
# Università degli Studi di Padova

Dipartimento di Ingegneria Industriale DII

*Corso di Laurea Magistrale in Ingegneria Aerospaziale*

# HIGH ACCURACY POINTING FOR STRATOSPHERIC BALLOON-BORNE TELESCOPES: SIMULATING THE POINTING ARCHITECTURE OF THE DICOS MISSION

***Relatore***
Carlo Bettanini Fecia Di Cossato

***Studente***
Alexandru Andrei Avram
1217521

*Anno Accademico 2021-2022*

To my family and to those who have always supported me

# Abstract

Stratospheric balloons have been used for many years as platforms to carry scientific payloads into the upper layers of the atmosphere. Compared to satellites and launchers, they are cost-efficient, reusable, flexible in their deployment and operability, able to carry heavy payloads and impose few mechanical constraints on them. DICOS is a balloon experiment under development at CNES that aims to demonstrate the feasibility of coronagraph imaging to study exoplanets aboard a balloon-borne gondola. Such an ambitious mission requires very accurate pointing performance under the tenth of an arcsecond. To assess the pointing architecture and validate the different transitions of DICOS pointing modes before its expected flight in 2024-2025, a generic simulator was developed in MATLAB/Simulink with a full dynamical model of the flight chain and different models of sensors, actuators, and flight software. This thesis presents the activities performed during the traineeship at the CNES site of Toulouse and the improvements made to the simulator. Several models have been implemented, and their effects on the entire simulator have been analyzed and discussed.

# Sommario

In questa tesi vengono descritte le attività di tirocinio svolte presso l'agenzia spaziale francese (CNES) con sede a Toulouse, in Francia. Il tirocinio si è svolto all'interno del programma Erasmus+ Traineeship e ha avuto una durata di sei mesi durante i quali lo studente ha svolto le sue attività nel dipartimento "Palloni Stratosferici". Esso si occupa del design di missioni scientifiche a bordo di palloni stratosferici, i quali sono capaci di portare la gondola, su cui vengono montati i payload, ad una quota operativa di circa 40 km. L'utilizzo dei palloni stratosferici, come alternativa ai satelliti, offre numerosi vantaggi: i primi infatti sono più economici, riutilizzabili, in grado di trasportare carichi utili pesanti imponendo loro meno vincoli meccanici, e capaci di essere operati più semplicemente.

DICOS è una missione attualmente in corso di sviluppo presso il CNES che mira a dimostrare la fattibilità dello studio di pianeti extrasolari per mezzo di un coronografo. Un obiettivo di missione così ambizioso richiede un puntamento molto accurato, con prestazioni inferiori a un decimo di arcosecondo. Per valutare l'architettura di puntamento e convalidare le diverse transizioni tra le varie modalità di puntamento prima del volo previsto nel 2024-2025, è stato sviluppato un simulatore in MATLAB/Simulink che riproduce il modello dinamico completo della catena di volo, dei sensori, degli attuatori e del software di volo.

Durante il tirocinio sono stati migliorati ed implementati diversi modelli, ed i loro effetti sull'intero simulatore sono stati analizzati e discussi. In particolare, sono stati creati i modelli per i controllori ed i filtri che gestiscono i segnali di input alle ruote di reazione montate sulla gondola, un nuovo modello per le ruote di reazione che tiene conto degli effetti dell'attrito, un filtro che tiene conto dell'angolo di attorcigliamento della catena di volo, un nuovo modello cinematico per correggere le inconsistenze rilevate durante lo studio, ed un modello preliminare per le camere montate a bordo.

# Contents

# Listing of acronyms

**BPS** . . . . . Stratospheric Pressurized Balloon

**BSO** . . . . . Open Stratospheric Balloon

**(CE)** . . . . . Cross-elevation frame

**CNES** . . . . . Centre national d'études spatiales

**CTS** . . . . . Toulouse Space Centre

**DICOS** . . . . . Démonstrateur d'Imageur COronographique Spatial

**DoF** . . . . . Degree of Freedom

**(E)** . . . . . Elevation frame

**FC** . . . . . Flight Chain

**FIREBALL** . . . . . Faint Intergalactic Redshifted Emission Balloon

**FOG** . . . . . Fibre-Optic Gyroscope

**IMU** . . . . . Inertial Measurement Unit

**LoS** . . . . . Line of Sight

**MIR** . . . . . Infrared Hot Air Balloon

**MPA** . . . . . Module Pointage Azimut

**(NA)** . . . . . Nacelle frame

**(RTL)** . . . . . Local Terrestrial Reference frame

**RW** . . . . . Reaction wheel

# 1 Introduction

The Balloons division of CNES is in charge of the gondola and flight chain design and development. The activities described in this thesis were performed during the internship in the before-mentioned division at the CNES site in Toulouse, France.

In the next paragraphs, an overview of the CST and the Balloons division is provided. Moreover, the DICOS mission is presented, focusing on the mission goals and on the MATLAB/Simulink simulator that was development to simulate the different pointing stages and modes transitions.

## 1.1 The Space Centre of Toulouse

The CST was created in 1968 and currently hosts nearly 2500 people, including CNES employees, staff from subsidiary companies and interns. It is a unique centre in Europe in terms of size, diversity and sectors of activity. In fact, during the last few years, the CST has acquired a high level of expertise in designing, developing and operating complex orbital systems and the knowledge necessary for the development of satellites and scientific payloads.

The activities of the CST are divided into two main areas:

- Support the French space industry and research activities in the national, European and international context by mastering the techniques necessary for the development of space systems
- Control and operation of existing systems on behalf of national, European and international institutions

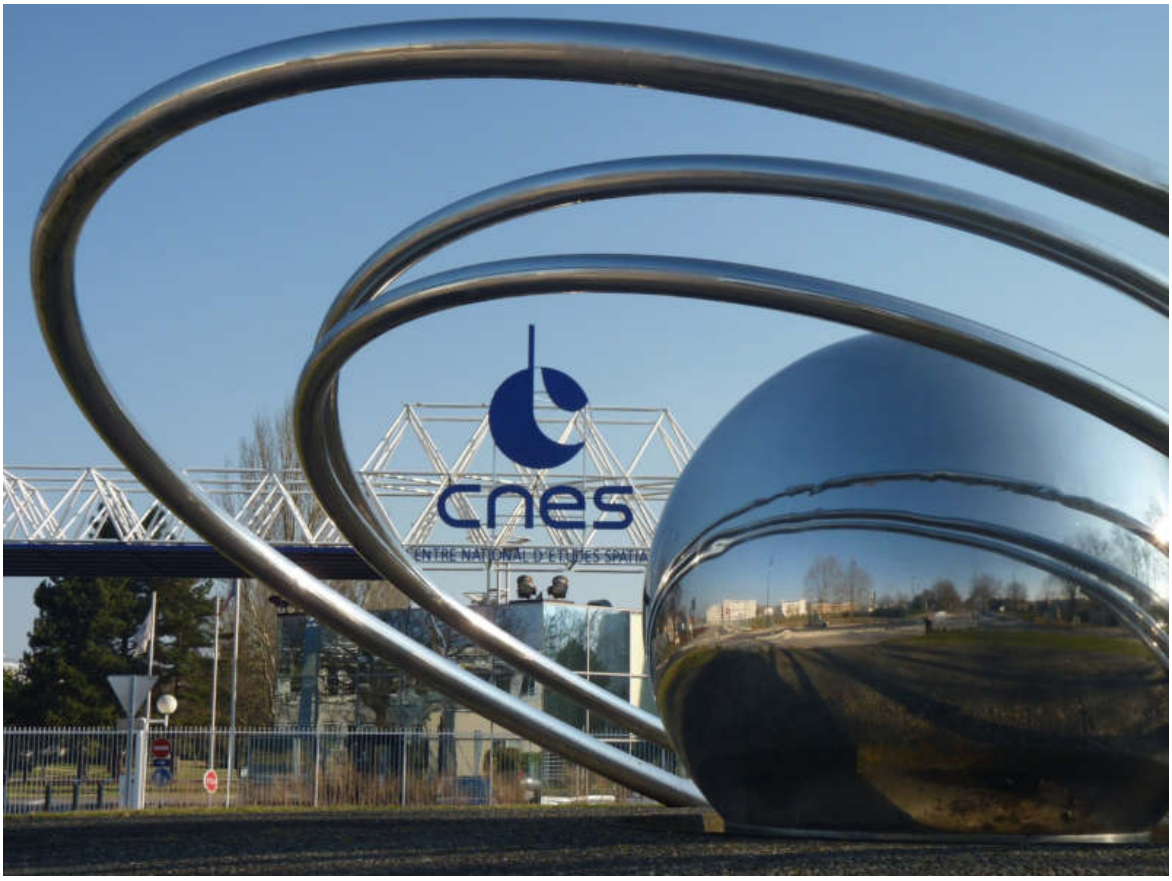



**Figure 1.1:** The Space Centre of Toulouse

## 1.2 The Balloons division

CNES also deals with the creation and operation of stratospheric gondolas dedicated to scientific or technological missions. These types of gondolas, which mass varies between 500 kg and 2500 kg, are carried into the stratosphere, through different possible types of balloons, at an altitude varying between 30 and 43 km, at which conditions similar to the ones in space are obtained. Balloon missions have many advantages over satellite missions: their cost is much lower, most of the equipment is reusable, if not heavily damaged during the landing, the deployment and operability are simpler and fewer mechanical constraints are imposed on payloads.

The missions performed by the Balloons division are mainly of two types:

1. Technological missions in which technologies intended for use on satellites are tested, calibrated and qualified (CASOLBA and innovative solar cells)

2. Scientific missions, that mainly belong to two areas:

    - Atmospheric chemistry studies that analyse the composition of the atmosphere
    - Astronomic studies in which telescopes sensitive to different wavelengths are used. Examples of this type of missions are: PILOT (sub-millimetre radiation), CLEAR (gamma rays) and FIREBALL (visible and UV radiation), which have been launched over the past years

In the astronomic missions, the pointing of the payloads consists in orienting the telescope line of sight in the desired direction by means of a two-stage fine pointing system that controls the azimuth of the entire gondola (orientation around the vertical axis) and the elevation and cross-elevation of the frame supporting the telescope itself.

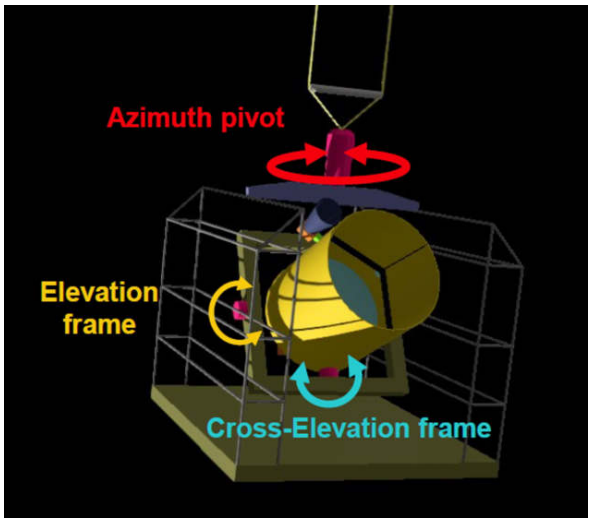



**Figure 1.2:** Representation of the telescope pointing control

Depending on the type of mission and, therefore, on the type of gondola and payload, different balloons could be implemented. The ascent of the balloon follows the Archimedes' principle: any body immersed in a fluid receives a force opposite to its weight and of value equal to the weight of the volume of fluid displaced. Therefore, several types of gas are used in the balloon envelope such as hot air, helium, or hydrogen. Considering the mass and the required operative altitude of scientific balloons, envelopes greater than one million cubic meters are required. CNES uses different types of balloons, depending on the specifications, mass, altitude, and mission duration. The most common types are the Infrared Hot Air Balloons (MIR), which

altitude depends on the daytime, Pressurized Balloons (BPS), which have a pressurized envelope allowing them to fly for a longer time, and Open Stratospheric Balloons (BSO), which have side sleeves to avoid any overpressure problem and consequently the bursting of the balloon once it reaches the final altitude.

Figure 1.3 shows a schematic representation of a BSO, in which the different elements of the flight chain can be identified.

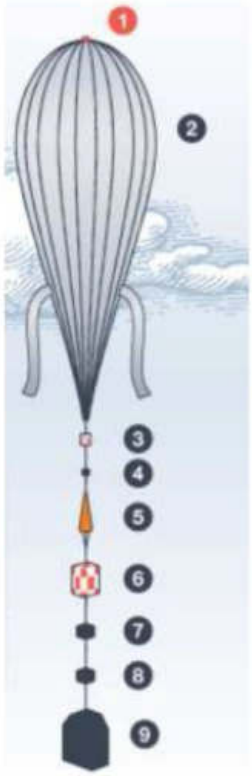



**Figure 1.3:** Schematic representation of a BSO. (1): Blowhole for the altitude control; (2): Balloon envelope; (3): GPS (4): Separator; (5): Parachute; (6): Flight avionics; (7): Antennas; (8): Strobe light; (9): Gondola.

## 1.3 The DICOS mission

The Balloon division of CNES is developing a new fine-pointing architecture in view of future astronomic missions whose themes are the analysis of the cosmic microwave background and intergalactic medium, and the study of exoplanets via stellar coronagraphy. To ensure the feasibility of this last type of mission, a technological demonstrator project called DICOS (Demonstrator of Space COronographic Imager) is under development at CNES. This project

requires very high-performance pointing capabilities, of the order of magnitude of a few hundredths of an arcsecond, and the implementation of a deformable mirror as the very-fine-pointing stage to achieve a precision that has never been obtained before. The long-term objective of a coronagraphic imager consists in producing an image, in the visible spectrum, of an exoplanet close to a star by collecting the maximum amount of light coming from the planet while rejecting the one emitted directly by the star using a mask that obscures it. During the procedure, it is also important to compensate the thermoelastic deformations of the instrument and the mechanical disturbances induced by the mechanisms aboard the gondola. DICOS will have to prove the feasibility of a mission that requires the same precision that would be needed to point, for several hours, a golf ball (4.85 cm) from a distance of 1000 km. To simulate the behaviour of the system and the transition among the different pointing stages, a dedicated simulator was created in the MATLAB/Simulink environment.

## 1.4 The Simulator

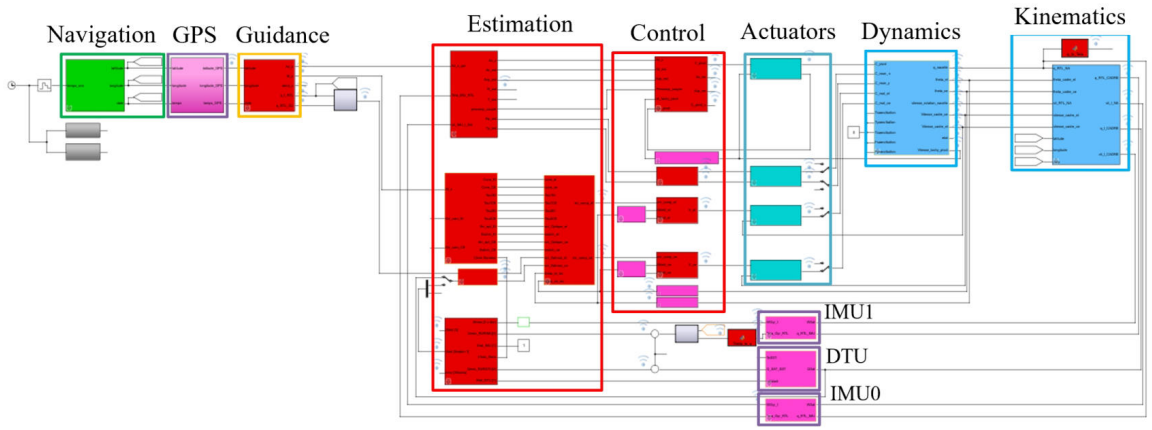



**Figure 1.4:** Representation of the simulator: the propagator in green; the flight software in red; the sensors in magenta; the actuators in cyan; the dynamic and Kinematic blocks in light blue

The structure of the simulator, reported in Figure 1.4, is briefly described below.

1. Navigation

   (a) Navigation Model

   - Input: Elapsed time sampled every second
   - Output: Latitude, longitude and actual date

- Remarks: The system is considered fixed with respect to the Earth, and coordinates are calculated with respect to the inertial reference frame

2. Sensors

(a) GPS

- Input: Latitude, longitude, actual date
- Output: Measured latitude, longitude, actual date
- Remarks: No model implemented

(b) IMU1

- Input: Speed of the cross-elevation frame with respect to the inertial one and quaternion representing the rotation between the cross-elevation frame and the local terrestrial frame
- Output: Measured speed of the cross-elevation frame with respect to the inertial one and estimated quaternion representing the rotation between the cross-elevation frame and the local terrestrial frame

(c) IMU0

- Input: Speed of the gondola with respect to the inertial reference frame and quaternion representing the rotation between the gondola reference frame and the local terrestrial frame
- Output: Measured speed of the gondola with respect to the inertial reference frame and estimated quaternion representing the rotation between the gondola reference frame and the local terrestrial frame

(d) DTU

- Input: Quaternion representing the rotation between the cross-elevation frame and the inertial frame
- Output: Estimated quaternion representing the rotation between the cross-elevation frame and the inertial frame

(e) Pivot tachometer

- Input: Angular speed of the azimuth pivot

- Output: Measured angular speed of the azimuth pivot

(f) Elevation encoder

- Input: Angular position of the elevation frame with respect to the gondola reference frame
- Output: Measured angular position of the elevation frame with respect to the gondola reference frame

(g) Cross-elevation encoder

- Input: Angular position of the cross-elevation frame with respect to the elevation frame
- Output: Measured angular position of the cross-elevation frame with respect to the elevation frame

(h) Elevation stroke end sensor

- Input: Angular position of the elevation frame with respect to the gondola reference frame measured by the elevation encoder
- Output:

(i) Cross-elevation stroke end sensor

- Input: Angular position of the cross-elevation frame with respect to the elevation frame measured by the cross-elevation encoder
- Output:

3. Flight software

(a) Guidance model

- Input: Measured latitude, longitude, actual date. Celestial coordinates of the star of interest
- Output: Azimuth, elevation and quaternions objective

(b) Gondola attitude estimation model

- Input: Angular position and velocity measured by IMUo

- Output: Azimuth, elevation and cross-elevation estimated angular position and velocity

(c) Azimuth control model

- Input: Azimuth objective, estimated azimuth and azimuth angular velocity, angular speed of the azimuth pivot, torque acting on the azimuth pivot
- Output: Electrical tension to the azimuth pivot

(d) Reaction wheel control model

- Input: Estimation of elevation and cross-elevation angular position and velocity
- Output: Electrical tension to the reactions wheels acting along the $X_{NA}$ and $Y_{NA}$ directions

4. Actuators

(a) Azimuth pivot

- Input: Electrical control voltage and angular speed of the azimuth pivot
- Output: Torque produced by the pivot

(b) XY Reaction wheels

- Input: Electrical control voltage for the X and Y reaction wheels
- Output: Torque produced by the X and Y reaction wheels

(c) Elevation motor

- Input: Electrical control voltage and angular speed of the elevation frame
- Output: Torque produced by the elevation frame motor

(d) Cross-elevation motor

- Input: Electrical control voltage and angular speed of the cross-elevation frame
- Output: Torque produced by the cross-elevation frame motor

5. Dynamic and kinematic models

(a) Dynamic model

- Input: Torques produced by the azimuth pivot, reaction wheels and elevation and cross-elevation motors. Aerodynamic perturbations
- Output: Gondola position quaternion in the local terrestrial reference frame, elevation and cross-elevation angular position and velocity, angular speed of the azimuth pivot and angular velocity of the gondola with respect to the local terrestrial reference frame
- Remarks: The aerodynamic perturbations have been considered null

(b) Kinematic model

- Input: Signals produced by the dynamic model, latitude and longitude of the system and actual date
- Output: Cross-elevation position quaternions in the local terrestrial and inertial reference frames, angular speed of the gondola and cross-elevation frames with respect to the inertial reference frame

# 2

# Reaction wheels control

The system consisting of the gondola and the entire flight chain can be seen, as explained in [1] and represented in Figure 2.1, as a multiple pendulum that is affected by a pendulum motion around the X and Y axes and a torsion motion around the Z axis. Since the three axes are coupled, the motion around one axis influences and can excite the motion around the others. The dynamic study and the attitude control of the gondola are, therefore, very complex.

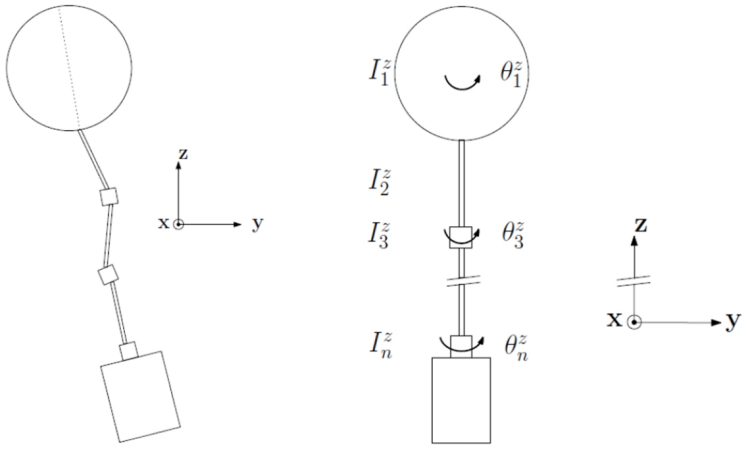



**Figure 2.1:** Schematic representation of the flight chain motion: Pendulum motion on the left and torsional motion on the right

To limit and balance the unwanted motion of the gondola, passive elements, that increase the system inertia around a specific axis, and active actuators, such as reaction wheels, are necessary.

This chapter reports the study conducted on the filter that filters the signal that controls the RWs used to compensate for the gondola pendulum motion around the X and Y axes. Figure 2.2 shows the RWs that will be used in the DICOS mission.

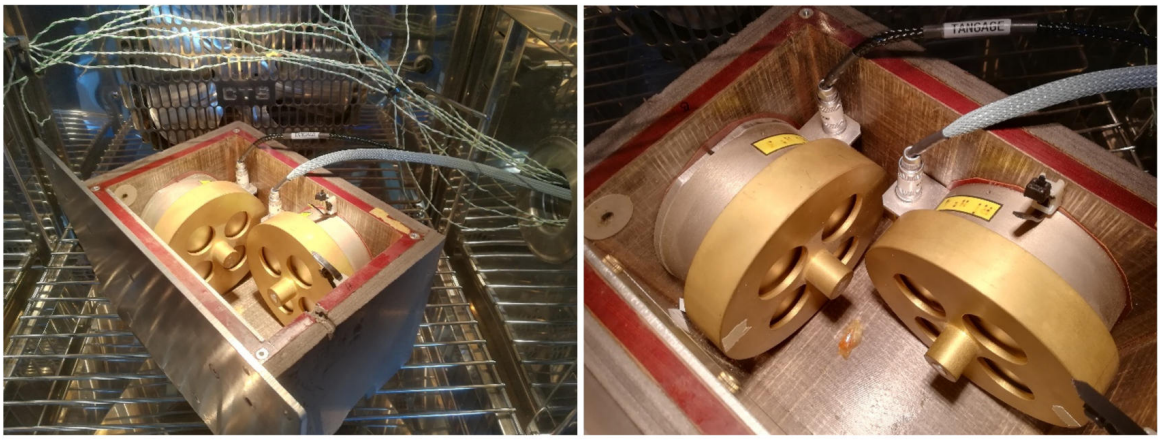


**Figure 2.2:** Photos of the Reaction Wheels in a thermal chamber

At first, the flight software implemented in the FIREBALL mission and responsible for the signal filtering was studied to create a Simulink model as similar as possible to reality; then, the obtained model was compared with the previous implementation to tune the filter parameters. Eventually, the comparison between the continue-time and discrete-time implementations has been done.

## 2.1 Flight software analysis

The filtering action is performed by the function *SrvRouReg.c* which is composed of several sub-functions, as indicated in Figure 2.3, that first initiate the necessary variables and then filter the input signal.

From the code reported below, which represents the most important part of the flight software, it is possible to notice that there are two inputs signals: the “fVitNum_degps” and the ”fAttNum_deg”, that is multiplied by the “fGainK”. These two signals are summed up together obtaining the input signal of the filter, X.

```
void SrvRouRegSendCmd ( ESrvRouRegDrv*RouDrv )
{
  float fTmp = 0.0f;
```

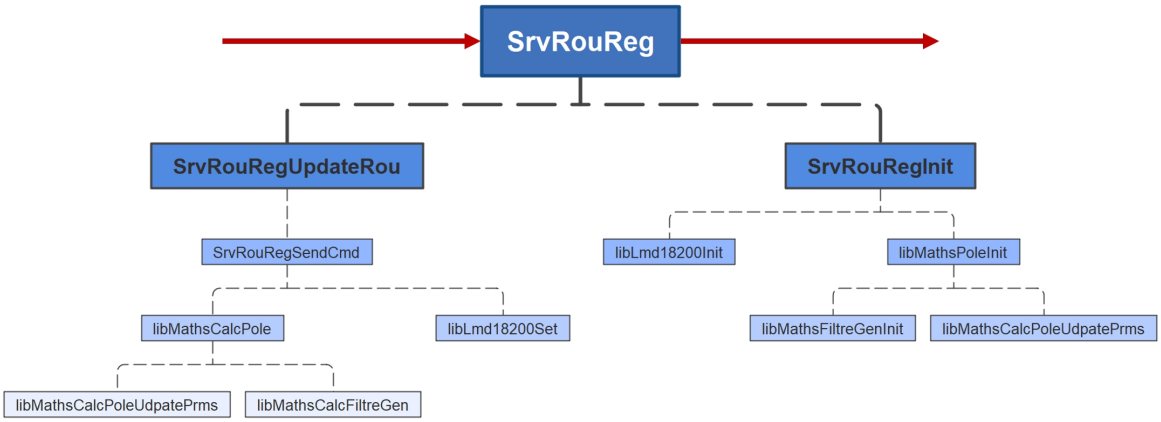



**Figure 2.3:** Schematic representation of the filter functions implemented in the flight software

```
  fTmp =  libMathsCalcPole( RouDrv->fVitNum_degps
                  + ( RouDrv->fGainK * RouDrv->fAttNum_deg ),
                  RouDrv->fPolK,
                  RouDrv->fPolF,
                  RouDrv->fPolZ,
                  RouDrv->pePolePrms )
                + RouDrv->fRoueCmdOffset;
  // Affectation si pas d'erreur
  if( !isnan( fTmp ) )
  {
    RouDrv->fRoueCmd = fTmp;
  }
  // Actualiser la sortie selon l'état de l'interrupteur
  fTmp = GET_VALUE_SEL( float, RouDrv->eSwitchRoue );
  // Appliquer sur la PWM
  libLmd18200Set( &RouDrv->ePwmOut, fTmp );
}
```

The implemented filter is the so-called digital filter which, in signal processing, is a system that performs mathematical operations on a sampled, discrete-time signal to enhance certain aspects of that signal itself. The transfer function for a linear, time-invariant, digital filter can be expressed as a transfer function in the z-domain, where $X(z)$ is the input and $Y(z)$ is the output. Its expression is reported in Equation 2.1.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{(-1)} + a_2 z^{(-2)} + ... + a_n z^{(-n)}}{b_0 + b_1 z^{(-1)} + b_2 z^{(-2)} + ... + b_m z^{(-m)}} \tag{2.1}$$

For an explanation on the z-domain and its advantages, see [2]. The implemented filter is of a second order, so the transfer function reduces to:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{(-1)} + a_2 z^{(-2)}}{b_0 + b_1 z^{(-1)} + b_2 z^{(-2)}} \tag{2.2}$$

The output signal is therefore:

$$Y(z)(b_0 + b_1 z^{(-1)} + b_2 z^{(-2)}) = X(z)(a_0 + a_1 z^{(-1)} + a_2 z^{(-2)}) \tag{2.3}$$

And its expression in the time domain is:

$$b_0 y(n) + b_1 y(n-1) + b_2 y(n-2) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) \tag{2.4}$$

That eventually becomes:

$$y(n) = \frac{(a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) - b_1 y(n-1) - b_2 y(n-2))}{b_0} \tag{2.5}$$

In which the different parameters $a_0$, $a_1$, $a_2$, $b_0$, $b_1$ and $b_2$ had been previously tuned for the FIREBALL mission.

## 2.2 Previous implementation

The MATLAB function and Simulink models that were implemented inside the simulator were not complete. In fact, the Simulink model reported in Figure 2.4, does not make use of the "selecteurs" and the parameters defined inside were not existent. Moreover, it evident that only the signals Rp_est and Tp_est, which are respectively the velocity around the X and Y axes, are used, while the instantaneous angles around the same axes are not present.

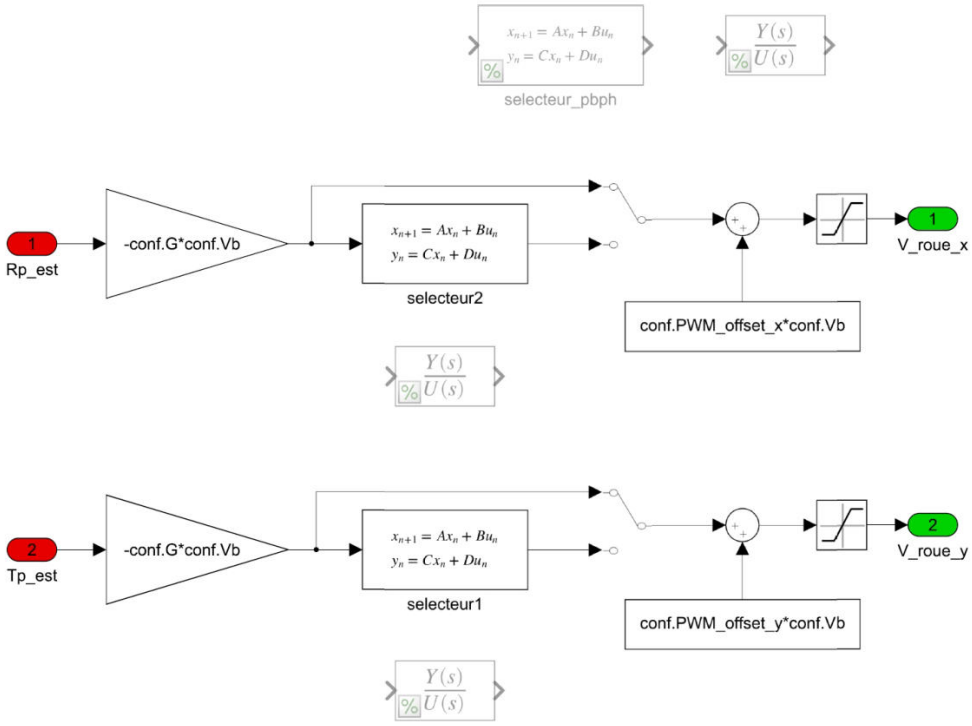



**Figure 2.4:** Previous filter model implemented in the simulator

## 2.3 New implementation

At first, no correspondence between the MATLAB function relative to the old filter and the flight software previously analysed was found. However, results (see also the next chapter) showed that they are quite similar. Therefore, the old MATLAB implementation was used to tune some parameters of the new filter that, otherwise, could not be estimated.

The MATLAB code first defines the wheels parameters (electrical resistance, inertia, gains, supply tension...) and the filter parameters (proper frequencies of the low pass filter and high pass filter, the middle frequency...).

The idea was to use a low pass filter and a high pass filter to create a band-bass filter. Its transfer function is reported in Equation 2.6:

$$G(s) = \frac{1}{1 + s\tau_{pb}} \frac{s\tau_{ph}}{1 + s\tau_{ph}} \tag{2.6}$$

The values of the different parameters are reported below:

- $R = 13.60\,\Omega$
- $I_r = 0.0074\,\mathrm{kg\,m^2}$
- $K_e = 0.3410\,\mathrm{V\,s\,rad^{-1}}$
- $K_m = 0.43\,\mathrm{N\,m\,A^{-1}}$
- $V_b = 28\,\mathrm{V}$
- $\tau_{pb} = 0.0080\,\mathrm{s}$
- $\tau_{ph} = 0.5305\,\mathrm{s}$

One can see that: $\tau_{pb} < \tau_{ph}$. The Bode plot of $G(s)$ is, as expected, the one presented in Figure 2.5. It is also interesting to notice that in the MATLAB code, the filter was normalized in order to obtain $0\,\mathrm{dB}$ gain at $\omega_0$.

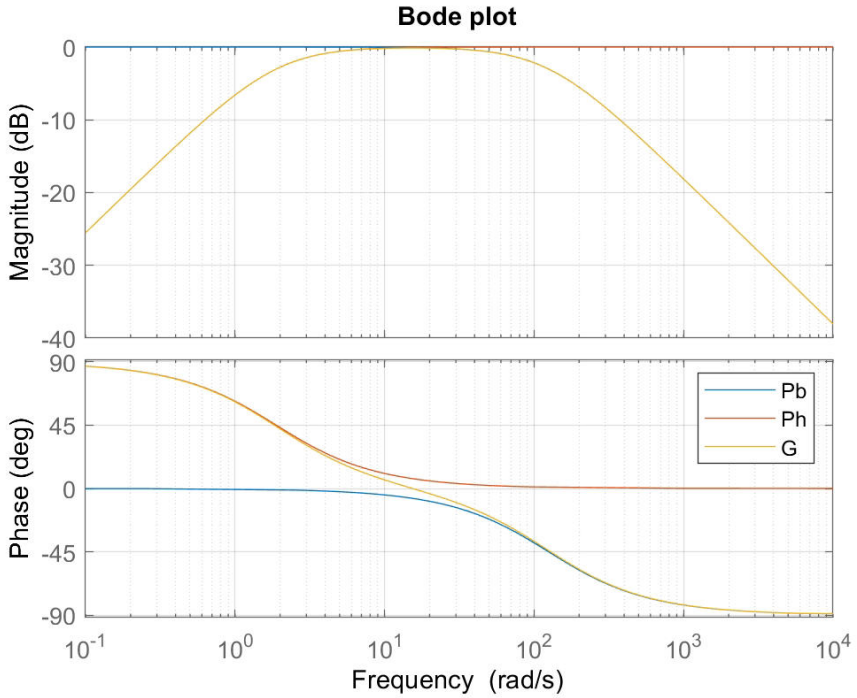



**Figure 2.5:** $G(s)$ Bode diagram

To compare the old and the new implementation, it was necessary to express $H(z)$ in the $s$-domain using the bilinear transform reported in Equation 2.7, and replacing the $z$ variable in the Equation 2.2. By considering that $a_1 = 0$, $a_2 = -a_0$ and $b_0 = 1$, one obtains:

$$s = \frac{2}{T_e}\frac{z-1}{z+1} \tag{2.7}$$

$$H(s) = \frac{a_2 + a_0\left(\dfrac{2+sT_e}{2-sT_e}\right)^2}{b_2 + b_1\left(\dfrac{2+sT_e}{2-sT_e}\right) + \left(\dfrac{2+sT_e}{2-sT_e}\right)^2} \tag{2.8}$$

And after some calculations:

$$H(s) = \frac{\left(\dfrac{2K\xi}{\omega_0}\right)s}{1 + 2\xi\dfrac{s}{\omega_0} + \left(\dfrac{s}{\omega_0}\right)^2} \tag{2.9}$$

By plotting the Equation 2.9, one obtains the Figure 2.6, which is quite similar to the one relative to the previous implementation.

