ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCHOOL OF ENGINEERING
Campus of Forlì

Bachelor's degree in
AEROSPACE ENGINEERING
Class L-9

GRADUATION THESIS IN
Satellites and Space Missions

Modeling and simulation of a CubeSat atmospheric re-entry trajectory

CANDIDATE
Francesco De Cecio

SUPERVISOR
Prof. Alfredo Locarini

Academic Year 2018/2019

Ai miei nonni, esempi di vita

ABSTRACT

In the recent years, a growing concern about space debris is forcing space agencies and space sector actors to think new solutions to limit this phenomenon. A valuable strategy is to let satellites in LEO to re-enter in the atmosphere at the end of their mission. Understand on which parameters to act in order to be compliant with international guidelines is becoming more and more important. Furthermore, a better comprehension of the physics involved in an atmospheric re-entry could help to propose innovative solutions for enabling small spacecrafts to survive it and perform a new kind of missions. The complexity of this problem is related also to the difficulty of recreating similar conditions on Earth to perform tests and analyses. For this purpose, in the framework of this thesis project, an academic re-entry simulation tool, with a particular focus on CubeSat applications, has been developed. The current version is represented by an open-source fully customizable MATLAB/Simulink implementation of existing models. Some models have been simplified to fit the needs of academic inexpert users and help those approaching the problem for the first time. A performance comparison with other available tools is provided as well as a list of known issues to be solved.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Acronyms

AoA	Angle of Attack
ARES	Academic Re-Entry Simulator
BC	Ballistic Coefficient
CIRA	Italian Centre for Aerospace Research
COTS	Commercial Off-the-Shelf component
DAS	Debris Assessment Software
DOF	Degrees of Freedom
ECSS	European Cooperation on Space Standardization
ESA	European Space Agency
EUV	Extreme Ultra-Violet emission
ISO	International Organization for Standardization
LEO	Low Earth Orbit
NASA	National Aeronautics and Space Administration
QARMAN	Qubesat for Aerothermodynamic Research and Measurements on AblatioN
RB	(atmosphere) Re-entry Body
S/C	Spacecraft
SFU	Solar Flux Unit
TPS	Thermal Protection System
VKI	Von Karman Institute for Fluid Dynamics

1. Introduction

1.1. Background

In the current evolution of space industry, one of the most important trends is represented by the spacecraft miniaturization process. A reduction in size of both bus platforms and payloads has allowed an increase of performances, which in turn reduces the overall mission costs. Constellations of several satellites have started to be delivered in orbit, replacing what more massive ones were previously achieving [1].

In this context, the CubeSat standard is becoming widely adopted in both academia and companies. Such a process aims at enabling new players to access space economy, breaking down typical barriers of a space mission. First of all, with a high level of standardization, the expertise needed in order to launch a CubeSat is not as huge as in the past. The use of COTS components is cutting the required time to have a spacecraft ready to launch. Furthermore, the industrial production of entire subsystems is reducing costs and enhance reliability of these platforms.

For these reasons, the number of small satellites is expected to increase exponentially in the next decades [2]. Despite these advantages, there is a concern that CubeSats may increase the number of space debris. In order to mitigate this potential problem, several debris removal possibilities are under investigation. So far, none of the tested removal strategies seems to be effective enough to tackle this problem. Therefore, to prevent the further generation of space debris during and after the useful lifetime of a spacecraft, NASA, ESA, and other space agencies have set several guidelines which must be met for missions commissioned by those agencies [3]. For example, the European Cooperation on Space Standardization (ECSS) adopted ISO-24113 in the space sustainability branch. In the next years, it is likely that the guidelines will be converted into actual regulations. While some countries have already taken this step and reflected space debris mitigation in their national regulations [4], worldwide implementation is still pending.

Planning to end a CubeSat mission with an atmospheric re-entry from LEO is one suggested method to be compliant with those requirements.

1.2. SCOPE OF THIS THESIS

The idea behind this work is to propose a strategy that puts together the need to de-orbit within a limited amount of time and the possibility to perform a partially controlled non-destructive re-entry. For this purpose, a very efficient thermal protection system (TPS) is essential. Developing a heat shield for CubeSat application is not an easy task, the mass should be kept as low as possible, but at the same time it must protect the spacecraft and its systems from severe heating conditions. In addition to this, to meet the market demand, it should be standardized, modular and easy to install.

In this way, several benefits can be obtained, for instance retrieval of parts or data designed to survive the re-entry is allowed. Moreover, this could enable CubeSats to perform new kind of missions and to conceive subsystems able to be flown more than one time, strongly cutting manufacturing costs.

1.3. RE-ENTRY MODELING

In order to dimension properly a heat shield, an open source spacecraft re-entry analysis tool is needed.

Since 1960's, many studies have been done by space agencies in understanding how to simulate and predict re-entry, for both strategical and human space flight concerns. Indeed, it is still difficult to get experimental data on Earth for the flow condition experienced by a s/c trough the atmosphere. Only in the recent years, the availability of powerful and big enough plasma wind tunnels (e.g. SCIROCCO facility at CIRA) is changing this paradigm and offering a tool to be correlated with numerical simulations, even though in most of the cases results must be scaled to be usable. In addition to this, doing accurate predictions about re-entry is extremely difficult, due to great variability of some factors, the most relevant of which is the atmospheric density of the upper atmosphere, changing from day to day in strong relation with solar activity [5], as better described in section 2.1.

In this respect, in the 1990's and 2000's numerous software like NASA's DAS and ORSAT, and ESA's SCARAB and DRAMA, have been developed. Unfortunately, none of this software is open source or user modifiable as well as being all released only to authorized users with a formal request to their institutions.

This thesis presents a simplified model, implemented in MATLAB and Simulink, for calculating the effects of the Earth's atmosphere on a CubeSat orbiting in LEO, with the benefit of being fully customizable. A rough estimation of both inertial and aerothermal loads is provided by the software, together with the dynamic and kinematic parameters of the whole re-entry trajectory. The presented codes are intended to be used in the early design phase of the S/C. Further and more refined analyses are needed in order to optimize and qualify a hypothetical TPS payload system. An overview of the simulator's capabilities is presented in Table 1.

Inputs	Outputs (a sample per integration step)
s/c mass	Altitude
s/c cross sectional area	Travelled distance
s/c drag coefficient	Time elapsed
s/c lift coefficient	Velocity (magnitude and components)
Initial time of simulation	Flight path angle
$F_{10.7}$ solar index	True anomaly
K_p geomagnetic index	Density
Orbital radius at apoapsis	Acceleration
Orbital radius at periapsis	Inertial load
Initial position	Aerothermal load
Initial velocity	Total time to demise
Initial flight path angle	
Initial true anomaly	
Initial delta V	
Integration step	

Table 1 - overview of simulator inputs and outputs

2. Models

Existing models have been selected for each aspect covered by this simulation tool.

Several assumptions have been done in order to reduce the huge complexity of the physical phenomena involved in this kind of problem. For each section, the most significant are listed.

For those who are interested in deepening the knowledge in modeling satellite aerodynamic drag a complete work is represented by *“A critical assessment of satellite drag and atmospheric density modeling”* (Vallado and Finkleman, 2014) [6].

2.1. Atmosphere

Modeling Earth atmosphere in a proper way is not an easy task, and requires taking into account several parameters, as can be found in section 3.5.1 of the book *Satellite Orbits* [7] and in section 8.6.2 of the book *Fundamentals of Astrodynamics and Applications* [8].

The most significant source of variability in predicting upper atmosphere density is represented by solar activity. When the Sun is particularly active, adds extra energy to the atmosphere heating it. Low density layers of air at LEO altitudes rise and are replaced by higher density layers that were previously at lower altitudes. Since drag force is closely related to density, in these conditions decay rate would increase.

The solar radio flux at 10.7 cm wavelength (2800 MHz) is an excellent indicator of solar activity and has proven very valuable in specifying and forecasting space weather. The F10.7 radio emissions originate high in the chromosphere and low in the corona of the solar atmosphere and it tracks other important emissions that form in the same regions of the solar atmosphere. It is well correlated with the sunspots number as well as Extreme Ultra-Violet (EUV) emissions that impact the ionosphere and modify the upper atmosphere. It is a long record and provides an history of solar activity over six solar cycles. Because this measurement can be made reliably and accurately from the ground in all weather conditions, it is a very robust data set with few gaps or calibration issues [9].

In addition to these long-term changes in upper atmospheric temperature and density caused by the 11 years solar cycle, interactions between the solar wind and the Earth’s magnetic field during geomagnetic storms can produce large short-term increases in upper atmosphere temperature and density. For this reason, the K-index quantifies disturbances in the horizontal component of earth's magnetic field with an integer in the range 0-9 with 1 being calm and 5 or more indicating a geomagnetic storm. It is derived from the maximum fluctuations of horizontal components observed on a magnetometer during a three-hour interval. The planetary 3-hour-range index K_p is the mean standardized K-index from 13 geomagnetic observatories between 44 degrees and 60 degrees northern or southern geomagnetic latitude.

Both these phenomena are difficult to predict accurately, so space weather forecasts are affected by an uncertainty growing with the time, in particular long-term predictions about solar maxima are particularly challenging [6], as can be seen in Figure 1.

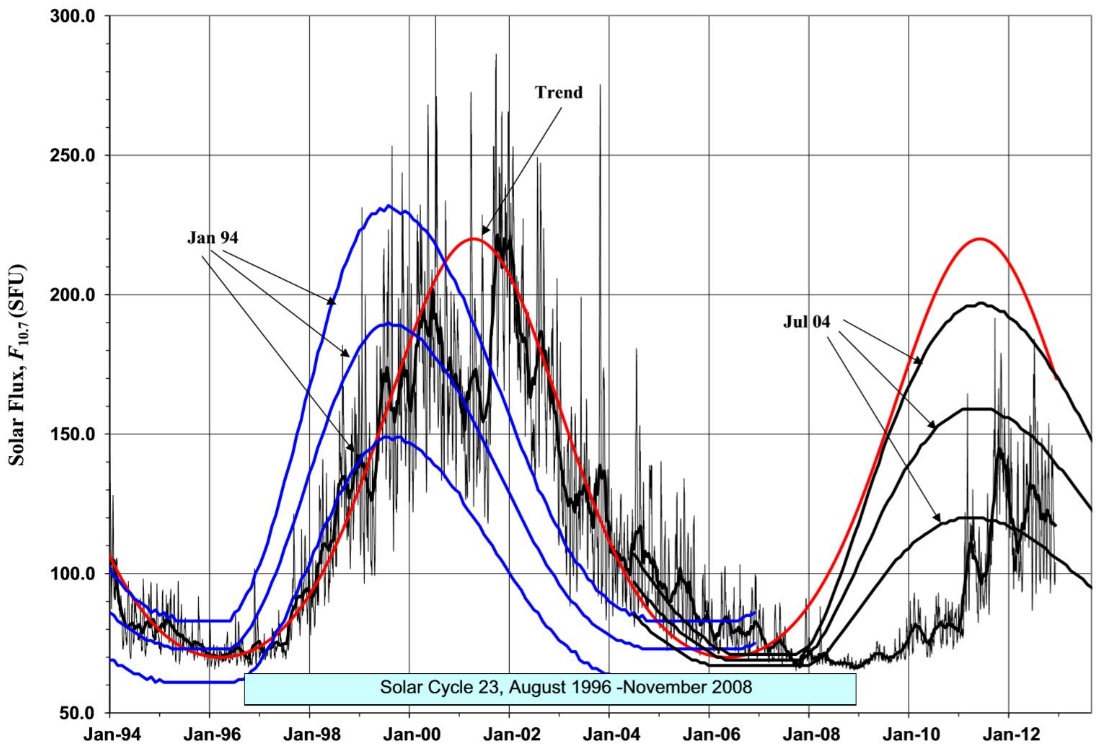


Figure 1 - solar flux predictions (source: Fig. 6 of [6])

Although very complex models have been developed (e.g. NRLMSISE-00), even two ‘’high-fidelity” models may produce markedly different results [6]. Considering this, two relatively simple models are implemented in the simulator: Jacchia J71 for upper atmosphere and Exponential Atmosphere for lower regions.

2.1.1. JACCHIA 1971

Jacchia J71 has been selected for calculating the density in a range of altitudes between 90 and 2500 km. The original model has been lightly simplified by some additional assumptions listed in the following section. This model has been chosen because it allows to choose the desired level of refinement of its results. All the basics computations performed by this model are quite simple and can be refined in a stepped way considering how many information are available. Reversely, Harris-Priester model has been discarded because it cannot be used unless the position vectors of both Sun and satellite in an Earth centred reference frame are known.

This model has been implemented in MATLAB in functions *J71_density* and *J71_density_simulink* (the latter compatible with Simulink and employed in the simulator) using the approach presented in [7], chapter 3.5.3. Atmosphere is assumed to be in diffusion equilibrium, where the constituents N_2, O_2, O, Ar, He and H_2 are considered. The computation of atmospheric density is performed as described below:

1. The exospheric temperature T_C is computed from data on solar activity [10]

$$T_c = 379.0° + 3.24°\bar{F}_{10.7} + 1.3°(F_{10.7} - \bar{F}_{10.7})$$

- $F_{10.7}$ is the average solar flux at 10.7 cm wavelength over the day before the date on which the density is evaluated;
- $\bar{F}_{10.7}$ is the average solar flux at 10.7 cm wavelength over the last three solar rotations of 27 days (both measured in Solar Flux Units SFU of $10^{-22}\frac{W}{m^2 Hz}$).

Hence, value of T_C is corrected for variations due to solar geomagnetic activity [10]

$$\Delta T_\infty^H = 28.0° K_p + 0.03° e^{K_p} \quad (H > 350\ km)$$

$$\Delta T_{\infty}^{L} = 14.0° K_p + 0.02° e^{K_p} \quad (H < 350\ km)$$

- K_p is the three-hourly planetary geomagnetic index for a time 6.7 hours earlier than the time under consideration;
- H is the altitude;
- a transition function f is implemented to avoid discontinuities at $H = 350\ km$

$$f = \frac{1}{2}\left(\tanh\left(0.04(H - 350)\right) + 1\right)$$

Finally, for corrected exospheric temperature T_{∞}

$$T_{\infty} = T_c + f\Delta T_{\infty}^{H} + (1 - f)\Delta T_{\infty}^{L}$$

2. Once T_{∞} is known, a temperature profile is assumed. Thus, diffusion equation should be integrated using the profile as input but turns out to be too time-consuming. For the scope of this thesis, a bi-polynomial fit for the computation of the standard density values proposed by Gill (1996) [11] has been used

$$log\ \rho\ (H, T_{\infty}) = \sum_{i=0}^{5}\sum_{j=0}^{4} c_{ij}\left(\frac{H}{1000\ km}\right)^{i}\left(\frac{T_{\infty}}{1000\ K}\right)^{j}$$

   - c_{ij} coefficients are provided by Gill's work and retrieved by MATLAB function *getCoeff(H, T_{∞})* available in the annexes of this document;
3. Some corrections are now applied to the density:
   - Additional geomagnetic activity correction below 350 km of height;
   - semi-annual density variation in thermosphere, introducing a time-dependent parameter.

The function in the end returns the corrected density value.

2.1.1.1. Additional assumptions

For the scopes of this thesis, in addition to Jacchia's hypothesis [10], further assumptions are introduced in order to have a simplified model.

- Diurnal variations of exospheric temperature are neglected;
- Seasonal-latitudinal density dependence is neglected.

Therefore, adding the latest assumptions, density is made constant over the Earth for a certain altitude. In this way a small error is introduced, but the knowledge of latitude and longitude is no more a requirement.

2.1.2. Exponential atmosphere

As regards lower atmosphere (below 100 km of height), a simpler exponential model has been chosen. The reason is due to less dependence on solar activity as the altitude decreases. Exponential model is static, and assumes an exponential decay of density from a starting given value, according to

$$\rho = \rho_0 e^{-\frac{H-H_0}{SH}}$$

- ρ_0 reference density;
- H_0 reference altitude;
- H actual altitude;
- SH scale height, which is the fractional change in density with height and it is given by $SH = \frac{k_B T}{mg}$ where k_B is the Boltzmann constant, T temperature, m molecular weight, g gravity.

