better interpret flight data.
2. Account for CO_2 pressure drop – e.g. by increasing CO_2 cartridge volume.
3. Change the IMU to give angles at a higher frequency – the 50 [Hz] limitation of [22] may be a limiting factor for future development.

In conclusion, this paper and the flight software [14] should help individuals interested in learning and helping advance rocket vertical landing efforts.

ACKNOWLEDGMENTS

This work was kindly sponsored by the Space Engineering Center at the École Polytechnique Fédérale de Lausanne.

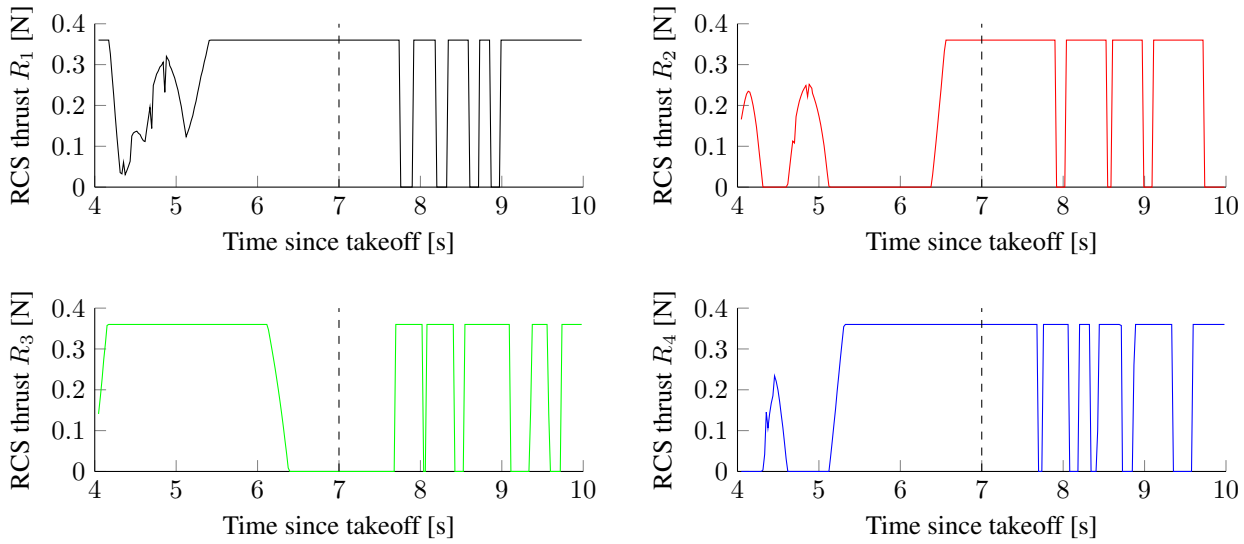

Figure 20. Commanded RCS thrusts (flight #2 data).

REFERENCES

[1] SpaceX. Reusability: The Key to Making Human Life Multi-Planetary. http://www.spacex.com/news/2013/03/31/reusability-key-making-human-life-multi-planetary, 2015.

[2] Spaceflight Now. ULA chief explains reusability and innovation of new rocket. http://spaceflightnow.com/2015/04/14/ula-chief-explains-reusability-and-innovation-of-new-rocket/, 2015.

[3] Space.com. Airbus' Adeline Project Aims to Build Reusable Rockets and Space Tugs. http://www.space.com/29620-airbus-adeline-reusable-rocket-space-tug.html, 2015.

[4] Adafruit. Raspberry Pi Model B+ 512MB RAM. http://www.adafruit.com/products/1914, 2015.

[5] Adafruit. Spy Camera for Raspberry Pi. https://www.adafruit.com/products/1937, 2015.

[6] Adafruit. PowerBoost 500 Basic - 5V USB Boost at 500mA from 1.8V+. https://www.adafruit.com/products/1903, 2015.

[7] Parker. VSO Miniature Proportional Valve. http://ph.parker.com/us/12051/en/vso-miniature-proportional-valve, 2015.

[8] SMC Pneumatics. http://www.smcpneumatics.com/.

[9] Leland Gas Technologies. Fixed Pressure Regulator, NR-10. http://www.lelandltd.com/regulating_valves_prd.htm, 2015.

[10] Leland Gas Technologies. 68 gram 1/2 threaded CO2. http://www.lelandltd.com/small_high_pressure.htm, 2015.

[11] HobbyKing. HD Wing Camera 1280x720p 30fps 5MP CMOS. http://hobbyking.com/hobbyking/store/__17200__HD_Wing_Camera_1280x720p_30fps_5MP_CMOS.html, 2015.

[12] Honeywell. HSC D LN N 100MD S A 5 temperature-compensanted pressure sensor. http://sensing.honeywell.com/honeywell-sensing-trustability-hsc-series-high-accuracy-board-mount-pressure-sensors-50099148-a-en.pdf, 2015.

[13] HobbyKing. Turnigy nano-tech 300mah 2S 35 70C Lipo Pack. http://hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=36220, 2015.

[14] Danylo Malyuta. Falco-4 open source code database. https://github.com/DanyloMalyuta/FlyARocket_GNC.

[15] Guilherme Vianna Raffo. *Robust Control Strategies for a Quadrotor Helicopter*. Doctoral thesis, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, 2011.

[16] Warren F. Phillips. *Mechanics of Flight*. Wiley, second edition, 2010.

[17] Herman Bruyninckx. Robot kinematics and dynamics. http://people.mech.kuleuven.be/~bruyninc/tmp/HermanBruyninckx-robotics.pdf, 2010.

[18] George J. Caporaso Mandell, Gordon K. and William P. Bengen. *Topics in Advanced Model Rocketry*. MIT, 1973.

[19] Sampo Niskansen. *OpenRocket technical documentation For OpenRocket version 13.05*, 2015.

[20] The Sky Dart Team. Open Rocket: Dynamic Stability Characteristics. https://sites.google.com/site/theskydartteam/openrocket\#TOC-Corrective-Moment-Coefficient, 2015.

[21] Jean-Pierre Moreau. Linear Programming in C/C++. http://jean-pierre.moreau.pagesperso-orange.fr/c_linear.html, 2014.

[22] Sparkfun. 9 Degrees of Freedom – Razor IMU. https://www.sparkfun.com/products/10736, 2015.