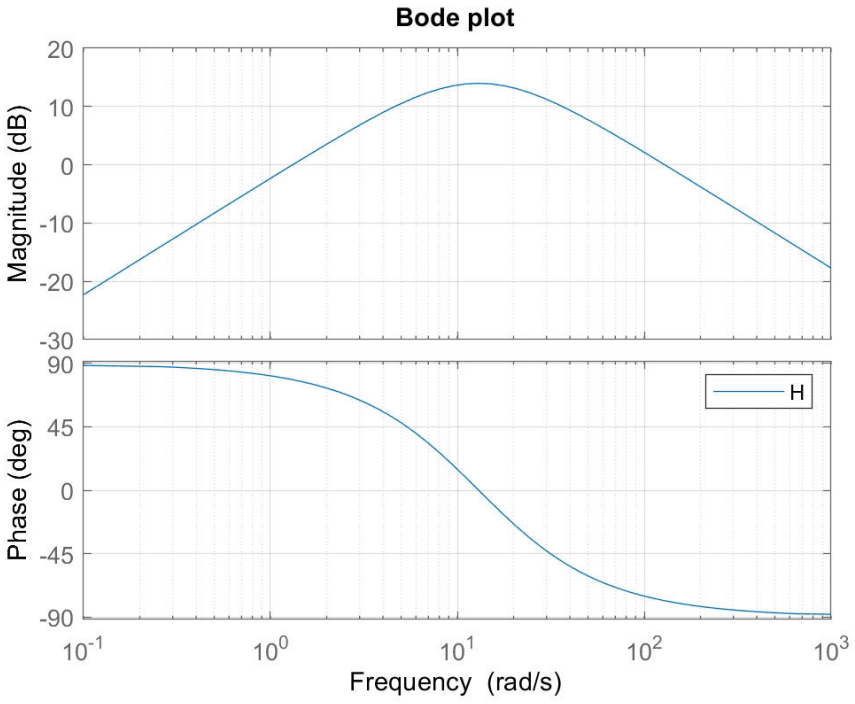



**Figure 2.6:** $H(s)$ Bode diagram

By comparing the obtained plot with the one relative to the filter previously implemented, the following parameters were found: $\omega_0 = 15\,\mathrm{rad\,s^{-1}}$ and $\xi = 5$. Moreover, the value of $K$ that could ensure $|H(\omega_0)| = 0\,\mathrm{dB}$ was investigated.

$$H(s) = \frac{H(S)}{|H(\omega_0)|} = \frac{\left(\frac{2\xi}{\omega_0}\right) s}{1 + 2\xi\frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2} \tag{2.10}$$

Therefore, $K = 1$ and the resulting transfer function is:

$$H(s) = \frac{150s}{225 + 150s + s^2} \tag{2.11}$$

The corresponding bode plot is reported in Figure 2.7, while the discrete time filter that was implemented in the simulator is reported in Figure 2.8

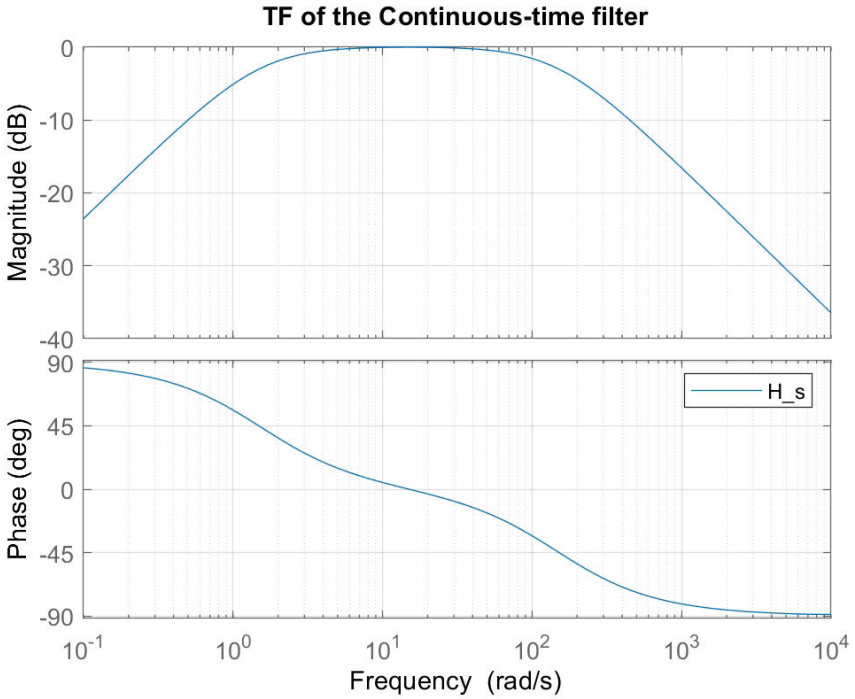
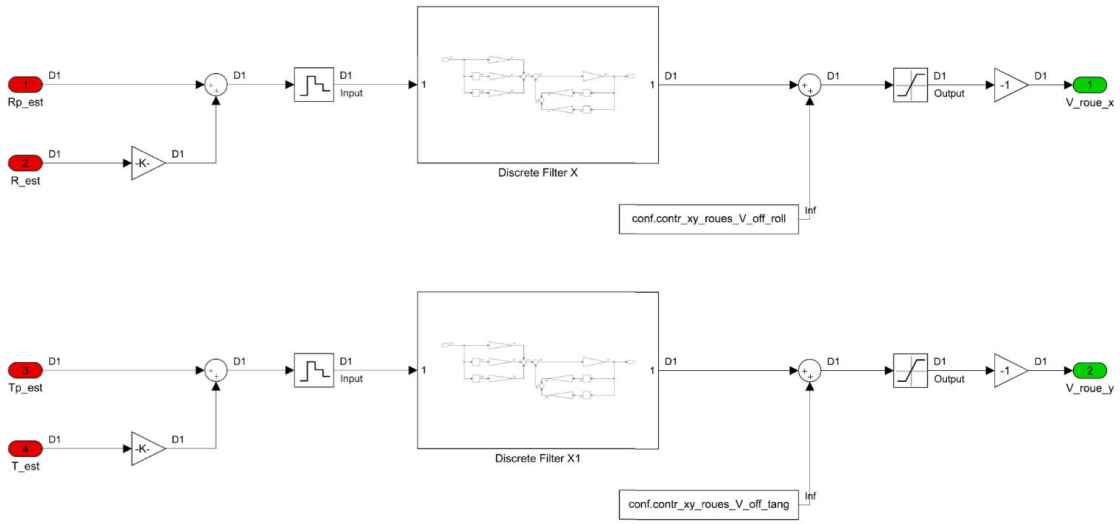
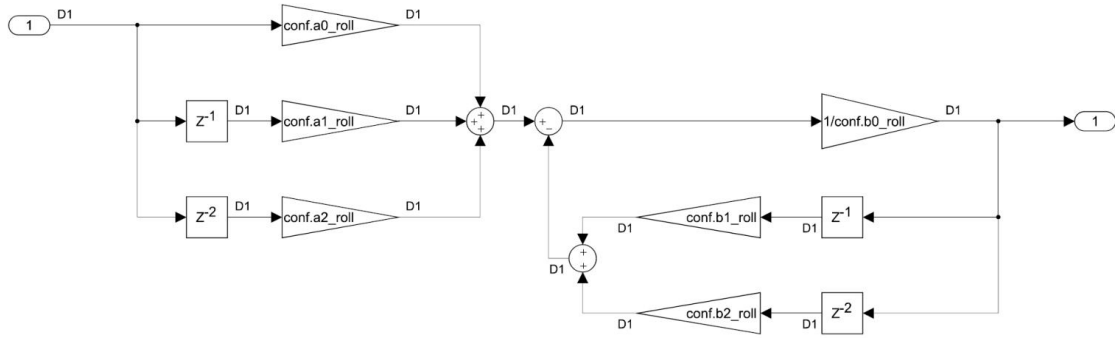



**Figure 2.7:** $H(s)$ Bode diagram considering $K = 1$




**(a)** RW control block model




**(b)** Discrete filter model

**Figure 2.8:** New filter model implemented in the simulator

## 2.4 Discussion and comparison of the results

### 2.4.1 Continue vs. Discrete implementation

As expected and showed in Figure 2.9, since the continue transfer function was correctly retrieved from the discrete one, the two output signals, given the same input signal, are basically the same. However, it is important to choose the correct sampling frequency with respect to the frequency of the input signal. In fact, working in the discrete time domain shall not introduce errors that could compromise the system performance. Therefore, an investigation on the sampling period was done.

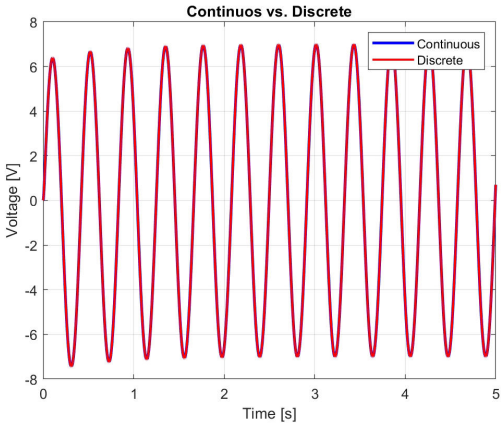



**Figure 2.9:** Continue vs. discrete time results

Considering that the filter is centered around $\omega_0 = 15\ \mathrm{rad\,s^{-1}}$, one could expect an input signal having that frequency. By looking at the Bode plot in Figure 2.10, the range of frequencies in which $|H(s)| > -3\ \mathrm{dB}$ was identified.

Between the two identified frequencies, the most problematic could be the higher one, which was therefore considered. The risk, in fact, is to have a sampling frequency not sufficiently high with respect to the input.

By considering $G = 2\ \mathrm{V\,rad^{-1}}$ and the same sinusoidal input having $\omega = 151.0\ \mathrm{rad\,s^{-1}}$, the error between the input and the output signals of the continuous and discrete time filters has been analysed varying the sapling period $T_e$. As visible from Figure 2.11, considering a sampling frequency of $f_e = 1$ Hz, yields to an absolute maximum error of about $0.7$ V, which is approximately 2.5% of the end of scale value (28 V) and that, therefore, can be considered sufficient.

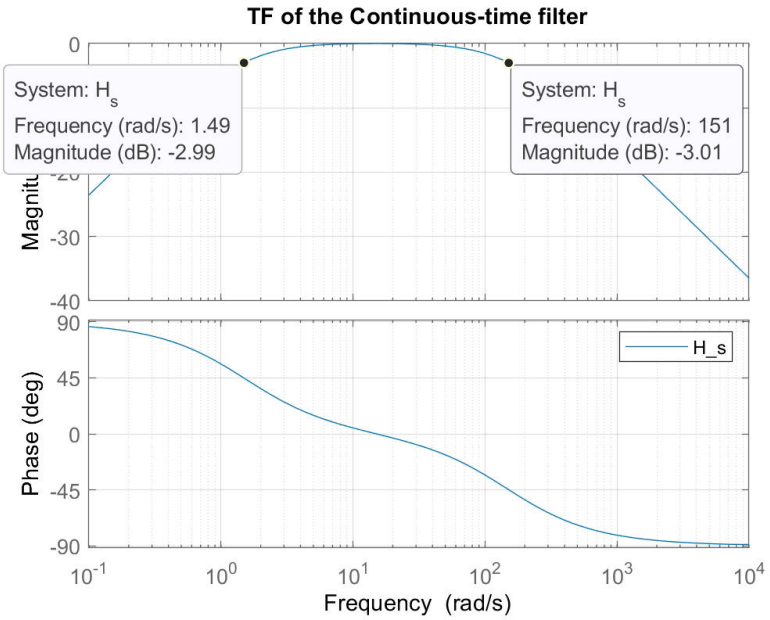
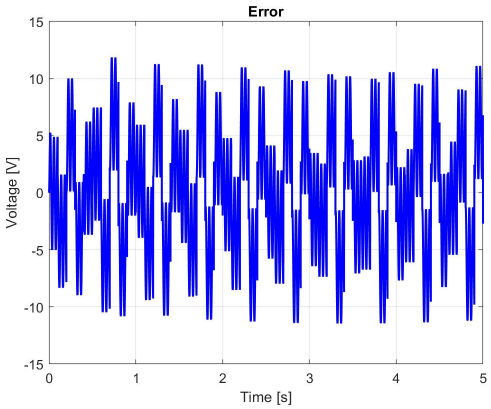
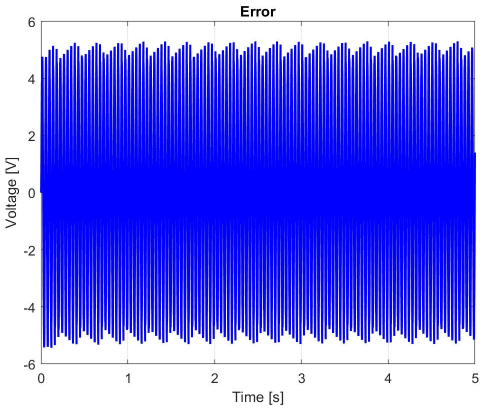


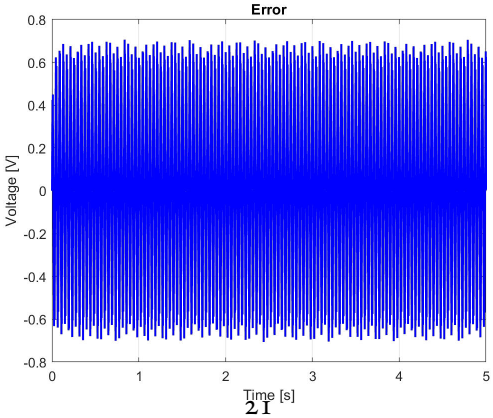

**Figure 2.10:** $H(s)$ Bode plot considering $K = 1$




**(a)** $T_e = 0.1s$




**(b)** $T_e = 0.01s$




**(c)** $T_e = 0.001s$

**Figure 2.11:** Continue vs. Discrete results depending on $T_e$

### 2.4.2 Output vs. Input

By comparing the output and input signals, it is possible to notice that the first one is shifted due to the offset applied to the signal soon after the filter, has different magnitude and phase due to the filter itself, and has an opposite sign due to the applied negative gain. Moreover, it would be possible to notice that, if the input signal is too high, the output signal will be capped by the saturation block, which limits its magnitude between $+28\ \mathrm{V}$ and $-28\ \mathrm{V}$, as desired.

# 3

# Attitude estimation of the gondola

The input signals to the filter analysed in chapter 2 are the output signals of the so-called complementary filter that combines the measurements of angle and angular velocities done by the IMU0 to estimate the attitude of the gondola. In this chapter, the before-mentioned filter is analysed: at first, the code implemented in the flight software of the FIREBALL mission is studied to retrieve an equivalent Simulink model and, then, the results are discussed. The complementary filter combines two separate data fluxes produced by the IMU0: the first one contains the three attitude angles that are retrieved from 3 accelerometers, while the second one is retrieved from 3 FOG. The accelerometers offer very good measurements at low frequency due to the processes performed inside the IMU, but are not reliable at high frequency. Vice versa, the angular velocity measurements made by the gyro inside the IMU are very good at high frequency and are not good at low frequency. The implemented complementary filter combines the two streams of data, since they refer to the same entity, and combines them to get a better estimation of the gondola attitude. The idea, in fact, is to give a bigger importance to the velocity data when the gondola rotates rapidly and bigger importance to the angle measurements when the gondola is rotating slowly.

## 3.1 Flight software analysis

The filtering action is performed in the *InterfacerImu90.stf* function. The process is divided into two main parts. In the first one, the information from the IMU is verified.

```
IF ( ( not IMU90Vit.Val ) OR ( not IMU90Att.Val ) ) THEN
  (* Si le MPF était opérationnel, on le sécurise *)
  IF ( etatPF = E_ETAT_PF_OPER ) THEN
    CommandeSFC_MPF := CMD_SECU ;
  END_IF;
  (* Si le MPA était ON, on le sécurise *)
  IF ( LibMpa.Etat = MPA_FBL_ETAT_MARCHE ) THEN
    CommandeSFC_MPA := CMD_SECU ;
END_IF;
```

If the information is valid, the attitude of the gondola is initialized using the angle measurements provided by the IMU. Roll, pitch and azimuth angles saved in the corresponding structures are converted by the function “any_to_lreal” in double precision real values and saved in the corresponding vectors.

```
AttitudeImu[1] := any_to_lreal( IMU90Att.Roulis );
AttitudeImu[2] := any_to_lreal( IMU90Att.Tangage );
AttitudeImu[3] := any_to_lreal( IMU90Att.Azimut ) ;
```

The instantaneous velocity is then calculated monitoring the angle variation in each sampling period $P_e$:

```
VitesseImu[1] := rad_to_deg_l ( any_to_lreal( IMU90Vit.DeltaThetaX1 ) / Pe ) ;
VitesseImu[2] := rad_to_deg_l ( any_to_lreal( IMU90Vit.DeltaThetaX2 ) / Pe ) ;
VitesseImu[3] := rad_to_deg_l ( any_to_lreal( IMU90Vit.DeltaThetaX3 ) / Pe ) ;
```

Then, there is an attitude estimator filter that aims to increase the attitude accuracy that can be either used or not. If the filter is active, the program checks if it was run for the first time, and in that case, the roll, pitch and azimuth angles are assigned to be equal to the “Sn” value of the corresponding structures:

```
roulis_est_prec := passebas_roulis_fbl.Sn;
tangage_est_prec := passebas_tangage_fbl.Sn;
azimut_est_prec := passebas_azimut_fbl.Sn;
```

The different structures are created by the following functions, in which one can notice that they take into account the sampling period, an indicator (TRUE/FALSE) that resets the integral

effect of the filter, the $\tau$ of the low pass filter and the input signal. The latter is given by the sum of the angles and the relative instantaneous velocities multiplied by the corresponding $\tau$.

```
passebas_roulis_fbl(Pe, TRUE, passebas_roulis_tau, AttitudeImu[1] +
    passebas_roulis_tau * VitesseImu[1] );
passebas_tangage_fbl(Pe, TRUE, passebas_tangage_tau, AttitudeImu[2] +
    passebas_tangage_tau * VitesseImu[2] );
passebas_azimut_fbl(Pe, TRUE, passebas_azimut_tau, AttitudeImu[3] -
    passebas_azimut_tau * VitesseImu[3] );
```

Eventually, the initialization indicator is placed equal to FALSE so that the initialization process is skipped during the successive iterations.

```
bInitFiltresAttitudeIMU := FALSE;
```

After the initialization, the new angles are calculated:

```
passebas_roulis_fbl(Pe, FALSE, passebas_roulis_tau, AttitudeImu[1] +
    passebas_roulis_tau * VitesseImu[1] );
passebas_tangage_fbl(Pe, FALSE, passebas_tangage_tau, AttitudeImu[2] +
    passebas_tangage_tau * VitesseImu[2] );
passebas_azimut_fbl(Pe, FALSE, passebas_azimut_tau, AttitudeImu[3] -
    passebas_azimut_tau * VitesseImu[3] );
```

The gondola angular velocity components are calculated considering the difference between the new and old values of the angles and the sample period $P_e$

```
VitesseNacelle[1] := ( passebas_roulis_fbl.Sn - roulis_est_prec ) / Pe ;
VitesseNacelle[2] := ( passebas_tangage_fbl.Sn - tangage_est_prec ) / Pe ;
VitesseNacelle[3] := NormaliserDeltaAzimut_L( azimut_est_prec -
    passebas_azimut_fbl.Sn ) / Pe ;
```

Note that due to the convention chosen for the yaw and azimuth directions, the result in the $Z$ direction has to be multiplied by $-1$. Moreover, one can see that the function *NormaliserDeltaAzimut_L* is used to bring its input in the range of $\pm 180\,^{\circ}$. Then, the previous values of roll, pitch and yaw are updated for the next iteration:

```
roulis_est_prec := passebas_roulis_fbl.Sn;
tangage_est_prec := passebas_tangage_fbl.Sn;
```

```
azimut_est_prec := passebas_azimut_fbl.Sn;
```

Eventually, the output values (attitude angles and angle rates) are calculated:

```
AttitudeNacelle[1] := roulis_est_prec;
AttitudeNacelle[2] := tangage_est_prec;
AttitudeNacelle[3] := NormaliserAzimut_L( azimut_est_prec + AzBiaisImuNac );
```

Note that the function *NormaliserAzimut_L.c* is used to bring its input in the range of $0 - 360\,^{\circ}$.

On the other hand, if one does not want to use the filter, the attitude angles and angular velocities of the gondola are simply the ones measured by the IMU, and the following equations are applied:

```
AttitudeNacelle[1] := AttitudeImu[1] ;
AttitudeNacelle[2] := AttitudeImu[2] ;
AttitudeNacelle[3] := NormaliserAzimut_L( AttitudeImu[3] + AzBiaisImuNac );

VitesseNacelle[1] := VitesseImu[1] ;
VitesseNacelle[2] := VitesseImu[2] ; (* - 0.0015; *)
VitesseNacelle[3] := VitesseImu[3] ;
```

## 3.2 Previous implementation

The filter originally implemented was composed by a discrete low pass filter and a discrete high pass filter that have been expressed, via a MATLAB function in their space vector representation. The Simulink model previously implemented in the simulator is reported in Figure 3.1.

This implementation had to be tested and, anyhow, did not perfectly represent what is performed in the flight software.

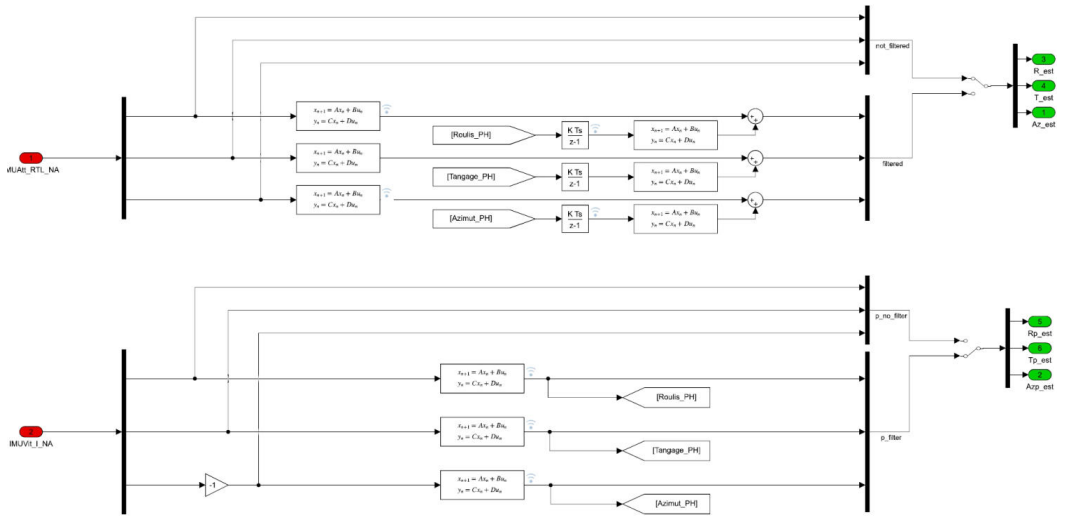



**Figure 3.1:** Previous complementary filter model implemented in the simulator

## 3.3 New implementation

By considering the analysis done before on the flight software, it is possible to notice that the velocities are first multiplied by the corresponding $\tau$ and, then, summed with the measured angles. The results are later filtered with a low pass filter, which expression in the $z$-domain is reported in Equation 3.1.

$$H(z) = \frac{T_e(z+1)}{(T_e+2\tau)z + (T_e-2\tau)} \tag{3.1}$$

The velocity can be computed as reported in Equation 3.2 that can be expressed in the $z$-domain as showed in Equation 3.3

$$v_k = \frac{\Delta u}{\Delta t} = \frac{u_k - u_{k-1}}{T_e} \tag{3.2}$$

$$v(z) = \frac{u(z) - zu(z)}{T_e} = \frac{1-z}{T_e}u(z) \tag{3.3}$$

The complete Simulink model is reported in Figure 3.2. It is possible to notice that the filter can be turned on or off completely: it is not possible to filter only the angle signal or the velocity signal, as it was wrongly done in the previous implementation.

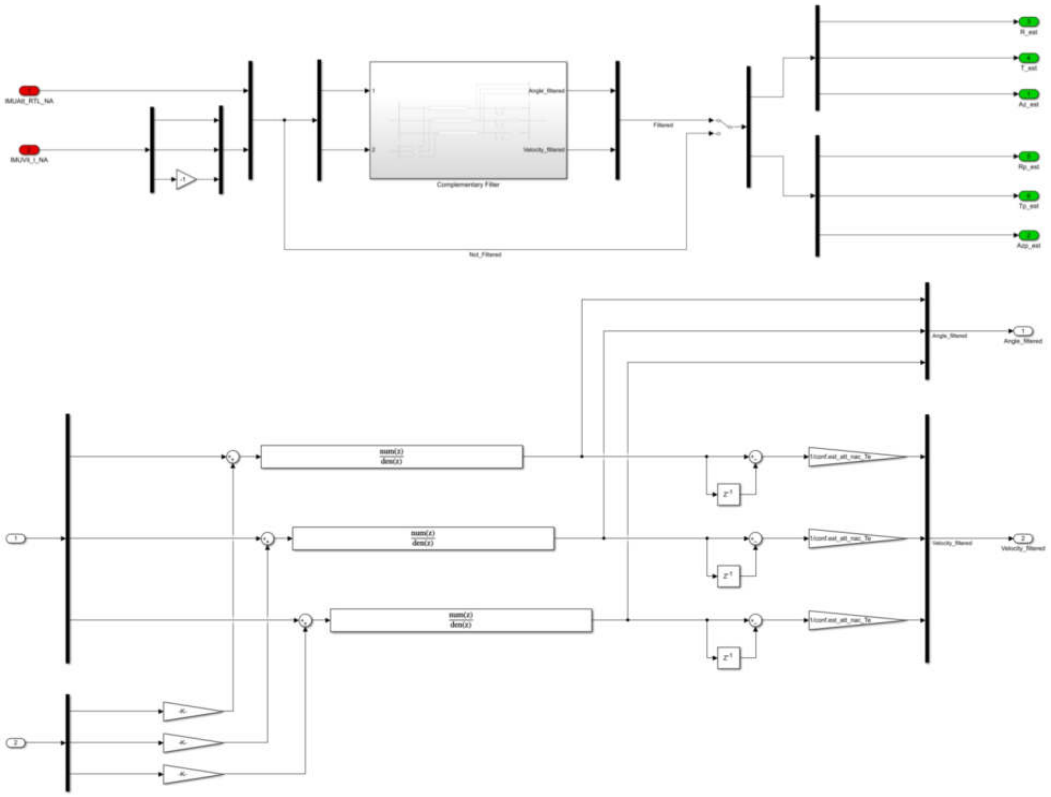


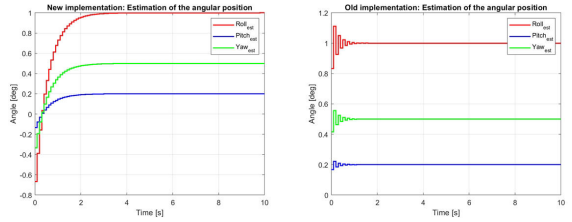
**Figure 3.2:** New complementary filter model implemented in the simulator

## 3.4 Discussion and comparison of the results

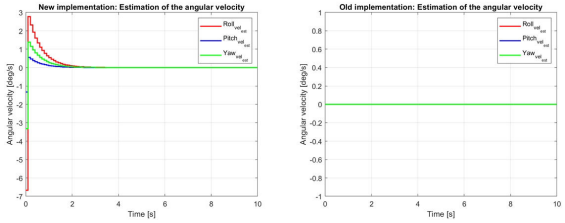
### 3.4.1 Old vs. new implementations

To compare the results, the following values have been considered for the different parameters: $T_e = 0.1$ s, $\tau_{roll} = \tau_{pitch} = \tau_{azimuth} = 0.01$ s. Figure 3.3, Figure 3.4 and Figure 3.5 report the comparison between the output of the new (on the left) and old (on the right) filters for different combination of inputs.

From Figure 3.3a one can see that the angular profiles are not the same. In fact, they differ at the beginning (in the transient) and tend to the same steady state values. From Figure 3.3b it can be seen that the two models differ again. In fact, in the old implementation, the velocity of the gondola was computed filtering with a high pass filter the measurements of velocity done by the IMUo, so it was strongly dependent to the input velocity that, in this case, is zero. In the flight software and, therefore, in the new implementations, the velocity components are computed

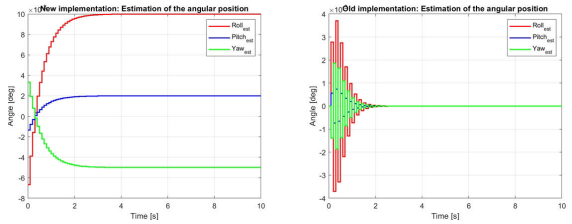



**(a)** Angular response. New model on the left, previous model on the right

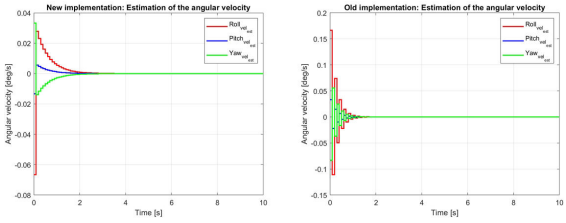



**(b)** Velocity response. New model on the left, previous model on the right

**Figure 3.3:** Input signals: $\overrightarrow{x_{IMU}} = [1.0\,0.2\,0.5]\,\mathrm{rad}$ and $\overrightarrow{\dot{x}_{IMU}} = [0\,0\,0]\,\mathrm{rad\,s^{-1}}$




**(a)** Angular response. New model on the left, previous model on the right

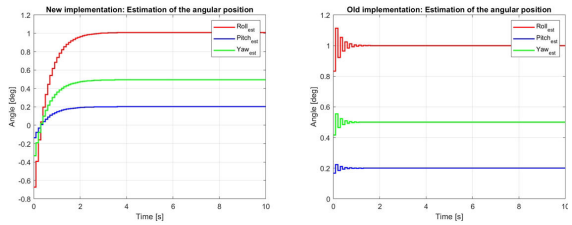



**(b)** Velocity response. New model on the left, previous model on the right

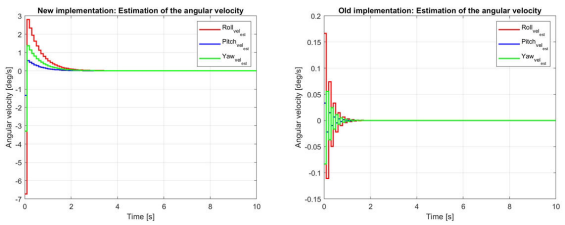
**Figure 3.4:** Input signals: $\overrightarrow{x_{IMU}} = [0\,0\,0]\,\mathrm{rad}$ and $\overrightarrow{\dot{x}_{IMU}} = [1.0\,0.2\,0.5]\,\mathrm{rad\,s^{-1}}$

calculating the discrete-time derivative of the attitude angles of the gondola. Therefore, the angular and angular velocity signals are more correlated one to another.

From Figure 3.4 it can be seen that the two models differ again. In the old implementation, the input velocity is filtered with a high pass filter to obtain the output velocity. Since the frequency tends to $0\,\mathrm{rad\,s^{-1}}$, the magnitude tends to $-\infty\,\mathrm{dB}$ and, therefore, the signal is attenuated. The same filtered signal is also integrated to obtain the angular profile. In the new implementation instead, the low pass filter filters the input velocity and yields directly to a (angular) signal that is




**(a)** Angular response. New model on the left, previous model on the right




**(b)** Velocity response. New model on the left, previous model on the right

**Figure 3.5:** Input signals: $\overrightarrow{x_{IMU}} = [1.0\,0.2\,0.5]\,\mathrm{rad}$ and $\overrightarrow{\dot{x}_{IMU}} = [1.0\,0.2\,0.5]\,\mathrm{rad\,s^{-1}}$

not attenuated to zero since, in this case, the magnitude tends to $1$ dB when the frequency tends to $0\,\mathrm{rad\,s^{-1}}$. The output velocity is computed as a consequence of the angle variation and, therefore, the two streams of signals are correlated.

Analogous considerations can be done for the third case, which plots are reported in Figure 3.5.

### 3.4.2 Continue vs. discrete implementations

To compare the discrete and the continue-time implementation and study the filter performance, the test model represented in Figure 3.6 was created, and the following values were considered $T_e = 0.001$ s, $\tau_{roll} = \tau_{pitch} = \tau_{azimuth} = 0.01$ s.

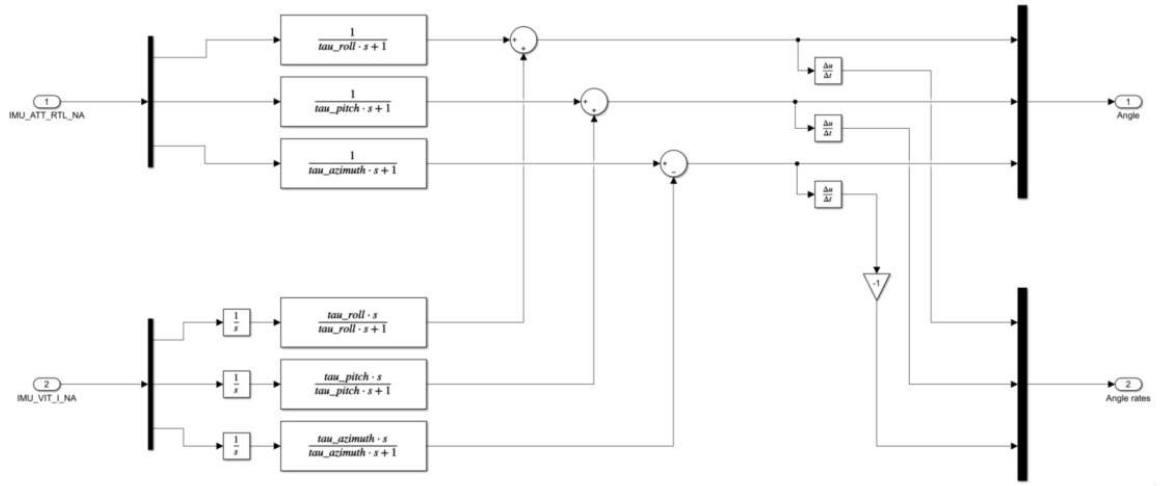
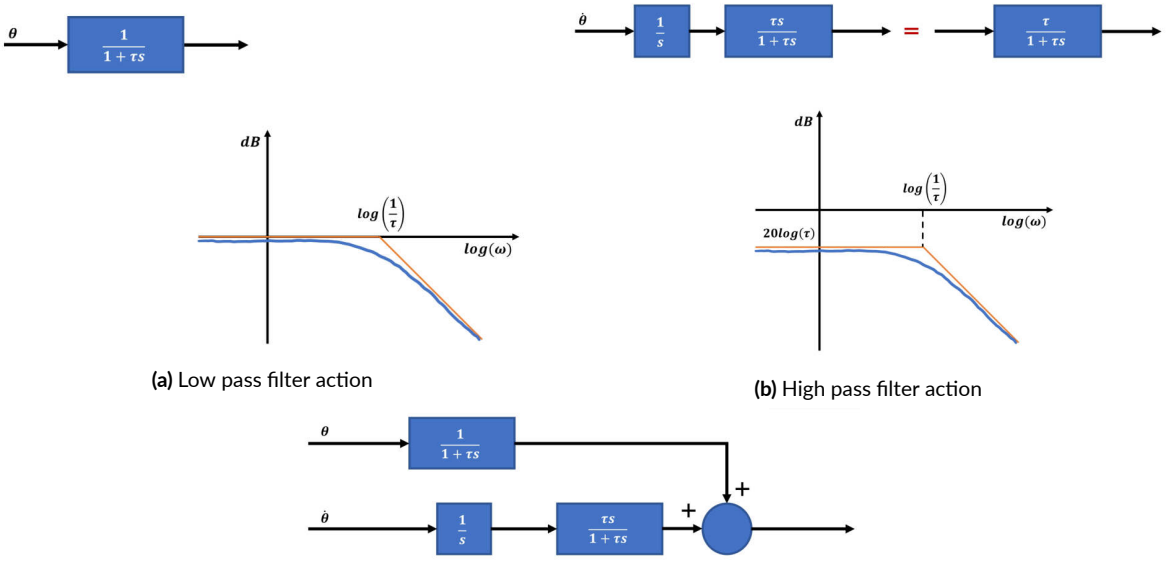



**Figure 3.6:** Simulink model used to test the continue and discrete implementation

As reported schematically in Figure 3.7, both the attitude signal and attitude rate signal are filtered with a low pass filter that has a time constant $\tau$. However, the attitude rate signal is also multiplied by $\tau$ itself, which, consequently, attenuates the signal by $20log(\tau)$ dB.