This trivial formula is implemented in *expAtm(H)* function, where the only input is altitude and the only output is density. As a consequence of the great simplicity, accuracy is limited. To improve model performances, altitudes from 0 to 100 km are split in 9 bands, for each of them constants employed in the formula are updated. These values are available in Table 2.

Altitude H (km)	Reference Altitude H_0 (km)	Reference Density ρ_0 (kg/m^3)	Scale Height SH (km)
0-25	0	1.225	7.249
25-30	25	3.899×10^{-2}	6.349
30-40	30	1.774×10^{-2}	6.682
40-50	40	3.972×10^{-3}	7.554
50-60	50	1.057×10^{-3}	8.382
60-70	60	3.206×10^{-4}	7.714
70-80	70	8.770×10^{-5}	6.549
80-90	80	1.905×10^{-5}	5.799
90-100	90	3.396×10^{-6}	5.382

Table 2 - Exponential Atmospheric Model reference values

Table extracted from *table 8-4* present in [8], which in turn is based on the work of Wertz (1978) [12], using the *U.S. Standard Atmosphere* (1976) for 0-25 km and CIRA-72 for 25–100 km.

2.1.3. Continuity check

A test script has been used to verify that predictions from J71 and exponential atmosphere were close to each other in the overlap band of heights (from 90 to 100 km). Results show a difference (at 95 km of altitude) of 26% during solar maxima and 23% during minima.

Density over altitude profile from both models has been plotted to check their functionality, the results are available in Figure 2 and Figure 3. For Jacchia J71 two runs are performed:

- solar minimum – March 2019
- solar maximum – March 2014

Even if the difference is not excessive, keeping in mind that it is the result of a comparison between completely different atmospheric models, better performances could be achieved joining the two curves by means of a polynomial fitting technique.

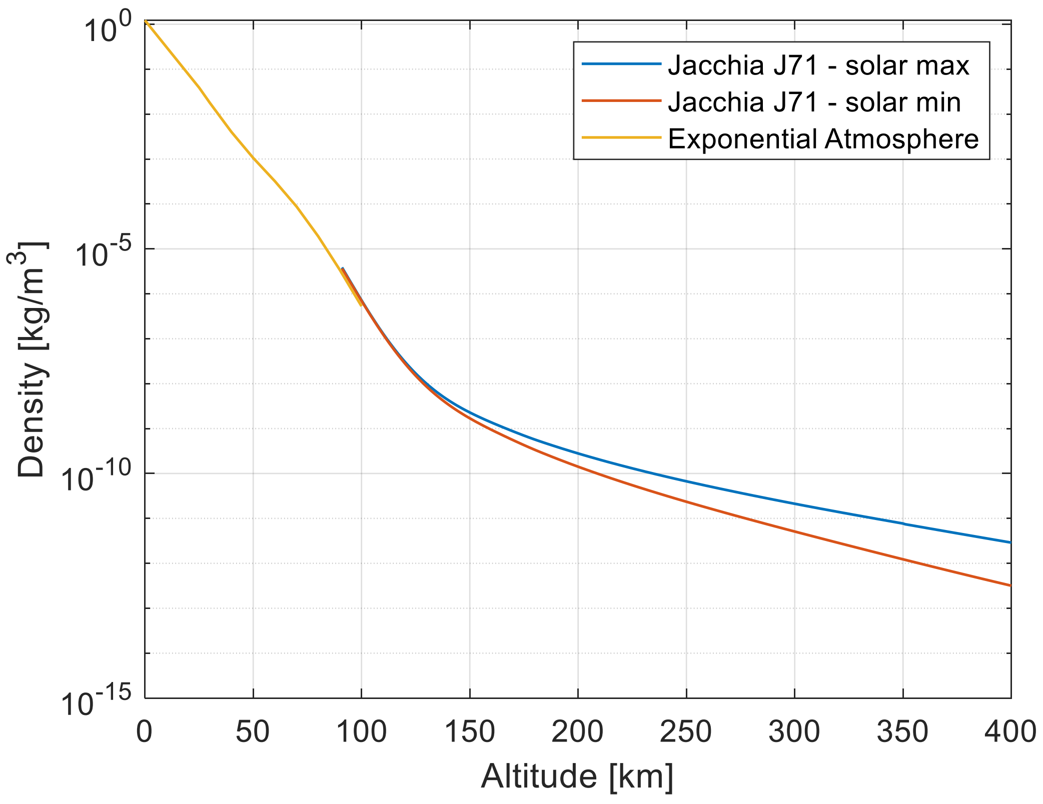


Figure 2 - Jacchia J71 and Exponential Atmosphere comparison

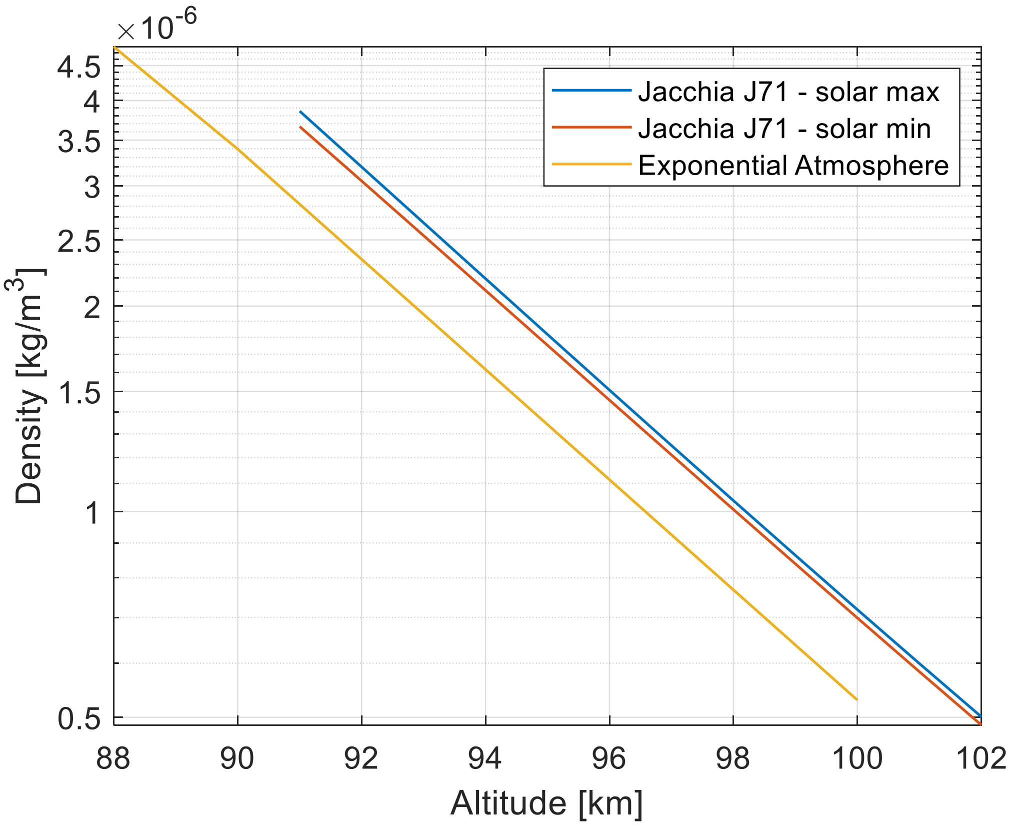


Figure 3 - zoom of previous figure on [90,100] km range

2.2. Dynamics

Determining parameters like speed, position and acceleration of the entire re-entry trajectory with good accuracy is fundamental for a reliable simulation result. When considering aerodynamic forces, a Keplerian description of the motion is reductive. This work is aimed at providing an adequate assessment of the inertial and aerothermal loads acting on an atmosphere re-entering body (RB), with the purpose of being useful to whom are involved in structural design. Therefore, the use of kinematic and dynamics equations of motion has been deemed necessary. For the section 2.2.1 chapter 7 of *Dynamics of Atmospheric Re-Entry* [13] has been used as a guideline.

2.2.1. Equations of planar motion

The purpose of this chapter is to explain what equations, derived from rigid body dynamics equations, drives the motion of RB in the simulator. Planar motion hypothesis, which leads to a great simplification of the complete equations, is adopted. Although it is a huge hypothesis, if the RB is subjected to drag only, or ballistic flight, this restriction results in no loss of generality; nevertheless, if the RV is capable of generating lift forces, then restricting it to a planar trajectory does limit the utility of the results [13].

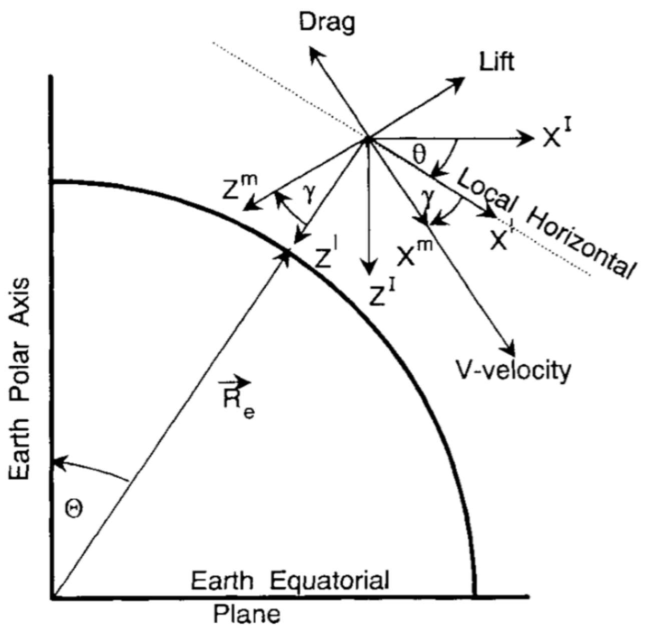


Figure 4 - planar re-entry: reference frames (source: Fig. 7.1 of [13])

The forces acting on the RB that has been considered in this project are the aerodynamic lift, drag, and the gravitational one. Three reference frames, centred in the spacecraft centre of gravity and presented in Figure 4, are used in the description:

1. (X^I, Y^I, Z^I) - An inertial frame such that the (X^I, Z^I) plane contains the velocity vector V throughout the motion;
2. (X^L, Y^L, Z^L) - A local frame such that the Z^L axis is along the local vertical;
3. (X^B, Y^B, Z^B) - A body frame attached to the RB such that the (X^B, Z^B) plane is coincident with the trajectory plane and the X^B axis is always aligned with the velocity vector.

Keep also in mind that: flight path angle γ is considered positive when the velocity vector is below the local horizontal, the gravitational acceleration vector is coincident with the Z^L axis. The aerodynamic forces are in the XZ plane (all axis systems), with drag along the negative X^B axis and lift normal to the X^B axis and aligned with negative Z^B axis. Because the trajectory is confined to a plane, the Y axes of all three systems are coincident and positive in the outgoing direction from the sheet.

Mathematical formulation of the equation for planar motion starts from vector formulation of Newton's Second Law (vectors are in bold)

$$\boldsymbol{\Sigma F^I_{ext}} = m\boldsymbol{a^I}$$

Where the term on the left side

$$\boldsymbol{\Sigma F^I_{ext}} = \boldsymbol{F^I_{aer}} + \boldsymbol{F^I_{grav}}$$

To simple represent forces components, each force is known in a convenient reference frame

$$\boldsymbol{F^B_{aer}} = [-D, 0, -L]^T$$

$$\boldsymbol{F^L_{grav}} = [0, 0, g]^T$$

Making use of pre-multiplied rotation matrices $\boldsymbol{T^T_S}$ (rotation from source frame to target frame)

$$\boldsymbol{F}_{aer}^{I} + \boldsymbol{F}_{grav}^{I} = \boldsymbol{T}_{B}^{I}\boldsymbol{F}_{aer}^{B} + \boldsymbol{T}_{L}^{I}\boldsymbol{F}_{grav}^{L}$$

In body frame Newton's equation becomes

$$\boldsymbol{a}^{B} = \frac{\boldsymbol{F}_{aer}^{B}}{m} + \boldsymbol{T}_{L}^{B}\frac{\boldsymbol{F}_{grav}^{L}}{m}$$

Where the acceleration can be expressed through Poisson's relation. Substituting, the final vector formulation is obtained

$$\dot{\boldsymbol{V}}^{B} + \boldsymbol{\omega}_{B}^{I} \times \boldsymbol{V}^{B} = \frac{\boldsymbol{F}_{aer}^{B}}{m} + \boldsymbol{T}_{L}^{B}\frac{\boldsymbol{F}_{grav}^{L}}{m}$$

Where $\boldsymbol{\omega}_{B}^{I}$ is relative angular speed between body and inertial frame. Writing in components vectors and matrices and separating, the previous formula gives the following two scalar equations for planar motion:

$$\frac{dV}{dt} = -\frac{D}{m} + g\ sin(\gamma)$$

$$V\left(\frac{d\theta}{dt} + \frac{d\gamma}{dt}\right) = -\frac{L}{m} + g\ cos(\gamma)$$

Aerodynamic forces can be expressed with the conventional notation:

$$L = \frac{1}{2}\rho V^{2} S C_{L}$$

$$D = \frac{1}{2}\rho V^{2} S C_{D}$$

S is the reference area, for a controlled re-entry is the cross-sectional area of the s/c in the re-entry attitude, for an uncontrolled one is the maximum cross-sectional area (tumbling s/c approximation). For the C_L, C_D coefficients values see section 2.2.5.2. For re-entry studies, many authors adopt the so-called *Ballistic Coefficient* (BC)

$$BC = \frac{m}{SC_{D}}$$

In addition to the two dynamics equation, which relates velocity magnitude V, flight path angle (or velocity direction) γ, and central angle θ to the forces acting on the satellite, some kinematic relationships are necessary. These can be derived from the geometry associated with the constrained motion.

For a circular orbit, looking at Figure 3, it is possible to use

$$\frac{d\theta}{dt} = \omega_L^I = \frac{V cos(\gamma)}{R_\oplus + h}$$

$$\frac{dh}{dt} = -V sin(\gamma)$$

Where h is the altitude of RB and $R_\oplus$ is Earth radius.

This system of four first-order ordinary nonlinear differential equations are the core of Simulink model, in which are integrated, giving the values of state variables.

2.2.2. Gravitational force

Really accurate ways of modeling Earth's gravitational field are available in literature, nevertheless considering all the hypothesis introduced so far, even a basic model results adequate. Thus, the adopted formula is the Newton's one

$$g(h) = \frac{\mu \times 10^3}{\left(R_\oplus + h\right)^2}$$

- $\mu = GM = 398600.44\ \frac{km^3}{s^2}$ is the standard gravitational parameter of Earth;
- $R_\oplus = 6371.0088\ km$ is Earth mean radius;
- h is the altitude of the satellite $[km]$;
- g is the resulting gravitational acceleration $\left[\frac{m}{s^2}\right]$.

2.2.3. Orbital motion

This simulation tool is conceived to provide data on the s/c motion since its release into orbit, as well as computing the total time until de-orbiting. Hence, considerations about orbital motion have been done. Planar motion hypothesis reduces the study only to variations inside orbital plane. The central angle θ assumes the value of true anomaly throughout the orbits. Satellite orbit is constrained by the definition of initial conditions needed for numerical integration of the equation and by the forces acting on the body.

Due to the simplifications present in this thesis, every effect predicted by the simulator is spherically symmetric around Earth, so, in this early version, it is not needed to fully constrain the orbit by asking to the user to provide inclination, right ascension of ascending node and argument of periapsis.

The simple kinematic relation that gives $\frac{d\theta}{dt}$ restricts model possibilities only to circular or low-eccentricity elliptic orbits ($e < 0.1 \div 0.2$). In a more advanced phase, that relation it is supposed to be updated to a more general one.

Initial conditions are calculated from orbital mechanics relationships. Given

- r_a radius at apoapsis
- r_p radius at periapsis
- θ_0 initial true anomaly

$$a = \frac{r_a + r_p}{2}; \quad e = \frac{r_a - r_p}{r_a + r_p}$$

Where a is the semimajor axis of the ellipse and e the eccentricity.

$$r_0 = \frac{a(1 - e^2)}{1 + e\cos(\theta_0)}; \quad h_0 = r_0 - R_\oplus$$

$$V_0 = \sqrt{\mu\left(\frac{2}{r_0} - \frac{1}{a}\right)}$$

While testing the model, it has been found out the importance of a correct coupling between initial condition and gravitational force modeling (predominant during early orbital phase of the simulation). At first, the equation presented in the previous paragraph, was replaced by the one proposed in [13]

$$g(h) = \frac{R_\oplus^2 g(0)}{\left(R_\oplus + h\right)^2}; \quad with\ g(0) = 9.80665\frac{m}{s^2}$$

This turned out in meaningless simulation results, because the calculation of g and V_0 were performed with a slightly different gravitational model.

2.2.4. Attitude

As for the choice of a planar motion reduction, even for the attitude a strong simplification is introduced. It has been assumed that the objects are at a constant attitude throughout the trajectory. This allows the removal of six differential equations from the simulation (three for angular position, three for angular velocity), which greatly reduces computational time. Anyhow it is extremely unlikely for an uncontrolled s/c to have a constant attitude during its descent. These choices lead to development of 3 degrees of freedom (3-DOF) point mass model.

2.2.5. Aerothermodynamics

2.2.5.1. Angle of Attack

The angle of attack for simulated body is assumed to zero, meaning that no lift is generated (assuming the body is symmetric). However, if the aerodynamic efficiency is different from zero, there is the possibility to simulate the effects of it (under the hypothesis of planar motion).

2.2.5.2. Coefficients

Drag and Lift coefficients are assumed to be constant during the whole simulation. This obviously cannot be true because of the substantial different in flow regimes encountered by the s/c during its motion from orbit to ground. In particular for C_D, which is much more important, a more accurate model should consider three regimes:

- Free molecular regime;
- Transitional regime;
- Continuum regime.

For each of these, identified by Knudsen number, a different approach for evaluating C_D is suggested, as can be found in [6], [14]. To have an idea of the range variability, see Figure 5.

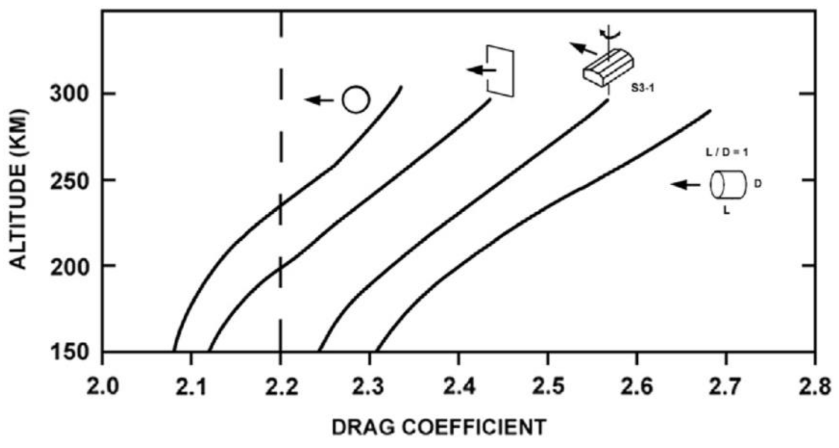


Figure 5 - Coefficient of drag values (source: Fig 15 of [6])

When choosing a constant C_D value, 2.2 is commonly assumed. It is a crude approximation but yields fairly good results in term of total time-to-demise prediction.

2.2.5.3. Thermodynamics and Heating

In the ideal assumption of a constant attitude during the descent, with the blunt nose of the TPS facing flow direction (continuum regime), the surrounding flowfield would be something similar to a bow shock wave detached from the CubeSat nose, as shown in Figure 6.

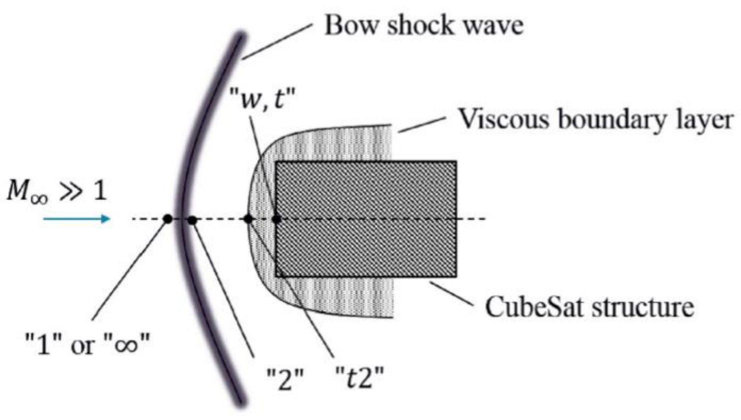


Figure 6 - shock wave in front of the CubeSat nose (source: Fig. 5 of [14])

To conduct an accurate analytic investigation of the thermodynamics involved in this process (and thus to have an idea of the heat absorbed by TPS), several parameters about the flow should be known (temperature, pressure, gases composition and specific heat capacity). This is currently out of the possibilities of this model.

In order to provide a likely estimate of the heat flux at the wall, which is closely related to the shape of the s/c nose, Tauber's engineering formula [15] has been used

$$\dot{q} = 1.83 \cdot 10^{-4}\, V^3 \sqrt{\frac{\rho}{R_C}}$$

- $\dot{q}$ stagnation point heat flux $\left[\frac{W}{m^2}\right]$;
- ρ free-stream density $\left[\frac{kg}{m^3}\right]$;
- V flight velocity $\left[\frac{m}{s}\right]$;
- R_C nose curvature radius $[m]$.

2.2.6. Inertial loads

For a proper structural design, the knowledge of inertial loads introduced by aerodynamic braking is of great importance. For a 3-DOF and zero AoA simulator the only deceleration that can be computed is the axial one, which in a controlled re-entry is the most significant.

Thus, the axial load factor is calculated from the related dynamic equation presented in section 2.2.1 and normalized with the gravitational acceleration at sea level

$$n_{ax} = \frac{dV}{dt}\frac{1}{g_0}$$

Where $g_0 = 9.80665\,\frac{m}{s^2}$

3. Model implementation on Simulink

Starting from equations discussed in chapter 2, a Simulink Model has been developed. This approach allowed an easy setup and rapid results analysis. Furthermore, the user-friendly interface could help future improvements and customization by inexperienced users.

Then, the complete simulator has been integrated within a single script named *ARES* (Academic Re-Entry Simulator) which represent the actual version of the simulator. The script, which is attached to this document, is composed by three main sections, which are described below.

3.1. Input of initial conditions

The first section is reserved to the definition of constants used in the simulation and to the input of initial conditions by the user. In the attached version of the script there are already inserted some real scenario values which are going to be described in section 4. The variables needed by the Simulink model to perform a simulation are listed in Table 3. For some of them, multiple inputs methods are possible. Others are driven by the chosen ones and computed in the script using the above presented formulas.

Variable name	Description	Value (pre-set)	Unit of meas.	Type
mi	Earth standard gravitational parameter	398600.44	km^3/s^2	constant
Re	Earth mean radius	6371.0088	km	constant
g0	gravitational acceleration (sea level)	9.80665	m/s^2	constant
Spacecraft parameters				
m	mass of the spacecraft	3	kg	user defined

A	s/c cross-sectional area (re-entry attitude)	0.01	m^2	user defined
Cd	drag coefficient	2.2		user defined
E	aerodynamic efficiency	0		user defined (if one is defined the other can be calculated)
CL	lift coefficient	0		
BC	ballistic coefficient	136.36	kg/m^2	driven
rcurv	TPS nose curvature radius	0.1	m	user defined
Space Environment parameters				
date	time at simulation start [y,m,d,h,m,s]	[2006,01,01, 12,00,00]		user defined
jdate[1]	Julian date at simulation start	2453737	days	computed by built-in function *juliandate()*
F107_avg[2]	F 10.7 cm solar flux (average of 3 solar rotations of 27 days before **date**)	90.85	SFU	user defined (databases or forecasts)
F107_day[2]	F 10.7 cm solar flux (average of the day before **date**)	86.0	SFU	user defined (databases or forecasts)
Kp[3]	three-hourly planetary geomagnetic K-index	1		user defined (databases or forecasts)
Orbital parameters[4]				
r_a	radius at apoapsis	6771.0088	km	user defined /driven (a couple **r_a, r_p** OR **a, e** can be defined)
r_p	radius at periapsis	6771.0088	km	
a	semimajor axis	6771.0088	km	
e	eccentricity	0		

Dynamic Equations initial conditions				
x0	travelled distance	0	m	user defined
gamma0	flight path angle	0	rad	user defined
theta0[5]	true anomaly	0	rad	user defined - initial position (if e=0 **theta0** is independent)
r0[5]	position vector length	6771.0088	km	
h0[5]	altitude	400	km	
V0[5]	velocity	7.67	km/s	driven (orbital speed)
Manoeuvres				
dV[6]	de-orbiting impulsive retrograde burn ΔV	0	m/s	user defined
V0	effective velocity (after ΔV)	7.67	km/s	driven
Integration method				
solv_kep	solver algorithm for the Keplerian phase of motion	ode4 Runge-Kutta	fixed step method	user defined
step_kep[7]	Keplerian phase integration step	30	s	user defined
stop_h[7]	threshold altitude for switching from 1st to 2nd integration	150	km	user defined
solv_atm	solver algorithm for the atmospheric phase of motion (re-entry)	ode4 Runge-Kutta	fixed step method	user defined
step_atm[7]	atmospheric phase integration step	0.1	s	user defined

Table 3 - detailed user inputs for simulator

Useful notes for users:

1. Julian day is the number of days passed from noon on Monday, January 1, 4713 BC (integer part of the number). The Julian date of any instant is the Julian day number plus the fraction of a day since the preceding noon in Universal Time. A built-in converter is available in MATLAB Aerospace Toolbox using the function *juliandate([y,m,d,h,m,s])*. It is used by the J71 density calculator function;
2. To find the values of the two parameters regarding solar activity (for historical, present or future uses) extensive information can be retrieved from https://www.swpc.noaa.gov. Both databases and forecasts are available (for accuracy about predictions see [6]);
3. Planetary geomagnetic index can be retrieved from the above presented website or from https://www.gfz-potsdam.de/en/kp-index/ which is the institution that has introduced the K-index;
4. Even if the kinematic equations of the model are derived for circular orbits, there is the possibility to simulate the behavior of an object in a low eccentricity orbit. The orbital description is limited to the shape of the orbit on its plane because of the planar motion hypothesis;
5. The only effective initial conditions (among these three) which are needed in the simulation are theta0, h0 and V0. Consequently, they can be either directly defined, if known, either calculated from a known parameter using the simple orbital mechanics relationships which are presented in section 2.2.3 implemented in the script);
6. There is the possibility to model a controlled re-entry by defining the entity of a hypothetical impulsive ΔV generated at t=0 s by a de-orbiting motor, under the hypothesis of a thrust vector aligned with the speed and with the opposite direction;
7. The simulation is split into two subparts as better described in the following section. It recommended to use a small enough step during atmospheric re-entry, characterized by rapid changes in states.

3.2. SIMULATION

Once the initial conditions are set, the simulation is performed into two different steps by calling a Simulink model (named *SatSim_ARES*) from the script using *sim()* built-in function. The first section is the one which simulates the s/c motion during the so-called Keplerian phase, where the prevailing force is the gravitational one. In these conditions drag is considerable as a perturbation, the variations in time of states are small and only connected to the geometry of the constrained motion. Therefore, a moderately big integration step can help to reduce total computational time. Exceeding in the step size could obviously result in a loss of accuracy.

This first simulation stops as the altitude reaches a certain threshold *stop_h*, which can be chosen by the user in input section. At this point, the simulation output is collected in a structure named *kepOut*, where all the variables are stored as arrays. Each array stores the evolution of a single variable through time. It means that for every variable a sample per integration step is memorized as an element of the respective array. Accessing to all the arrays of *kepOut* with the same index *ii* returns the complete state of the model at time $t = ii * integration\ step$ from the simulation beginning.

The last value of each variable becomes the initial input for the 2nd run. The Simulink model is the same, but the simulation is performed using a smaller integration step, in order to have a higher resolution and precision during the atmospheric flight phase. As the altitude decreases, the prevailing force becomes the aerodynamic one. Thus, all the related phenomena occur in this phase, which is the most important to analyse for designing a re-entry capable CubeSat. This run ends by default when altitude reaches 0 km, the output is now collected into *atmOut*, organized as described above. All the data are now ready to be processed and analysed.

Splitting the simulation in two, enables to obtain in a reasonable time both the information about total time to de-orbit (1st part) and about re-entry predictions (2nd part, with proper accuracy).

3.2.1. SIMULINK MODEL DESCRIPTION

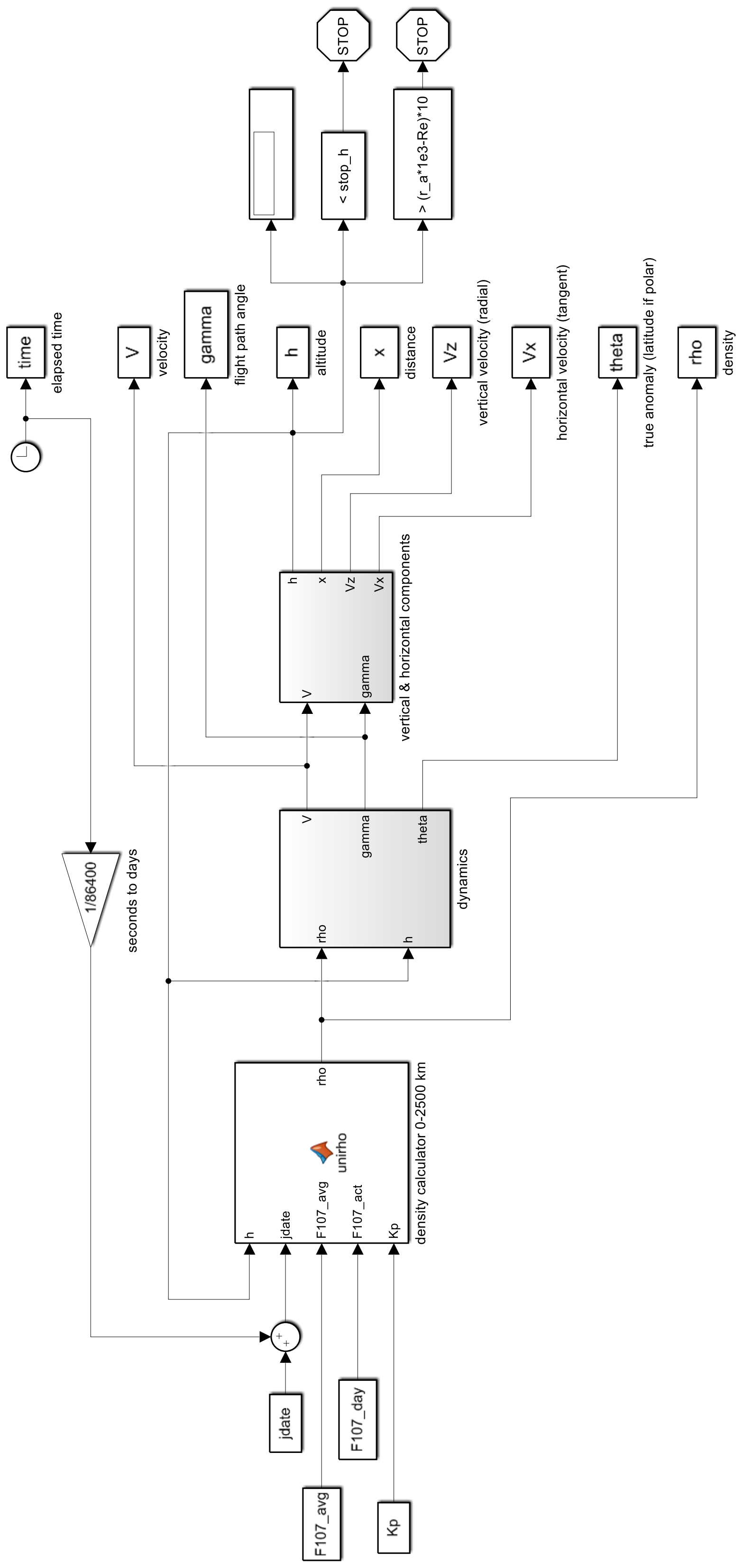


Figure 7 - Simulink model overview

In Figure 7 is presented the Simulink model which performs all the simulations. The key feature of this software is the possibility to organize data flow in a visual manner. The fundamental unit of a Simulink model is the *block.* Each block has a different feature, the most used blocks for this thesis are *sources* and *user defined function.* The following paragraphs describe what is the role of each of those, always referring to the above presented overview.