Figure 3.8 and Figure 3.9 report the output of the discrete and continue-time filter for different inputs.

As expected from Figure 3.8, the amplitude of the output signals in the discrete and continue time domains, are attenuated of $-20$ dB/dec for frequencies higher than $1000$ Hz (since in this case $\tau = 0.001$ s). In addition, it is possible to notice that the discrete time signal, for $f = 2000$ Hz has an amplitude smaller than the continuous one. This is due to the sampling period that is too high.




**(a)** Low pass filter action

**(b)** High pass filter action

**(c)** Schematic representation of the complementary filter

**Figure 3.7:** Complementary filter action

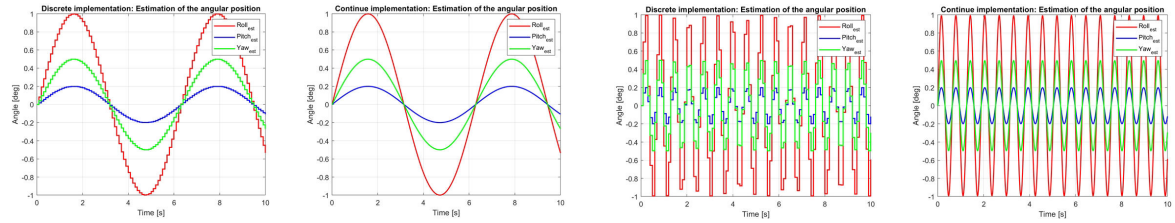
Figure 3.9 shows that, as mentioned before, the input signal, that in this case is a sinusoidal velocity, is attenuated by $20log(\tau)$ at any frequency.

Moreover, assuming that the output signal could depend on the ratio of the two input frequencies $\omega_1$ and $\omega_2$, an analytical analysis was performed. Given two sinusoidal inputs $x_1 = Asin(\omega_1 t)$ and $x_2 = Bsin(\omega_2 t)$, one can write that:

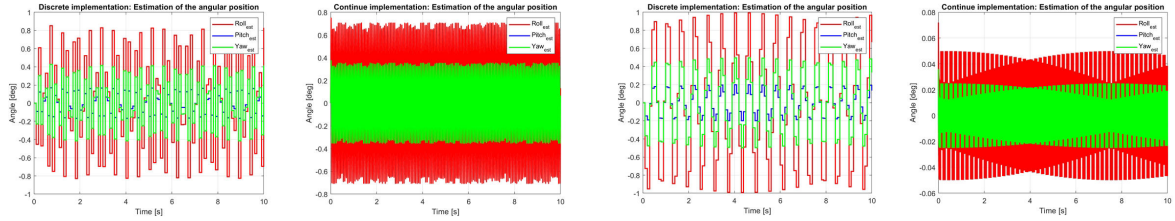
$$y(t) = x_1(t) + x_2(t) = (A - B)sin(\omega_1 t) + 2Bsin\left(\frac{\omega_1 + \omega_2}{2}t\right)cos\left(\frac{\omega_1 - \omega_2}{2}t\right) \quad (3.4)$$

The first addend of the Equation 3.4 is a simple sinusoidal function that is zero at $t_{k_1} = \frac{k_1\pi}{\omega_1}$, where $k_1 = 1, 2, 3....$ Figure 3.10a reports its graph.

The second addend of the Equation 3.4 is given by the product between a $sin$ and a $cos$. The resulting function is zero at $t_{k_2} = \frac{2k_2\pi}{\omega_1 + \omega_2}$, where $k_2 = 1, 2, 3...$, and at $t_{k_3} = \frac{\pi(1 + 2k_3)}{|\omega_1 - \omega_2|}$, where $k_3 = 0, 1, 2, ....$

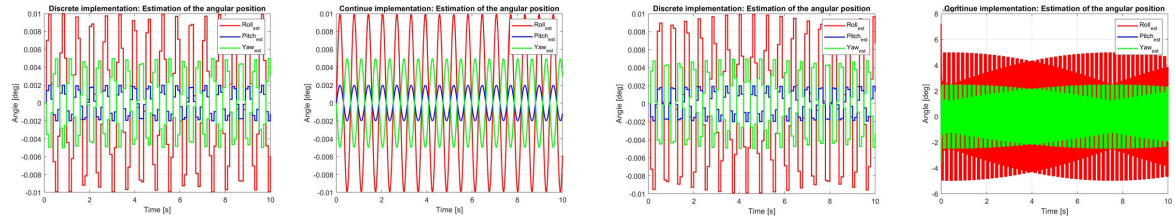



**(a)** $f = 1\,\text{Hz}$ **(b)** $f = 10\,\text{Hz}$



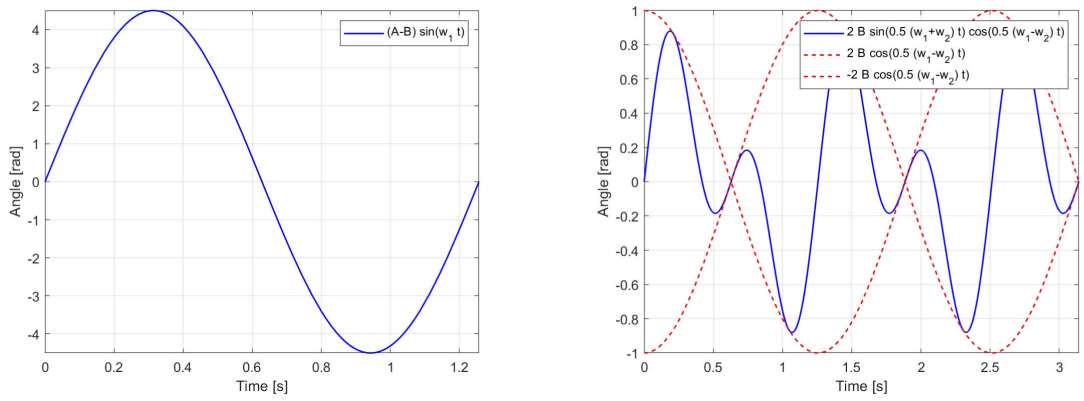

**(c)** $f = 100\,\text{Hz}$ **(d)** $f = 2000\,\text{Hz}$

**Figure 3.8:** Input signals: $\overrightarrow{x_{IMU}} = [1.0\,0.2\,0.5] sin(2\pi f t)\,\text{rad}$ and $\overrightarrow{\dot{x}_{IMU}} = [0\,0\,0]\,\text{rad}\,\text{s}^{-1}$




**(a)** $f = 10\,\text{Hz}$ **(b)** $f = 2000\,\text{Hz}$

**Figure 3.9:** Input signals: $\overrightarrow{x_{IMU}} = [0\,0\,0]\,\text{rad}$ and $\overrightarrow{\dot{x}_{IMU}} = [1.0\,0.2\,0.5] sin(2\pi f t)\,\text{rad}\,\text{s}^{-1}$




**(a)** First addend, where $A = 5, B = 0.1A, w_1 = 5\,\text{rad}\,\text{s}^{-1}$ **(b)** Second addend, where $A = 5, B = 0.1A, w_1 = 5\,\text{rad}\,\text{s}^{-1}$ and $w_2 = 10\,\text{rad}\,\text{s}^{-1}$

**Figure 3.10:** First and second addend of Equation 3.4

Considering for example that $x_1$ is the attitude signal and $x_2$ the attitude rate signal, one has that $A \gg B$. Moreover, assuming that $\omega_1 < \omega_2$, it was possible to study the relationship between the different $t_{k_i}$. Figure 3.10b reports the plot of the second addend obtained considering $A = 5$, $B = 0.1A$, $w_1 = 5\,\mathrm{rad\,s^{-1}}$ and $w_2 = 10\,\mathrm{rad\,s^{-1}}$.

Comparison of the different $t_{k_i}$, which result is reported in Figure 3.11 :

- $t_{k_1}$ vs. $t_{k_2}$

$$\begin{aligned} t_{k_1} &= \frac{k_1 \pi}{\omega_1} \\ t_{k_2} &= \frac{2k_2 \pi}{\omega_1 + \omega_2} \\ t_{k_1} > t_{k_2} &\rightarrow \omega_1 < \frac{\omega_1 + \omega_2}{2} \rightarrow \omega_1 < \omega_2 \end{aligned} \tag{3.5}$$

  Since $\omega_1 < \omega_2$ by hypothesis, $t_{k_1} > t_{k_2}$ always.

- $t_{k_1}$ vs. $t_{k_3}$

$$\begin{aligned} t_{k_1} &= \frac{k_1 \pi}{\omega_1} \\ t_{k_3} &= \frac{\pi(1 + 2k_3)}{\omega_2 - \omega_1} \\ t_{k_1} > t_{k_3} &\rightarrow \omega_1 < \frac{\omega_2}{2} \end{aligned} \tag{3.6}$$

  So, if $\omega_1 < \dfrac{\omega_2}{2}$, $t_{k_1} > t_{k_3}$

- $t_{k_2}$ vs. $t_{k_3}$

$$\begin{aligned} t_{k_2} &= \frac{2k_2 \pi}{\omega_1 + \omega_2} \\ t_{k_3} &= \frac{\pi(1 + 2k_3)}{\omega_2 - \omega_1} \\ t_{k_2} > t_{k_3} &\rightarrow \omega_1 < \frac{\omega_2}{3} \end{aligned} \tag{3.7}$$

  So, if $\omega_1 < \dfrac{\omega_2}{3}$, $t_{k_2} > t_{k_3}$

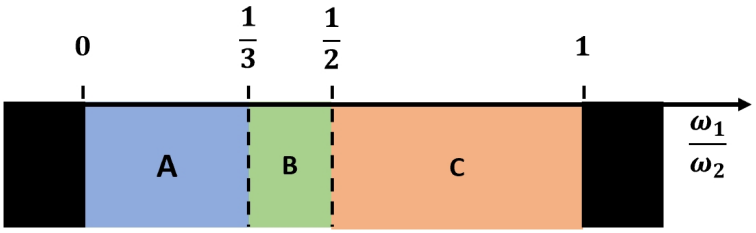



**Figure 3.11:** Schematic representation of the identified zones

Figure 3.12 and Figure 3.13 show the behaviour of the function $y(t)$, depending on the zone considered. More in details:

- Zone C $\rightarrow \frac{1}{2} < \frac{\omega_1}{\omega_2} < 1$

  As visible from Figure 3.12a, the two frequencies are quite similar and, consequently, the blue signal is deformed in a smooth way.

- Zone C/B $\rightarrow \frac{\omega_1}{\omega_2} = 0.5$

  As visible from Figure 3.12b,the behavior is almost the same, but in this case, the blue signal is anti-symmetric with respect to the semi-period of the green one.

- Zone B $\rightarrow \frac{1}{3} < \frac{\omega_1}{\omega_2} < \frac{1}{2}$

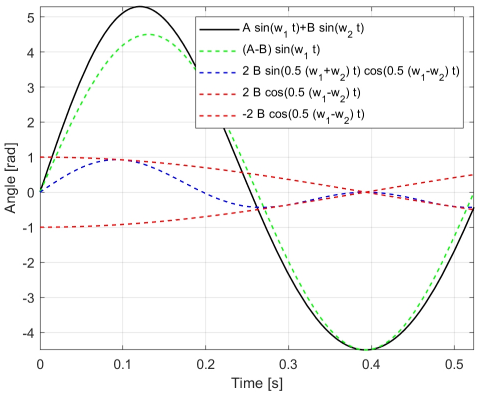
  The symmetry is lost again. The frequency of the blue signal is becoming greater and the deformation of the black signal are more localized.

- Zone A/B $\rightarrow \frac{\omega_1}{\omega_2} = \frac{1}{3}$

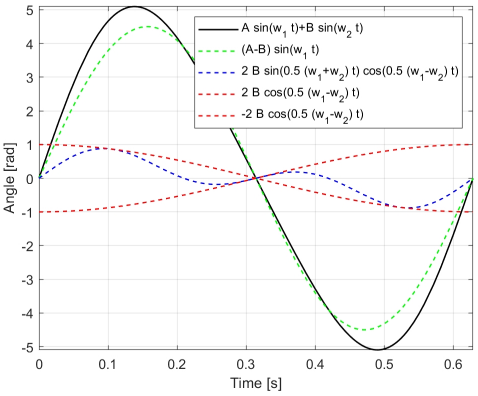
  There is a new symmetry. The blue signal is symmetric with respect to the $\frac{T_1}{4}$ and anti-symmetric with respect to the $\frac{T_1}{2}$.

- Zone A $\rightarrow \frac{\omega_1}{\omega_2} < \frac{1}{3}$

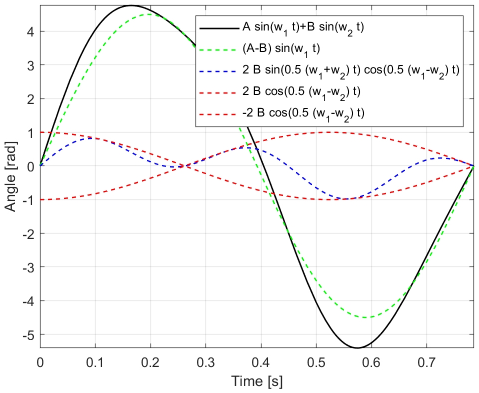
It is evident that the lower the ratio $\frac{\omega_1}{\omega_2}$ is and the more localized the variations produced by the blue signal on the green one are, so the more the effects of $x_2$ on $x_1$ becomes local rather than secular. On the other hand, the effects on $x_1$ are less prominent since $B \ll A$.



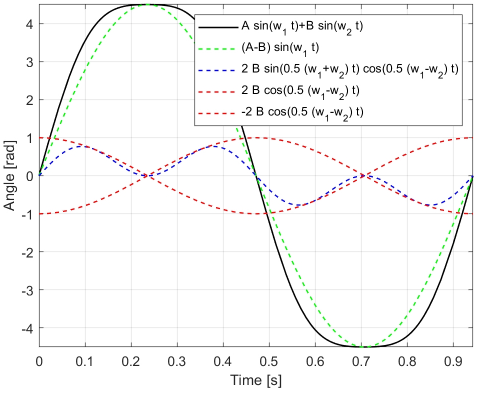

**(a)** Zone C: $\frac{\omega_1}{\omega_2} = 0.6$




**(b)** Zone C/B: $\frac{\omega_1}{\omega_2} = 0.5$

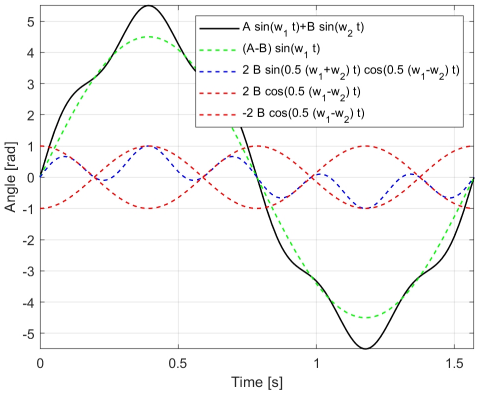



**(c)** Zone B: $\frac{\omega_1}{\omega_2} = 0.4$

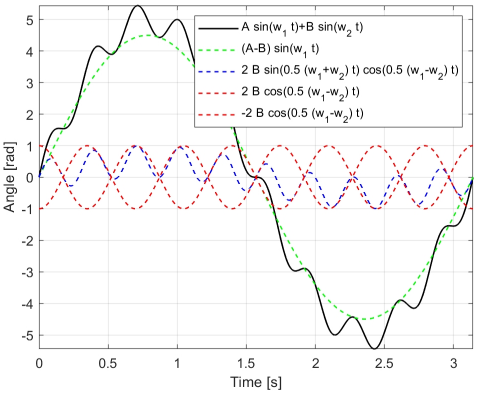



**(d)** Zone A/B: $\frac{\omega_1}{\omega_2} = \frac{1}{3}$

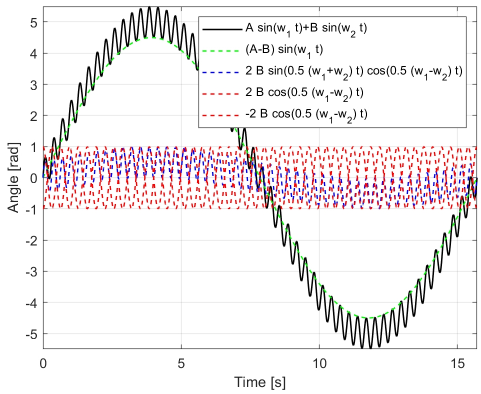
**Figure 3.12:** Zones to C to B




(a) $\frac{\omega_1}{\omega_2} = 0.2$




(b) $\frac{\omega_1}{\omega_2} = 0.1$




(c) $\frac{\omega_1}{\omega_2} = 0.02$

**Figure 3.13:** Zone A: $\frac{\omega_1}{\omega_2} < \frac{1}{3}$

# 4

# Flight chain control

The azimuth signal filtered by the complementary filter analysed in the chapter 3 is used to control the orientation of the gondola around the vertical axis. In addition to the azimuth control of the gondola, it is necessary to control the flight chain angle. In fact, along it there are wires and sensitive components that shall not twist, and therefore an active control that maintains the flight chain angle in a certain range is required. In this chapter, the before-mentioned filter is analysed. At first, the code implemented in the flight software of the FIREBALL mission is studied to retrieve an equivalent Simulink model and, then, the results are discussed.

## 4.1 Flight software analysis

The flight chain control is performed by the function *SrvAngCdv.c* that is composed by several sub-functions, as indicated in the Figure 4.1. In the next lines, a general explanation of the filter is provided.

At the beginning, the function controls if the surveillance is active or inactive. In the first case, the actual angular position of the flight chain is monitored, otherwise it is not. Then, the low pass filter is initialized. The function verifies that the sampling period $T_e > 0$, the surveillance is active, and the condition reported in Equation 4.1 is verified, where $f$ and $f_e$ are respectively the cutoff frequency of the low pass filter and the sampling frequency.

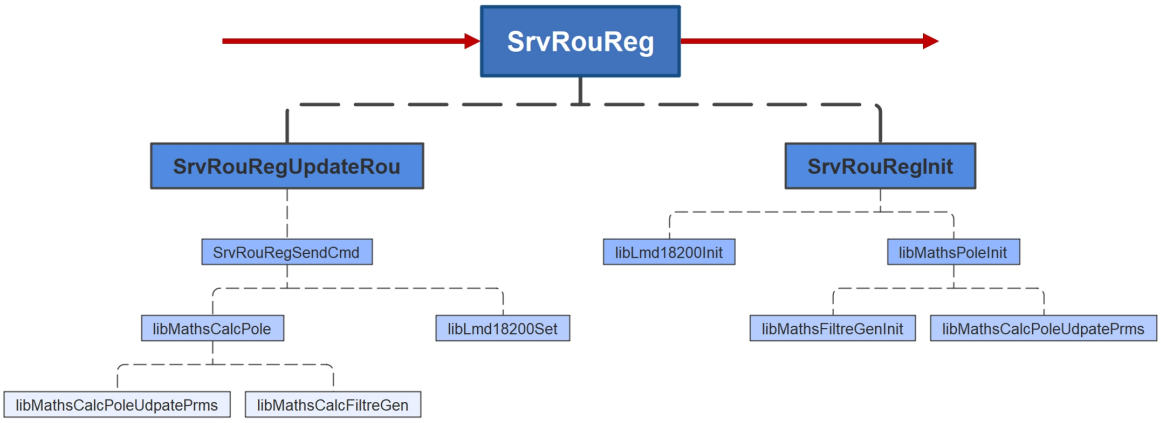



**Figure 4.1:** Schematic representation of the filter functions implemented in the flight software

$$f < \pi \frac{f_e}{2} \tag{4.1}$$

This is indeed to remove frequencies above the half sampling frequency (also called Nyquist frequency), that would otherwise get folded back into frequencies below the Nyquist frequency, corrupting the signal. To remove those frequencies, the cutoff frequency shall be as lower as possible with respect to the sampling frequency, and obligatory smaller than the Nyquist frequency. If these conditions are respected, the low pass filter is initiated and the filter parameters $A_0$ and $A_1$ are computed as reported in Equation 4.2 and Equation 4.3.

Considering the transfer function of a low pass filter and the bilinear transform reported in Equation 2.7, one obtains:

$$H(s) = \frac{V(s)}{U(s)} = \frac{1}{1+s\tau} = \frac{T_e + T_e z}{(T_e - 2\tau) + (T_e + 2\tau)z} = \frac{\frac{T_e}{T_e+2\tau} + \frac{T_e}{T_e+2\tau} z^{-1}}{1 + \frac{T_e - 2\tau}{T_e + 2\tau} z^{-1}}$$

From which:

$$\begin{aligned} V(n) &= U(n)\frac{T_e}{T_e+2\tau} + U(n-1)\frac{T_e}{T_e+2\tau} - V(n-1)\frac{T_e-2\tau}{T_e+2\tau} \\ &= \frac{T_e}{T_e+2\tau}(U(n)+U(n-1)) + \frac{2\tau - T_e}{T_e+2\tau}V(n-1) \end{aligned}$$

And eventually:

$$A_0 = \frac{2\tau - T_e}{T_e + 2\tau} \tag{4.2}$$

$$A_1 = \frac{T_e}{T_e + 2\tau} \tag{4.3}$$

The input signal to the filter is given by the angular velocity of the flight chain times the time constant of the low pass filter. In fact, as mentioned later, in reality a high pass filter should be implemented, but the $s$ at the denominator of the integrator operator would be simplified by the one a the numerator of the high pass filter and, therefore, what remains is a $\tau$ that multiplies a low pass filter. The input signal is filtered and the output is monitored to determine whether the angle of the flight chain exceeds the range.

## 4.2 New implementation

In this case, there was not a previous model implemented in the simulator. From the flight software, two main parts could be identified:

1. filtering of the flight chain angle
2. comparison between the flight chain angle and the threshold value

Considering the needs of the DICOS mission, a threshold value of $360\,^\circ$ and a $\tau = 150\,\mathrm{s}$ were considered. To understand precisely how to integrate the model in the simulator, it is useful to remember that the pivot, indicated in Figure 4.2, has to satisfy two objectives:

1. point the gondola towards the desired direction, meaning that the azimuth of the gondola shall be equal to the azimuth objective
2. keep the velocity of the flight chain at zero for stability reasons

Therefore, the torque generated by the pivot is given by the sum (with sign) of the two torques that satisfy the two objectives respectively as reported in Equation 4.4, where $K_{D,\theta}$, $K_{P,\theta}$, $K_{P,v}$, $K_{I,v}$, $\theta_z$, $\theta_c$ and $\theta_{FC}$ are respectively the derivative gain for azimuth orientation, proportional gain for azimuth orientation, proportional gain for FC damping, integral gain for FC damping, azimuth of the gondola, azimuth objective and azimuth of the FC.

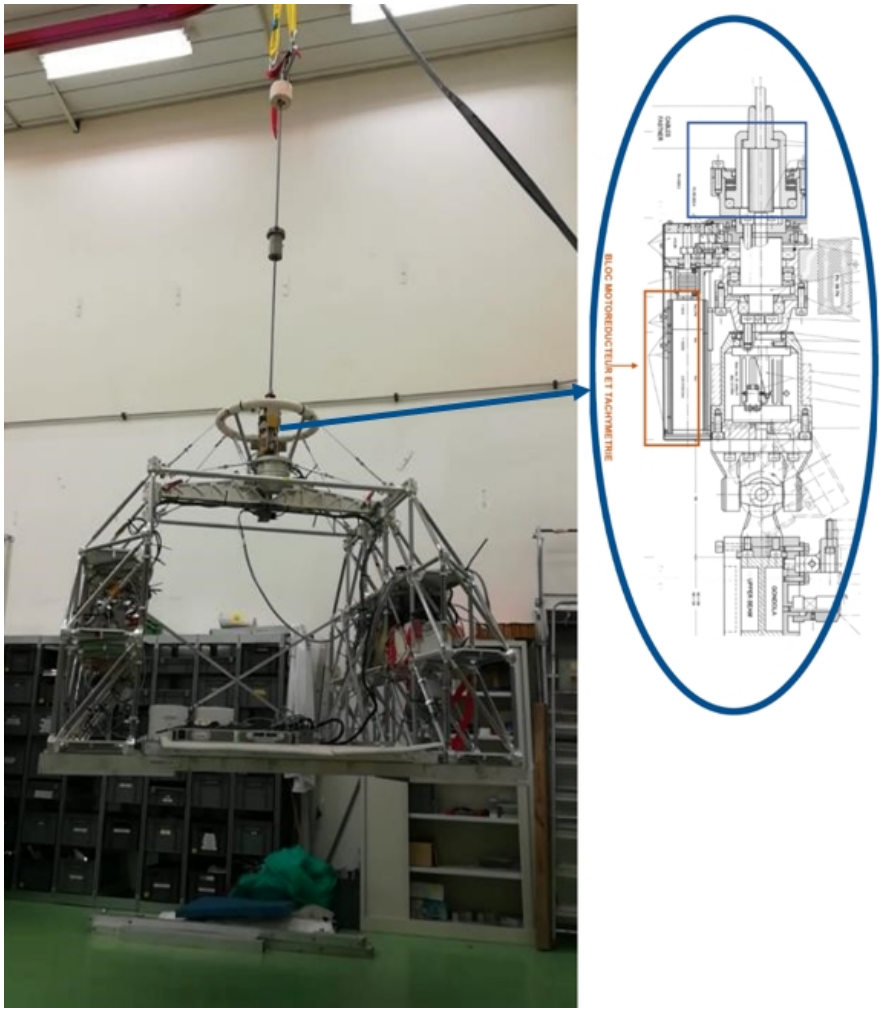



**Figure 4.2:** Photo of the gondola Carmencita and detail of the pivot

$$\begin{aligned} C_{mot} &= C_{Azimuthcontrol} - C_{FCdamping} \\ &= (K_{D,\theta}(\dot{\theta_z} - \dot{\theta_c}) + K_{P,\theta}(\theta_z - \theta_c)) - (K_{P,v}\dot{\theta_{FC}} + K_{I,v}\theta_{FC}) \end{aligned} \tag{4.4}$$

The choice of using a proportional and derivative solution for $C_{Azimuthcontrol}$ and a proportional and integral solution for $C_{FCdamping}$ was based on considerations that were out of the scope of this study, but that aimed to satisfy the objectives while opposing to the external disturbances without affecting the stability of the system.

In the azimuth pointing loop, it is possible to identify the following actuators and sensors,

relevant for this study:

- a pivot, which is a DC motor that could be connected to a gear box in some applications and that connects the gondola to the rest of the flight chain.
- a tachometer, which is a second motor that generates a tension proportional to the rotation speed. It is mounted inside the pivot. Its voltage is then elaborated by the MPA unit to determine the rotation speed of the flight chain. The speedometer measures the relative velocity between the gondola and the flight chain $\dot{\theta_{tacho}}$, while the IMUo measures the $\dot{\theta_z}$. Therefore, the Equation 4.5 can be retrieved:

$$\dot{\theta_{tacho}} = \dot{\theta_{FC}} - \dot{\theta_z} \rightarrow \dot{\theta_{FC}} = \dot{\theta_{tacho}} + \dot{\theta_z} \tag{4.5}$$

  A schematic representation is reported in Figure 4.3.

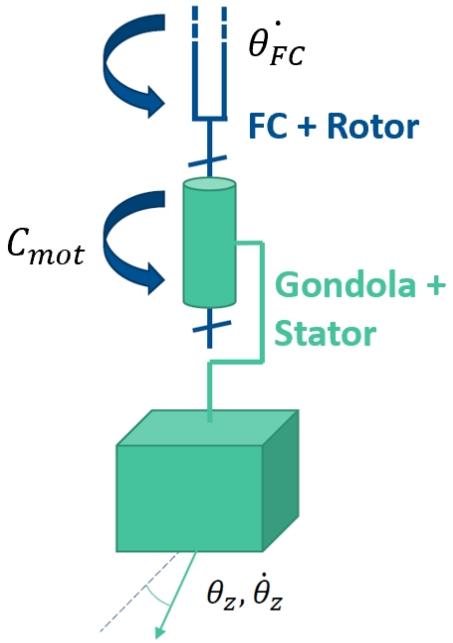



**Figure 4.3:** Schematic representation of the pivot

The low pass filter studied in the previous section is used to filter the $\dot{\theta_{FC}}$ to obtain, as output, the angle of the flight chain $\theta_{FC}$, as if the input signal was filtered by a high pass filter. Figure 4.4 summarizes this process, while Figure 4.5 presents the Simulink models that have been created and implemented in the simulator and the equivalent digital filter.

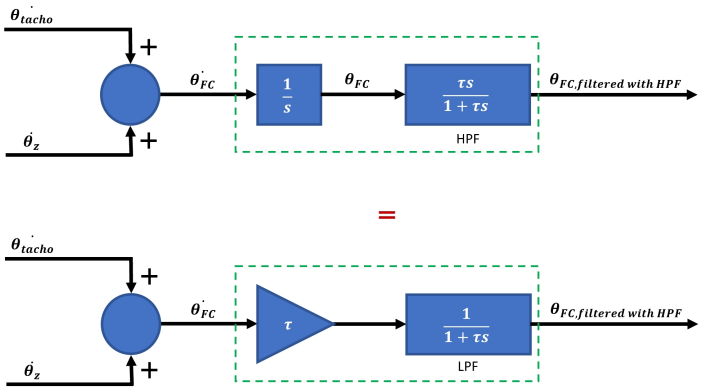



**Figure 4.4:** Schematic representation of the filter

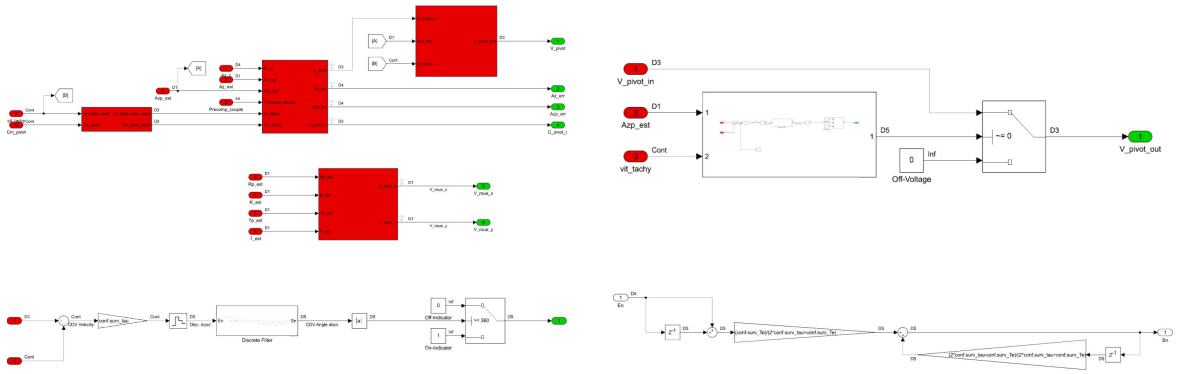



**Figure 4.5:** Simulink models implemented in the simulator

## 4.3 Discussion of the results

Figure 4.7 reports the plots obtained after running the simulation. It is possible to notice that the flight chain velocity has a pseudo-sinusoidal behaviour that is particularly disturbed at the beginning of the simulation. After approximately 10 s, the disturbances are eliminated and the profile converges to zero as wanted and mentioned before. Moreover, it is also possible to notice that the velocity magnitude is significantly smaller than previously hypothesize and that the angle of the flight chain stabilizes to a value which is not zero. In fact, as mentioned before, the two 1st order high pass filters, make the system lose the capability of controlling the flight chain direction. That is why it may occur situations in which the angle exceeds the limit of 360 °. Moreover, it is

possible to notice that the angle signal is relatively smaller than the threshold value. In fact, the voltage applied to the pivot is never forced to zero, since the threshold angle is never reached.

To check what happens if the flight chain has an angle greater than 360°, its value was manually forced to exceed the range after 2 s from the beginning of the simulation. As expected, the tension to the pivot became zero and the system could not reach the azimuth objective. As visible from Figure 4.6, since no automatic action that could reactivate the pivot was implemented in the current model, the motor remains off until the end of the simulation. In the next versions, different actions could be done on the pivot when the FC angle exceeds the range.