3.2.1.1. UNIRHO

As can be seen at the left of the overview some of the inputs from MATLAB script are loaded to be used in the density computation block. This block function, named *unirho,* combines Jacchia J71 and Exponential Atmosphere and is presented below.

```matlab
%This function combines Jacchia J71 and Exponential Atmosphere for
%calculating density in the band [0;2500] km
%A transition function should be introduced to avoid discontinuities

function rho = unirho(h,jdate,F107_avg,F107_act,Kp)
h=h/1000; %conversion [m] to [km]
rho=0;

%from 0 to 100 km of altitude use Exponential Atmosphere
    if h<100 && h>=0
        rho=expAtm(h);
    end

%from 100 to 2500 km of altitude use Jacchia J71
    if h>=100 && h<=2500
        rho=J71_density_simulink(h,jdate,F107_avg,F107_act,Kp);
    end

%A warning is printed if a negative altitude is predicted by
%the simulation (due to Simulink discrete stopping criterion)
    if h<0
        fprintf('Warning: mismatched density, altitude: %3.2e km',h)
        rho=1.225;
    end

end
```

The scope of this block is to compute density for a given value of altitude *h* and solar activity data. One of the issues of the simulator is that the values of F 10.7 and Kp are considered constant for the whole time. This is a limit for the reliability of results.

3.2.1.2. *DYNAMICS*

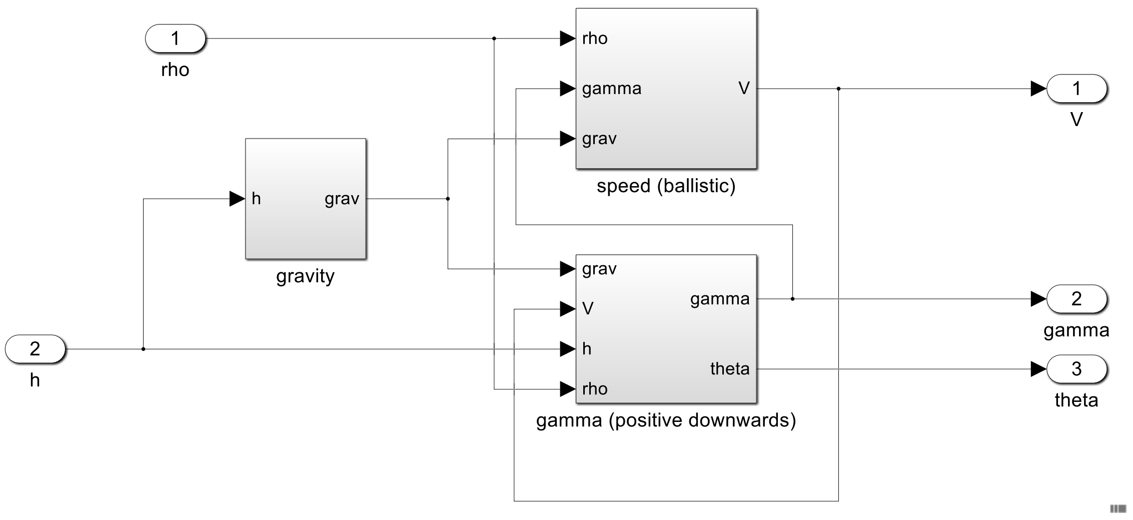


Figure 8 - Dynamics block

Figure 8 block is related to dynamics computation, starting from density and altitude. It presents three sub-blocks. The first is the gravitational force block, in Figure 9.

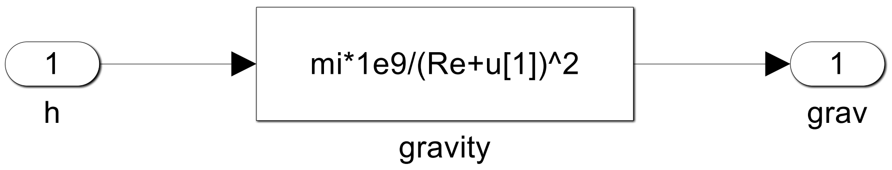


Figure 9 - Gravity sub-block

Please note that *u[i]* is the i-th inputs to *user defined function* block (arrows coming from left side). In the central square there is implemented the equation of chapter 2.2.2.

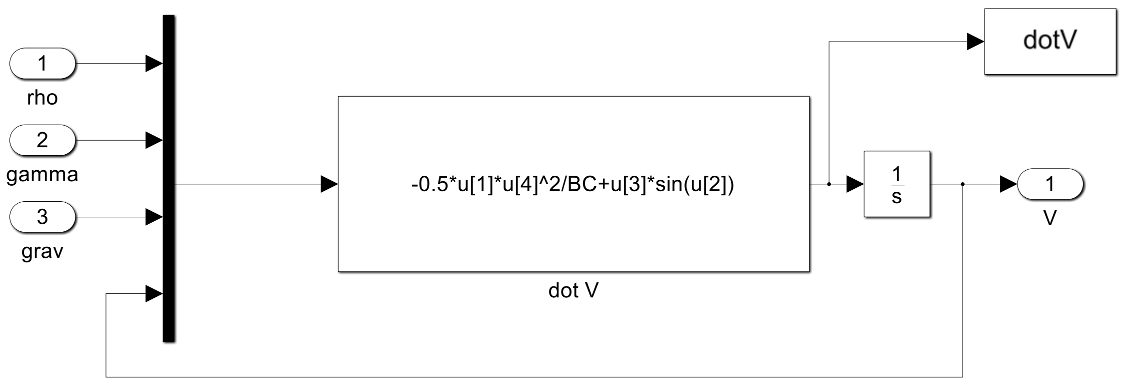


Figure 10 - speed sub-block

The second one, in Figure 10 is the block in charge of velocity computation. It is the implementation of $\frac{dV}{dt}$ equation (chapter 2.2.1). An integrator bloc is present (1/s) to integrate $\frac{dV}{dt}$ and obtain $V(t)$. That kind of block requires an initial condition, which is $V0$. The derivative value $dotV$ is collected as well in order to calculate deceleration and axial load factor.

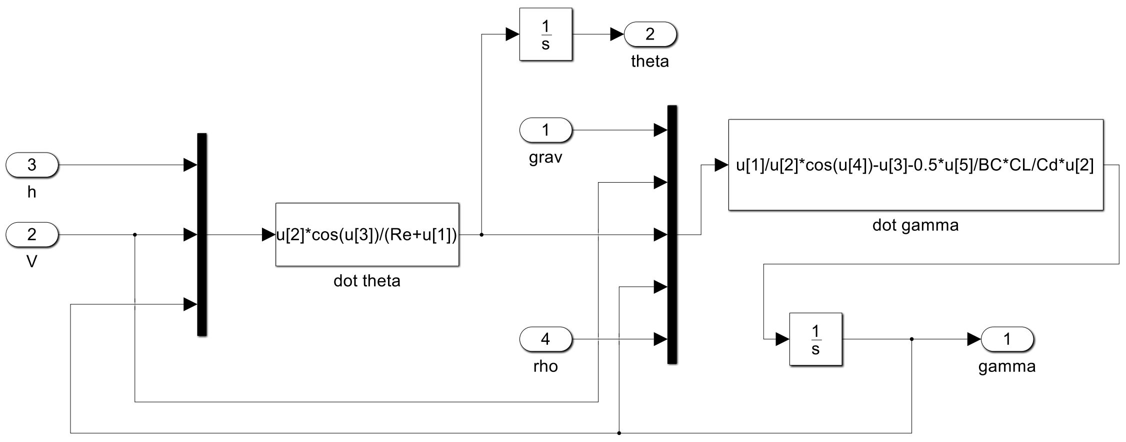


Figure 11 - theta and gamma sub-block

The third and last sub-block, in Figure 11, regards the computation of true anomaly theta and flight path angle gamma, making use of chapter 2.2.1 relations and two integrators (with the respective initial conditions).

3.2.1.3. Vertical & Horizontal components

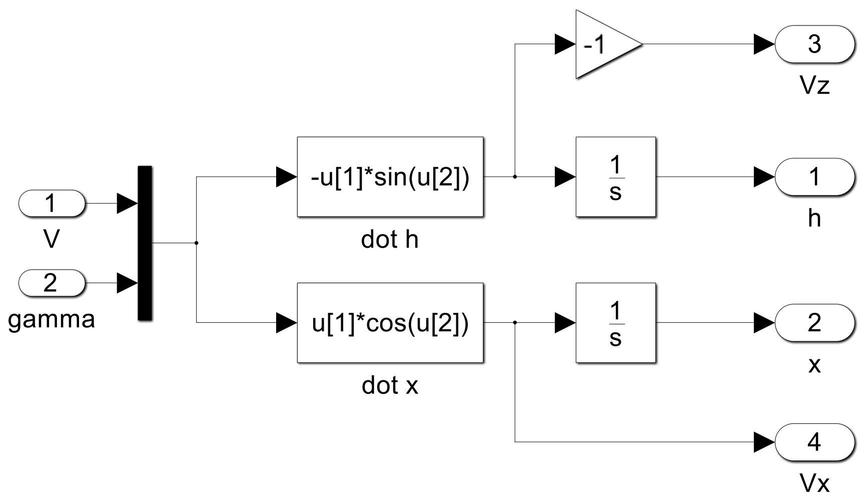


Figure 12 - vertical and horizontal components block

The vertical (radial) and horizontal (tangent) component of velocity (Vz and Vx with respect to the local vertical reference frame) are obtained multiplying velocity magnitude respectively by cosine and sine of flight path angle gamma, as shown in Figure 12. Through an integration altitude and travelled distance are calculated. The presence of the gain block (-1) is due to the choice of the reference system. Introducing it positive vertical velocity is toward the center of the Earth.

3.2.1.4. Output blocks and stopping criteria

At the right side of the model there are several square blocks organized in a column. Each block sends to the MATLAB workspace an array of values (one per integration step) named as the respective block. The output description is going to be the focus of the next section of this document.

Two stopping criteria are present to stop the simulation:

- when altitude decreases below the threshold stop_h;
- when altitude increases above a safety limit (default 10 times altitude at apoapsis). That prevents from wasting time in case of an error in initial conditions and aborts a senseless simulation.

All the inputs and outputs from this Simulink model, as well as all the computations, are performed using International System of Units. Thus, some conversions are applied when needed both in the *ARES* script and in the Simulink model.

3.3. Results post-processing and analysis

The third and last section of the *ARES script* is the one in which all the results coming from the two runs of the model are processed, analyzed and plotted.

Firstly, *kepOut* and *atmOut* outputs structure are unified and organized into arrays. In order to keep only consistent data, arrays are filtered with the aim of discarding spurious values. For example, last value of each array is always associated with a negative altitude and needs to be discarded. This is because the stopping criterion (if altitude < 0 then stop simulation) needs to read a regularly computed negative value to become effective.

It is now possible to calculate *total time to de-orbit*, *axial load factor* and *heat flux* at stagnation starting from output dynamics and kinematics parameters, using relationships from chapter 2.2.6 and 2.2.5.3 respectively. At this point, all the output values are converted from I.S. base units into more suitable units of measurement.

A complete list of the output data arrays is presented in Table 4. The length of all the arrays is the same, it is linked both with the total elapsed time from simulation start to re-entry and with the integration step (remember: each array element is a sample taken at each integration step)

Variable name	Description	Unit of measurement
h	altitude	km
x	travelled distance	km
time	elapsed time	min
V	velocity magnitude	km/s
Vx	horizontal velocity (tangent)	km/s
Vz	vertical velocity (radial)	km/s
gamma	flight path angle	deg

theta	true anomaly	rad/deg
rho	density profile	kg/m^3
dotV	axial deceleration	m/s^2
gload	axial load factor	nondimensional [g]
heat1	stagnation heat flux	W/m^2

Table 4 - Simulator complete output

The proposed method to analyze the results of the simulation is to plot the most significant charts, and to print on MATLAB workspace some basic information about peak loads (inertial and thermal) and total time to de-orbit. This group of plots does not want to be representative of all the capabilities of the simulator. Users can choose whether to use default plots, to add new ones combining the outputs presented above or to analyze numerical data arrays writing their own code.

The plotted results are (y-axis vs x-axis):

▪ Altitude [km]	vs	Time [years];
▪ Velocity [km/s]	vs	Time [years];
▪ Altitude [km]	vs	Flight path angle [deg];
▪ Altitude [km]	vs	Velocity magnitude [km/s];
▪ Altitude [km]	vs	Velocity components [km/s];
▪ Heat flux [W/m^2]	vs	Altitude [km];
▪ Axial load factor	vs	Altitude [km];
▪ Trajectory shape evolution (polar plot).		

Further information and examples about plots are available in chapter 4, where the simulator script has been applied to a real mission scenario.

4. Real case application

4.1. QARMAN mission overview

Despite the concept of a heat shield enabling a CubeSat to survive re-entry, acting in a first phase as a de-orbiting device increasing drag, is not something new [16], [17], a similar mission has not been performed so far. QARMAN - Qubesat for Aerothermodynamic Research and Measurements on AblatioN - developed by *Von Karman Institute for Fluid Dynamics (VKI)* [18] and planned to be launched in 2020 is something close to the idea. It is a 3U CubeSat equipped with a TPS composed by a P50 ablative cork nose (1U) and a ceramic layer of SiC for side panels.

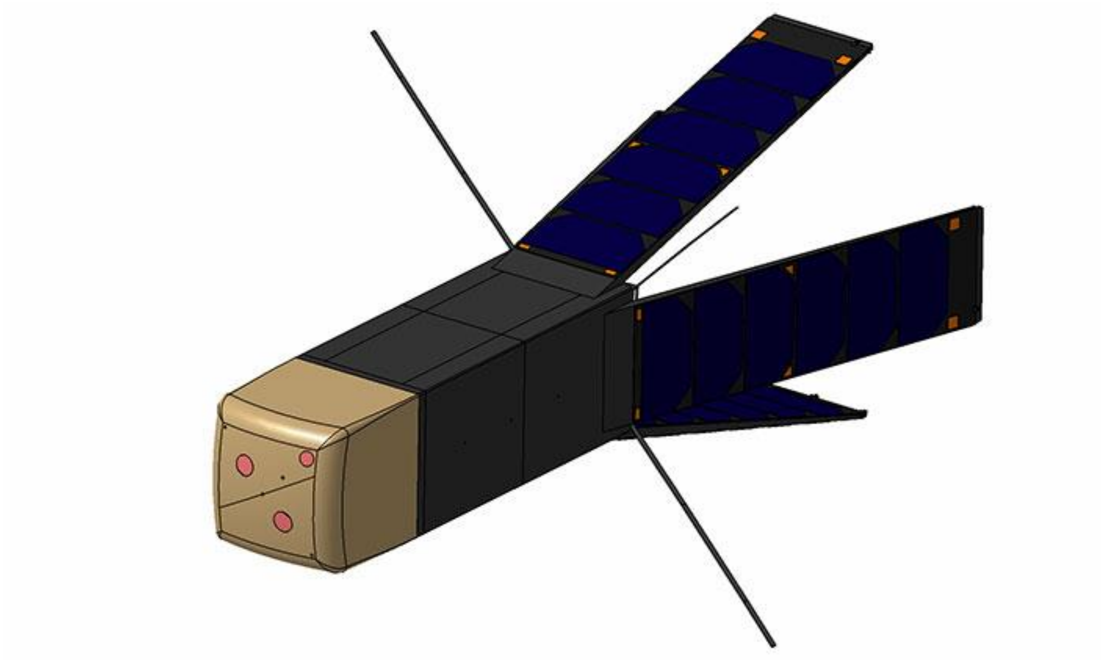


Figure 13 - QARMAN CubeSat (source: [18])

At the end of the orbital phase of its mission, the side panels will open to rest at an angle of 15 degrees with respect to the satellite axis, as can be seen in Figure 13. This results in an increase of aerodynamic drag, thus a decrease of velocity. Hence, the satellite will slowly de-orbit. During re-entry the s/c is expected to collect data on the ablation process. To protect the electronics during the re-entry, several layers of aerogel insulation are implemented on the satellite.

All the collected data are transmitted towards the Iridium constellation, providing valuable information for future atmospheric research. The mission ends with a crash-land on ground.

4.2. Simulation of the mission

4.2.1. Introduction

Starting from the available information about QARMAN presented in Table 5, a full simulation of the model has been performed using *ARES script*. In order to assess simulation performances, some of the results are compared with those obtained from DAS, while others are compared with simulations performed at VKI.

Description	Value
mass	3 kg
perigee altitude	400 km
apogee altitude	400 km
inclination	51.6° (ISS orbit)
cross-sectional area (closed configuration)	0.01 m^2

Table 5 - QARMAN parameters

It must be considered that the main scope of this chapter is to test simulator's reliability and not to predict precisely what will be the evolution of the QARMAN mission. Therefore, some of the necessary parameters to run the simulation are hypothesized values, but every comparison with other software is made under the same conditions. As regards the *ARES,* complete inputs list used to perform the following discussed simulation is the one of Table 3. All the satellite parameters are assumed to be constant during the mission.

4.2.2. Results and comparisons

The results of the simulation are presented hereafter. Firstly, are presented the *ARES* output plots, as described in chapter 3.3. For each of the plots, the most significant things to be noticed are described. Then a comparison with DAS is proposed, regarding total time to de-orbit and altitude history over mission time. In the last part of this section, some heat flux vs altitude charts from VKI and the results of an experimental testing session at CIRA are commented.

4.2.2.1. Ares results

In Figure 14 is represented the decrease of the satellite altitude caused by atmospheric drag, which acts dissipating orbital energy. This simulation starts in a fictitious date: 1st Jan 2006.