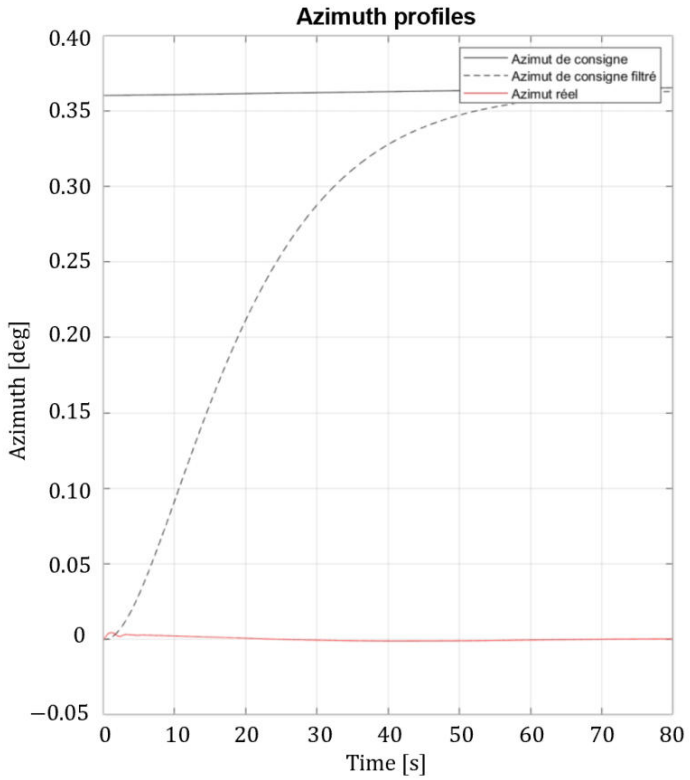
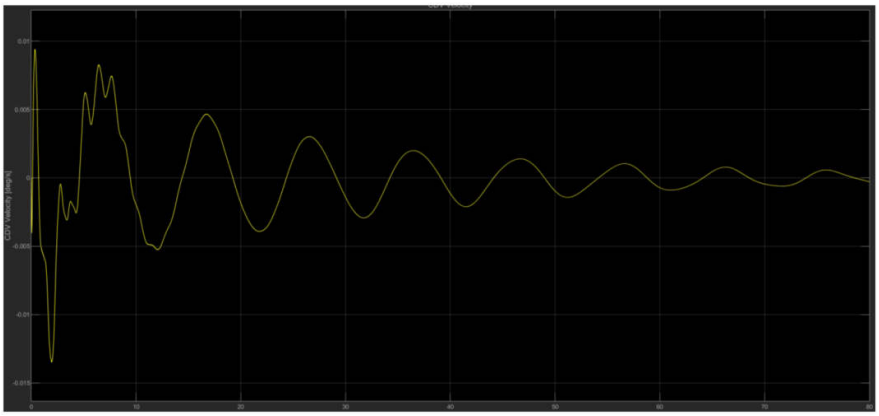
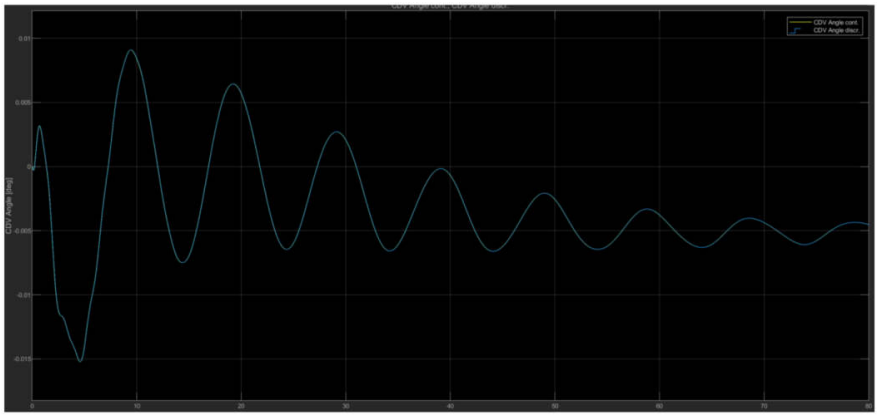


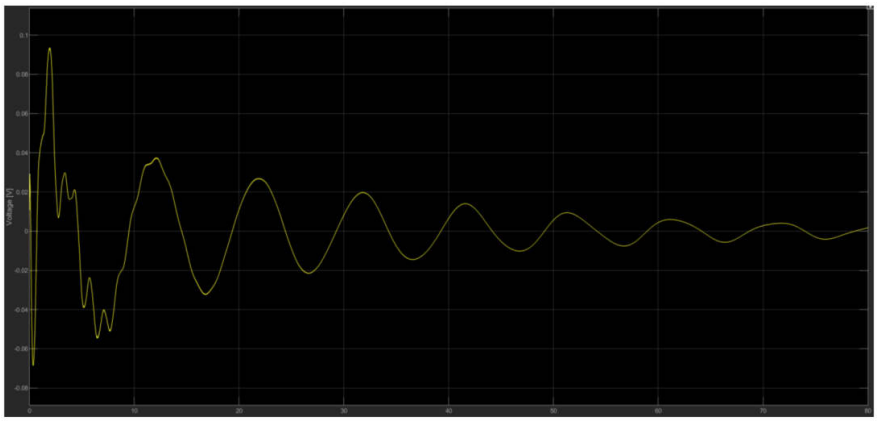

**Figure 4.6:** Azimuth profile



**(a)** $\dot{\theta_{FC}}$




**(b)** $\theta_{FC}$



**(c)** Pivot suply voltage

**Figure 4.7:** Simulation results

# 5

# Reaction wheels model

Reaction wheels are the actuators responsible for the azimuth control and pendulum motion damping described in the previous chapters. Considering their crucial role, an alternative model to the one previously implemented in the simulator is investigated, considering different possible friction models.

## 5.1 Previous implementation

The RWs considered in this study are those mounted on the gondola and having the spinning axis aligned respectively with the $X_{NA}$ and $Y_{NA}$. Since they are used to compensate the oscillations of the gondola around the homonyms axes, their velocity profile crosses the 0 often and therefore friction effects, such as stiction, should be kept into account to obtain a good model.

Figure 5.1 reports the model previously implemented in the simulator that was tested to verify if there were margins of improvement. To do so, a dedicated model in a separated Simulink file was created and a ramp or sinusoidal input function was considered as $\theta_{objective}$. Since the RW model required an electrical tension as input, a PI controller was implemented to convert the error between $\theta_{objective}$ and $\theta_{real}$ into a tension signal. To tune the PI parameters, a step function was used, as shown from Figure 5.2.

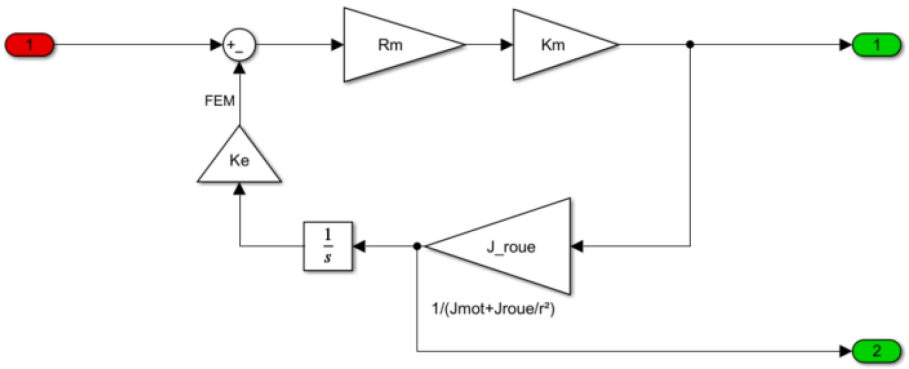



**Figure 5.1:** Previous RW model implemented in the simulator

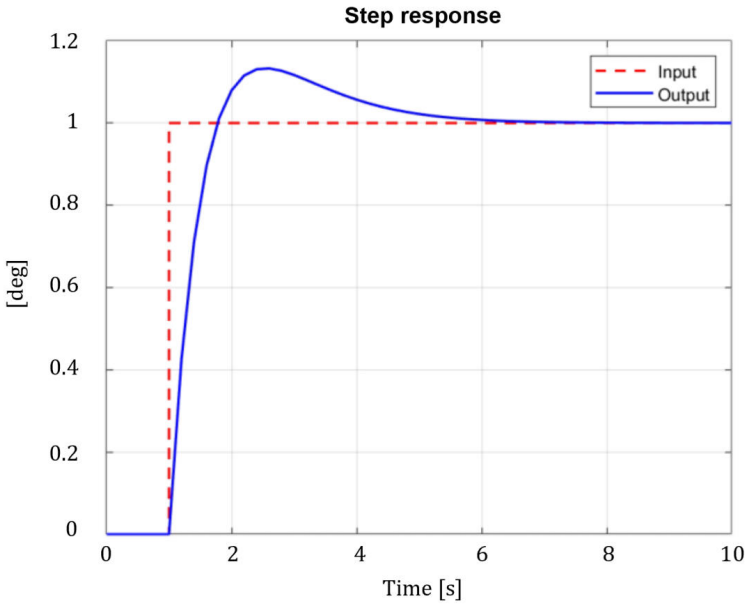



**Figure 5.2:** Response to a step function input

By comparing the output and input signals of the model, it was possible to assert that there was not any friction model implemented in this the previous RW model. In fact the response of the system has a certain time delay due to the motor inertia, but does not present any plateau or irregular behaviors.

## 5.2 New implementation

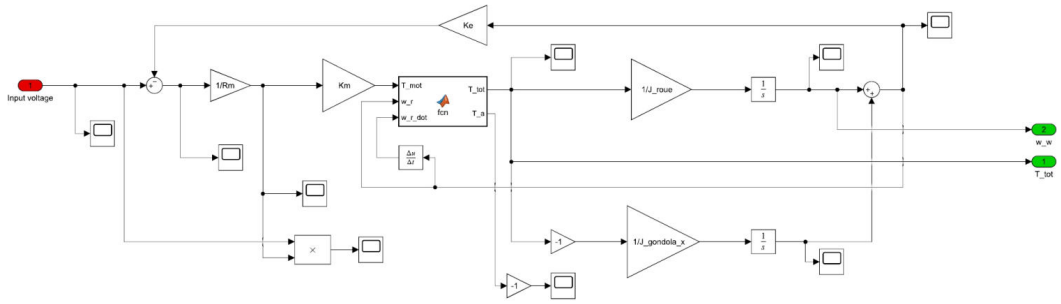



**Figure 5.3:** New RW model

Since, as mentioned at the beginning of the chapter, friction is very important for this kind of application, the model reported in Figure 5.3 was created.

The total torque generated by the wheel (which calculations is explained later) acts on the gondola and on the wheel itself, but with opposite sign. Considering the inertia of the wheel and the estimated one of the gondola, it is possible to compute both the angular acceleration of the wheel and gondola. By integrating those signals, one obtains the angular velocity of the wheel and of the gondola, which is much lower since its inertia is considerably higher. By summing these signals, one obtains the relative angular velocity that is used to compute the torque to the motor and the friction torque.

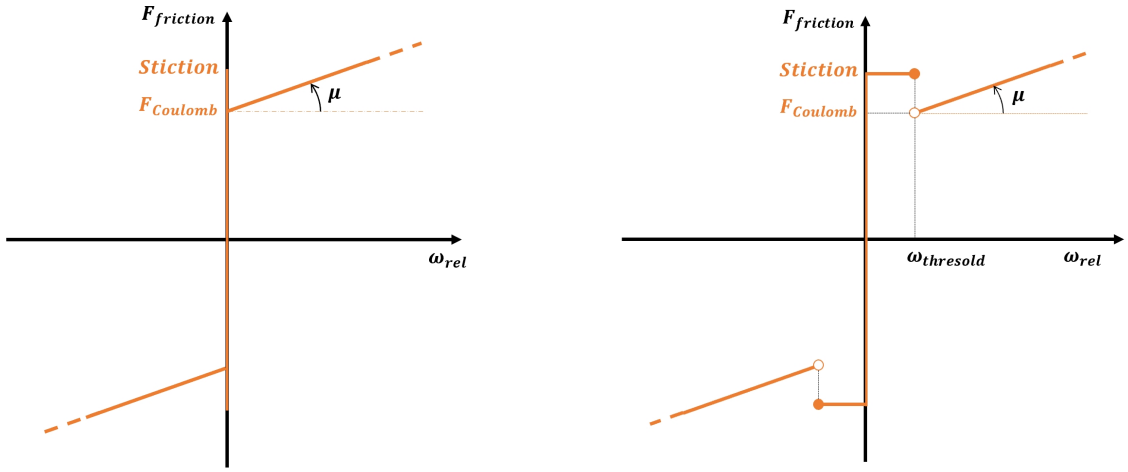
## 5.3 Friction models

Several friction models are available in the literature and each of them present advantages and disadvantages. In this section, two possible friction models and their implementation are analyzed.

### 5.3.1 First model

As reported in Figure 5.4a this model considers the stiction, that is the friction force at zero velocity, the Coulomb friction, which value is constant at any velocity, and the viscous friction that increases linearly with the relative velocity.

To avoid numerical errors during the simulation, a threshold velocity $w_{threshold} \neq 0$ was used as upper limit of the velocity range in which one can consider the stiction presence. Moreover,



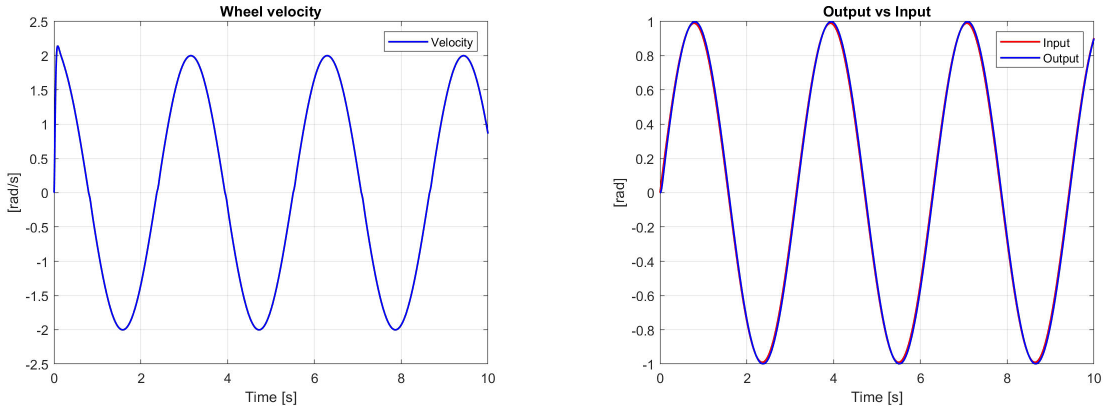

**(a)** Theoretical model

**(b)** Implemented model

**Figure 5.4:** Representation of the first friction model

to avoid issues in the determination of $sign(w_{rel})$ when $w_{rel} \ll 1$, a solution that considers the numerical precision of the simulation $\varepsilon$ is used. Figure 5.4b shows a schematic representation of the model.

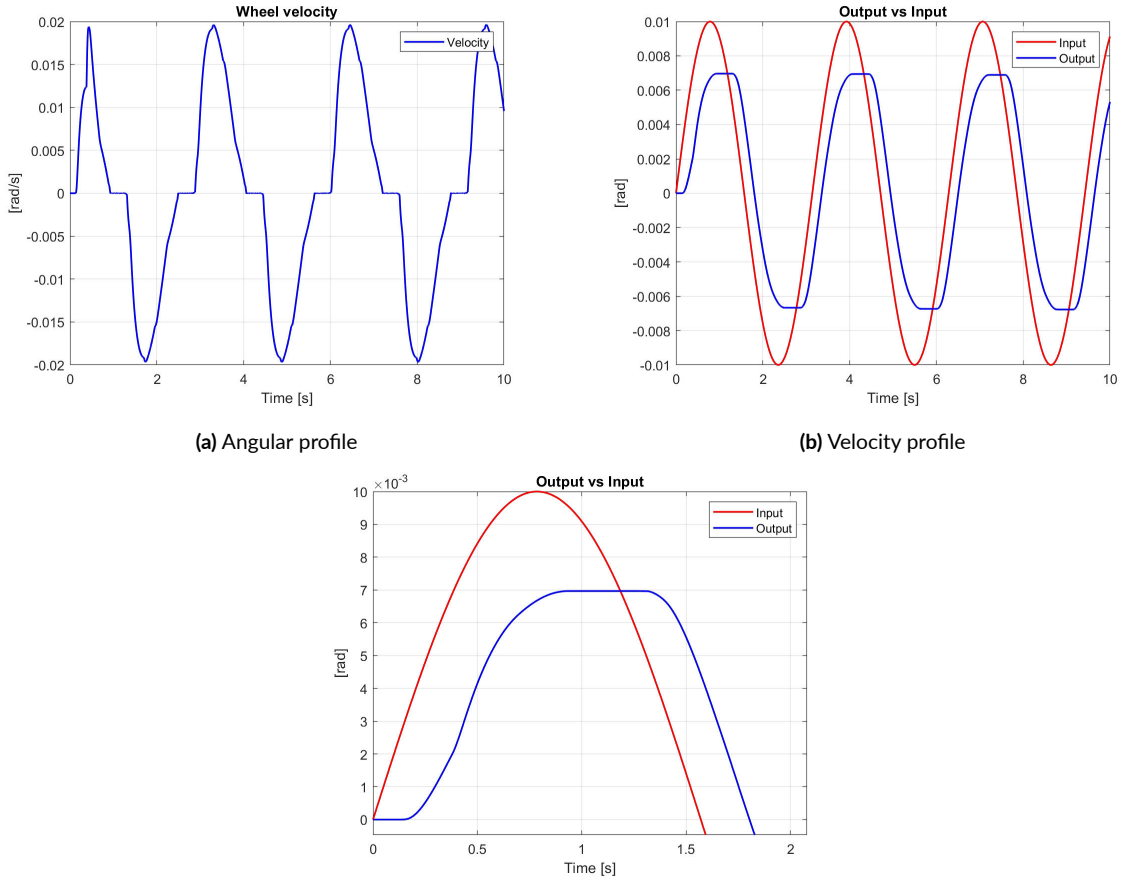
The parameters used in this model have been retrieved from the wheels datasheet and from papers in which experiments have been done using similar wheels. However, for a better characterization, practical experiments may be required to correctly represent the actual behavior of our wheels. Nevertheless, one can see in Figure 5.5 the simulation results for a sine wave input and overestimated friction coefficients.




**(a)** RW relative velocity

**(b)** RW angular profile

**Figure 5.6:** Output signals for realistic friction coefficients




**(a)** Angular profile

**(b)** Velocity profile

**(c)** Zoom in the velocity profile

**Figure 5.5:** Output signals for overestimated friction coefficients

The motor rotates effectively when the applied torque is bigger than the total friction torque. Therefore, the $\omega_{rel}$ profile presents some plateaus soon after the zero crossing. Clearly, this phenomenon becomes more problematic if $\dot{\theta}_{objective}$ crosses 0 often.

To select realistic values for the friction coefficients, the RW datasheet (T-2215-C motor by Kollmorgen) was used, from which the following information can be extrapolated:

- Static friction $T_f = 0.0233\ \mathrm{N\,m}$
- No load speed $\omega_{NL} = 54\ \mathrm{rad\,s^{-1}}$
- Peak electrical power $P_p = 41\ \mathrm{W}$
- Peak torque $T_p = 0.76248\ \mathrm{N\,m}$

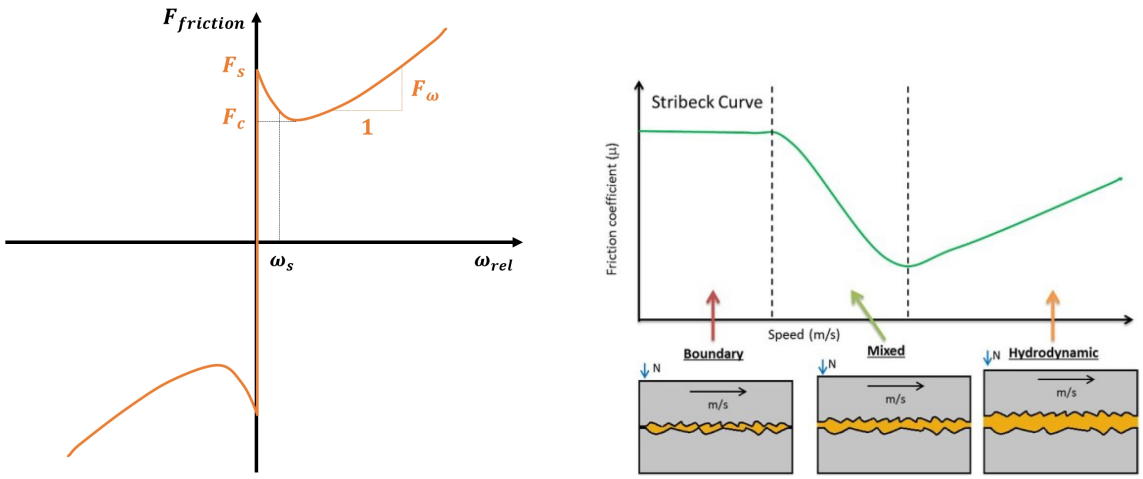
By reading from the datasheet that the static friction is the sum of the brush-commutator friction and the magnetic friction (that includes both the cogging torque and the hysteresis drag), and by knowing its order of magnitude, the different drag coefficients in the model have been tuned to respect the before-mentioned values.

Figure 5.6 shows the relative velocity of the wheel and the corresponding angular profile.

In conclusion, one could say that the obtained results are realistic and that the model could be implemented in the simulator as representative of a possible friction model.

### 5.3.2 Second model

Even if the previous model, once correctly tuned, represents sufficiently well the real behavior of the RW, it does not represent the reality in case of lubricated surfaces and/or small motion. Generally, one could decide to create or use a model that tries to capture as many friction characteristics as possible. However, this does not necessary mean it is the best solution, since it may involve some side effects such as non-linearities, high computational costs, etc. A better solution is to use a model sufficiently precise to represent the most important aspects of the system under study. The model implemented in this chapeter is the so-called LuGre model, that incorporates the Stribeck model and the pre-sliding behavior,considering plastic deformation for the asperities.




**(a)** Friction profile

**(b)** Friction coefficient

**Figure 5.7:** Representation of the second friction model

As shown in Figure 5.7, the Stribeck model consists of three lubrication regions:

1. The boundary lubrication: the surfaces are in direct contact and the load is supported mainly by the surfaces asperities.

2. The mixed lubrication: both the asperities and the liquid lubricant support the load. The surfaces slide on a thin layer of liquid lubricant (almost) without touching each other.

3. The hydrodynamic lubrication: the load is supported mainly by hydrodynamic pressure. The viscous friction becomes important since the velocity is high enough.

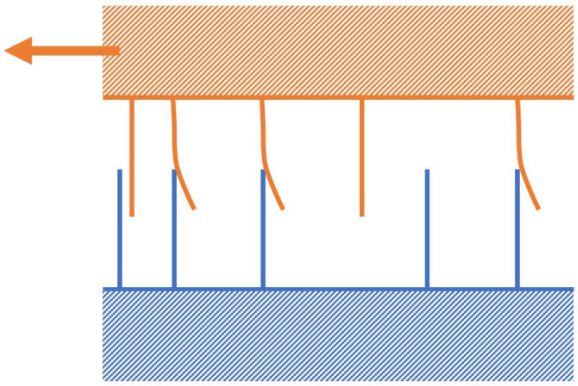


**Figure 5.8:** Schematic representation of the two surfaces touching in the asperities

By focusing more on the first region and doing a zoom in, one would see the two surfaces touching each other in the asperities, as schematically shown in Figure 5.8 that was reported from [3] and [4]. If the external force/torque is not zero, but not sufficiently high to overcome the static friction, the asperities will experience deformation and the two contact surfaces will face a pre-sliding motion. Only when the external force/torque becomes high enough, the two surfaces start sliding one on the other. From [3] it is possible to extrapolate the Equation 5.1.

$$\begin{cases} F = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{x} \\ \dot{z} = \dot{x} - \sigma_0 \dfrac{|\dot{x}|}{g(\dot{x})} z \\ g(\dot{x}) = F_c + (F_s - F_c) \exp\left(-|\dfrac{\dot{x}}{\omega_s}|\right) \end{cases} \tag{5.1}$$

Where

- $F$ is the friction force
- $\sigma_0$ is the bristle stiffness

- $\sigma_1$ is the bristle damping coefficient
- $\sigma_2$ is the viscous friction coefficient
- $z$ is the average deflection of the bristle
- $\dot{x}$ is the sliding velocity
- $g(\dot{x})$ is a function proposed to describe the Stribeck effect
- $F_s$ is the static friction
- $F_c$ is the coulomb friction
- $\omega_s$ is the Stribeck velocity. It is empirically determined and generally has a value $\in [10^{-5}, 10^{-2}]\,\mathrm{m\,s^{-1}}$
- $j$ is the Stribeck shape factor. Its value is usually 2

Figure 5.9 shows the Simulink implementation.

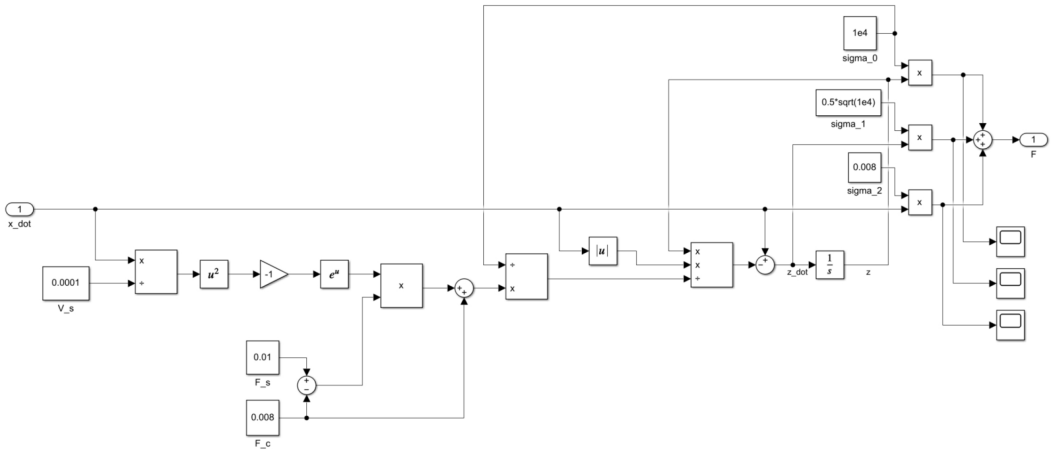



**Figure 5.9:** Simulink model of the second friction model

LuGre model is based on a group of complex nonlinear equations with six parameters: four static parameters ($F_s$, $F_c$, $\omega_s$, $\sigma_2$) and two dynamic parameters ($\sigma_0$, $\sigma_1$). These parameters are difficult to estimate since they are coupled and they cannot be directly measured. However, they could be estimated empirically. Static parameters could be obtained through a series of experiments, in which the velocity of the motor is kept constant in a region where it does not experience the stick-slip behavior. By monitoring the torque, it is possible to obtain the empirical

Stribeck curve (see the next paragraphs) that will be fitted by tuning the static parameters of the LuGre model. In the pre-sliding process, the friction is mainly due to $\sigma_0 z$ and $\sigma_1 \dot{z}$; therefore, in order to estimate the dynamic parameters, one can measure the angle output of the motor and compare it with the theoretical one obtained for a combination of $\sigma_0$ and $\sigma_1$. Since during the traineeship this data was not available and it was not possible to conduct any experiment, values that have been used in similar motors and and in the literature have been used, checking again that the results respected the datasheet of the motor.

## Steady State analysis

In the steady state, when the sliding velocity is kept constant, the bristle deformation is constant too and, therefore, $\dot{z} = 0$. From the second equation of Equation 5.1, one can obtain:

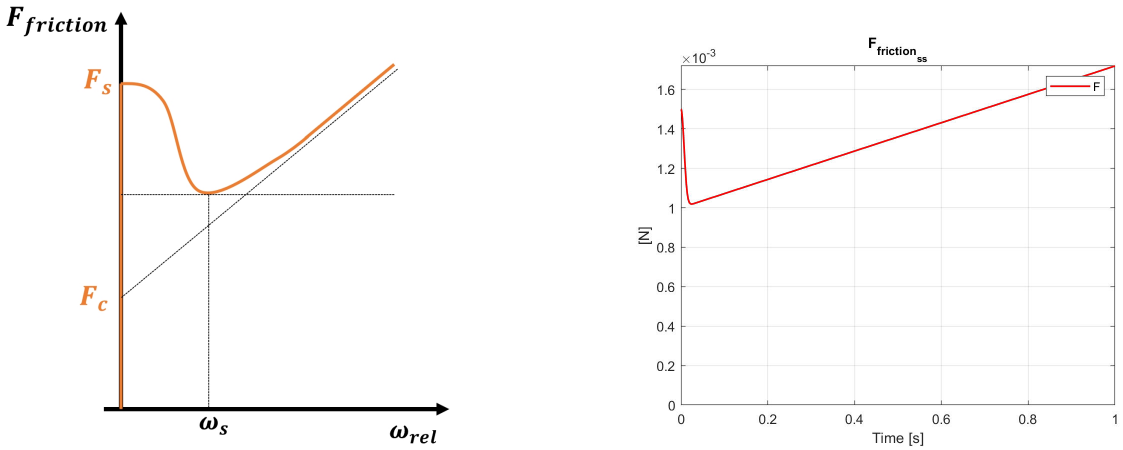
$$\dot{x} - \frac{|\dot{x}|}{g(\dot{x})} z = 0 \rightarrow z_{ss} = g(\dot{x}) sign(\dot{x})$$

Therefore, the friction force becomes:

$$F_{ss} = \sigma_0 z_{ss} + \sigma_2 \dot{x} = F_c sign(\dot{x}) + (F_s - F_c) exp\left(-\frac{\dot{x}}{\omega_s}\right)^2 sign(\dot{x}) + \sigma_2 \dot{x} \tag{5.2}$$

Which corresponds to the friction force of the Stribeck's model.

Figure 5.10 reports a comparison between the theoretical and empirical curves.




**(a)** Theoretical curve

**(b)** Curve obtained through Simulink

**Figure 5.10:** Comparison between the theoretical steady state friction profile and the one retrieved from simulations

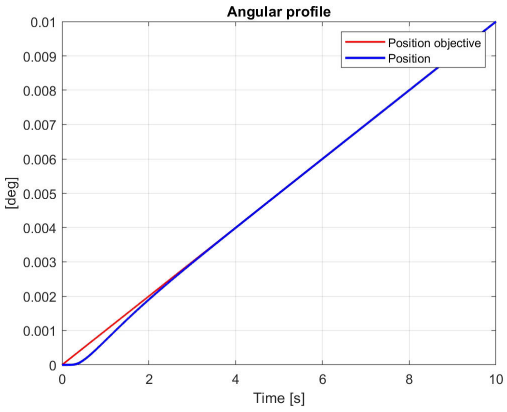
Studying the steady state dynamics was useful to understand the influence of the static parameters. Basically, once one defines $\sigma_2$, $F_s$ and $F_c$, which are respectively the steepness of the straight line, the force at $\dot{x} = 0$ and the point of intersection between the straight line extension and the $y$-axis, the only parameter that remains to be defined is the $\omega_s$. The lower this parameter is, the more the position of the minimum point of $F_{ss}$ moves to the bottom-left corner.

## Dynamic analysis

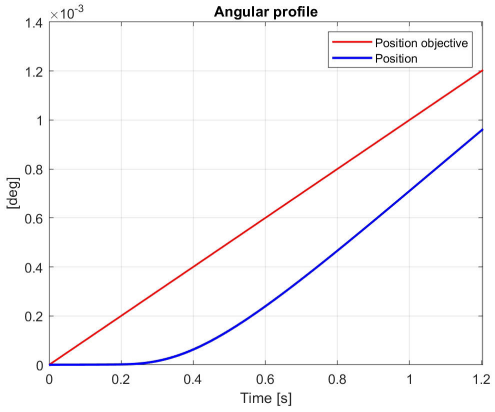
As also mentioned in [3], when the sliding velocity is not constant, the behavior of the system in terms of pre-sliding displacement and break-away force will depend on the shape of the function $g(\dot{x})$ and on the whole nonlinear differential equations (rather than on a single parameter). Different typical cases have been studied in the before mentioned paper. In the this section, the considerations done after investigating the wheel dynamics when it has to move linearly with a constant speed of $10^{-3}\,\mathrm{rad\,s^{-1}}$ have been reported.

Considering a linear angular profile allowed to understand more easily the behavior of the system. Figure 5.11 reports the obtained results. From Figure 5.11a and Figure 5.11b, it is possible to see that the surfaces seem to start sliding after almost $0.2\,\mathrm{s}$. In fact, the wheel velocity in the same region is not zero, but positive and increasing, as shown in Figure 5.11c and Figure 5.11d. In fact, the error ($\epsilon = \theta_{objective} - \theta$) is increasing in time and, therefore, the applied torque does the same. This deforms the surface asperities more and more causing a micro-motion process. As shown in Figure 5.11f, the friction torque increases in time, since the average deformation of the bristles keeps increasing. Thanks to this study, it was possible to understand more the influence of some parameters:

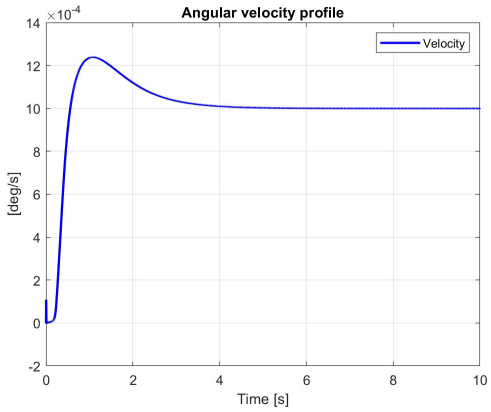
- $\sigma_0 \rightarrow$ the higher it is, the higher the friction torque is, since for the same deformation $z$, the material generates a higher resistance
- $\sigma_1 \rightarrow$ the higher it is, the bigger the resistance against rapid $\theta$ changes is
- $\sigma_2 \rightarrow$ does not have a big influence on this region, since it is linked to the sliding velocity, which is low
- $F_s \rightarrow$ defines the breaking force/torque, after which the two surfaces start sliding (Force value at $1.6\,\mathrm{s}$)
- The other parameters respect what was already said



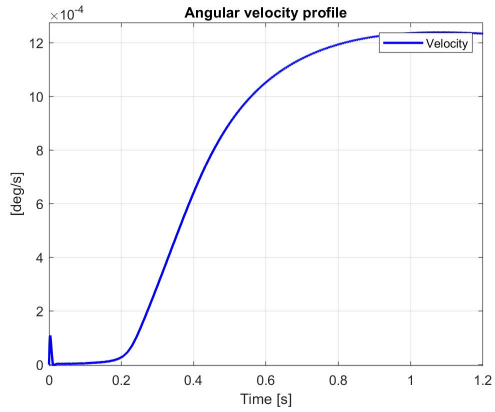

**(a)** Angular profile



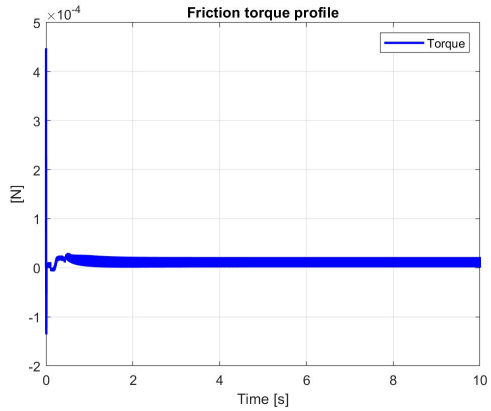

**(b)** Zoom in in the angular profile



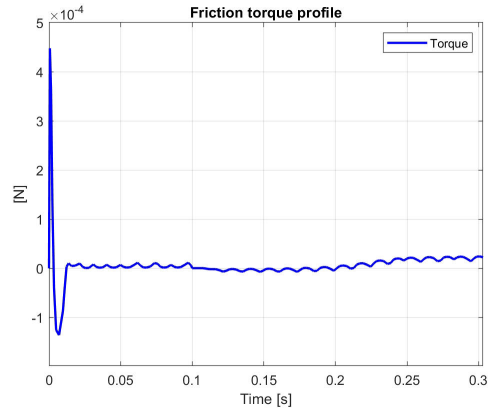

**(c)** Angular velocity profile




**(d)** Zoom in angular velocity profile




**(e)** Friction torque profile

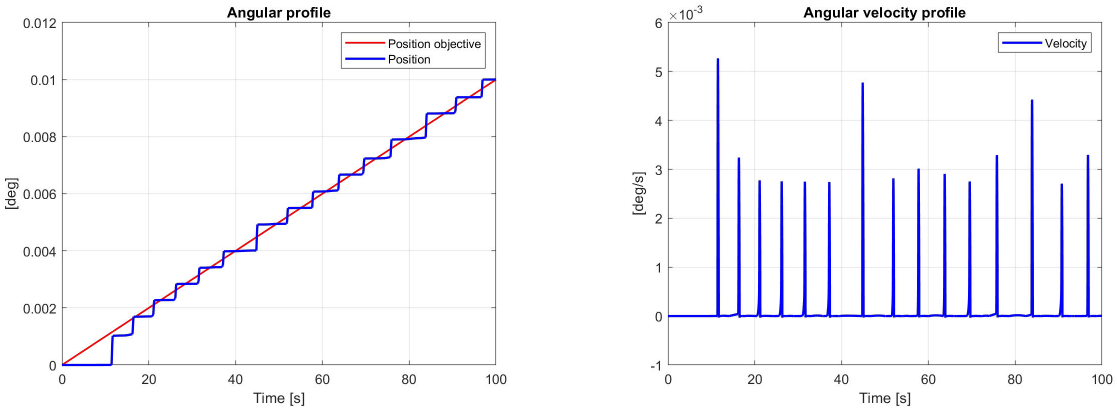



**(f)** Zoom in friction torque profile

**Figure 5.11:** Results for a constant angular velocity $\dot{\theta}_{objective} = 10^{(-3)}\ \mathrm{rad\,s^{-1}}$

Once the applied torque reaches the $F_s$, the surfaces start sliding one over the other and the

motor starts moving. The friction drops and, for a certain $\Delta t$, the applied torque is much higher than the friction torque. This accelerates the motor and produces the step that can be seen in the velocity profile. Since there is a sudden variation in the bristle deformation, $\sigma_1$ has a big influence on the behavior of the system in this region of the plot. After this transitory, there is the viscous region, in which $\sigma_2$ has a big influence and the viscous friction is prominent. Once the system reaches the steady state, the wheel speed and the friction torque remain constant, as shown in Figure 5.11d and Figure 5.11e respectively. This study also permitted to highlight the great influence that the input velocity has on the wheel dynamics. More precisely, the lower the speed of the wheel is, the worse it is since the wheel operates in the stick-slip region, where there is stiction and therefore a discontinuous motion. In fact, by considering an input velocity of $10^{-4}$ $\mathrm{rad\,s^{-1}}$, one obtains the angular profile reported in Figure 5.12a.