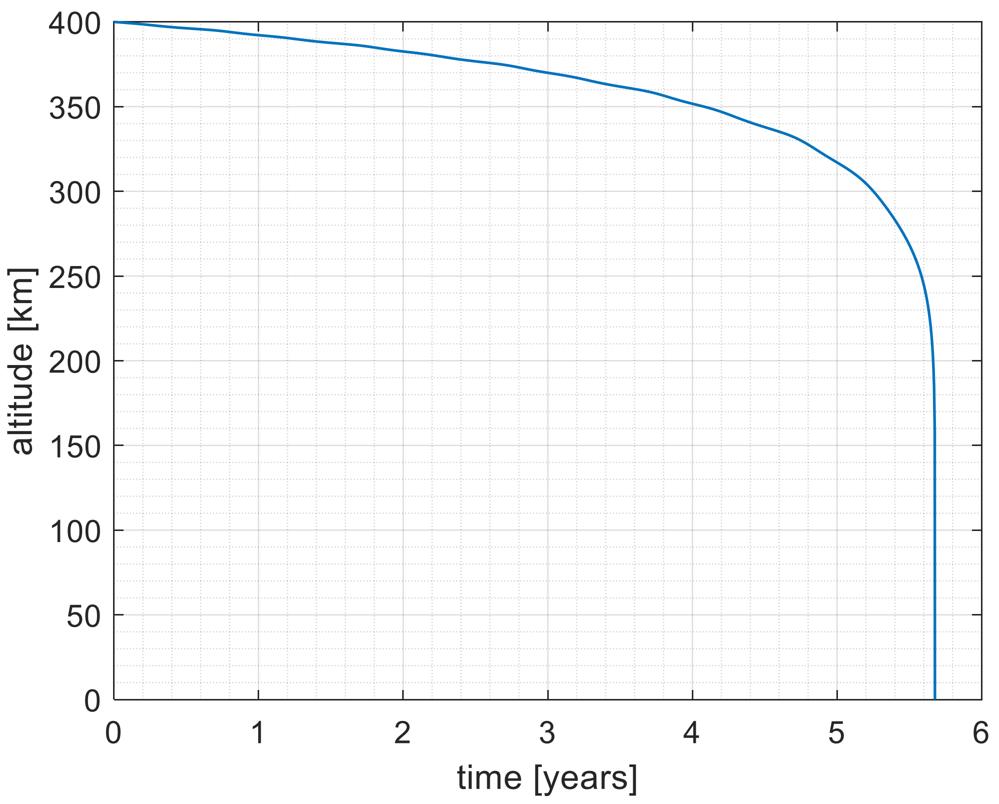


Figure 14 - s/c altitude vs time from mission beginning

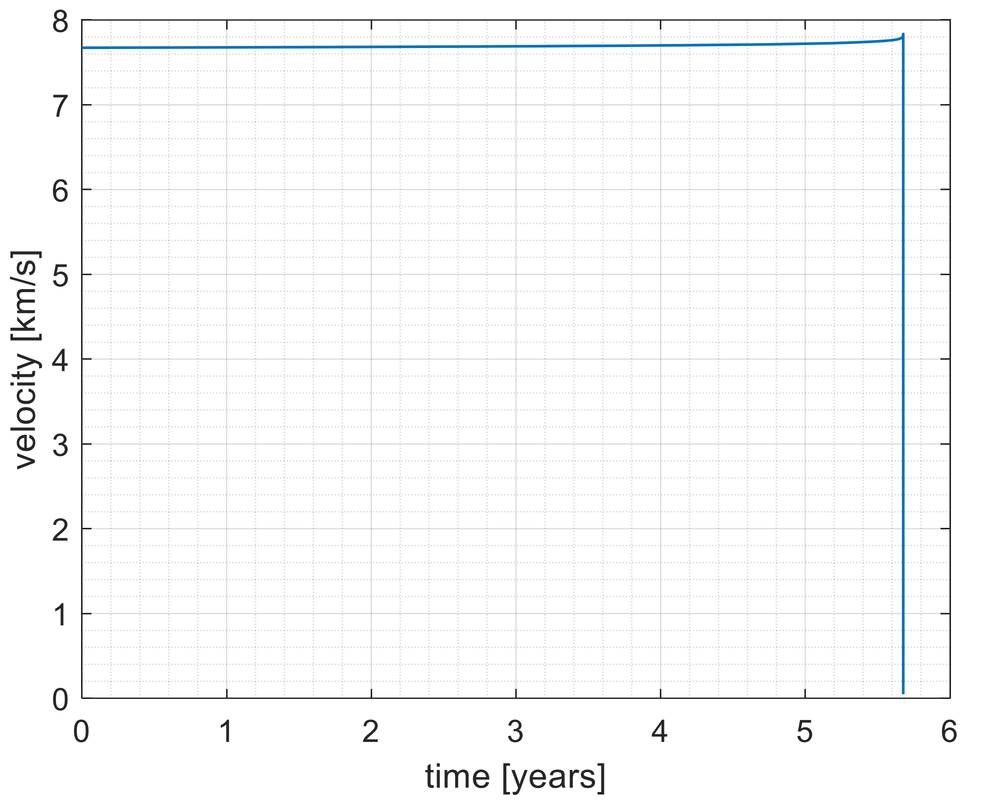


Figure 15 - velocity magnitude vs time from mission beginning

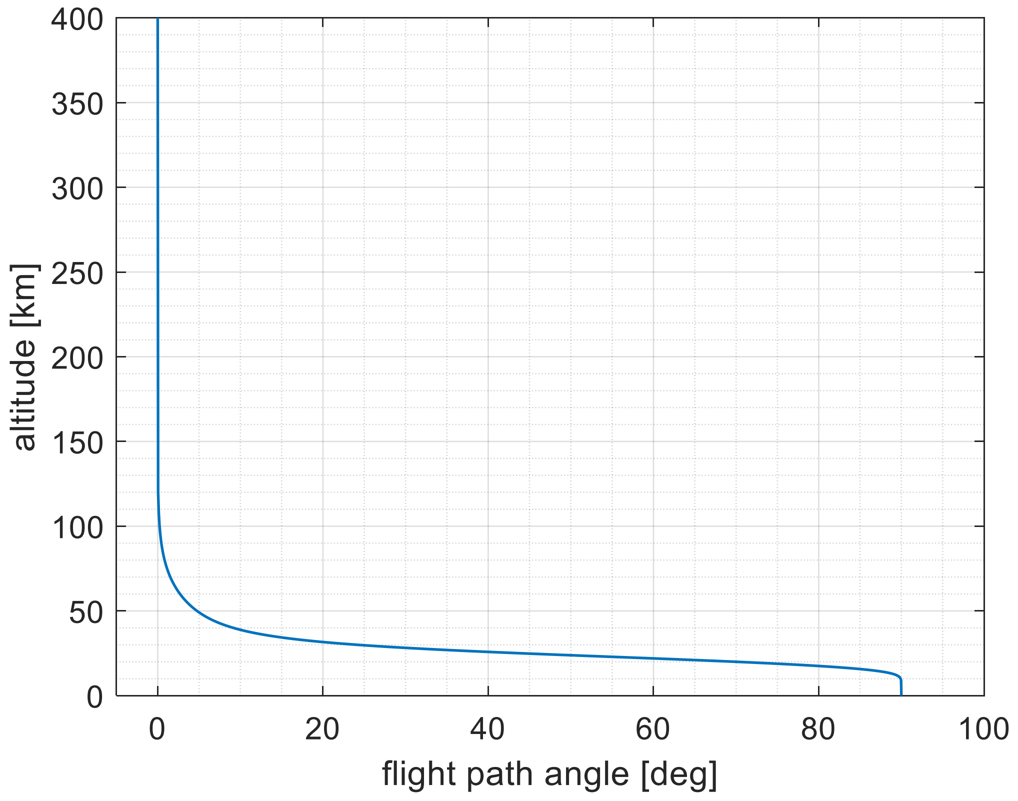


Figure 16 - altitude vs flight path angle

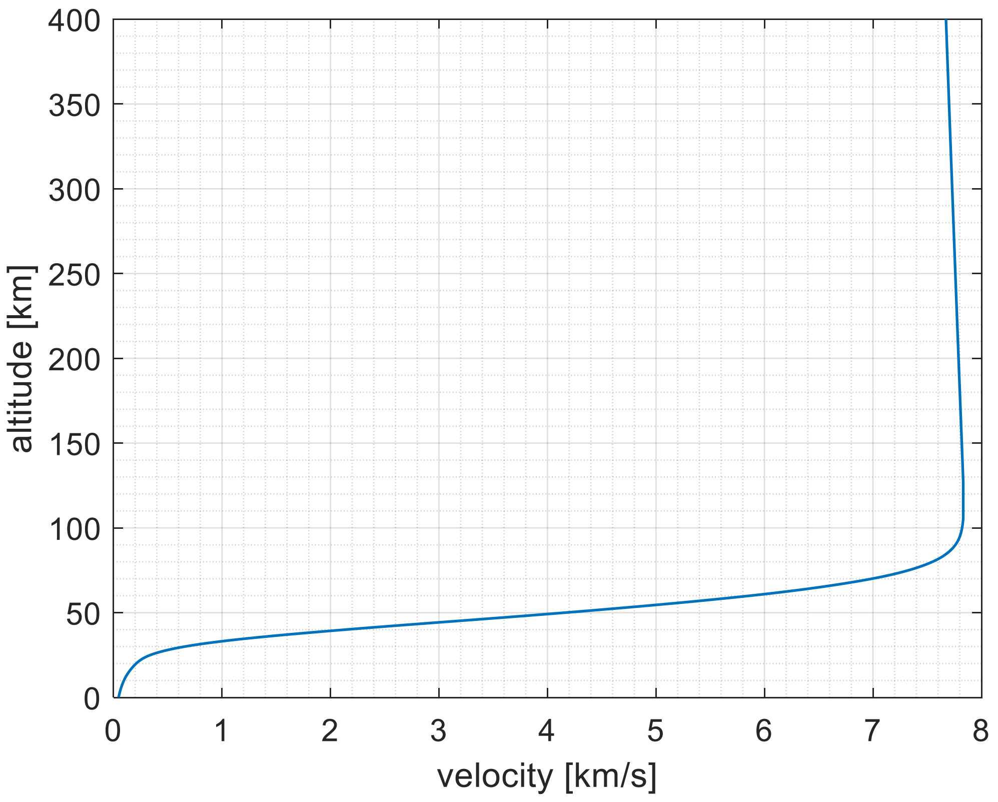


Figure 17 - altitude vs velocity magnitude

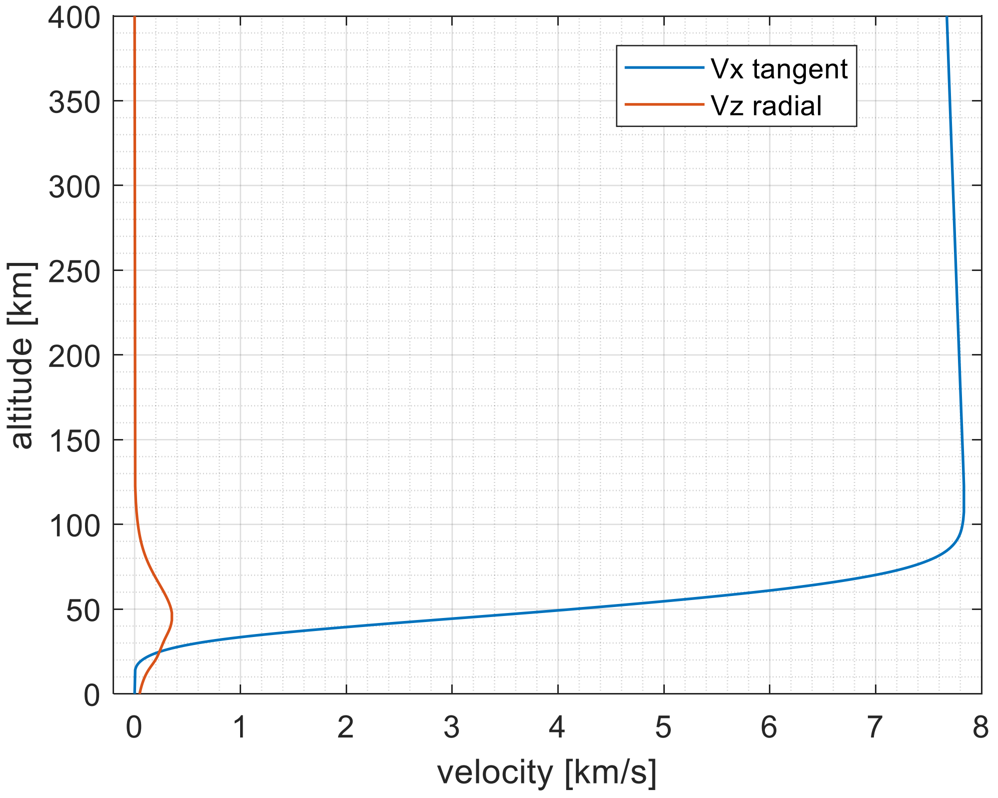


Figure 18 - altitude vs velocity components

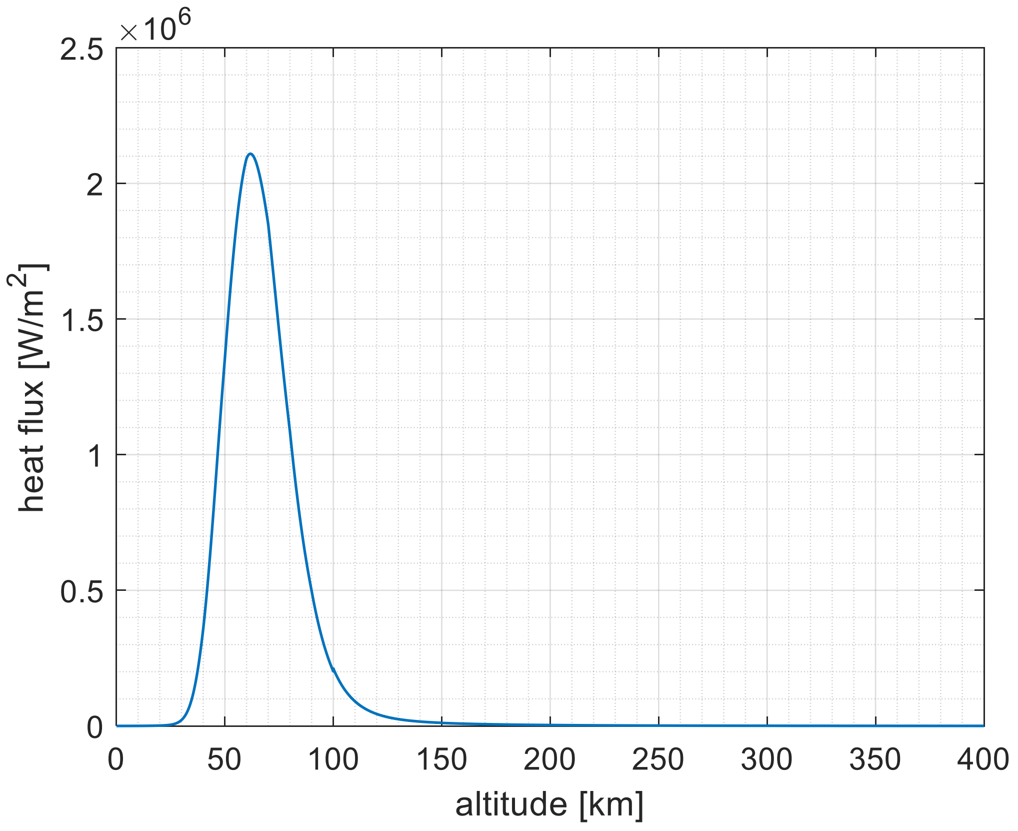


Figure 19 - stagnation point heat flux vs altitude

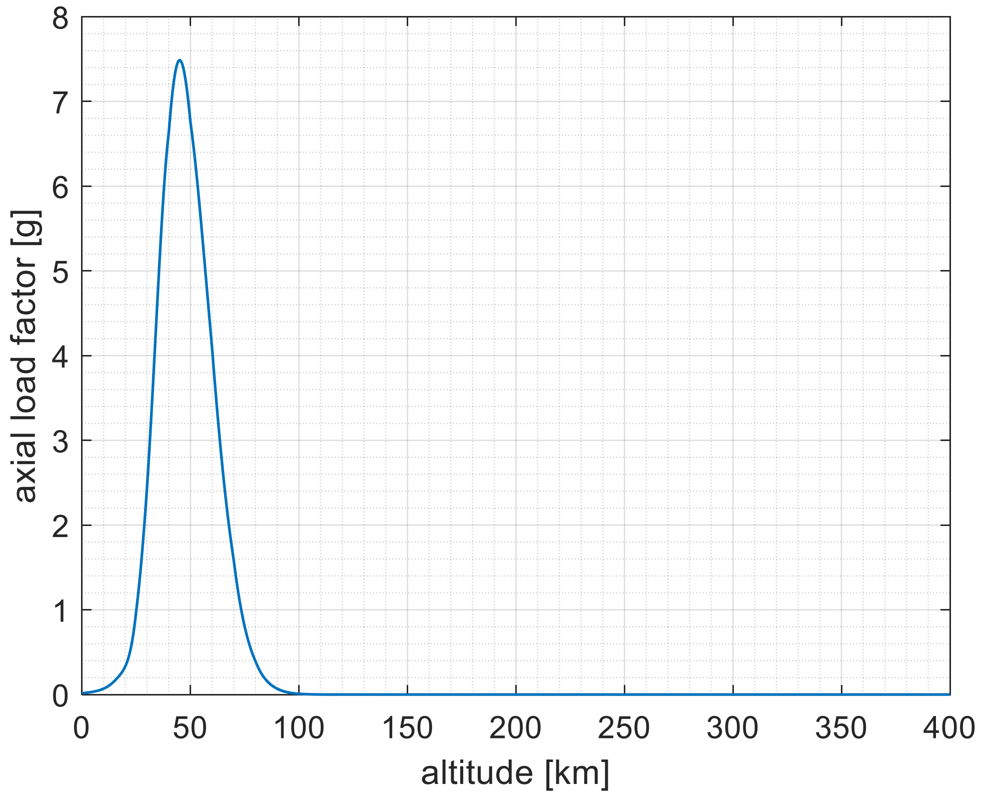


Figure 20 - axial load factor vs altitude

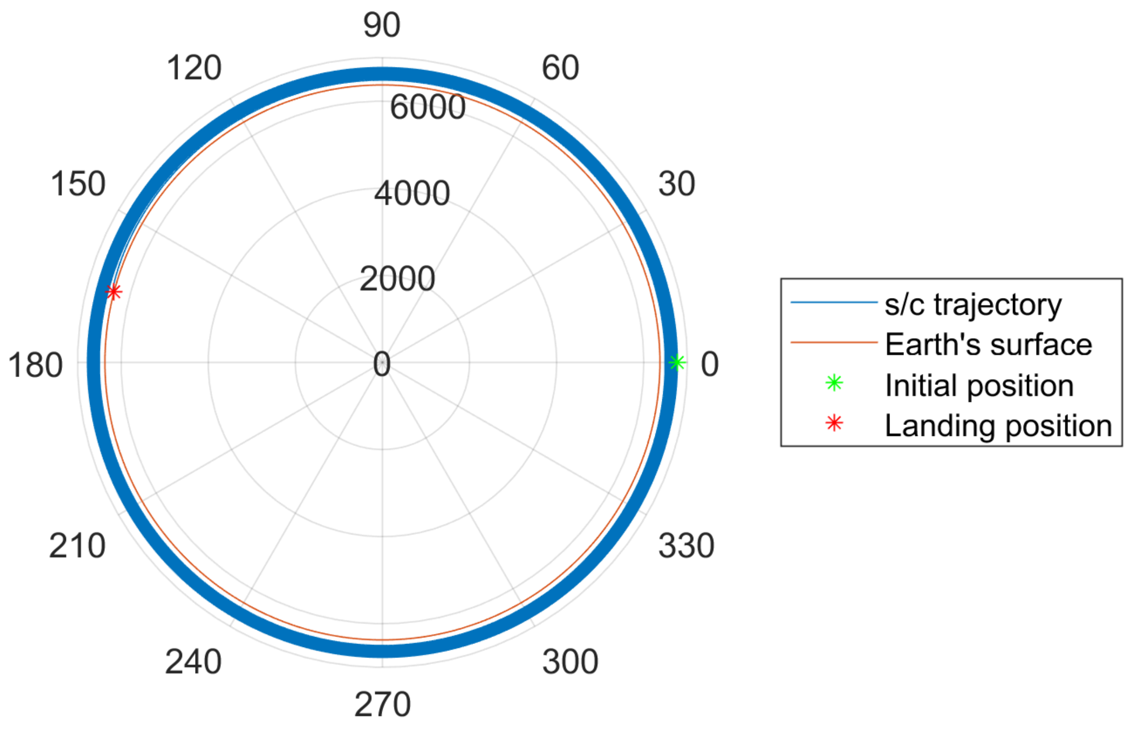


Figure 21 - trajectory shape evolution

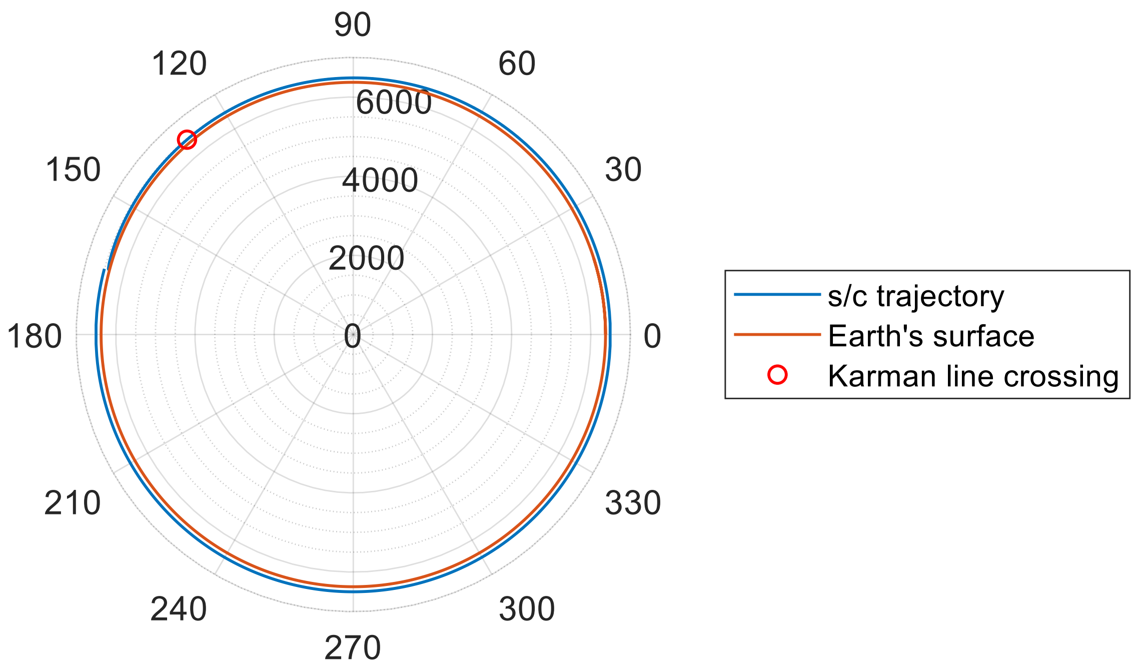


Figure 22 - re-entry trajectory

Event	Value	Altitude of event
Max axial load factor	7.5 g	45.0 km
Max stagnation heat flux	2.11 MW/m^2	61.8 km
Total time to de-orbit	5.68 years	400 km to 0 km

Table 6 - workspace output

In Figure 15, Figure 17 and Figure 18 it is possible to analyse how velocity changes both in magnitude and direction during the mission. At first there is a slight increase due to the reduction of the semimajor axis of the orbit, secondly the drag effect becomes stronger and a severe decrease occurs. It must be noticed that all the stronger variations are confined to the very last part of the mission, meaning that the s/c encounters the atmosphere as a near discontinuity.

In Figure 16 is represented the angle formed by the direction of the velocity and the local horizon. The “rotation” of the velocity vector is due to dissipation of the horizontal component of the velocity (local vertical frame) operated by drag, as can be seen also in Figure 18. The residual velocity at landing is aligned with the local vertical and has a magnitude of 169.7 km/h. This confirms that even surviving re-entry QARMAN is likely to crash on the ground.

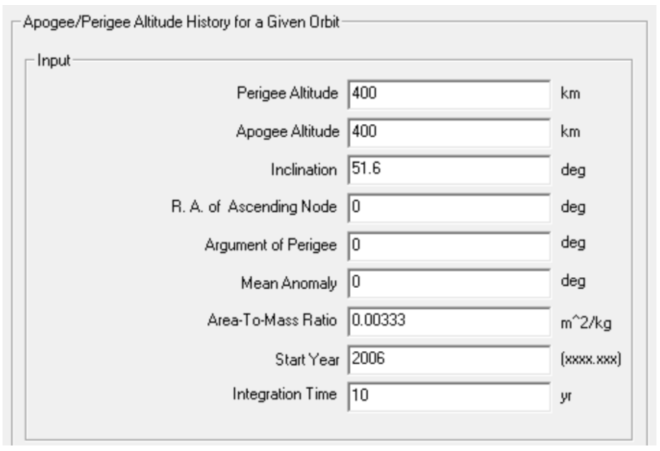
In Figure 19 and Figure 20 is possible to see that the most of the loads, both thermal and inertial, are experienced in a relatively narrow band of altitudes.

In Figure 21 and Figure 22 a graphical representation of the trajectories is available. In the first the whole mission is represented, hence the trajectories are so similar that is impossible to distinguish a single orbit. In the second only the last portion of the mission is extracted, the last orbit together with the effective re-entry trajectory. From simulation data output is possible to find out the duration of this last orbit: 92.4 minutes.

Additional results are printed to MATLAB workspace and presented in Table 6.

4.2.2.2. DAS Simulation

To check the reliability of the total time to de-orbit and the altitude vs time chart provided by *ARES* script, a simulation with the DAS tool has been performed. It has been used the *apogee/perigee altitude history for a given orbit* function, with Figure 23 input parameters.



Apogee/Perigee Altitude History for a Given Orbit

Input

Perigee Altitude	400	km
Apogee Altitude	400	km
Inclination	51.6	deg
R. A. of Ascending Node	0	deg
Argument of Perigee	0	deg
Mean Anomaly	0	deg
Area-To-Mass Ratio	0.00333	m^2/kg
Start Year	2006	(xxxx.xxx)
Integration Time	10	yr

Figure 23 - DAS simulation input

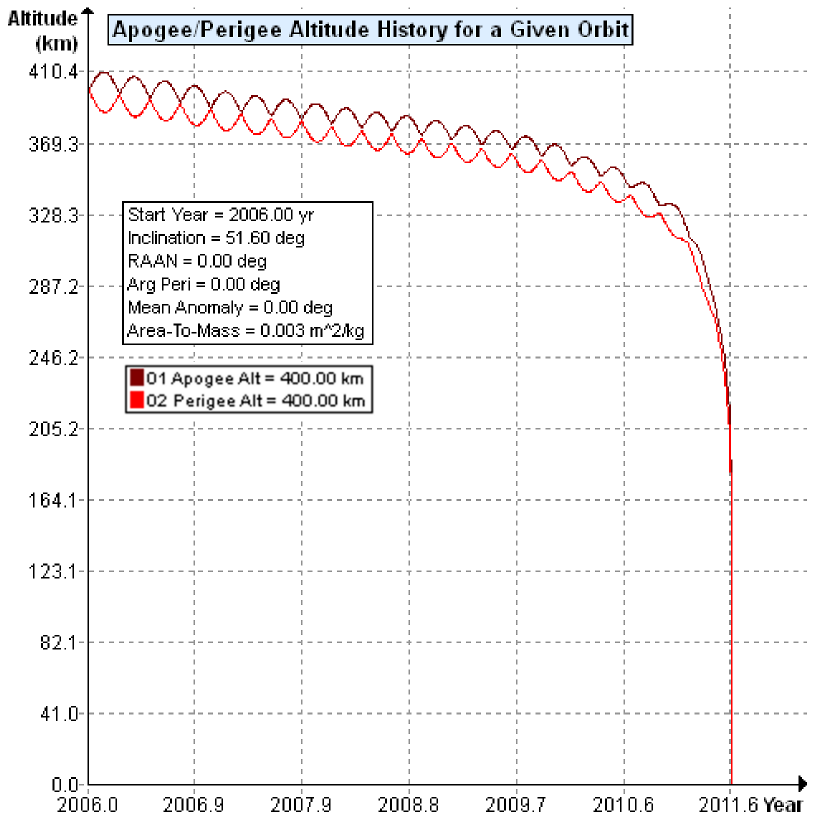


Figure 24 - DAS altitude vs time

Figure 14 and Figure 24 are comparable, except from the oscillations present in the latter (probably due to second order effects of gravitational force). Furthermore, DAS predicted orbital lifetime is 5.58 years, which turns in a 1.8% difference with the ARES estimation. Thus, results comparability can be considered a limited but significant proof of the reliability of the presented simulator.