**(a)** Angular profile

**(b)** Velocity profile

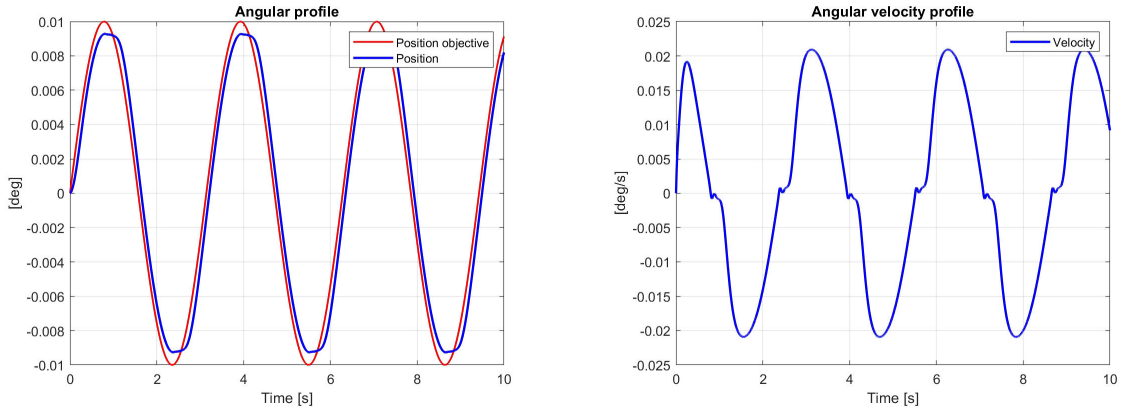
**Figure 5.12:** Results for a constant angular velocity $\dot{\theta}_{objective} = 10^{-4}\ \mathrm{rad\,s^{-1}}$

By looking at the friction profile in the first 8 s, it is possible to recognize the same behavior described before: micro motion until the input torque reaches the breaking point and, then, viscous motion. Soon after the wheel starts sliding, the velocity reaches a value too high and, therefore, the PID controller regulates the tension in order to reduce the error. The applied torque decreases and the velocity of the wheel goes to zero, but in this way the wheel enters again the stick region, and cannot leave it until the applied torque is again sufficiently high. This requires the error to be high enough and, therefore, a considerable amount of time. As a result, the velocity profile is the one reported in Figure 5.12b.

A possible way to prevent this behavior, in applications that requires small velocities, is to better tune the PID controller (for example increasing the proportional gain), or by choosing a motor

capable of producing a higher torque with the same current.

The same considerations can be done also for a sinusoidal velocity, as input, with amplitude of $10^{-2}\,\mathrm{rad\,s^{-1}}$. The Figure 5.13 gathers the obtained results.




**(a)** Angular profile

**(b)** Velocity profile

**Figure 5.13:** Results for a sinusoidal angular velocity with amplitude $A = 10^{(-2)}\,\mathrm{rad\,s^{-1}}$

In conclusion, one can say that the LuGre's model seems to be accurate, if correctly tuned, and that it can take into account the pre-sliding effect, which is important if the velocity is low (it makes physically sense, since the motor is departing from the stick region very slowly). On the other hand, the model is not linear and consequently:

- changes of the output are not proportional to the changes of the input
- the solution strongly depends on initial and boundary conditions
- the non-linear dynamical equations are difficult to solve and this increases the computational cost and time

In addition, the different parameters have to be tuned properly, since they heavily influence the system response. To do so, one could do some tests on the RWs. During the traineeship, there was not this possibility and information from different tests and reports have been used to compare the transfer functions of the two different models.

One of the conducted studies tried to estimate the electrical and mechanical frequencies of the RW under study by interpolating the theoretical transfer function, reported in Equation 5.3, with the empirical one, retrieved as showed in Figure 5.14 and Equation 5.6. Results showed that $f_{electric} = 41$ Hz and $f_{mechanic} = 598$ Hz at $+50°\mathrm{C}$. These two values have been used to deduce the inductance of the RW $L$ and the ratio $\mathrm{J/K_m}$ as indicated below.

$$H_{theoretic}(\omega, f_{mechanic}, f_{electric}) = \frac{G_{static} i\omega}{\left(1 + \frac{i\omega}{2\pi f_{electric}}\right)\left(1 + \frac{i\omega}{2\pi f_{mechanic}}\right)} \tag{5.3}$$

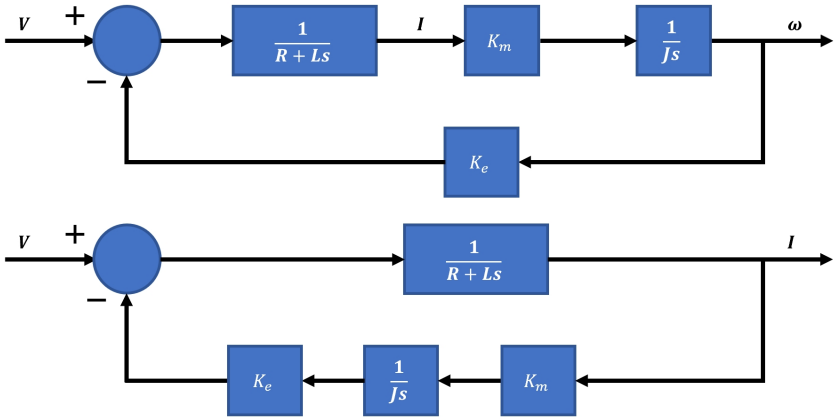



**Figure 5.14:** Schematic representation of the transfer function $H = \frac{I}{V}$

$$\begin{aligned} H = \frac{I}{V} &= \frac{1 + \frac{1}{R + Ls}}{1 + \frac{1}{R + Ls}\frac{K_m K_e}{Js}} \\ &= \frac{Js}{K_m K_e + JRs + JLs^2} \\ &= \frac{\frac{J}{K_m K_e}s}{1 + \frac{JR}{K_m K_e}s\frac{JL}{K_m K_e}s^2} \end{aligned} \tag{5.4}$$

And, by considering that normally $\Delta > 0$ and $\tau_{electric} \ll \tau_{mechanic}$, one can obtain:

$$\frac{I}{V} = \frac{\frac{J}{K_m K_e}s}{\left(1 + \frac{L}{R}s\right)\left(1 + \frac{RJ}{K_m K_e}s\right)} \tag{5.5}$$

Therefore,

$$
\begin{aligned}
f_{electric} &= \frac{R}{2\pi L} \rightarrow L = \frac{R}{2\pi f_{electric}} = 0.0036\,\mathrm{H} \\
f_{mechanic} &= \frac{K_m K_e}{2\pi R J} \rightarrow \frac{J}{K_m} = \frac{K_e}{2\pi R f_{mechanic}} = 0.00012\,\mathrm{kg\,m^2\,A\,N^{-1}\,m^{-1}}
\end{aligned}
\tag{5.6}
$$

Although the first result is realistic, the second one is too small. In fact, by considering $J = 7.3610^3\ \mathrm{kg\,m^2}$ from the datasheet, one would obtain that $K_m = 59.97\mathrm{N\,m\,A^{-1}} \gg K_e$. Since it seemed there was an inconsistency in the data used to conduct the previous study and there was no possibility of further investigation, the study was interrupted here.

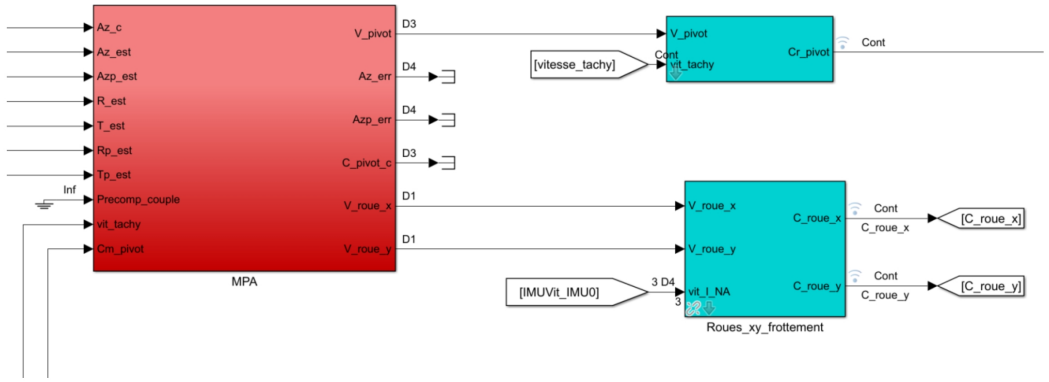
## 5.4 New implementation

As shown in Figure 5.15a and Figure 5.15b, the RW model was implemented in the simulator. As mentioned before, the different parameters reported below need to be correctly and accurately tuned to obtain realistic and consistent results.

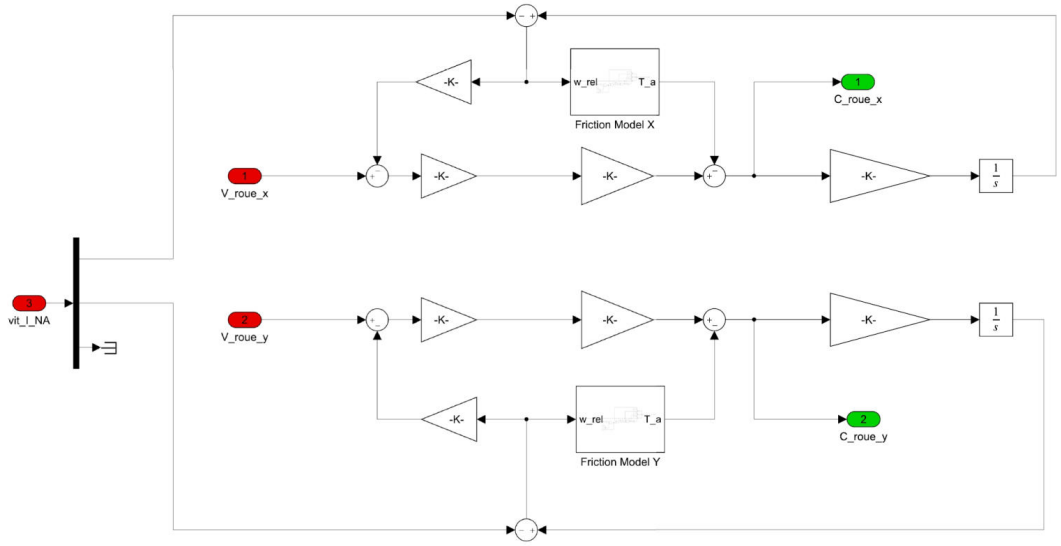
```
% ROUES XY FROTTEMENT
conf.rou_xy_frott_Rm_x = 13.6; %[Ohm]
conf.rou_xy_frott_J_roue_x = 7.36e-3; %[kg.m^2]
conf.rou_xy_frott_Ke_x = 0.431; %[Vs/rad]
conf.rou_xy_frott_Km_x = 0.43; %[Nm/A]
conf.rou_xy_frott_F_c_x = 5e-4; %[N.m]
conf.rou_xy_frott_F_s_x = 1e-3; %[N.m]
conf.rou_xy_frott_Sigma_0_x = 0.0140; %[N.m/rad]
conf.rou_xy_frott_Sigma_1_x = 0.007; %[N.m/(rad/s)]
conf.rou_xy_frott_Sigma_2_x = 7.189e-04; %[N.m/(rad/s)]
conf.rou_xy_frott_V_s_x = 0.01; %[rad/s]

conf.rou_xy_frott_Rm_y = 13.6; %[Ohm]
conf.rou_xy_frott_J_roue_y = 7.36e-3; %[kg.m^2]
conf.rou_xy_frott_Ke_y = 0.431; %[Vs/rad]
conf.rou_xy_frott_Km_y = 0.43; %[Nm/A]
conf.rou_xy_frott_F_c_y = 5e-4; %[N.m]
conf.rou_xy_frott_F_s_y = 1e-3; %[N.m]
conf.rou_xy_frott_Sigma_0_y = 0.0140; %[N.m/rad]
```

```
conf.rou_xy_frott_Sigma_1_y = 0.007; %[N.m/(rad/s)]
conf.rou_xy_frott_Sigma_2_y = 7.189e-04; %[N.m/(rad/s)]
conf.rou_xy_frott_V_s_y = 0.01; %[rad/s]
```




**(a)** RW model in the simulator




**(b)** RW model

**Figure 5.15:** New RW model implemented in the simulator

# 6 Kinematic model

While debugging the simulator, a problem in the implemented kinematic block emerged. In fact, the signal obtained from the quaternions manipulation seemed to not be consistent with the one obtained from the velocities manipulation, even if they describe the same entity, namely the motion of the cross-elevation frame with respect to the inertial frame.

In this chapter, an alternative convention to the one previously used to name velocities is introduced and every single part of the block is analyse in deep. Eventually, the new model has been implemented in the simulator and the results have been discussed.

## 6.1 The new convention

Understanding exactly the nature of a signal was quite difficult since it was impossible to presume that the previous implementation, reported in Figure 6.1, was correct. Therefore, it was necessary to perform tests to verify that every signal was correct and expressed in the correct reference frame. Thus, a new and common convention to express unambiguously the velocity signals was required. To completely characterize the velocity vector, there are three important elements to define:

1. The body which the velocity is referred to
2. The second body (or system of reference) with respect to the velocity is calculated
3. The reference frame in which the vector is expressed

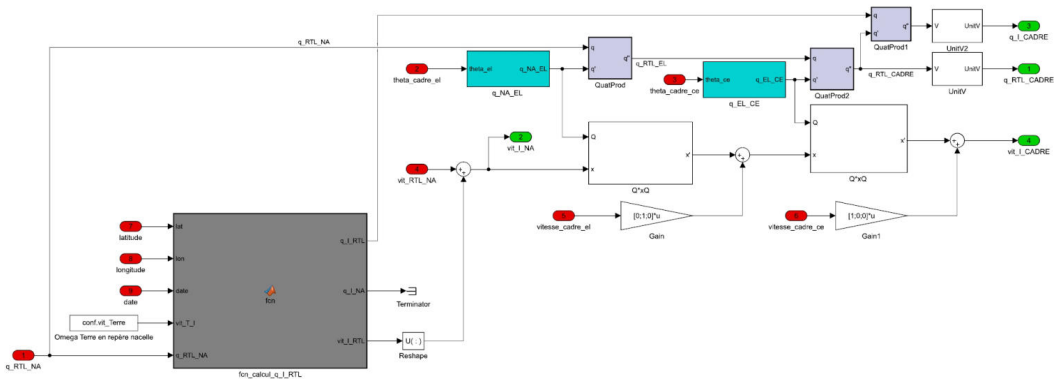



**Figure 6.1:** Previous Kinematic model implemented in the simulator

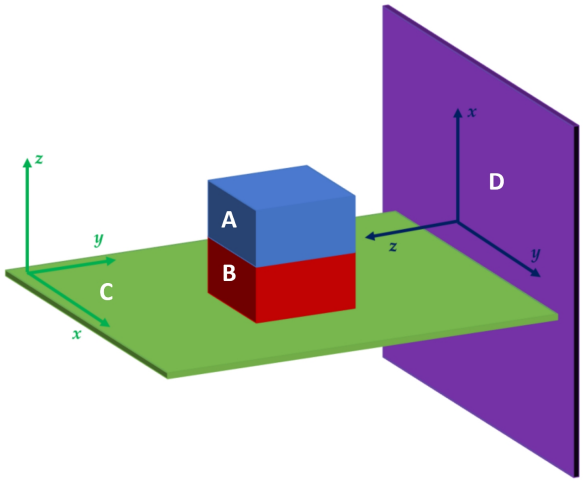



**Figure 6.2:** New kinematic convention

In fact, one could consider as an example the system represented in Figure 6.2 in which there are the body A solidly attached to the body B, a table C and a wall D. A and, therefore, B can rotate on the table, while both C and D are still. Supposing one is interested in expressing the velocity of the body A (1), it will be numerically different if the said velocity is expressed with respect to B or C/D (2). In the first case, in fact, it will be zero, while in the second case it will not. It is also important to notice that C and D are still, but they have different reference frames. Therefore, the velocity of A will have different Cartesian coordinates depending on the reference frame in which the vector is projected (3). With that said, it was possible to introduce the following convention to indicate the velocity of the body A with respect to B and expressed in the reference frame of C: $Vit^{[C]}_{B,A} \iff Vit\_B\_A(C)$

## 6.2 Velocity of the Earth

The first thing done was to check the velocity of the Earth, named in the model as "Omega Terre en repère nacelle", that was defined in the function *cCinematique_Dicos.m* as follows:

```
conf.vit_Terre= [(2*pi)/86164.1 0 0];
```

Although the name suggests that the velocity is expressed in the (NA) reference frame, it is a constant vector, which does not make sense since the gondola can rotate with respect to (RTL) and, therefore, to (I). Moreover, the velocity cannot be defined in the (RTL) reference frame since the actual latitude and longitude of the system are not taken into account. In conclusion, considering also the equations implemented in the "fcn_calcul_q_I_RTL" block (studied in the next section), it was possible to conclude that the velocity had to be defined as indicated below and that, therefore, the correct name should have been $Vit\_I\_T(I)$.

```
conf.vit_Terre= [0 0 (2*pi)/86164.1];
```

## 6.3 $q_{I \to RTL}$ computation

The latitude, longitude and mission date are used inside the "fcn_calcul_q_I_RTL" block to compute the quaternion relative to the transformation from (I) to (RTL) $q_{I \to RTL}$. The before-mentioned quaternion is multiplied by the quaternion relative to the transformation from (RTL) to (NA) $q_{RTL \to NA}$, according to Equation 6.1 and the quaternion operations rules that can be easily found in the literature.

$$q_{I \to NA} = q_{I \to RTL} * q_{RTL \to NA} \tag{6.1}$$

Therefore, the evolution in time of $q_{I \to NA}$ depends on the motion of (RTL) with respect to (I), since the system rotates around $Z_I$ with the same velocity the Earth rotates around its rotation axis, and on the motion of (NA) with respect to (RTL) due to the system dynamics.

The last input of the "fcn_calcul_q_I_RTL" block is $Vit\_I\_T(I)$, that was discussed in the previous section. As mentioned before, this is exactly the same velocity that the system must have to maintain its orientation with respect to the surface of Earth. Different velocities, in fact, would not permit the (RTL) reference frame to remain tangent to the surface and correctly oriented, therefore $Vit\_I\_T(I) == Vit\_I\_RTL(I)$. Figure 6.3 offers a schematic representation of the

two frames of reference.

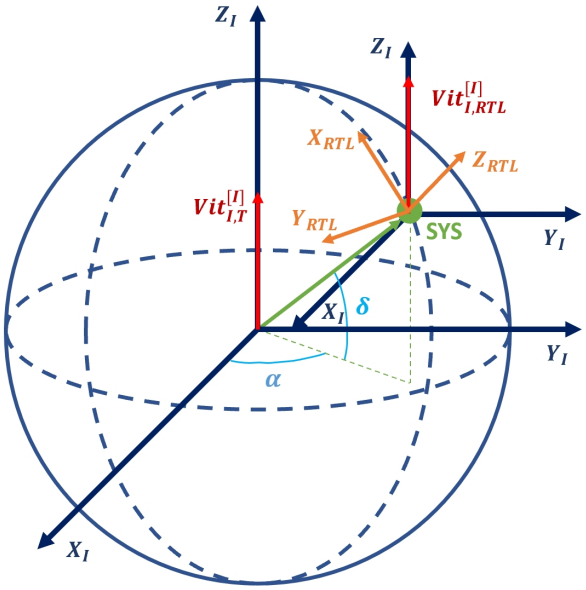



**Figure 6.3:** I and RTL reference frames representation

The Equation 6.2 is used to express the before-mentioned velocity in the (NA) reference frame, where $q^*_{I\to NA}$ is the conjugated of $q_{I\to NA}$, and $q_{Vit^{[NA]}_{I,RTL}}$ and $q_{Vit^{[I]}_{I,RTL}}$ are the quaternions obtained placing in front of the relative arrays a 0 as additional entry.

$$q_{Vit^{[NA]}_{I,RTL}} = q^*_{I\to NA} * q_{Vit^{[I]}_{I,RTL}} * q_{I\to NA} \tag{6.2}$$

This action is performed in the *calcul_vit_T_NA.m* function:

```
vit_T_NA = quatrot_mat(1,vit_T_I,q_I_NA);
```

By opening the before-mentioned function, one would see that since the first index of the function is 1, the operation performed inside the function is the following one:

```
vec3 = Q * vece3 * conj(Q)
```

However, the correct equation that should be implemented is the following one:

```
vec3 = conjQ * vece3 * (Q)
```

In fact, by considering for example $q_{I\to NA} = [-0.778 - 0.1970.295 - 0.518]$, that for a 321 rotation sequence corresponds to a rotation around each axis of $[0° - 41.5394°67.2652°]$, one

obtains the rotations reported in Figure 6.4.

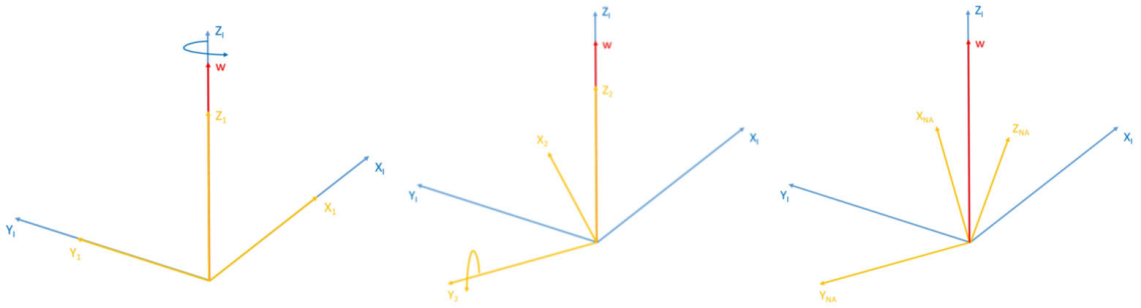
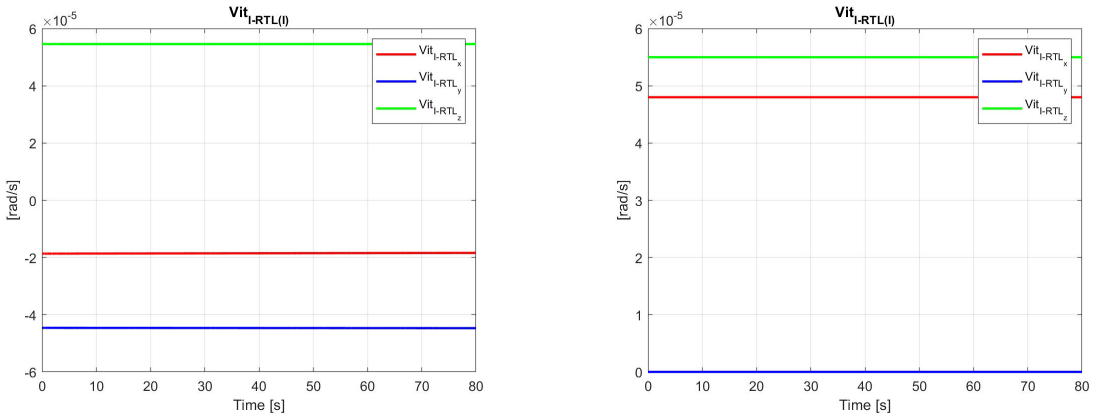


**Figure 6.4:** Representation of $q_{I \to NA}$ rotation sequence

Since $Y_{NA}$ remains in the $(XY)_I$ plane, the $\omega_{y,NA}$ should be roughly zero. However, by plotting $Vit^{[I]}_{I,RTL}$, one obtains the Figure 6.5a. On the other hand, using the correct formula brings to Figure 6.5b which seems to be correct since the velocity vector is decomposed almost equally along the $X_{NA}$ and $Z_{NA}$ axes.




**(a)** Wrong angular disposition

**(b)** Correct angular disposition

**Figure 6.5:** Wrong and correct angular disposition

## 6.4 Sampling frequency

By investigating the different signals inside the kinematic block, it was possible to notice that $q_{I \to RTL}$ is very dependant on the sampling period. In fact,

- if $T_e \leq 1$ s, the angles retrieved from the quaternion update every 1 s

- if $T_e = t > 1$ s, the angles retrieved from the quaternion update every $t$ s

This is due to the way the date is calculated inside the *OBC* block. As visible from Figure 6.6, the time is sampled every $T_e$ seconds. The discretized signal is used inside the *Modele navigation* block to compute the actual date in the following format: $[YYYY, MO, DD, S]$, where $S$ is the total number of seconds elapsed from the beginning of the day $DD$. Consequently, the seconds are represented by a discrete signal that presents discontinuities every $1$ s, even if the sampling period was smaller than that. As a consequence, the quaternion and the corresponding angles present the same behavior and, therefore, their time derivative may change a lot depending on the chosen $T_e$.

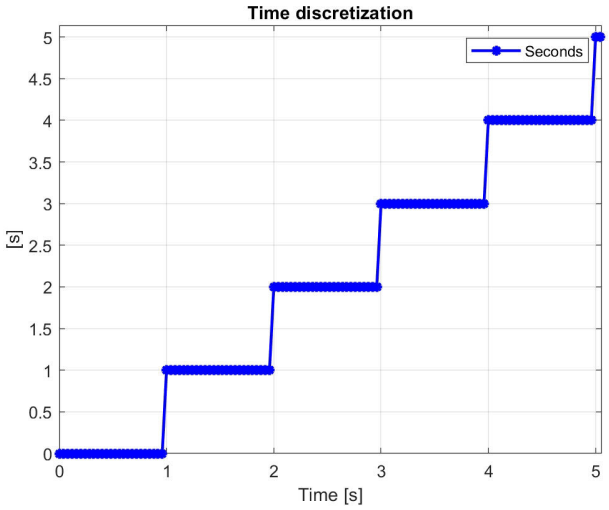
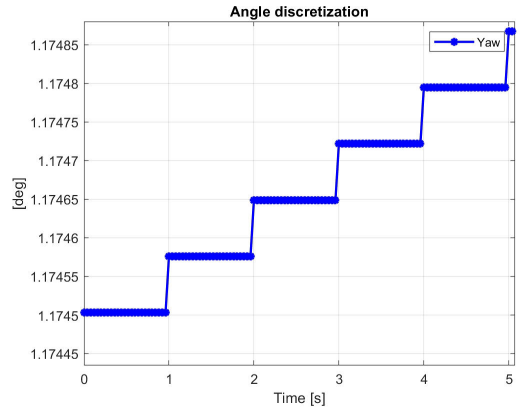



**Figure 6.6:** Discretization of the time signal

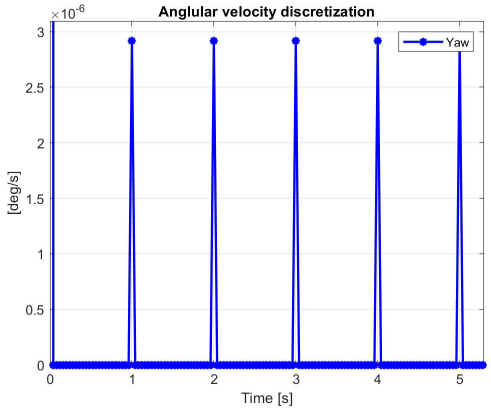
In fact, by considering for example $T_e = 0.2$ s, one would obtain the angles reported in Figure 6.7a that shows that the signal is actually updated every $1$ s. Moreover, by computing the time derivative of the reported signal, one would not obtain a constant positive value, as expected since the signal is monotonous growing, but the signal reported in Figure 6.7b. In fact, the time derivative of a signal $u$ is given by Equation 6.3.

$$\frac{\Delta u}{\Delta t} = \frac{u_t - u_{t-1}}{T_e} \tag{6.3}$$

By considering the plateau zones of Figure 6.7a, one would have that the derivative is zero since $\Delta u = 0$. On the other hand, by considering the discontinuity zones, it is possible to see that $\Delta u$ is always the same and, therefore, the peaks are grater the smaller $T_e$ is.




**(a)** Zoom in the discretization of the angular signal

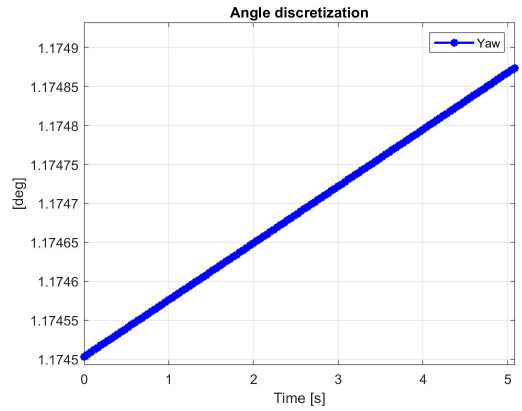



**(b)** Discretization of the angular velocity signal

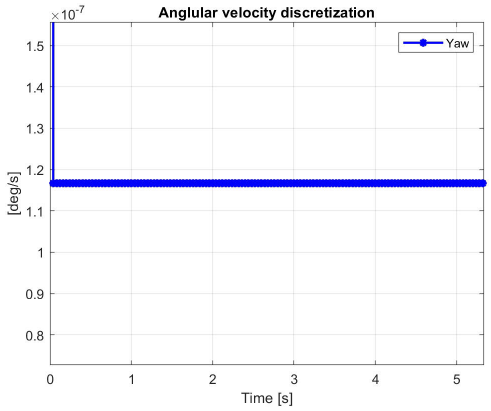
**Figure 6.7:** Discretization of the angular and angular velocity signals

To avoid this computational error, a different procedure was implemented by creating the "Date_to_quaternion" block to compute the Sidereal Time, which is at the basis for the calculation of the quaternion $q_{I \to RTL}$. The block converts the initial date in a datetime type variable in order to add proficiently the elapsed seconds during the simulation and then converts it in the vector $[YYYY, MO, DD, H, MI, S]$. This vector is later used to compute the Julian Days, the Sidereal Time and, eventually, the quaternion $q_{I \to RTL}$.

As a result, the zoom in the 3rd angle profile retrieved from the quaternion and its time derivative are reported in Figure 6.8a and Figure 6.8b respectively.




**(a)** Zoom in the new discretization of the angular signal




**(b)** New discretization of the angular velocity signal

**Figure 6.8:** Discretization of the new angular and angular velocity signals

It is evident that this approach makes the time discretization effectively useful and yields to a

velocity signal that is physically consistent. Therefore, the new model was implemented in the *OBC* block, as shown in Figure 6.9.

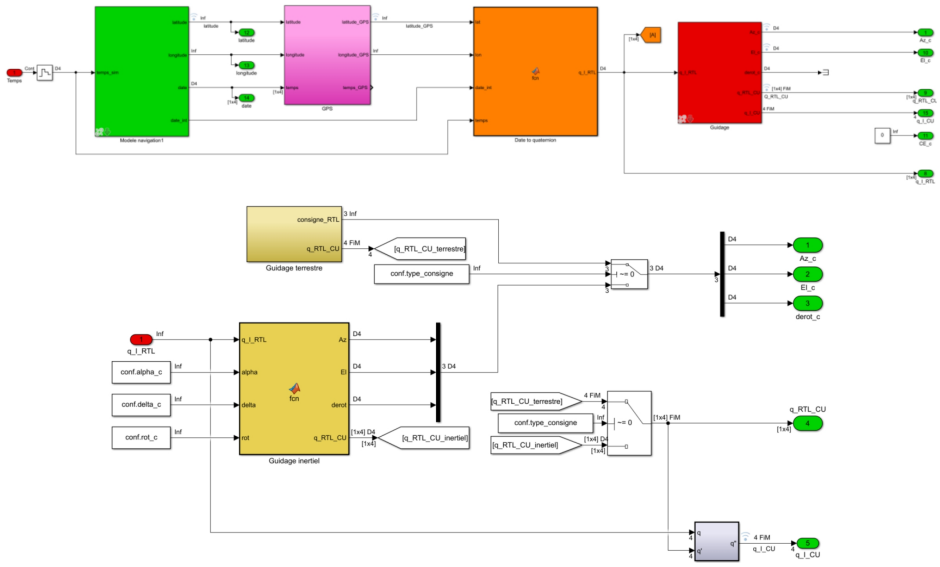



**Figure 6.9:** New OBC model implemented in the simulator

## 6.5 Local analysis

Since as seen in the previous section, the kinematic model could be badly influenced by other models in the simulator, a local analysis of the whole simulator was performed to investigate the reasons of the signals inconsistency. The relations showed in Equation 6.4 have been particularly useful to verify the nature of the signals.