4.2.2.3. VKI STUDIES

The critical functionality of the QARMAN TPS has been assessed with both numerical and experimental analysis. Multiple preliminary studies has been performed at VKI [19], [20]. Some of the results are below reported.

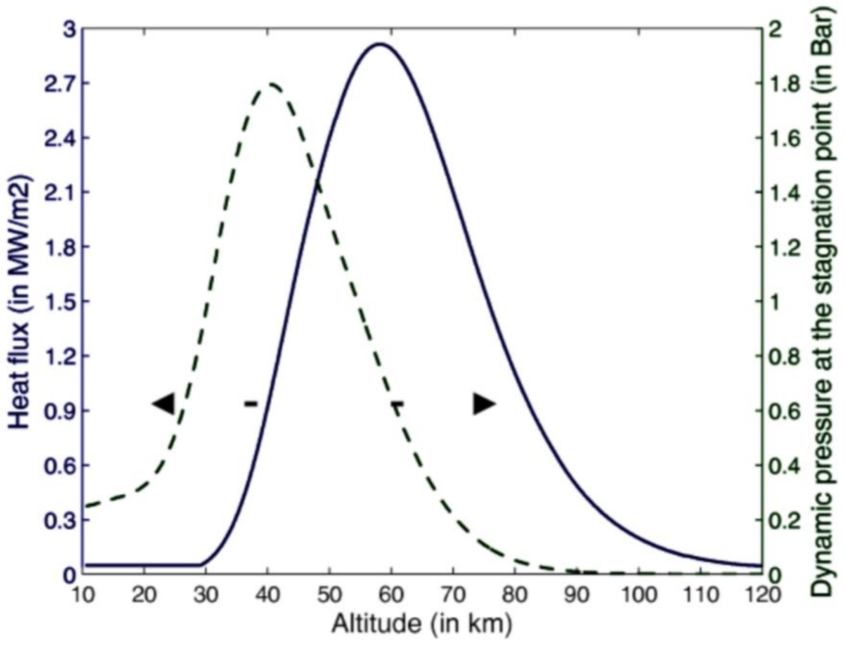


Figure 25 - VKI stagnation point heat flux vs altitude (source: Fig. 7 of [19])

Both Figure 25 and Figure 26 look very similar to Figure 19 and Figure 20 respectively. This means that the employed simulation techniques are not very different, and since the data from VKI has been used to design the actual TPS of the QARMAN, it is proven that the simulator presented in this thesis is capable of providing a likely estimate of the loads acting on a s/c during re-entry.

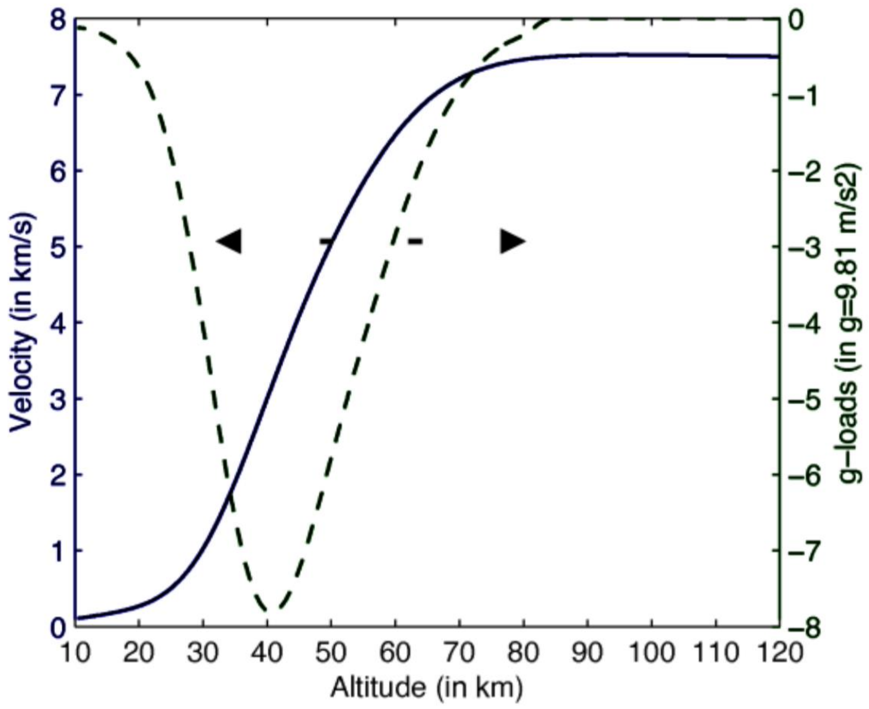


Figure 26 - VKI g-loads and velocity vs altitude (source: Fig. 6 of [19])

In 2018, in order to validate thermal modelling of TPS and duplicate on ground the integral heat load of re-entry phase, the full-scale satellite has been tested in SCIROCCO plasma wind tunnel at CIRA, as shown in Figure 27. For the first time in the world, in an arc jet plant, instead of single components at a time, a complete and full-scale spacecraft was tested, taking a huge step forward in experimental analysis of re-entries.

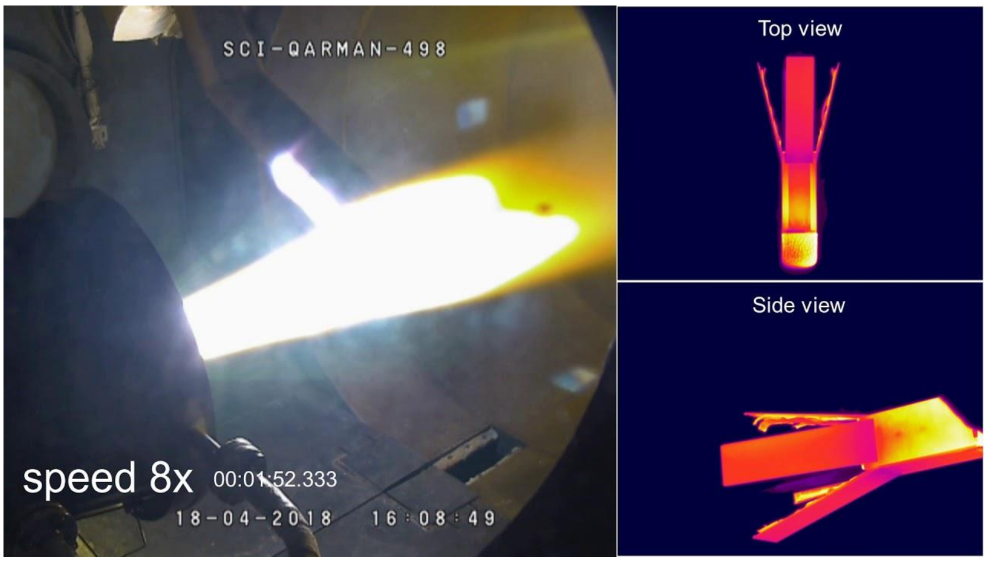


Figure 27 - SCIROCCO test of QARMAN (source: VKI)

Description	Target	Measured
Probe Stagnation Heat Flux	2120 kW/m^2	2178 kW/m^2
Probe Stagnation Pressure	40 mbar	39.6 mbar
Air Mass flow rate	0.65 kg/s	0.65 kg/s
Argon Mass flow rate	0.03 kg/s	0.03 kg/s
Total pressure	3.7 bar	3.7 bar
Test Duration	390 sec	395 sec
Mach number	7	
Velocity	~ 6 km/s	

Table 7 - SCIROCCO test conditions (source: [21])

The measured heat flux during the SCIROCCO test, as described in Table 7, is close to the value of 2.11 MW/m^2 predicted by the simulation script. The P50 heat shield of QARMAN shows in Figure 28 that an ablative material can withstand these conditions. Therefore, even if further investigations are needed, a heat flux value close to the above stated one could be used for heat shield design purposes.

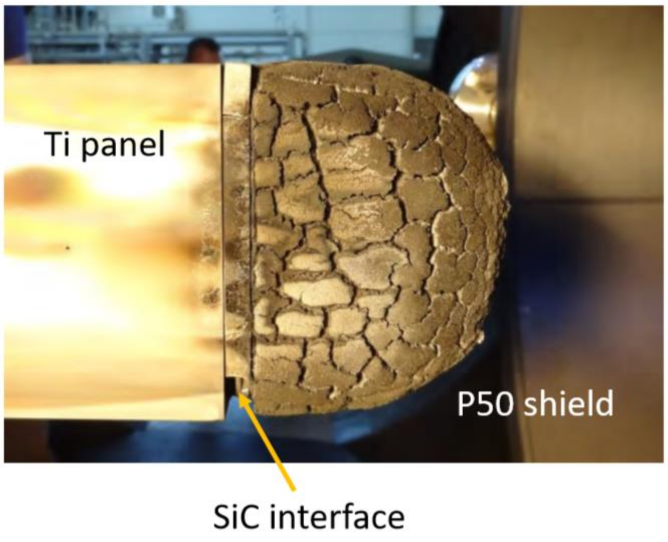


Figure 28 - ablation of P50 (source: [21])

5. Conclusions

The simulation script presented in this work, without any pretense of being an extremely accurate and comprehensive tool for the modeling of all the physical phenomena involved, has been intended by the author as a starting point to understand the problem of re-entry modeling. Firstly, further model-validation tests must be carried out in order to assess both performance and accuracy of the simulator. This has been a major concern during this thesis work, but unfortunately it is clearly very difficult to perform an experimental campaign in this frame. In addition to this, only few data from real missions are available, and even fewer regarding CubeSats. The comparison presented in chapter 4.2.2 must be considered only a preliminary result in terms of model reliability. However, since the obtained results are perfectly comparable with other works, the aim of providing a starting point for those interested in designing a re-entry capable CubeSat can be considered achieved.

Working on this topic, has allowed the author to have a small but significant overview of such a complex topic. Although way more complete models are available in literature (e.g. the one presented by Bevilacqua and Rafano Carnà [14]), this work is the proof that even very difficult problems can be divided into smaller challenges and tackled by inexperienced students.

5.1. Future work

The future work regarding this thesis topic could go in two different directions:

1. The first one is the simulator improvement. A list of things to be revised and added is presented below. After that, an extensive model validation campaign must be conducted;
2. The second kind of development could be the preliminary design of a CubeSat 1U module shielding system, which could enable potentially every CubeSat to survive to atmospheric re-entry.

5.1.1. Simulator improvement

In order to be improved, the simulator should be revised. Starting from scratch is the suggested approach for developing a complete and enhanced simulator. For that purpose, the description of the motion should be performed from an Earth centred reference frame, using complete dynamics equations. However, it is possible to improve the performance of the current version of this simulator by removing the following hypothesis and working on the proposed points:

Atmosphere modeling

- consider diurnal variations in exospheric temperature;
- include seasonal-latitudinal density dependence;
- preserve continuity from Jacchia J71 to Exponential model with a polynomial fit;

Dynamics

- remove planar motion hypothesis;
- include rigid body 6 DoF equations set;
- enhance gravity model with second order terms;
- add Earth rotation effects;
- allow for high ellipticity orbits simulation (change related kinematic equations);
- add possibility of setting a real orbit (remove restriction to orbital plane only);

Aerothermodynamics

- introduce different flow regimes (C_d not constant);
- remove 0° AoA limitation;
- model wind effects;
- refine stagnation point heat flux calculations in different flow regimes;
- add equations for modeling ablation process;
- consider mass reduction and possible s/c demise due to thermal loads;

Simulator

- automatic update of solar activity parameters during simulation (an ftp connection to *SWPC* can be adopted);
- introduce a user-friendly interface for the simulator;
- look for a strategy to reduce computational time;
- implement a different data analysis system based on MATLAB structures array (i.e. each element of the array is a structure with the complete state of the model).

5.1.2. Re-entry module concept

Starting from basic analysis that can be performed by current version of the simulator, it is possible to start the design of a likely 1U CubeSat module for enabling re-entries. This module should be easily attachable to third part developed CubeSats and should protect the s/c as a TPS, together with acting as a de-orbit drag increasing device. It should also slow down the satellite enough to make it land softly, giving the possibility to the owner to recover its spacecraft.

Such a system, as described in the introduction, could help to tackle space debris problem by introducing a smart solution which is no more a non-repayable investment for s/c owner. Innovative missions could be planned, such as low-cost experiments involving payload recovery.

BIBLIOGRAPHY

[1] R. Sandau, “Status and trends of small satellite missions for Earth observation,” *Acta Astronaut.*, vol. 66, no. 1–2, pp. 1–12, 2010.

[2] T. Villela, C. A. Costa, A. M. Brandão, F. T. Bueno, and R. Leonardi, “Towards the thousandth CubeSat: A statistical overview,” *Int. J. Aerosp. Eng.*, vol. 2019, 2019.

[3] Inter-Agency Space Dedbris Coordination Committee, “IADC Space Debris Mitigation Guidelines,” *IADC Sp. Debris Mitig. Guidel.*, 2007.

[4] B. Lazare, “The French Space Operations Act: Technical Regulations,” *Acta Astronaut.*, vol. 92, no. 2, pp. 209–212, Dec. 2013.

[5] C. Pardini and L. Anselmo, “On the accuracy of satellite reentry predictions,” in *Advances in Space Research*, 2004.

[6] D. A. Vallado and D. Finkleman, “A critical assessment of satellite drag and atmospheric density modeling,” *Acta Astronaut.*, vol. 95, no. 1, pp. 141–165, 2014.

[7] O. Montenbruck and E. Gill, *Satellite Orbits*. Springer, 2000.

[8] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, Fourth Edi. Hawthorne, CA: Microcosm Press, 2013.

[9] “F10.7 cm Radio Emissions,” *Space Weather Prediction Center*. [Online]. Available: https://www.swpc.noaa.gov/phenomena/f107-cm-radio-emissions. [Accessed: 09-Sep-2019].

[10] L. G. Jacchia, “Revised static models of the thermosphere and exosphere with empirical temperature profiles,” 1971.

[11] E. Gill, “Smooth Bi-Polynomial Interpolation of Jacchia 1971 Atmospheric Densities For Efficient Satellite Drag Computation,” *DLR-GSOC IB 96-1*, 1996.

[12] J. R. Wertz, *Spacecraft Attitude Determination and Control*. 1978.

[13] F. J. Regan and S. M. Anandakrishnan, “Re-Entry Vehicle Particle Mechanics,” in *Dynamics of Atmospheric Re-Entry*, 1993, pp. 179–222.

[14] S. F. Rafano Carná and R. Bevilacqua, “High fidelity model for the atmospheric re-entry of CubeSats equipped with the Drag De-Orbit Device,” *Acta Astronaut.*, vol. 156, no. May 2018, pp. 134–156, 2019.

[15] M. E. Tauber, “Review of high-speed, convective, heat-transfer computation methods,” *NASA Tech. Pap.*, 1989.

[16] V. Carandente and R. Savino, “New Concepts of Deployable De-Orbit and Re-Entry Systems for CubeSat Miniaturized Satellites,” *Recent Patents Eng.*, vol. 8, no. 1, pp. 2–12, 2014.

[17] J. Andrews, K. Watry, and K. Brown, “Nanosat Deorbit and Recovery System to Enable New Missions,” in *25th Annual AIAA/USU Conference on Small Satellites*.

[18] “QARMAN - Qubesat for Aerothermodynamic Research and Measurements on AblatioN,” *Von Karman Institute for Fluid Dynamics*. [Online]. Available: http://www.qarman.eu/. [Accessed: 21-Sep-2019].

[19] G. Bailet, C. O. Asma, J. Muylaert, and T. Magin, “Feasibility Analysis and Preliminary Design of an Atmospheric Re-Entry Cubesat Demonstrator,” in *7th Aerothermodynamics Symposium, Brugge, Belgium*, 2011.

[20] G. Bailet, I. Sakraker, T. Scholz, and J. Muylaert, “Qubesat for Aerothermodynamic Research and Measurement on AblatioN,” 2013.