By considering :

$$\dot{q(t)} = \frac{1}{2} q(t) * q_{\Omega(t)}$$
$$\frac{q(t) - q(t - dt)}{dt} = \frac{1}{2} q(t) * q_{\Omega(t)}$$

It is possible to obtain :

$$q_{\Omega(t)} = \frac{2}{dt} q^*(t) * (q(t) - q(t - dt))$$

And eventually :

$$q_{\Omega(t)} = \frac{2}{dt} \left( \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - q^*(t) * q(t - dt) \right) \tag{6.4}$$

where $q_{\Omega(t)}$ is the quaternion obtained placing in front of the velocity vector $\Omega(t)$ a 0 as additional entry.

It is important to notice that, since the integration time and, therefore, $dt$ are not constant during the simulation, Equation 6.4 can only be applied once the simulation is over and the data is gathered.

The "Dynamic" block was supposed to be correct, and as such, it was not modified. However, a small study on the signals $q\_nacelle$ and $vitese\_rotation\_nacelle$ was done since the two signals were not named properly.

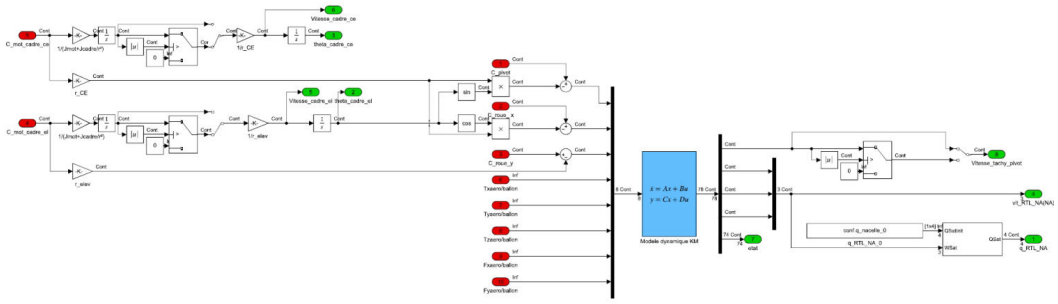


**Figure 6.10:** New angular and angular velocity discretization

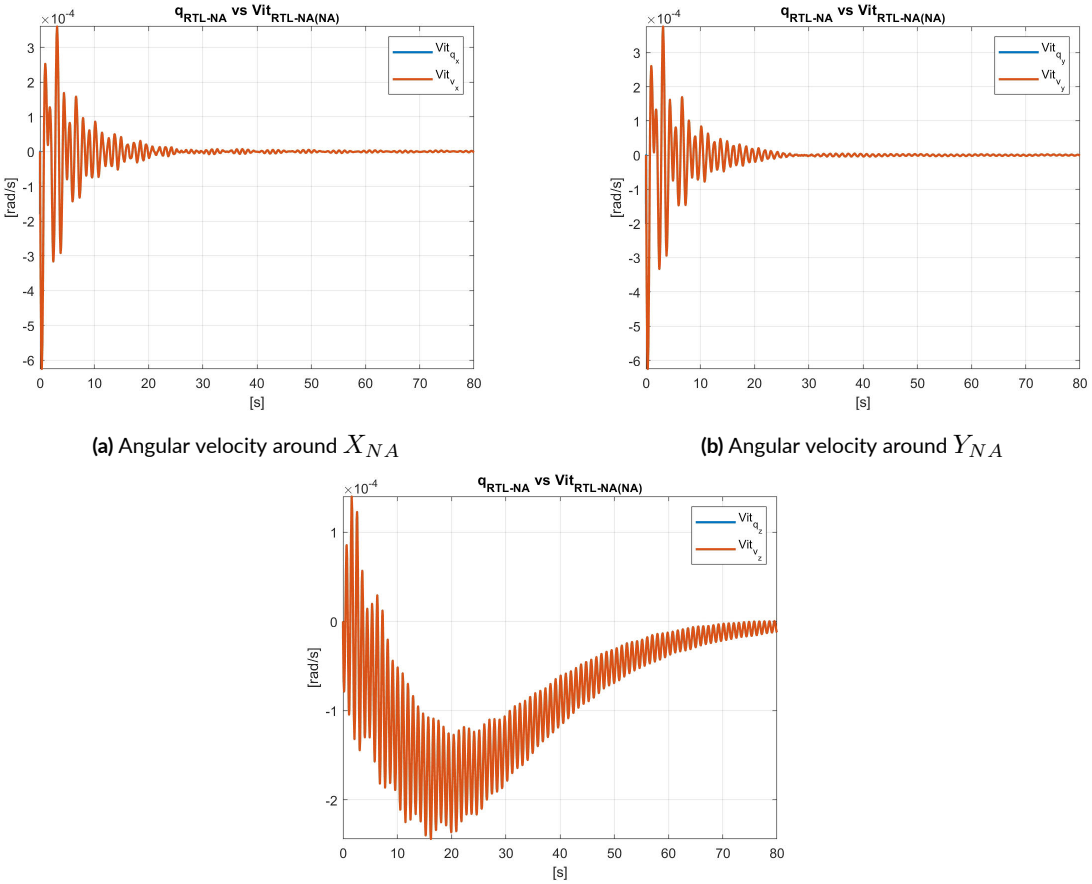
From Figure 6.10, it is evident that $q\_nacelle$ is computed directly from the $vitese\_rotation\_nacelle$ through the Equation 6.5 and, thus, the two signals had necessarily to be consistent.

$$\dot{q(t)} = \frac{1}{2} q(t) * q_{\Omega(t)} \tag{6.5}$$

Moreover, by comparing Equation 6.5 with Equation 6.6, it was possible to understand that $vitese\_rotation\_nacelle$ was $vit\_RTL\_NA(NA)$, and the $q\_nacelle$ the $q_{RTL \to NA}$.

$$\dot{q_{A \to B}}(t) = \frac{1}{2} q_{A \to B}(t) * q_{\Omega^{[B]}_{A,B}(t)} \tag{6.6}$$

In addition, the obvious consistency among the before-mentioned signals offered the possibility to test the applicability and correctness of Equation 6.4. Figure 6.11 shows in fact a perfect consistency among the angular velocities around the three axes retrieved from the velocity and the quaternion respectively.




**(a)** Angular velocity around $X_{NA}$

**(b)** Angular velocity around $Y_{NA}$

**(c)** Angular velocity around $Z_{NA}$

**Figure 6.11:** Angular velocities components comparison

To check the Kinematic block, a new implementation that could represent the behavior of the system was created in a different Simulink project. The aim was to use input signals that were surely consistent, combine them, check whether the consistency was preserved and, if not, try to understand the reasons. Thanks to this new implementation, it was also possible to better understand how to combine signals and change reference frame. Moreover, it brought a reduction in the required simulation time and some margins of enhancement.

The new implementation could be divided in three parts:

1. Motion of (NA) with respect to (I)

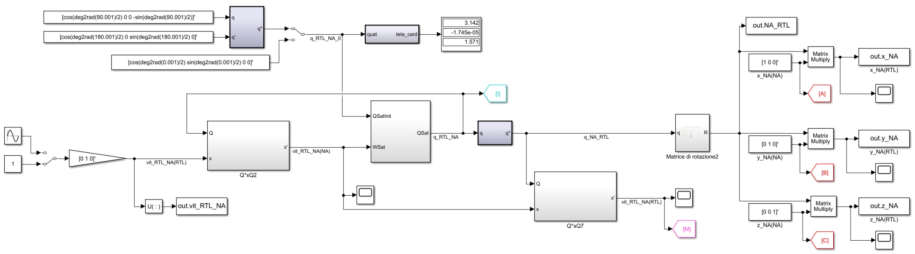


**Figure 6.12:** Motion of NA with respect to RTL

As showed in Figure 6.12, the considered model offers the possibility to choose different possible initial orientation of the NA reference frame with respect to the RTL reference frame by modifying the quaternion $q_{RTL\_NA_0}$. Several initial orientations have been tested and they did not affect the consistency of the results. The quaternion is then propagated using the Equation 6.6 and the $Vit^{[NA]}_{RTL,NA}$ retrieved from Equation 6.7.

$$q_{Vit^{[NA]}_{RTL,NA}} = q^*_{RTL \to NA} * q_{Vit^{[RTL]}_{RTL,NA}} * q_{RTL \to NA} \tag{6.7}$$

The right side of the figure was created to obtain, through MATLAB, a visual representation and simulation of the relative motion of the two reference frames.

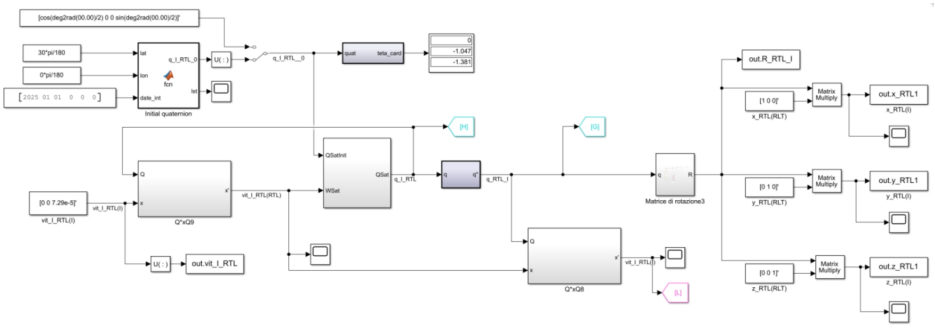


**Figure 6.13:** Motion of RTL with respect to I

As shown in Figure 6.13, the analogous implementation has been done to study the motion of (RTL) with respect to the I reference frame. Unlike the previous case, now the the initial quaternion $q_{I\_RTL_0}$ could be either set arbitrary or retrieved from the latitude, longitude and mission date.

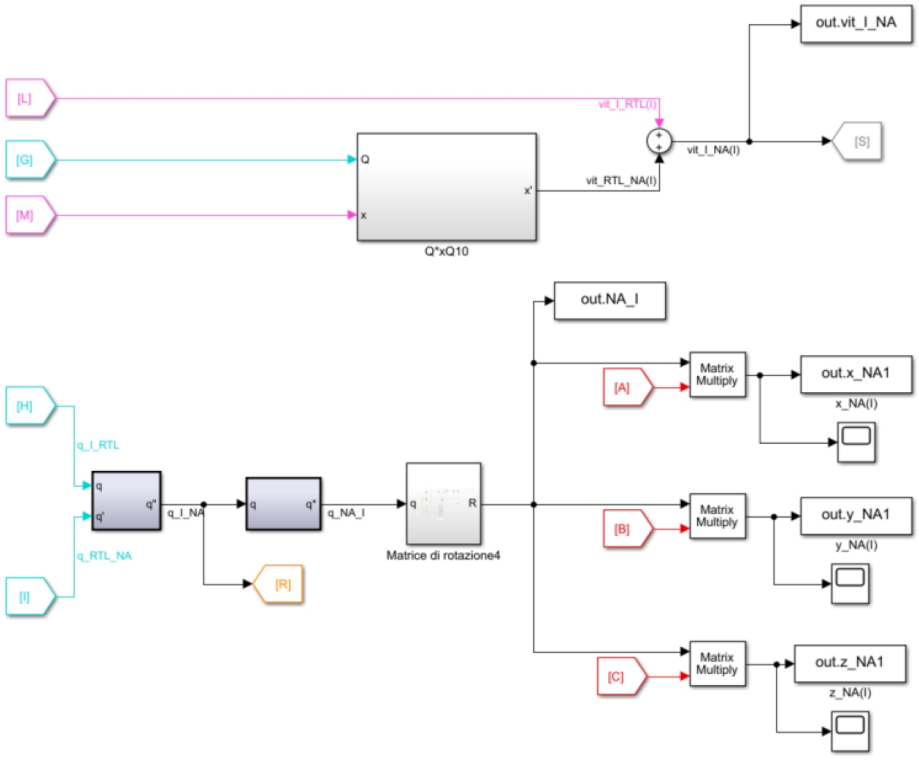



**Figure 6.14:** Motion of NA with respect to I

As shown in Figure 6.14, the model combines the other two previously analysed and verifies the consistency of the obtained signals. On the upper part, the $Vit^{[I]}_{RTL,NA}$ is calculated and

then used to calculate $Vit^{[I]}_{I,NA}$. On the lower part of the figure, the $q_{I\_RTL}$ and $q_{RTL\_NA}$ are combined to obtain $q_{I\_NA}$, that should contain the same information bought by $Vit^{[I]}_{I,NA}$. In fact, simulations demonstrated that the two streams of data were consistent with each other. Figure 6.15 reports a screenshot of the motion of NA with respect to I

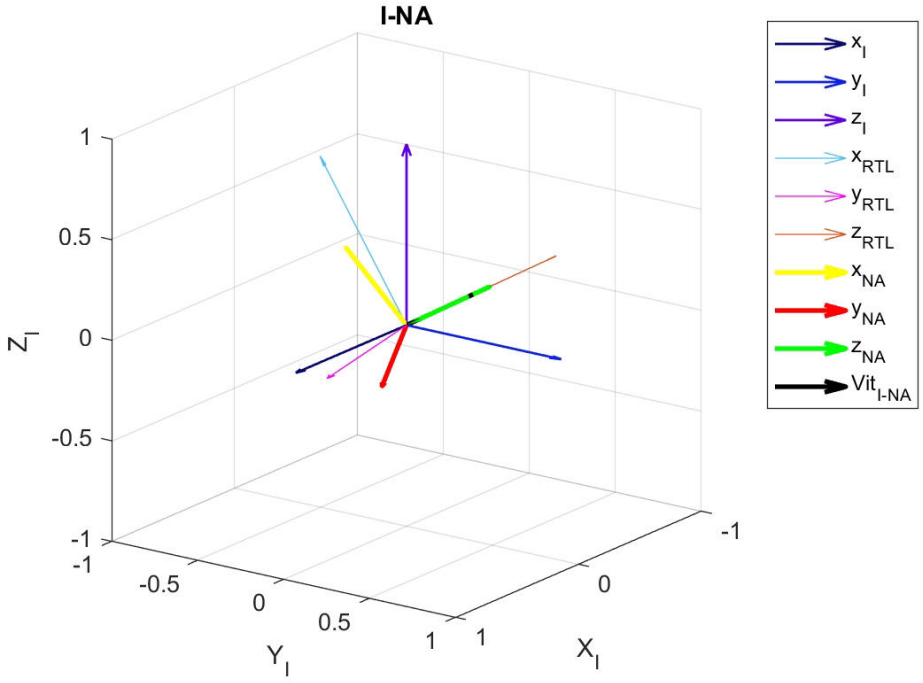
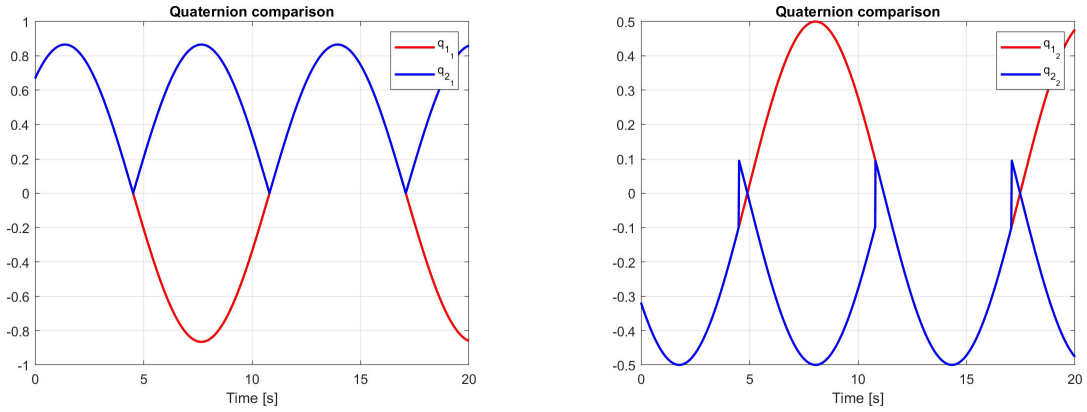



**Figure 6.15:** Screenshot of the motion of NA with respect to I

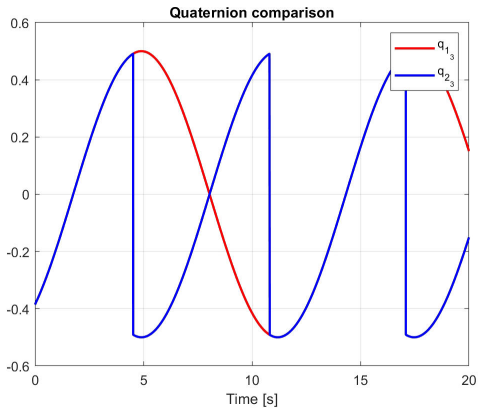
It is dutiful also to say that, instead of comparing the velocities, one could compare the quaternions $q_{I\_NA}$ and the one obtained converting in quaternion the angles obtained via integration of the velocity $Vit^{[NA]}_{I,NA}$. By doing so, the plots in Figure 6.16 are obtained.



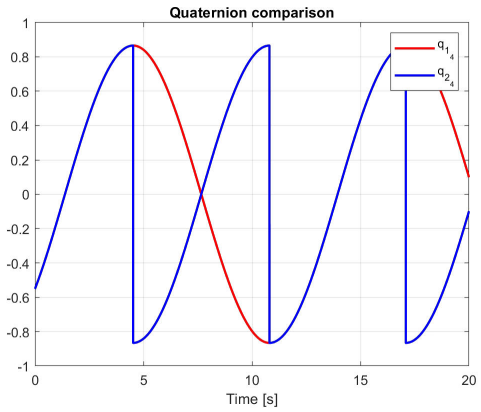

**(a)** $q_1$




**(b)** $q_2$




**(c)** $q_3$




**(d)** $q_4$

**Figure 6.16:** Quaternions components comparison

It is possible to notice that the components of the two quaternions are exactly the same, hence consistent, up to $4.19$ s. Then, they become specular but remain consistent as demonstrated in Equation 6.8.

$$
\begin{aligned}
q = \begin{bmatrix} cos\left(\frac{\alpha}{2}\right) \\ q_1 sin\left(\frac{\alpha}{2}\right) \\ q_2 sin\left(\frac{\alpha}{2}\right) \\ q_3 sin\left(\frac{\alpha}{2}\right) \end{bmatrix} &= \begin{bmatrix} cos\left(\frac{\alpha+2\pi}{2}\right) \\ q_1 sin\left(\frac{\alpha+2\pi}{2}\right) \\ q_2 sin\left(\frac{\alpha+2\pi}{2}\right) \\ q_3 sin\left(\frac{\alpha+2\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} cos\left(\frac{\alpha}{2}+\pi\right) \\ q_1 sin\left(\frac{\alpha}{2}+\pi\right) \\ q_2 sin\left(\frac{\alpha}{2}+\pi\right) \\ q_3 sin\left(\frac{\alpha}{2}+\pi\right) \end{bmatrix} \\
&= - \begin{bmatrix} cos\left(\frac{\alpha}{2}\right) \\ q_1 sin\left(\frac{\alpha}{2}\right) \\ q_2 sin\left(\frac{\alpha}{2}\right) \\ q_3 sin\left(\frac{\alpha}{2}\right) \end{bmatrix} = -q
\end{aligned}
\tag{6.8}
$$

2. Motion of (CE) with respect to (NA)

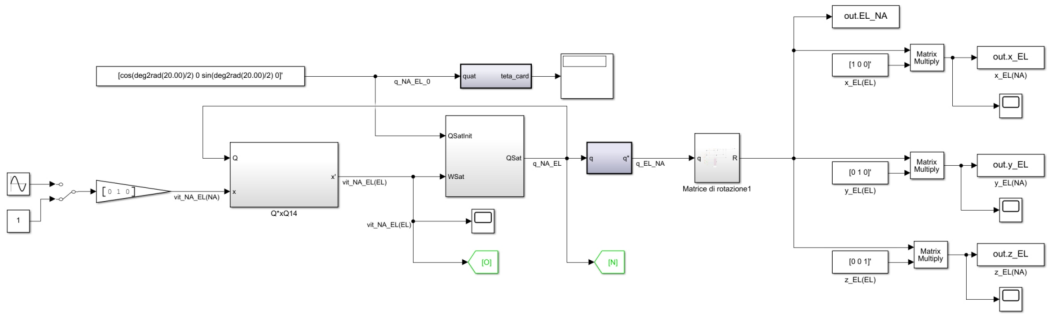



**Figure 6.17:** Motion of EL with respect to NA

Figure 6.17 shows the model implemented to retrieve $Vit_{NA,EL}^{[EL]}$ and $q_{NA \to EL}$.

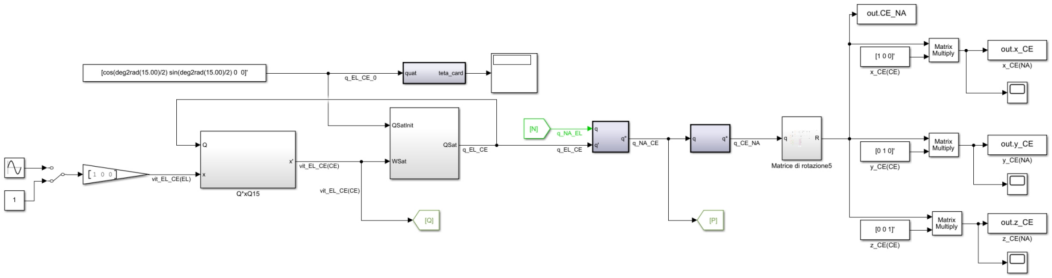
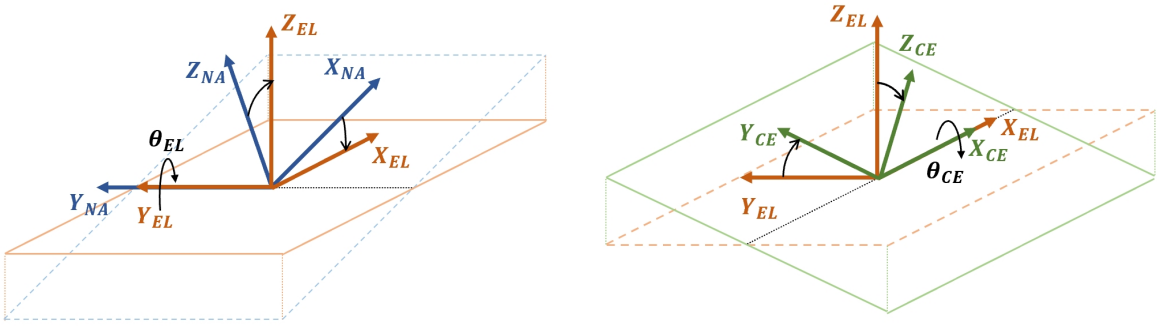



**Figure 6.18:** Motion of CE with respect to NA

Analogously, Figure 6.18 shows the model implemented to retrieve $Vit^{[CE]}_{NA,CE}$ and $q_{NA \to CE}$.

Thanks to these two schemes, it is possible to choose two arbitrary velocities for the EL and CE frames, combine them, and obtain the velocity of the CE frame with respect to NA. By considering the way the two frames move one in the other, schematized in Figure 6.19, it is possible to observe that $Vit^{[NA]}_{NA,EL}$ will be always directed along $Y_{NA}$, while $Vit^{[NA]}_{NA,CE}$ will always be directed along $X_{NA}$. Conducting several simulations in which both the velocity and initial configurations were changed always yielded to consistent results.




**(a)** EL frame **(b)** CE frame

**Figure 6.19:** Schematic representation of the EL and CE frames

Before continuing, two remarkable things could be highlighted: it is very convenient to work with quaternions rather than Euler's angles for their well-known advantages, but also because they involve less problems with the time derivation; quaternions represent the instantaneous rotation of a reference frame with respect to an other, and they are not specifically related to any rotation sequence.

3. Motion of (CE) with respect to (I)

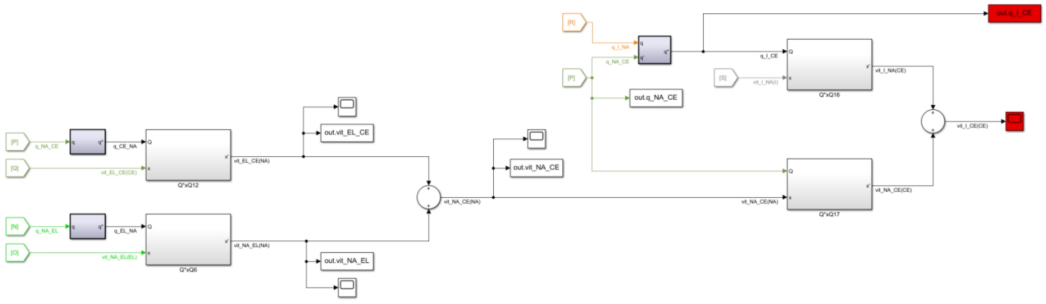


**Figure 6.20:** Motion of CE with respect to I

Figure 6.20 shows the last part of the model, in which $Vit_{I,CE}^{[CE]}$ and $q_{I \to CE}$ are retrieved and compared.On the left side of the scheme, the velocities calculated before are brought in the NA reference frame and then summed to obtain $Vit_{NA,CE}^{[NA]}$. Since the IMU1 needs as input a velocity expressed in CE, both $Vit_{I,NA}$ and $Vit_{NA,CE}$ are brought in the correct frame and then summed to obtain $Vit_{I,CE}^{[CE]}$. Simultaneously, the quaternion $q_{I \to CE}$ is computed using $q_{I \to NA}$ and $q_{NA \to CE}$.

By comparing $Vit_{I,CE}^{[CE]}$ and $q_{I \to CE}$ through the usual method, one would get the error signal showed in Figure 6.21 that is relatively small if compared to the previous implementation.

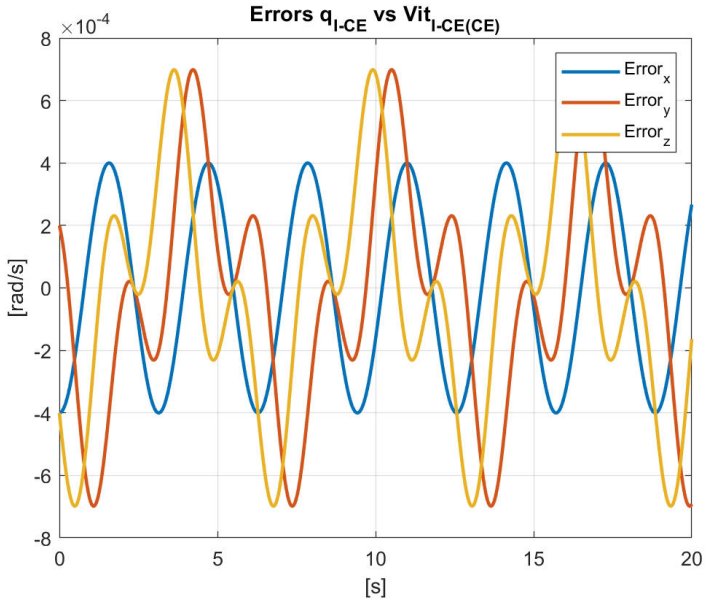



**Figure 6.21:** Error between $Vit_{I,CE}^{[CE]}$ and $q_{I \to CE}$

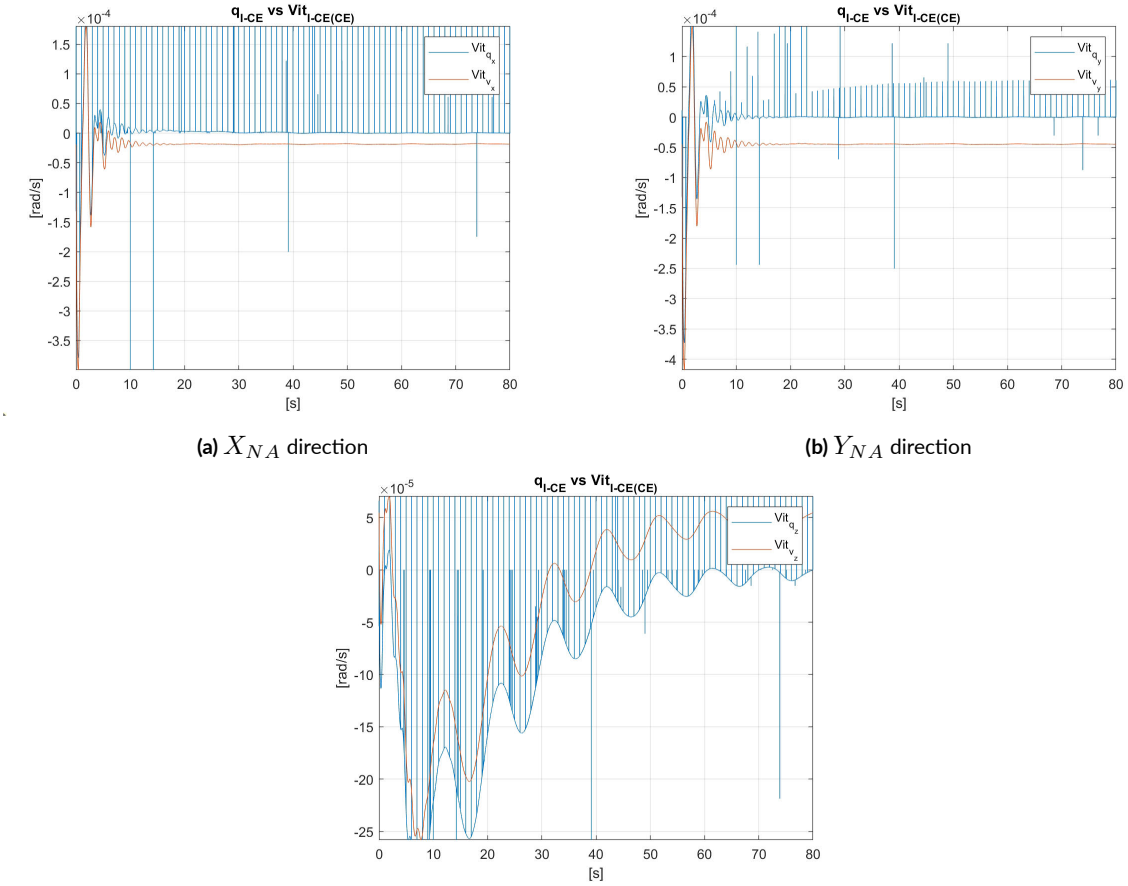
This result showed that the model is capable of preserving the consistency of the signals. In fact, although the velocity and quaternions are not two completely distinct streams of data since the first ones are brought in the different reference frame through the second ones, this method is a good and valid alternative to the other option that would have seen the quaternion $q_{I \to CE}$ calculated directly with Equation 6.5 since, in this case, the consistency would have been force.

By considering that the results are acceptable, the model has been simplified (the simulation in MATLAB has been removed) and it was integrated in a new kinematic block inside the simulator.

In the future, an additional study could be done to understand the effects that these errors have on the Kalman filter and on the system performance.

## 6.6 Discussion and comparison of the results

From Figure 6.22 it is possible to notice from the previous plots that the speed retrieved from the quaternions is very disturbed and presents some peaks that could also reach the order of magnitude of $10^8$. These peaks are probably due to numerical integration errors, but what is more worrying and problematic is the offset between the different pairs of signals.




(a) $X_{NA}$ direction

(b) $Y_{NA}$ direction

(c) $Z_{NA}$ direction

**Figure 6.22:** Comparison between $Vit_{I,CE}^{[CE]}$ and $q_{I\_CE}$

Moreover, from Figure 6.23 that reports the error between the real state and state predicted by the Kalman filter, it is evident that the errors in the kinematic models influence also the performance of the Kalman filter.

On the contrary, Figure 6.24 shows that in the new implementation the different pairs of signals are consistent and besides the peaks, that are smaller than before, they are basically the same. As

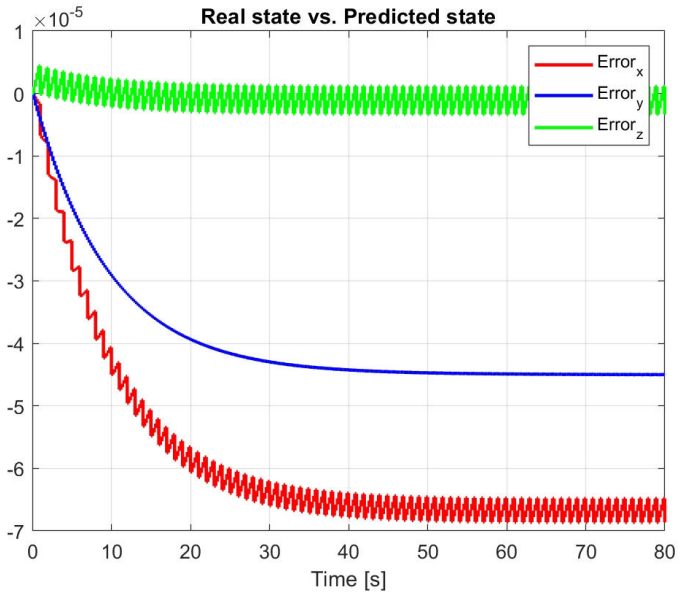
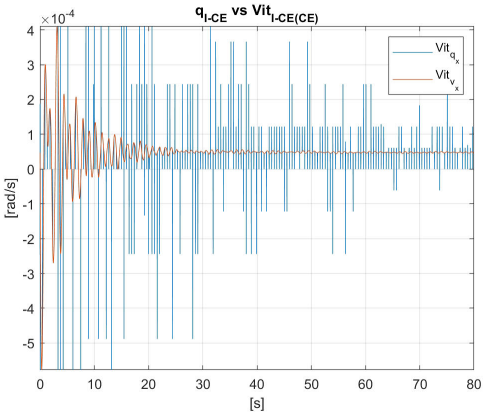


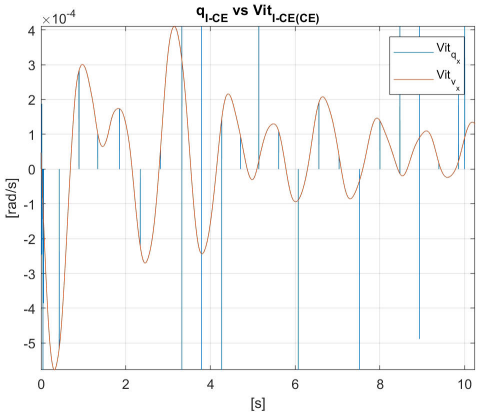

**Figure 6.23:** Error between the real state and state predicted by the Kalman filter

a consequence, the error between the real and predicted states is one order of magnitude smaller than before, and converges to 0 or oscillates around it.

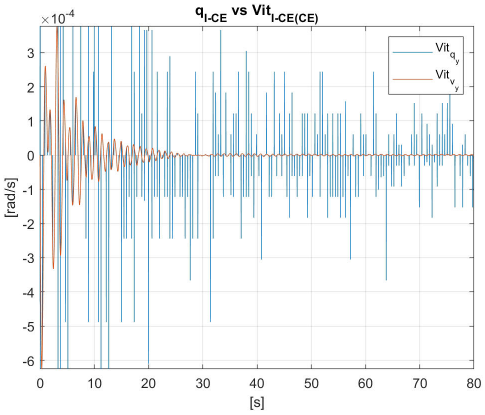
In the future, an additional analysis could be done to understand the origin of the peaks, that at the moment were suspected to be numerical, and to evaluate their effects on the Kalman filter.



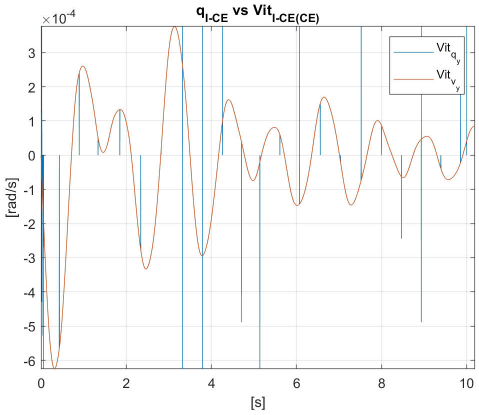

**(a)** $X_{NA}$ direction



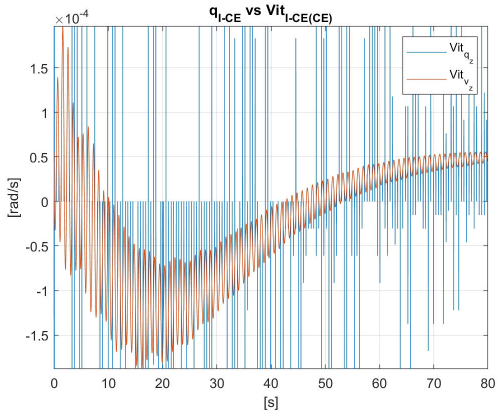

**(b)** Zoom: $X_{NA}$ direction



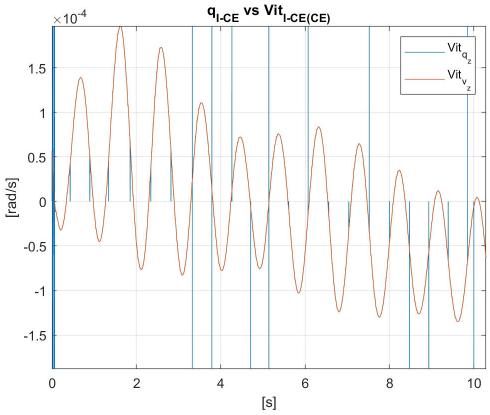

**(c)** $Y_{NA}$ direction




**(d)** Zoom: $Y_{NA}$ direction




**(e)** $Z_{NA}$ direction



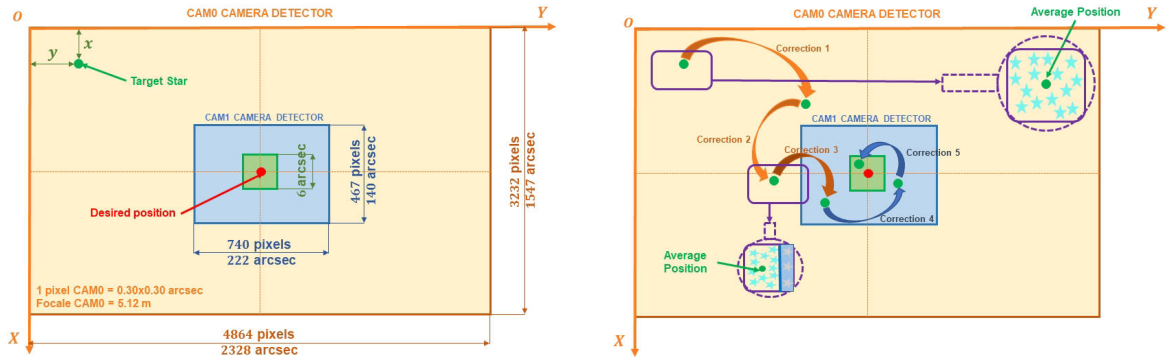

**(f)** Zoom: $Z_{NA}$ direction

**Figure 6.24:** New implementation: comparison between $Vit_{I,CE}^{[CE]}$ and $q_{I\_CE}$

# 7

# Camera model

The fine pointing is performed using the CAM0 and CAM1 mounted on the CE frame. The idea, as mentioned, is to use the DTU, the RWs, and the EL and CE actuators to bring the star (or better said, the centroid of the cloud) inside the FOV of CAM0. Then, using the pixel position of the star on the camera detector, the quaternion required to align the camera LoS with the star vector is computed and a command is sent to the before mentioned actuators. Once the centroid enters the CAM1 FOV, the action of the before mentioned actuators is interrupted and the deformable mirror starts to work in order to bring the centroid at the center of the camera FOV. Figure 7.1 summarizes the process.




**(a)** Characteristics of the cameras

**(b)** Pointing sequences

**Figure 7.1:** CAM0 and CAM1 pointing process

## 7.1 New Implementation

Since it was necessary to implement a first simple camera model that could also take into account some kind of distortions that could affect the camera lens, the Pinhole camera model was chosen. Given the coordinates of the target in the World reference frame, the aim is to compute its projection position on the focal plane of the camera. Figure 7.2 reports a schematic representation of the four reference frames utilized:

- UVW: the World reference frame
- XYZ: the Camera reference frame
- xy: the Image reference frame
- uv: the Pixel reference frame

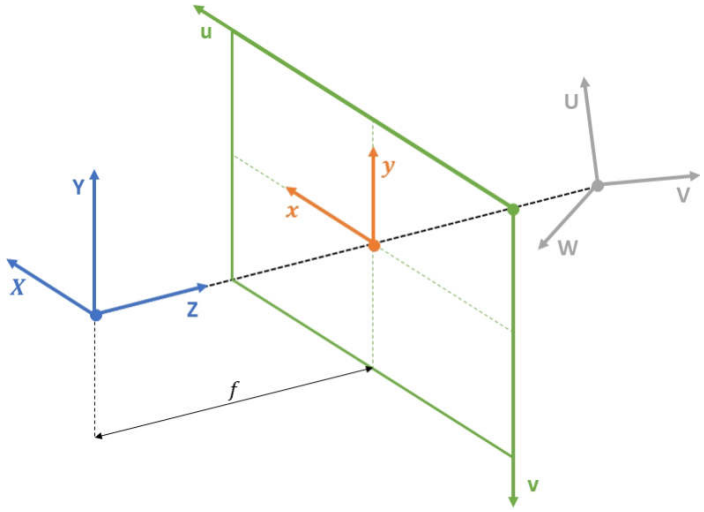



**Figure 7.2:** Schematic representation of the different reference frames

It is evident that each reference frame is centered in a different point in space and has a different orientation. Moreover, it is possible to notice that the focal plane is positioned at $Z = f$, where $f$ is the focal length of the pinhole camera. Equation 7.1 represent the so-called perspective projection equations, which project a 3D point, with coordinates expressed in the XYZ reference frame, in the xy plane.

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z} \tag{7.1}$$

To represent this transformation using the matrix notation, it is necessary to obtain first linear equations. To achieve this, it is sufficient to use the so-called homogeneous coordinates in which it is necessary to introduce a fictitious third non-zero coordinate, as shown in Equation 7.2.

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{7.2}$$

Therefore, the perspective projection equations can be written as shown in Equation 7.3.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{7.3}$$

Then, to obtain the real image coordinates, it is sufficient to apply the Equation 7.4.

$$x = f\frac{x'}{z'}$$
$$y = f\frac{y'}{z'} \tag{7.4}$$

Generally, the position of the target is not expressed in the XYZ reference frame, but in the UVW reference frame. Therefore, it is convenient to introduce a transformation matrix between the two reference frames. As shown in Equation 7.5 he transformation matrix contains both the rotation matrix and the translation vector (that represents the position of one reference frame with respect to the other).

$$[P_2] = \begin{bmatrix} [R]_{UVW \to XYZ} & \vec{T} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & c_x \\ r_{21} & r_{22} & r_{23} & c_y \\ r_{31} & r_{32} & r_{33} & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.5}$$

The parameters contained in $[P_2]$ are the so-called extrinsic parameters, since they are not dependent on the camera characteristics.

An additional consideration can be done on the so-called intrinsic matrix $P_1$, which takes into account the transformation between the xy and uv reference frames, considering both the rotation and the translation, the scaling and the skew angle (respectively due to the different size of the pixels along the x and y direction, and to the angle at the corner of the pixel). The intrinsic matrix has the expression shown in Equation 7.6:

$$[P_1] = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{s}{f_x} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.6}$$

The intrinsic parameters depend on the camera characteristics.

Moreover, it would be possible to consider the lens distortion (like the radial one) using a polynomial expression. However, since the errors produced by such distortions could be corrected after a successful calibration of the camera, they have been neglected in this study.

Considering again the homogeneous coordinates, it is possible to obtain Equation 7.7.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{7.7}$$

So, altogether the transformation is reported in Equation 7.8.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & c_x \\ r_{21} & r_{22} & r_{23} & c_y \\ r_{31} & r_{32} & r_{33} & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{7.8}$$

From which it is possible to retrieve the pixel position using Equation 7.9.

$$u = f\frac{x'}{z'}$$
$$v = f\frac{y'}{z'} \tag{7.9}$$

Once the pixel coordinates of the target are computed, it is possible to calculate the quaternion $q_{star,CAM0 \to LoS,CAM0}$ using the procedure schematized in Figure 7.3.

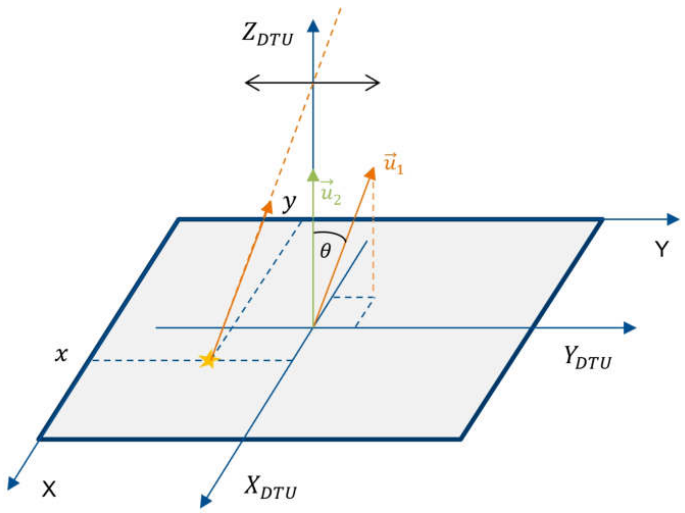



**Figure 7.3:** Schematic representation of the method used to compute the pixel coordinates

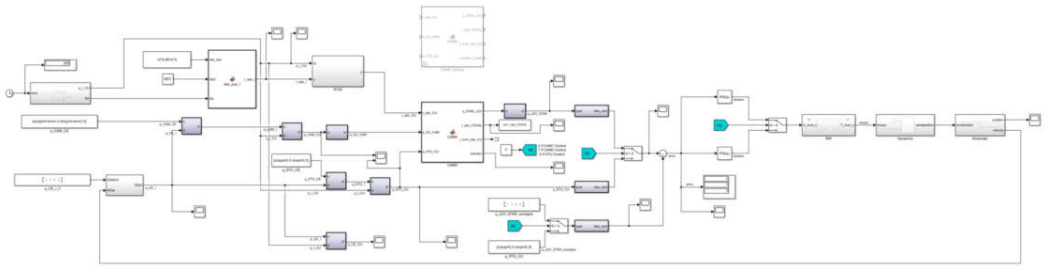


**Figure 7.4:** Simplified model of the gondola kinematics

To verify the behavior of the model, a simplified implementation, reported in Figure 7.4, of the gondola and flight chain was created. A Right Ascension $RA$ and Declination $DEC$ was considered to identify a hypothetical star in the sky. Moreover, only the CE frame, on which the DTU and cameras are mounted, was considered. The system inertia was represented by a simple

matrix of inertia, and 3 independent RWs controlled by 3 PID controllers have been considered as actuators. The model took into account the different orientation of CAM0 with respect to the CE reference frame hypothetically, due to the temperature effects or other casual errors that affect the system after the camera calibration, by using the variable "error" that could be arbitrarily changed. The aim of the simulation, was to match the LoS of the sensors with the direction vector between the star and the sensor itself, considering the different pointing stages described at the beginning of this chapter. Different initial configuration have been tested, and the model seemed to behave fine.

## 7.2 Discussion and comparison of the results

Figure 7.5 reports the results obtained after a simulation from which it can be seen that the system managed to bring the star inside the FOV of CAM1.

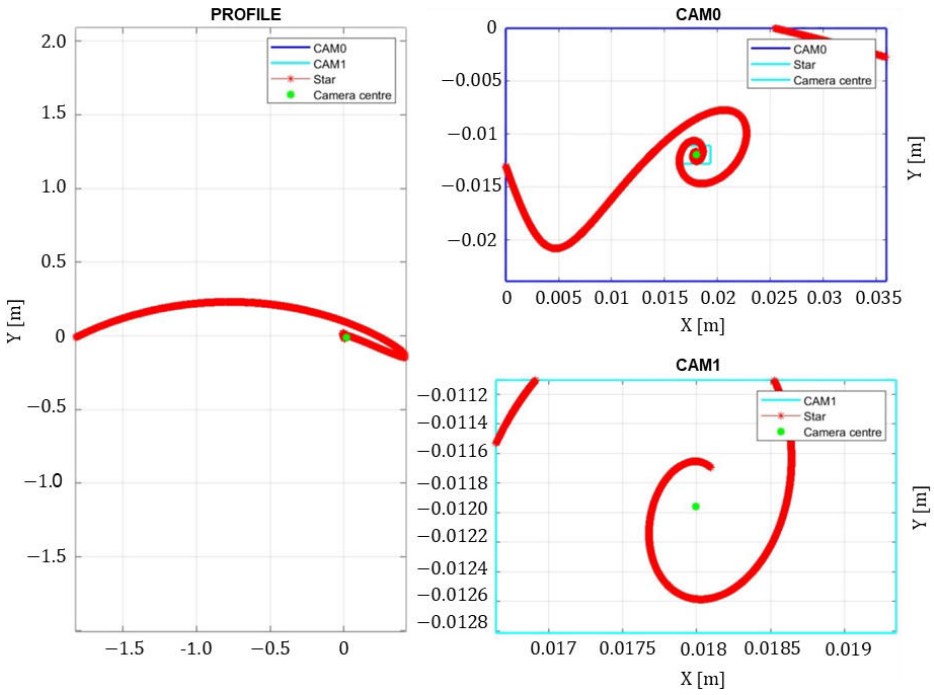



**Figure 7.5:** Simulation results

In addition, a model that could take into account the biases introduced by the thermal gradients was implemented. The idea was to consider an hotspot on the camera CCD and a consequent thermal conduction flux that was able to deform the focal plane characterized by $nm$ poles equally

distributed. To determine the temperature in each of the poles, the Fourier's law was applied: it was sufficient to solve a system of $nm$ equations in $nm$ unknowns. Eventually, the temperature distributions was interpolated linearly to obtain the results showed in Figure 7.6.

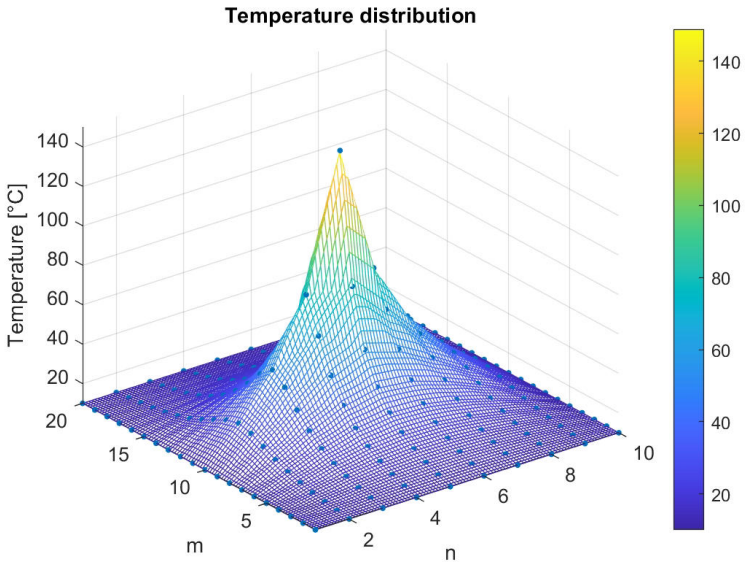



**Figure 7.6:** Temperature distribution

However, since this activity was done at the end of the internship, there was not enough time to test exhaustively the model and implement a deformation model.

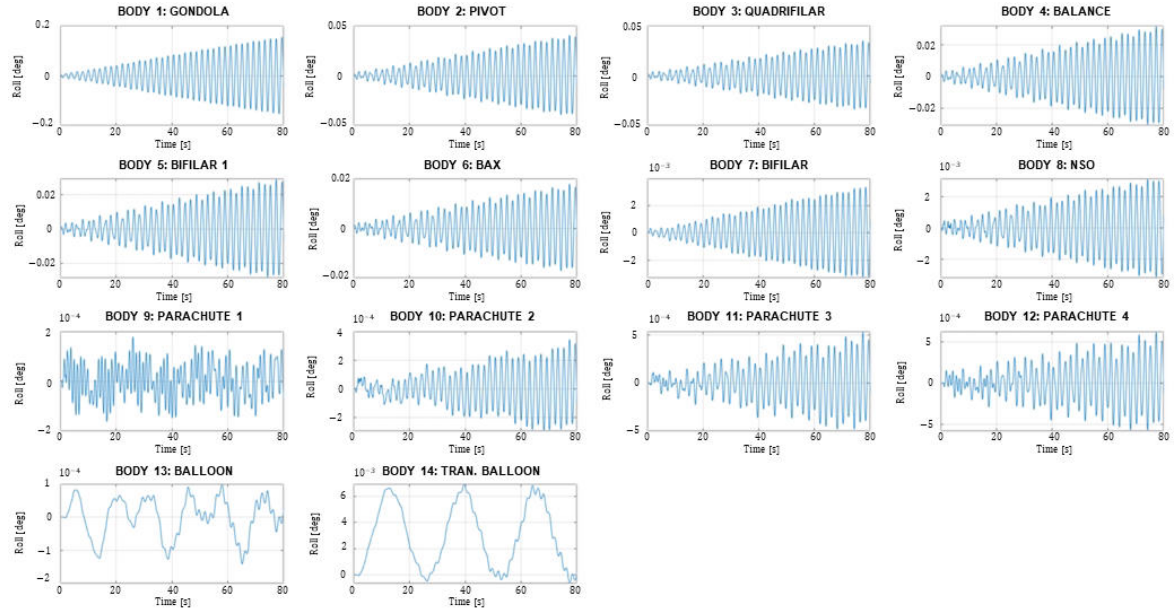
# 8

# Discussion of the results

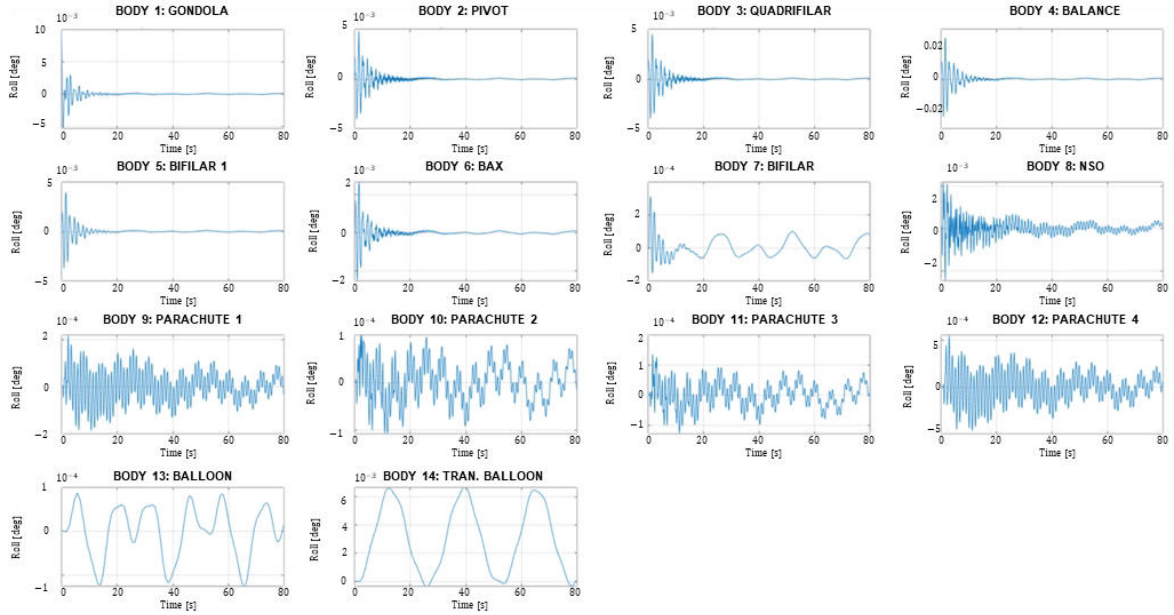
The previous chapters reported at their end the results relative to the implementation of the corresponding model. To test their effect on the entire simulator, a simulation of $80s$ has been done, and the corresponding results have been reported in this chapter.

Figure 8.1, Figure 8.2 and Figure 8.3 show the comparison between the angular position of the different elements of the flight chain, around the $X_{NA}$, $Y_{NA}$ and $Z_{NA}$ respectively, obtained from the old and new implementations. By observing the plots, it is possible to observe a general enhancement of the system performance. In fact, by looking at the divergent angular profiles, it is evident that the system was previously unstable. This was probably due to multiple factors, such as the inconsistency found in the kinematic model during the study that affected negatively the behaviour of the system. Thanks to the new implementation and the correction of several minor errors, it was possible to obtain a stable system that could converge, as visible from **??**, after roughly $30s$. On the other hand, an increase of the computational cost was observed. This behaviour is due mainly to the implementation of the new reaction wheel model because, as discussed in the dedicated chapter, consists in a set of non-linear equations which are difficult to solve.

Figure 8.4 reports a set of screenshots of the final state of the simulator, focusing mainly on the parts modified during the traineeship.

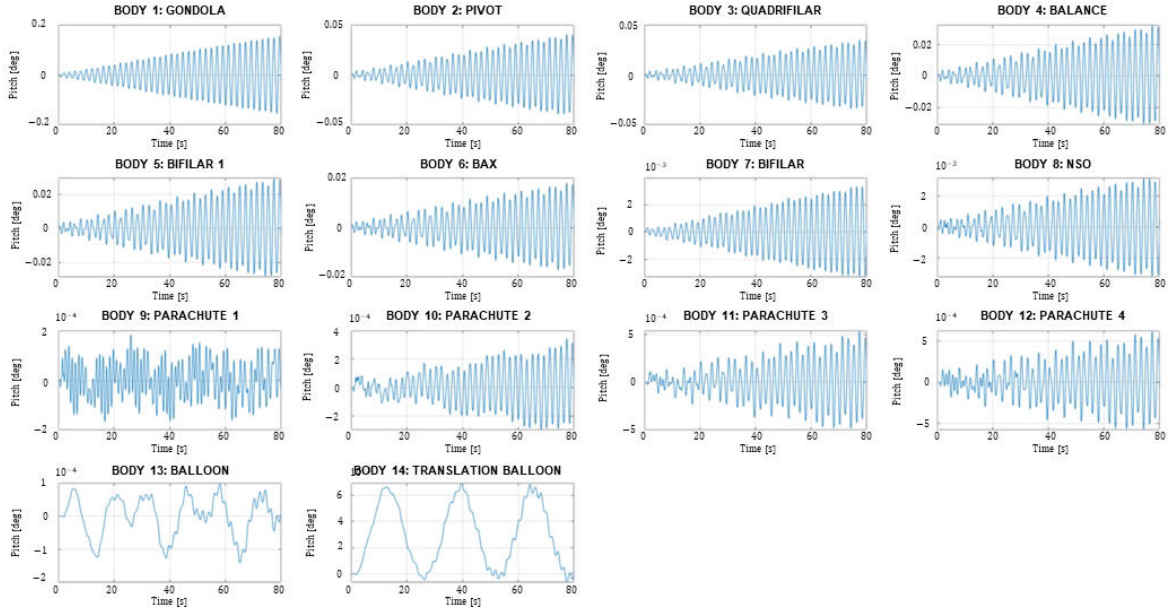



**(a)** Old implementation

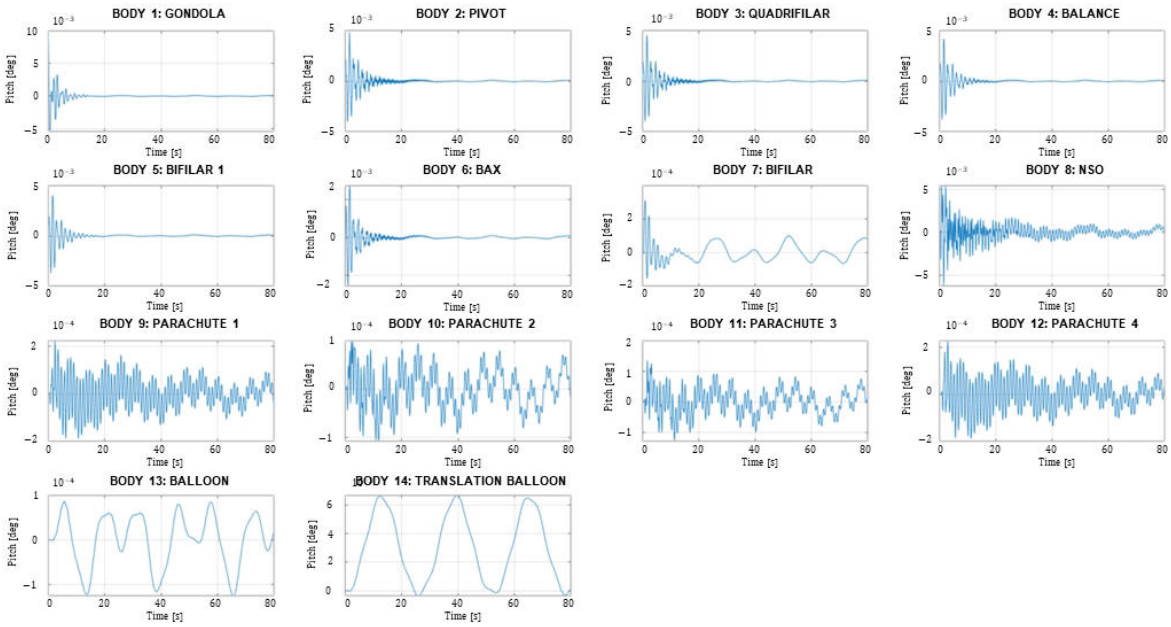



**(b)** New implementation

**Figure 8.1:** Simulation results: roll

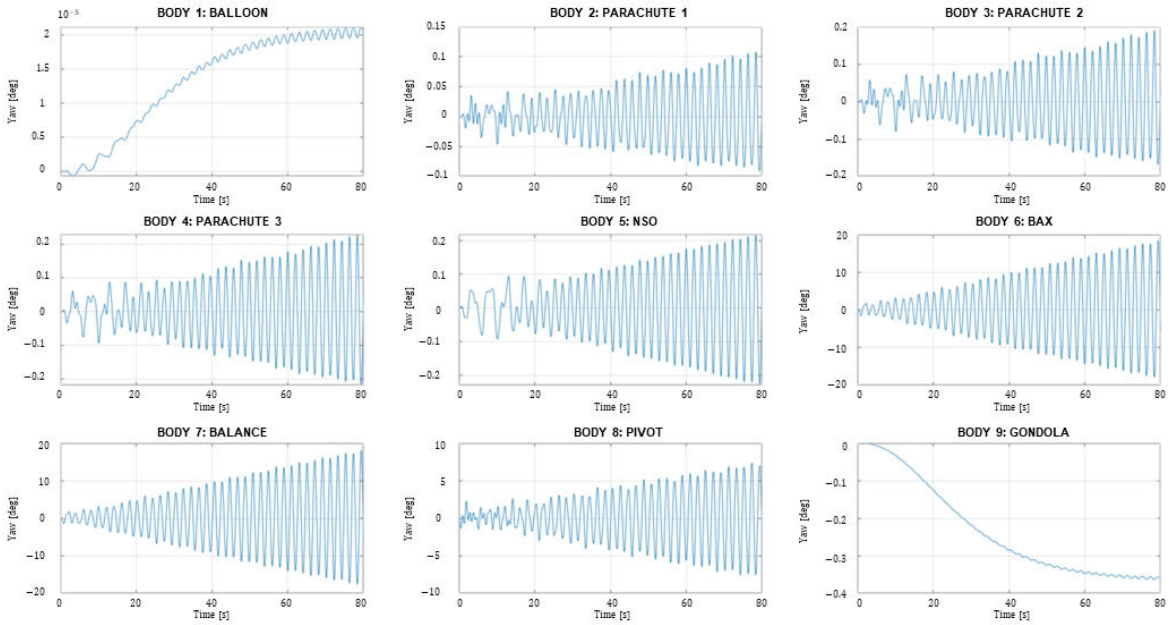



**(a)** Old implementation

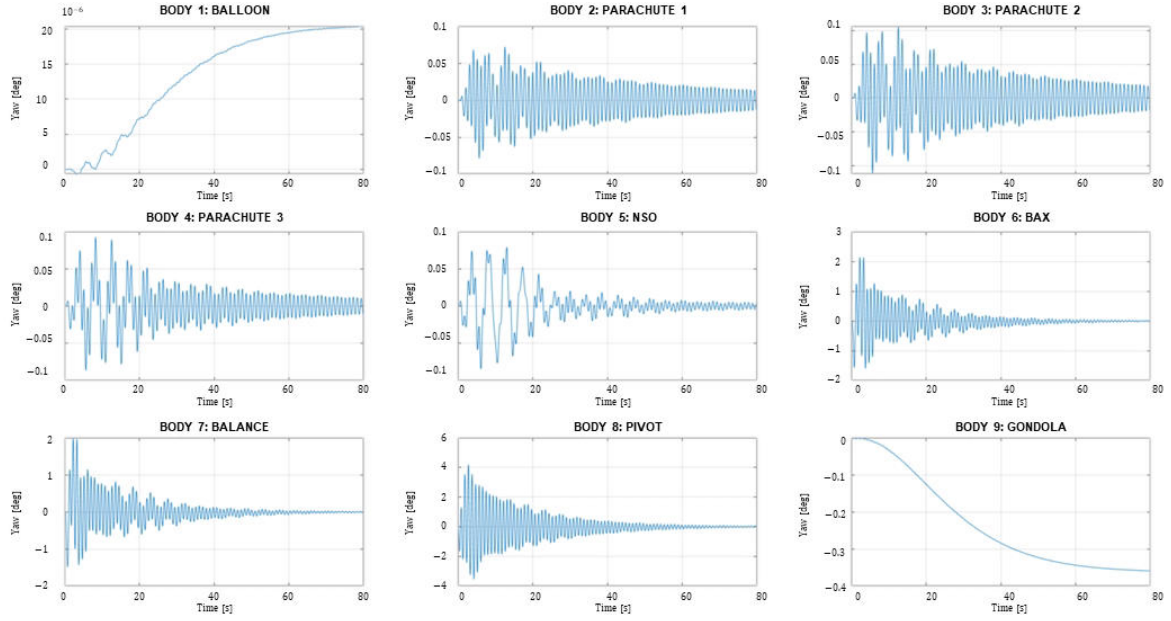



**(b)** New implementation

**Figure 8.2:** Simulation results: pitch

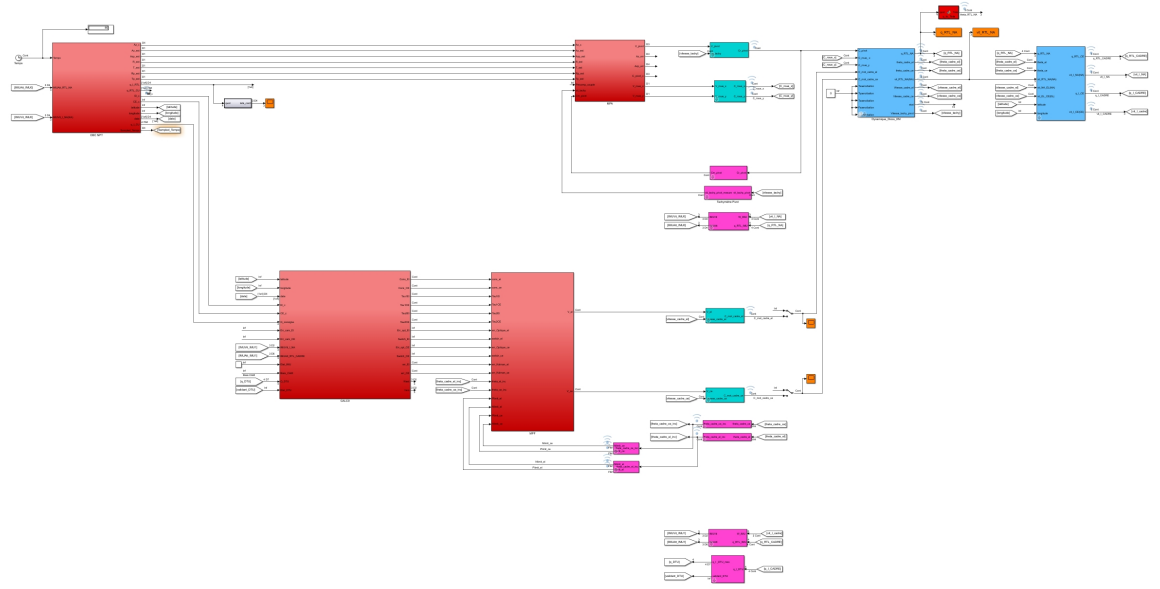



**(a)** Old implementation

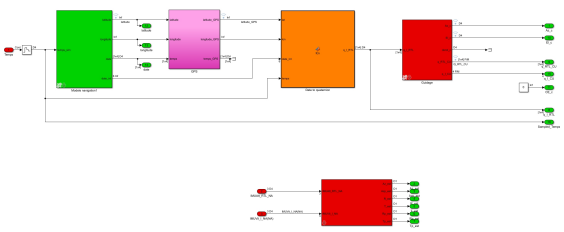



**(b)** New implementation

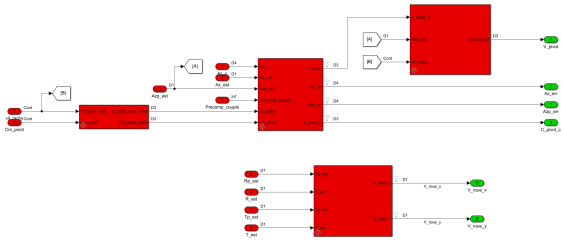
**Figure 8.3:** Simulation results: yaw



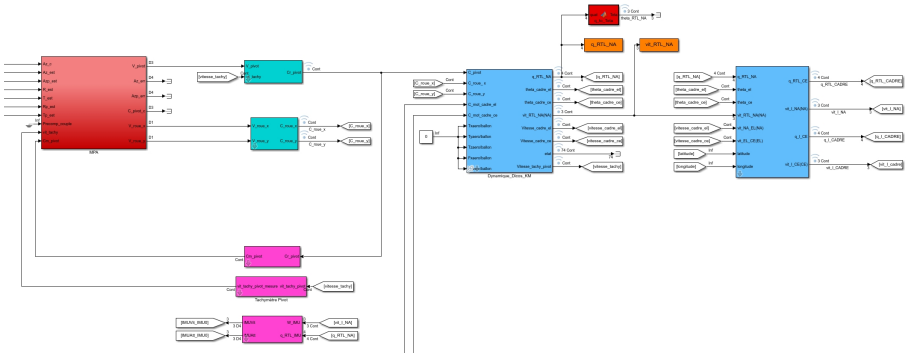
**(a)** Overview



**(b)** OBC Block



**(c)** MPA Block



**(d)** Dynamic and Kinematic blocks

**Figure 8.4:** Updated simulator

# 9 Conclusion

During the traineeship, different features of the simulator have been treated and enhanced, and several tasks have been completed. This thesis reports the activities done describes the procedure that was followed for the creation and implementation of new models that aimed to enhance the simulator performance and correspondence to the reality.

The chapter 2 reports the study that aimed to create a Simulink model as similar as possible to the one implemented in the flight software. The different parameters of the filter have been tuned considering the model previously implemented in the simulator, and the results have been eventually discussed.

The chapter 3 treats the implementation of the complementary filter, that combines the attitude and attitude velocity signals obtained from the IMU0 to estimate the attitude of the gondola. The flight software was studied to obtain a Simulink model that could represent correctly the system behaviour and, then, the results have been analysed and discussed to assert the performance of the new implementation.

The chapter 4 deals with the implementation of the controller that monitors the twist angle of the flight chain and controls the azimuth motor. Its Simulink model has been created and its behaviour and effects on the simulator have been tested.

The chapter 5 reported two different models for the reaction wheels acting on along $X_{NA}$ and $Y_{NA}$ that could take into account the friction. Among the two, the second one seemed to be more suitable since it could represent more accurately the wheel behaviour when its velocity crosses 0. The corresponding model has been implemented in the simulator, but a precise tuning of the

friction parameters is required to obtain a realistic model.

The chapter 6 reports the study that aimed to solve the inconsistency problem among the output signals of the old Kinematic block by defining a new convention that could identify uniquely the velocities, correcting several minor errors that affected the previous version of the simulator, and creating a totally new kinematic model that could correctly sample the time and manipulate the different signals. The new model has been tested and eventually integrated in the simulator, observing a smaller inconsistency between the retrieved signals. Further investigations could be done to assert if the observed error is due to numerical error, as hypothesized.

The chapter 7 deals with the implementation of a first camera model to obtain the pixel coordinates of the star and, from them, the quaternion that represents the rotation between the camera line of sigh and the direction of the star. A simplified and separated model, that could represent the system kinematic, has been created and used to test the camera model and pointing procedures. Eventually, the results have been discussed highlighting that the model should be further tested to assert the model performance.

Among the different topics treated, the implementation of the friction models for the reaction wheels and the creation of the new kinematic model have been the most demanding. The first one required the resolution of several numerical errors, while the second one required the debugging of the entire simulator.

Considering the current state of the simulator, different possible features could be developed or further enhanced, such as:

- A better characterization of the friction phenomenon to correctly tune the friction parameters of the reaction wheel models
- A more precise camera model that could take into account the thermal deformations
- A different approach to solve the algebraic loops in the simulator

Nevertheless, the implemented models proved a general enhancement of the system performance, at a relatively low computational cost, and introduced several new features that were not present before.

Moreover, by considering the technical topics treated in this thesis and the additional accomplishments achieved during the traineeship, it is possible to assert that this experience was successful and extremely fulfilling.

# References


[1] E. Kassarian, F. Sanfedino, D. Alazard, H. Evain, and J. Montel, "Modeling and stability of balloon-borne gondolas with coupled pendulum-torsion dynamics," *Aerospace Science and Technology*, vol. 112, 2021.

[2] B. Guo and J. Zhu, *Signals and Systems*. Berlin, Boston: De Gruyter, 2018.

[3] C. de Wit, C., Olsson, H., Astrom, K.J., and P. Lischinsky, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

[4] X. Wang, S. Lin, and S. Wang, "Dynamic friction parameter identification method with lugre model for direct-drive rotary torque motor," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–8, 2016.

# A

# Additional Studies

This chapter gathers two additional and independent studies that have been done during the traineeship and that do not influence the performances of the simulator.

## A.1 Azimuth ramp error

This section reports the study conducted on the error between the azimuth objective and azimuth filtered by two low pass filters. An analytical approach is exposed, the results are discussed and eventually a possible solution that aims to reduce the error is proposed.

### A.1.1 Motivations

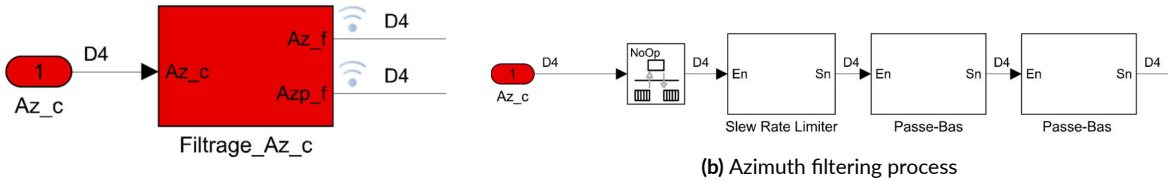
Keeping in mind that the main objective is to enhance the performance of the system in order to achieve a better pointing precision, the behavior of the Azimuth Control block contained in the MPA block was studied. More precisely, the effects that the filter has on the pointing accuracy have been investigated.

By running a simulation, one can observe that the azimuth objective, once that the system reaches the steady state and starts tracking the Sun or a star, varies in time like a sinusoidal function that has a period $T \approx 24$ h. Locally, this function can be approximated as a straight line and therefore, it can be seen as a ramp function.

Feeding the subsystem with a ramp signal as input yields to a filtered signal that presents a certain offset with respect to the original one, as deeper analysed later. The aim of this study is to estimate this steady state error and try to remove it.

### A.1.2 Analytical approach and implementation

Figure A.1 reports the filter block contained inside the MPA block. It is possible to notice that the azimuth objective signal is first limited by a Slew Rate Limiter and then filtered by two low pass filters that have the same time constant $\tau$.




**(a)** Filter block

**(b)** Azimuth filtering process

**Figure A.1:** Representation of the azimuth filter

The system can be represented as schematically shown in Figure A.2, where $E(s)$ is given by Equation A.1, where A is the slope of the ramp and $\gamma(t)$ is 0 for $t < 0$ and 1 otherwise.

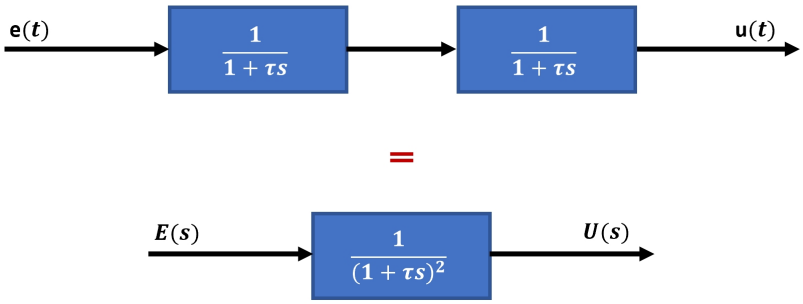



**Figure A.2:** Schematic representation of the filtering process

$$e(t) = At\gamma(t) \rightarrow E(s) = \frac{A}{s^2} \tag{A.1}$$

By using the Theorem of the Final Value, it is possible to compute the steady state error as indicated in Equation A.2.

$$\begin{aligned}\epsilon(t=\infty) &= \lim_{s\to 0} s\epsilon(s) \\ &= \lim_{s\to 0} s(E(s)-U(s)) \\ &= \lim_{s\to 0} s\left(\frac{A}{s^2}-\frac{A}{s^2}H(s)\right) \\ &= 2A\tau\end{aligned} \tag{A.2}$$

From which it is evident that the grater A and $\tau$ are, and the grater the error is. In fact, if the ramp is very steep, the initial error, that will be carried in the future, between the first point on the ramp and the state before will be great. Moreover, the grater $\tau$ is and the slower the response of the system will be. However, it is important to notice that the MPA works in the discrete time domain, and therefore, it is necessary to understand if the discrete time filtering adds an additional error to the one previously calculated. It is convenient to use the z-transform and move from the $s$ domain to the $z$ one. Many conversions are possible, but the most used one is the Tustin's transformation that is reported in Equation A.3.

$$s = \frac{2}{T_e}\frac{z-1}{z+1} \tag{A.3}$$

Equation A.4 reports the calculation of $H(z)$, starting from $H(s)$, while Equation A.5 the error between the input and output signals $\epsilon(z)$.

$$H(s) = \frac{1}{1+\tau s}\frac{1}{1+\tau s} = \frac{1}{(1+\tau s)^2} = \frac{1}{1+2\tau s+\tau^2 s^2}$$

And by using the Tustin's transformation, one obtains:

$$\begin{aligned}H(z) &= \frac{1}{1+2\tau\left(\frac{2}{T_e}\frac{z-1}{z+1}\right)+\tau^2\left(\frac{2}{T_e}\frac{z-1}{z+1}\right)^2} \\ &= 1-\frac{T_e^2(z^2+2z+1)}{(T_e^2+4\tau T_e+4\tau^2)z^2+(2T_e^2-8\tau^2)z+(T_e^2-4\tau T_e+4\tau^2)}\end{aligned} \tag{A.4}$$

$$\epsilon(z) = E(z) - U(z) = E(z)(1 - H(z))$$
$$= E(z)\left(1 - \frac{T_e^2(z^2 + 2z + 1)}{(T_e^2 + 4\tau T_e + 4\tau^2)z^2 + (2T_e^2 - 8\tau^2)z + (T_e^2 - 4\tau T_e + 4\tau^2)}\right)$$

And by remembering that the Theorem of the Final Value in the z-domain is $\epsilon(t = \infty) = \lim_{z \to 1}(z - 1)\epsilon(z)$, one obtains:

$$\epsilon(t = \infty) = 2A\tau \tag{A.5}$$

Equation A.5 proves that the steady state error does not depend on the chosen sampling period since the continuous and discrete time implementation are subjected to the same error. To compare the discrete time behavior with the continuous one, the Simulink model represented in Figure A.3, was created remembering the analogy between the z-domain and the discrete time domain ($x(n - k) = X(z)z^{-k}$).

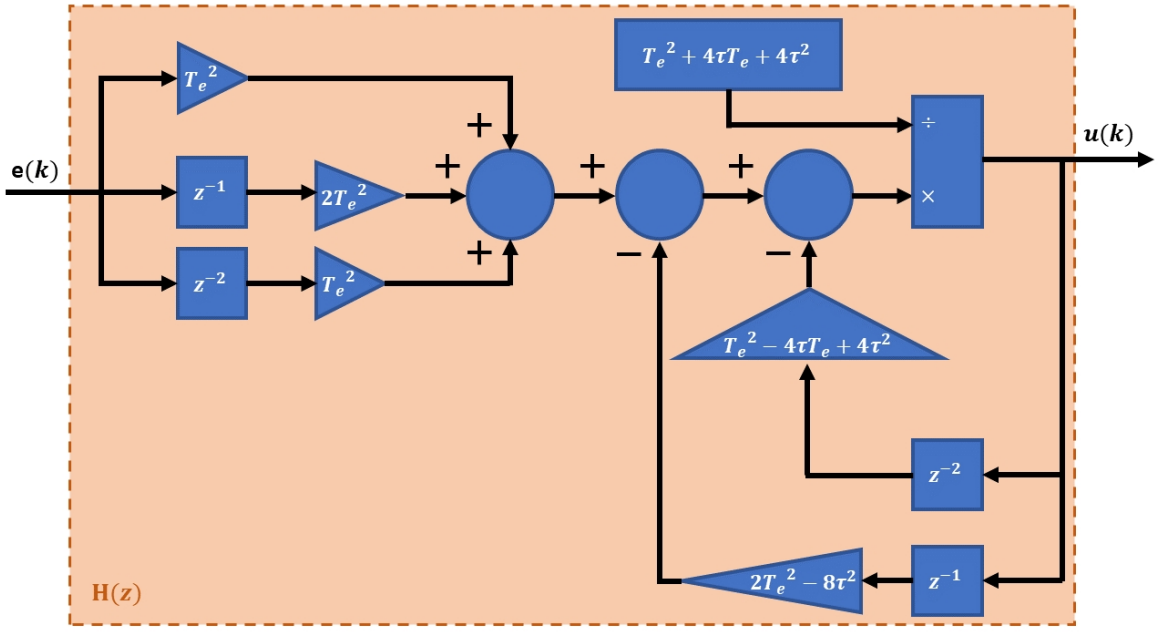
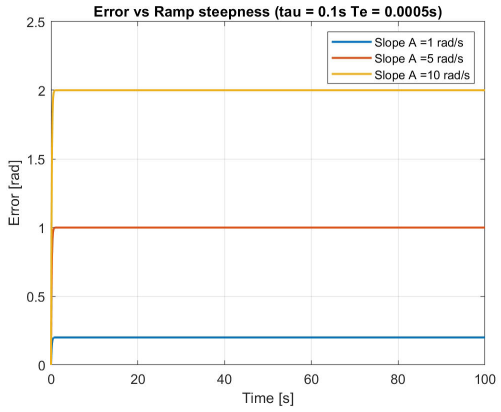
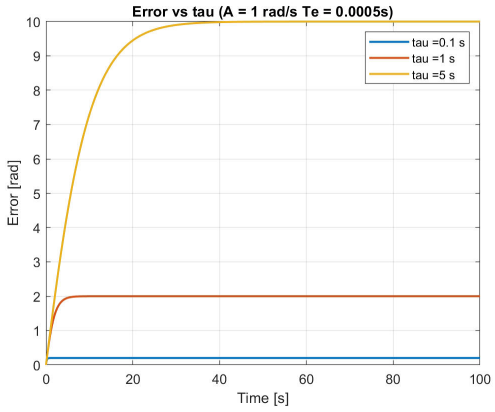


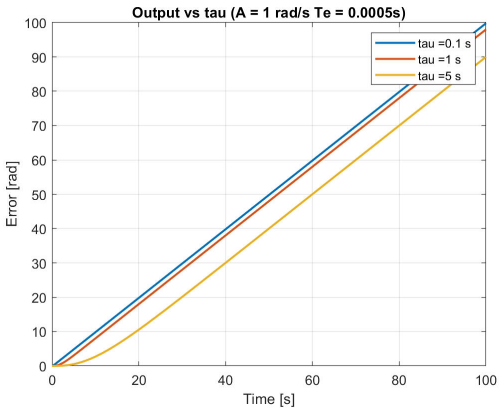

**Figure A.3:** Schematic representation of the discrete time filtering process




**(a)** Error vs. Ramp steepness




**(b)** Error vs. $\tau$




**(c)** Output signal

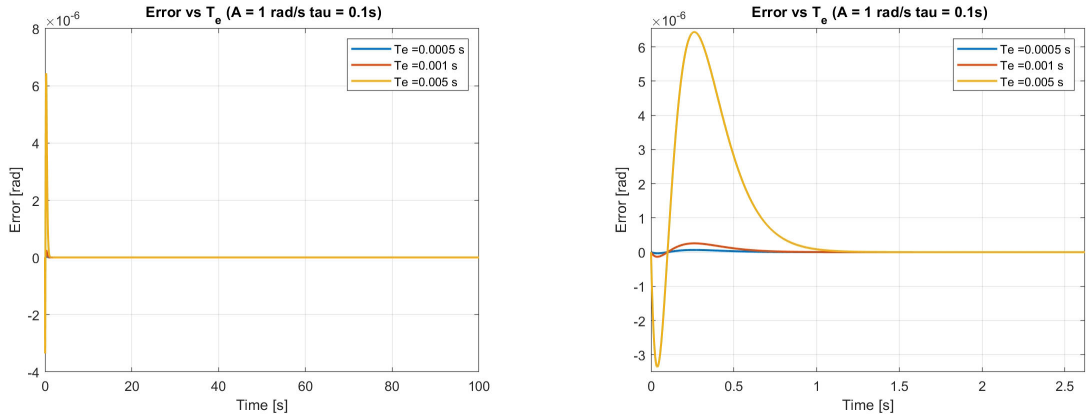
**Figure A.4:** Signal comparison

### A.1.3 Discussion and comparison of the results

It is possible to notice from Figure A.4a and Figure A.4b the linear dependence of the error $\epsilon(t)$ from the ramp steepness $A$ and time constant $\tau$. Moreover, Figure A.4c shows that increasing $\tau$ makes the system response slower.

As visible from Figure A.5, the steady state error is the same in all cases since there was no dependence from the sampling period. On the other hand, one can notice that the greater the sampling period is, the grater the error in the first instants of the simulation is. This is due to the fact that, the input chances quickly at the beginning and, therefore, the discrete time filter cannot "track" its trend.

In conclusion, one can say that there is no difference between the discrete and continuous time implementation if the sampling period is sufficiently small.




**(a)** Error vs. $T_e$

**(b)** Zoom on error vs. $T_e$

**Figure A.5:** Signal comparison

### A.1.4 Steady state error compensation

Since the steady state error is constant and depends only on the steepness of the ramp and the time constant of the low pass filters, a possible approach to compensate it was investigated.

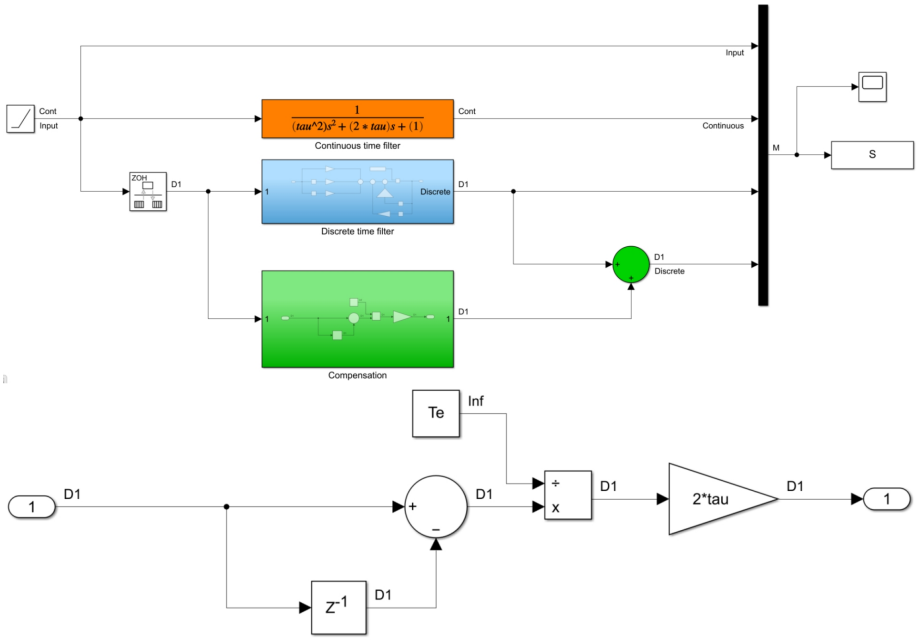



**Figure A.6:** Simulink model for the steady state error compensation

Figure A.6 represents the model implemented in Simulink, where the green elements represent

are responsible for the error compensation. The idea was to add to the input signal the steady state error calculated with Equation A.5, where $A = \frac{e(k) - e(k-1)}{T_e}$. As a consequence, the compensation removes effectively the steady state error but, on the other hands, creates a step at the beginning of the simulation, as visible from Figure A.7.

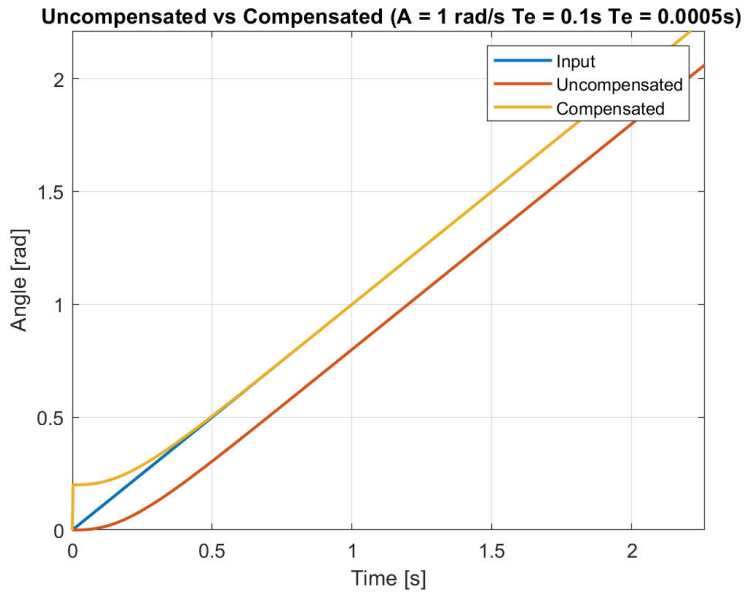



**Figure A.7:** Compensated and uncompensated output signals

The error of the uncompensated and compensated signals with respect to the input signal is reported in Figure A.8.

A possible solution to limit the initial step behavior of the compensated output and, therefore, the error at the beginning of the simulation, is to use a low pass filter as reported in Figure A.9.

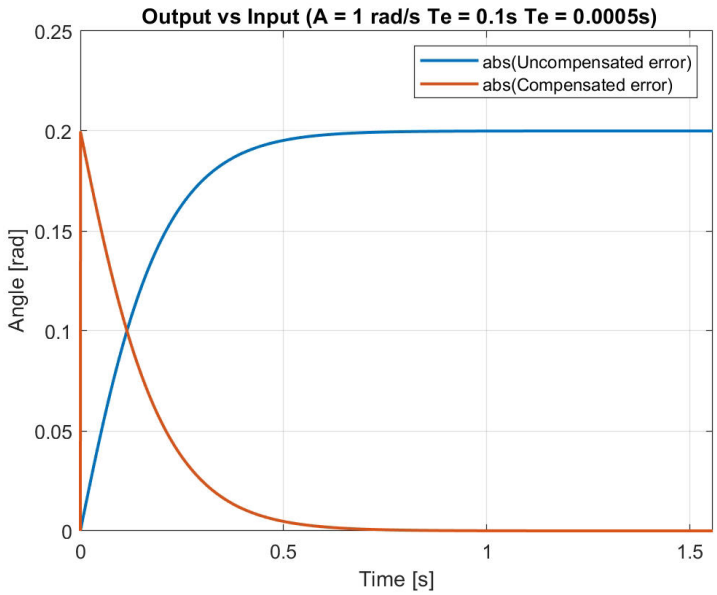



**Figure A.8:** Compensated and uncompensated errors

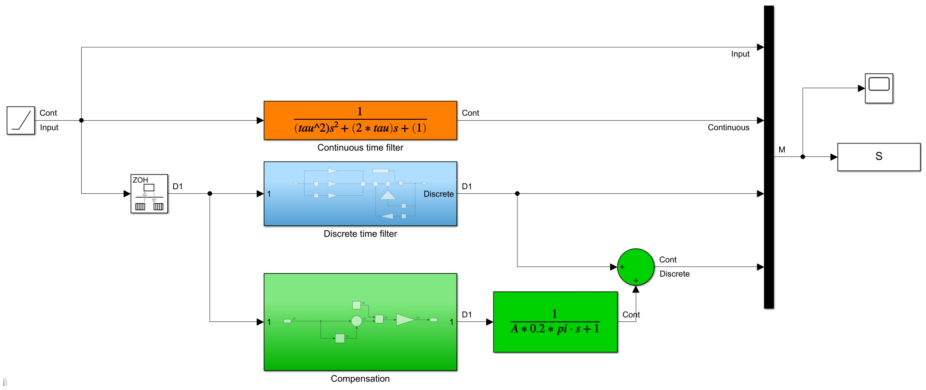



**Figure A.9:** Simulink model for the steady state error compensation with attenuated step behaviour

As a result, by using a time constant $\tau_{correction} = 0.2A\pi$, one obtains Figure A.10.

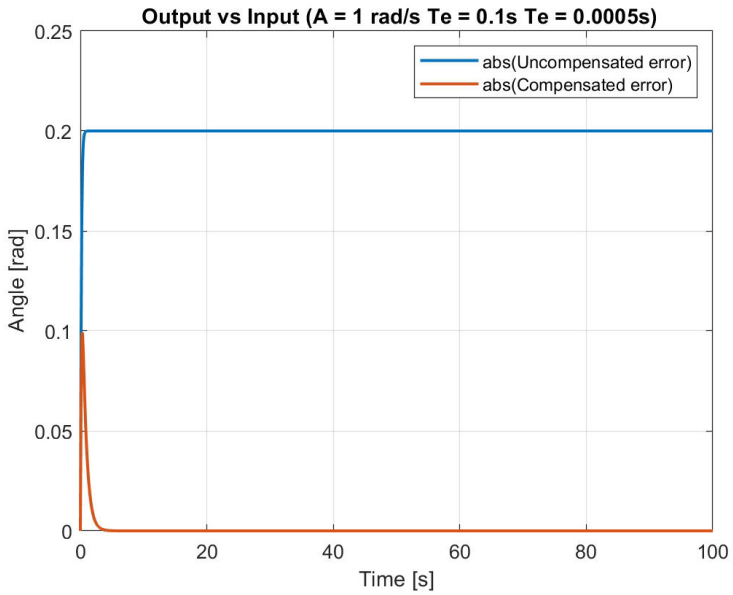



**Figure A.10:** Compensated and uncompensated errors with attenuated step behaviour

In conclusion, the conducted study showed that the action of filtering a ramp signal, representative of the azimuth objective, in the continuous or discrete time domains yields to the same steady state error for both cases. The discrete time filtering adds an additional error due to the sampling period that affects the output in the first stages of the simulation. To remove the steady state error, it is possible to add a compensator that, as drawback, adds an error in the first stages of the simulation. To remove it, an additional low pass filter can be added, tuning well the $\tau_{correction}$ in order to achieve a good compromise between the error magnitude and the time at which it reaches zero.

## A.2 Unwinding pendulum

In this section, a study that aimed to prove the feasibility of damping the pendulum motion of the gondola by simply unwinding the flight chain was conducted. An analytical approach is proposed, the results are presented and eventually the feasibility is discussed.

### A.2.1 Idea

The main idea was to approximate the whole system as a big pendulum that, taking into account the great inertia of the balloon, can be considered oscillating around a point $A$. Although more precise models that consider for example the flight chain composed b $n - bodies$ could be implemented, for a first feasibility study, the system reported in Figure A.11 was considered to be sufficiently accurate.

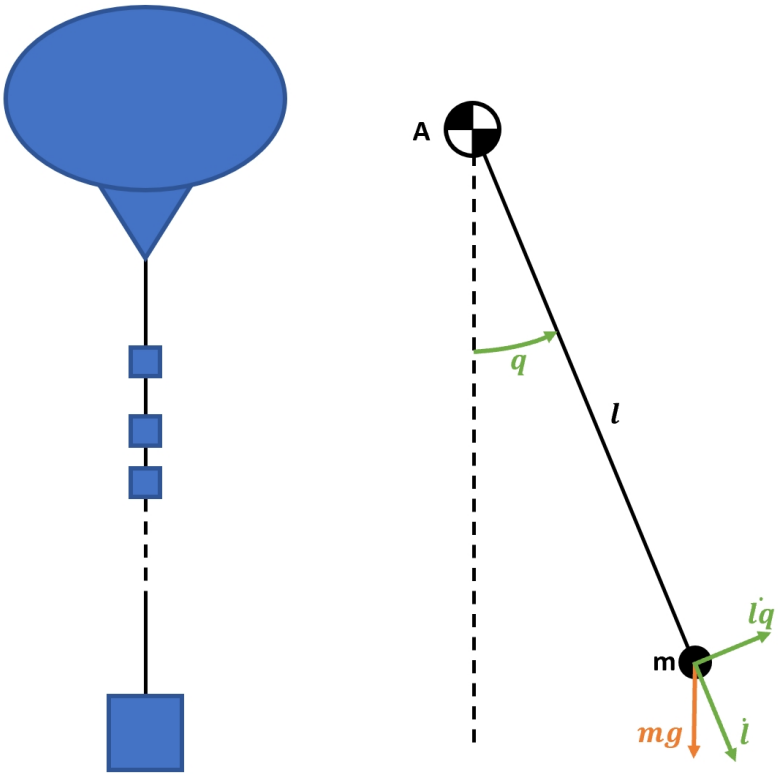



**Figure A.11:** System approximation

### A.2.2 Analytical approach and implementation

To study the phenomenon, one can use the Lagrange's equation, in which the kinetic and potential gravitational energy are respectively reported in Equation A.6 and Equation A.7.

$$T = \frac{1}{2}mV^2 = \frac{1}{2}m(l^2\dot{q}^2 + \dot{l}^2) \tag{A.6}$$

$$U = -mgl\cos(q) \tag{A.7}$$

Therefore, the equation of motion linearized for small angles can be retrieved as indicated in Equation A.8.

$$\begin{aligned}
&L = T - U = \frac{1}{2}m(l^2\dot{q}^2 + \dot{l}^2) + mgl\cos(q)\\
&\frac{\partial L}{\partial \dot{q}} = ml^2\dot{q}\\
&\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = ml^2\ddot{q} + mgl\sin(q) + 2ml\dot{q}\dot{l} = 0\\
&\ddot{q} + \frac{2\dot{l}}{l}\dot{q} + \frac{g}{l}\sin(q) = 0\\
&\ddot{q} + \frac{2\dot{l}}{l}\dot{q} + \frac{g}{l}q = 0
\end{aligned} \tag{A.8}$$

From Equation A.8 it is possible to define the equivalent stiffness coefficient $K_{eq} = \frac{g}{l}$ and the equivalent damping coefficient $C_{eq} = \frac{2\dot{l}}{l}$. The first one is always positive, while the second one has a sign that depends on the sign of $\dot{l}$:

- if $\dot{l} > 0$, $C_{eq} > 0$, the pendulum unwinds and the system is stable
- if $\dot{l} < 0$, $C_{eq} < 0$, the pendulum rolls up and the system is unstable. In fact, part of the energy provided to the system is used to increase $U$ (since $h$ increases), while the other part increase the oscillations of the pendulum.

Clearly, the friction with air would contribute to damp the oscillations since it would increase $C_{eq}$, but it can be neglected in this studt since at $40km$ of altitude its effect is limited. Moreover, increasing $\dot{l}$ reduces the required time to unwind the flight chain, and therefore the number of

oscillations, but increases their magnitude. Since this is a 2 DoF system, an equation for $l$ should have been written too. However, this would have been out of the scope of this study and, therefore, it was not considered.

The system dynamics has been simulated on MATLAB using Ode45. A constant unwinding velocity and the initial conditions reported below have been considered. The simulation automatically ends when the length of the flight chain reaches the final length.

```
% INITIAL CONDITIONS
ic_starting.angolo=deg2rad(5);          %[rad] - Initial angular displacement
ic_starting.vel_angolare=0;             %[rad/s] - Initial angular velocity
ic_starting.vel_lineare=e.vel_lineare;  %[m/s] - Initial linear velocity
% CONSTANTS DEFINITION
e.g=9.81; %[m/s^2]
e.lunghezza_finale=100;                 %[m]   - Final length of the chord
e.lunghezza_iniziale=10;                %[m]   - Initial length of the chord
e.vel_lineare=0.1;                      %[m/s] - Unrolling velocity of the chord
```

An initial amplitude of $5$ was considered. Note that, since the equation was linearized, the higher the initial angle is and the less accurate the results will be. The initial and final length of the flight chain are respectively $10\,\mathrm{m}$ and $100\,\mathrm{m}$, but they can be adjusted considering a more realistic scenario.

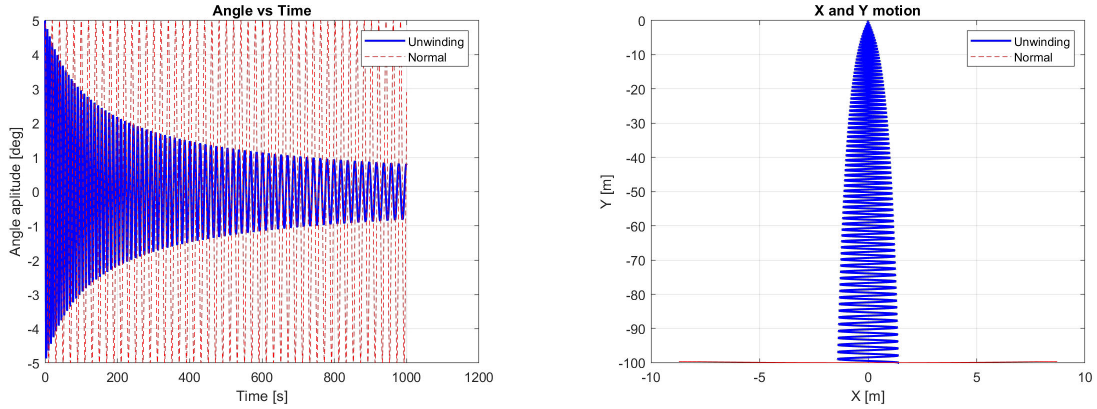
### A.2.3 Discussion of the results

The equation of motion of a normal pendulum is retrieved in Equation A.9.

$$
\begin{aligned}
&L = T - U = \frac{1}{2} m l^2 \dot{q}^2 + + mgl\cos(q) \\
&\frac{\partial L}{\partial \dot{q}} = ml^2\dot{q} \\
&\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \ddot{q}l + g\sin(q) = 0 \\
&\ddot{q}l + gq = 0 \\
&\ddot{q} + \frac{g}{l}q = 0
\end{aligned}
\tag{A.9}
$$

By propagating it and the equation relative to an unwinding pendulum, it was possible to obtain

the plots reported in Figure A.12. Figure A.12a clearly shows that using an unwinding pendulum reduces the angular amplitude of the oscillations, while from Figure A.12b it is clear that the linear oscillations along the X direction increases as a consequence of the flight chain length growth. However, the X amplitude is considerably low if compared to the one obtained if the pendulum would oscillate with a constant length of $100m$.




**(a)** Angular oscillations

**(b)** Linear oscillations

**Figure A.12:** Simulation results

The obtained results can be considered promising. However, the reality is more bitter since the flight chain is not properly a single simple body, since it is made of several pieces and parts that contains sensors and, sometimes, fragile components. Moreover, its length and nature make it hard to be folded and even harder to be outspread following a desired velocity profile. Therefore, additional studies are required to definitely determine the feasibility of the proposed idea.