[21] D. Masutti, E. Trifoni, E. Umit, A. Martucci, and D. Amandine, “QARMAN re-entry CubeSat : Preliminary Results of SCIROCCO Plasma Wind Tunnel Testing,” 2018.

APPENDIX

1. ARES script

```
% ARES - Academic Re-Entry Simulator - Sept 2019

% This script evaluates the decay of an Earth orbiting spacecraft exposed
% to atmospheric drag.
% Atmospheric density is calculated through:
% -Jacchia J71 model [100 to 2500 km of altitude]
% -Exponential atmosphere [0 to 100 km of altitude]

clear
close all
```

Initial conditions

```
%constants
mi=398600.44;           %km^3/s^2    %Earth G*M
Re=6371.0088;           %km          %Earth mean radius
g0=9.80665;             %m/s^2       %Gravitational acceleration (sea level)

%spacecraft parameters
m=3;                    %kg          %spacecraft mass
A=(10e-2)^2;            %m^2         %spacecraft cross-sectional area (re-entry
attitude)
Cd=2.2;                              %drag coefficient
E=0;                                 %aerodynamic efficiency
CL=Cd*E;                             %lift coefficient
BC=m/A/Cd;              %kg/m^2      %ballistic coefficient
rcurv=0.1;              %m           %TPS nose curvature radius

%space environment                   %to be refined: updating during simulation
$
date=[2006,01,01,12,00,00];          %intial time of simulation [y,m,d,h,m,s]
jdate=juliandate(date);              %conversion to julian date

F107_avg=90.85;         %SFU         %F10.7 average of 3x27 days before the
date under consideration
F107_day=86.0;          %SFU         %F10.7 average of day before the date
under consideration
Kp=1;                                %Kp three-hourly planetary geomagnetic
index

%orbital parameters
r_a=400+Re;             %km          %radius at apoapsis
r_p=400+Re;             %km          %radius at periapsis
a=(r_a+r_p)/2;          %km          %semimajor axis
e=(r_a-r_p)/(r_a+r_p);               %eccentricity

%re-entry path initial values (at t=0 s)
x0=0;                                     %m          %travelled distance
```

```
gamma0=0;                         %rad         %flight path angle
theta0=0;                         %rad         %true anomaly
r0=a*(1-e^2)/(1+e*cos(theta0));   %km          %position vector length
h0=r0-Re;                         %km          %height
V0=sqrt(mi*(2/r0-1/a));           %km/s        %orbital speed

%de-orbiting retrograde burn
dV=0;                               %km/s      %impulsive delta V obtained
V0=V0-dV;                           %km/s      %effective inital speed

%integration method
solv_kep='ode4';     %solver method for orbital phase. ode4 is runge kutta
step_kep='30';  %s   %keplerian integrator fixed step size
stop_h=150;     %km  %threshold altitude for switch from Kep. to Atm. phase

solv_atm='ode4';     %solver method for re-entry phase. ode4 is runge kutta
step_atm='0.1'; %s   %atmospheric integrator fixed step size

%conversions to I.S.
Re=Re*1000;                         %m
V0=V0*1000;                         %m/s
h0=h0*1000;                         %m
stop_h=stop_h*1000;                 %m
```

Simulation

```
%a 3 DOF simulator has been implemented in Simulink model SatSim_ARES

%Kepleran phase of the simulation, slow variations in states due to low
%density, thus low drag perturbation.
%Integration step: 30 s
%First run stops when altitude reaches stop_h threshold

kepOut = sim('SatSim_ARES','Solver',solv_kep,'FixedStep',step_kep);

%   updating initial conditions, 2nd run using ouput from 1st

%space environment
ndays1=kepOut.time(end)/60/60/24;  %days since intial time of simulation
jdate=jdate+ndays1;                %new julian date at beginning of 2nd run

F107_avg=90.85;          %SFU      %to be refined: updating
F107_day=86.0;           %SFU      %to be refined: updating
Kp=1;

%re-entry path initial values (at t=0 s)
x0=kepOut.x(end);                            %m         %travelled distance
gamma0=kepOut.gamma(end);                    %rad       %flight path angle
theta0=kepOut.theta(end);                    %rad       %true anomaly
h0=kepOut.h(end);                            %m         %height
V0=kepOut.V(end);                            %m/s       %orbital speed

%Re-entry simulation, atmospheric phase, more accuracy is needed.
```

```
%Smaller integration step required (e.g. 0.1 s)

%Simulation stops when altitude reaches 0 km (default)
stop_h=0;    %m          %final desired altitude

atmOut = sim('SatSim_ARES','Solver',solv_atm,'FixedStep',step_atm,...
    'StartTime','kepOut.time(end)');
```

Results analysis and plotting

```
%   union of 1st and 2nd run results

h=[kepOut.h;atmOut.h];
x=[kepOut.x;atmOut.x];
time=[kepOut.time;atmOut.time];
V=[kepOut.V;atmOut.V];
Vx=[kepOut.Vx;atmOut.Vx];
Vz=[kepOut.Vz;atmOut.Vz];
gamma=[kepOut.gamma;atmOut.gamma];
theta=[kepOut.theta;atmOut.theta];
rho=[kepOut.rho;atmOut.rho];
dotV=[kepOut.dotV;atmOut.dotV];


%   data filtering

%section to be revised
%excludes unsensed data due to discrete stopping criteria
%only (end) value is typically wrong (i.e. h(end)<0)

n_sa=length(time);
k=0;

for i=n_sa:-1:1
    if h(i)<0 %|| h(i)>1.05*(r_a*1000-Re)
        k=k+1;
        h(i)=[];
        x(i)=[];
        time(i)=[];
        V(i)=[];
        Vz(i)=[];
        Vx(i)=[];
        gamma(i)=[];
        theta(i)=[];
        rho(i)=[];
        dotV(i)=[];
    end
end

%   data analysis

%time to deorbit
nyears=time(end)/3.154e+7;
```

```
fprintf('Total time to de-orbit: %3.2f years\n',nyears)

%structural loading
gload=abs(dotV/g0);                %strutural loading [g]
gload_max=max(gload);
h_gload_max=h(gload==gload_max)/1000;
fprintf('Max load factor of %.1f g experienced at an altitude of %.1f
km\n',gload_max,h_gload_max)

%aerothermal load
heat1=1.83e-4*V.^3.*sqrt(rho./rcurv);
heat1_max=max(heat1);
h_heat1_max=h(heat1==heat1_max)/1000;
fprintf('Max heat flux of %3.2e W/m^2 experienced at an altitude of %.1f
km\n',heat1_max,h_heat1_max)

%conversions

h=h/1000;                          %[m] to [km]
x=x/1000;                          %[m] to [km]
time=time/60;                      %[s] to [min]
V=V/1000;                          %[m/s] to [km/s]
Vz=Vz/1000;                        %[m/s] to [km/s]
Vx=Vx/1000;                        %[m/s] to [km/s]
gamma=rad2deg(gamma);              %[rad] to [deg]
%theta=rad2deg(theta);             %[rad] to [deg]

%plotting

nfigure(1,2,3)
plot(V,h)
xlabel('velocity [km/s]')
ylabel('altitude [km]')
grid on
grid minor

nfigure(2,2,3)
plot(h,heat1)
xlabel('altitude [km]')
ylabel('heat flux [W/m^2]')
grid on
grid minor

nfigure(3,2,3)
plot(h,gload)
xlabel('altitude [km]')
ylabel('axial load factor [g]')
grid on
grid minor

nfigure(4,2,3)
plot(time.*60./3.154e+7,h)
xlabel('time [years]')
```

```
ylabel('altitude [km]')
grid on
grid minor

nfigure(5,2,3)
plot(time.*60./3.154e+7,V)
xlabel('time [years]')
ylabel('velocity [km/s]')
grid on
grid minor

nfigure(6,2,3)
plot(gamma,h)
xlabel('flight path angle [deg]')
ylabel('altitude [km]')
grid on
grid minor

%Earth's surface circle generation
circ_ang=0:0.01:2.1*pi;
lcirc=length(circ_ang);
circ_r=ones(1,lcirc).*6371;

nfigure(7,2,3)
polarplot(theta,h+6371,circ_ang,circ_r,...
    theta(1),h(1)+6371,'*g',theta(end),h(end)+6371,'*r')
title('Trajectory shape evolution')
legend('s/c trajectory','Earth''s surface',...
    'Initial position','Landing position')

%Karman line crossing detection
kar_mask=fix(mean(find(h<100.1 & h>99.9)));

%isolation of the last orbit
up=theta(end)-2*pi+0.01;
down=theta(end)-2*pi-0.01;
orb_mask=fix(mean(find(theta<up & theta>down)));

nfigure(8,2,3)
polarplot(theta(orb_mask:end),h(orb_mask:end)+6371,circ_ang,circ_r,...
    theta(kar_mask),h(kar_mask)+6371,'or')
title('Re-entry trajectory')
legend('s/c trajectory','Earth''s surface','Karman line crossing')

nfigure(9,2,3)
plot(Vx,h,Vz,h)
xlabel('velocity [km/s]')
ylabel('altitude [km]')
legend('Vx tangent','Vz radial')
grid on
grid minor
```

2. J71_density_simulink function

```
%This function computes the density using Jacchia J71 model
%rho=J71_density_simulink(H,jdate,F107_avg,F107_act,Kp)
%input: height [90-2500 km], julian date, F10.7 3-monthly average, F10.7
%three hourly average, Kp planetary index
%output: density value
%Reference: Satellite Orbits, section 3.5.3

function rho=J71_density_simulink(H,jdate,F107_avg,F107_act,Kp)
```

Exospheric Temperature

```
T_c = 379.0+3.24*F107_avg+1.3*(F107_act-F107_avg);




% Geographic corrections

% The actual exospheric temperature depends on the local hour angle
% of the Sun with respect to the satellite.
% It also depends, however, on the declination of the Sun and the
% geographic latitude of the satellite.
% The actual exospheric temperature T1 with the diurnal variations
% included can be computed from a more compliated formula available
% in the referenced book.

% Geomagnetic corrections
% using the three-hourly planetary geomagnetic index Kp for a time
% 6.7 hours earlier than the time under consideration
f = 0.5*(tanh(0.04*(H-350)+1));      %transition function
dT_infH = 28.0*Kp+0.03*exp(Kp);      %H>350km
dT_infL = 14.0*Kp+0.02*exp(Kp);      %H<350km
dT_gm = f*dT_infH+(1-f)*dT_infL;

T_e = T_c+dT_gm;   %geomagnetic corrected exospheric temperature
```

Standard Density

```
%Use bi-polynomial fit from Gill (1996)
coeff=getCoeff(H,T_e); %coefficient matrix
logRho=0;
for i=0:5
    for j=0:4
        logRho = logRho + coeff(i+1,j+1)*((H/1000)^i)*((T_e/1000)^j);
    end
end
```

Corrections

```
%Correction due to semi-annual density variation in thermosphere
Phi=(jdate - 2400000.5 -36204)/365.2422; %number of tropical years since Jan
1, 1958
```

```
tauSA=Phi + 0.09544*((0.5+0.5*sin(2*pi*Phi+6.035))^1.65 - 0.5);

fZ=(5.876e-7*H^2.331 + 0.06328)*exp(H*-2.868e-3);
gt=0.02835 + 0.3817*(1+0.4671*sin(2*pi*tauSA+4.137))*sin(4*pi*tauSA+4.259);

logRhoSA=fZ*gt;

%Correction due to geomagnetic activities
if H<350
    logRhoGM = (0.012*Kp + 1.2e-5*exp(Kp))*(1-f);%Transition function from
temp calc
else
    logRhoGM=0;
end

%Seasonal-latitude correction
%logRhoSL=0.014*(Z-90).*exp(-0.0013*(Z-90).^2)*sin(1*pi*Phi -
1.72)*sin(lat)^3/abs(sin(lat));

logRho=logRho+logRhoGM+logRhoSA;
rho=10^logRho;
end
```

3. getCoeff function

```
function c=getCoeff(Z,T)
%Retrieve coefficients for bi-polynomial fit taken from Tables 3.9 and 3.10
%of Satellite Orbits. These numbers were entered manually and may include
%some typing errors. This should be replaced by the process described in
%E. Gill, "Smooth Bi-Polynomial Interpolation of Jacchia 1971 Atmospheric
%Densities For Efficient Satellite Drag Computation," DLR-GSOC IB 96-1,
%German Aerospace Center (DLR), 1996
%This paper is unavailable at this time.

if Z<90 || Z>2500
    error('Z must be in range 90km < Z < 2500km')
end
if T<500 || T>1900
    error('T must be in range 500K < T < 1900K')
end

if T<850
    if Z<1000
        if Z<500
            if Z<180
                c=[-0.3520856e2  0.3912622e1 -0.8649259e2  0.1504119e3 -
0.7109428e2
                      0.1129210e4  0.1198158e4  0.8633794e3 -0.3577091e4
0.1970558e4
                     -0.1527475e5 -0.3558481e5  0.1899243e5  0.2508241e5 -
0.1968253e5
                      0.9302042e5  0.3646554e6 -0.3290364e6 -0.1209631e5
0.8438137e5
                     -0.2734394e6 -0.1576097e7  0.1685831e7 -0.4282943e6 -
0.1345593e6
                      0.3149696e6  0.2487723e7 -0.2899124e7  0.1111904e7
0.3294095e4];
            else
                c=[ 0.2311910e2  0.1355298e3 -0.8424310e3  0.1287331e4 -
0.6181209e3
                     -0.1057776e4  0.6087973e3  0.8690566e4 -0.1715922e5
0.9052671e4
                      0.1177230e5 -0.3164132e5 -0.1076323e4  0.6302629e5 -
0.4312459e5
                     -0.5827663e5  0.2188167e6 -0.2422912e6  0.2461286e5
0.6044096e5
                      0.1254589e6 -0.5434710e6  0.8123016e6 -0.4490438e6
0.5007458e5
                     -0.9452922e5  0.4408026e6 -0.7379410e6  0.5095273e6 -
0.1154192e6];
            end
        else
            c=[-0.1815722e4  0.9792972e4 -0.1831374e5  0.1385255e5 -
0.3451234e4
```

```
                   0.9851221e4 -0.5397525e5  0.9993169e5 -0.7259456e5
0.1622553e5
                  -0.1822932e5  0.1002430e6 -0.1784481e6  0.1145178e6 -
0.1641934e5
                   0.1298113e5 -0.7113430e5  0.1106375e6 -0.3825777e5 -
0.1666915e5
                  -0.1533510e4  0.7815537e4  0.7037562e4 -0.4674636e5
0.3516946e5
                  -0.1263680e4  0.7265792e4 -0.2092909e5  0.2936094e5 -
0.1491676e5];
        end
    else
        c=[ 0.3548698e3 -0.2508685e4  0.6252742e4 -0.6755376e4  0.2675763e4
             -0.5370852e3  0.4182586e4 -0.1151114e5  0.1338915e5 -0.5610580e4
             -0.2349586e2 -0.8941841e3  0.4417927e4 -0.6732817e4  0.3312608e4
              0.3407073e3 -0.1531588e4  0.2179045e4 -0.8841341e3 -0.1369769e3
             -0.1698470e3  0.8985697e3 -0.1704797e4  0.1363098e4 -0.3812417e3
              0.2494943e2 -0.1389618e3  0.2820058e3 -0.2472862e3
0.7896439e2];
    end
else
    if Z<1000
        if Z<500
            if Z<180
                c=[-0.5335412e2  0.2900557e2 -0.2046439e2  0.7977149e1 -
0.1335853e1
                       0.1977533e4 -0.7091478e3  0.4398538e3 -0.1568720e3
0.2615466e2
                      -0.2993620e5  0.5187286e4 -0.1989795e4  0.3643166e3 -
0.5700669e2
                       0.2112068e6 -0.4483029e4 -0.1349971e5  0.9510012e4 -
0.1653725e4
                      -0.7209722e6 -0.7684101e5  0.1256236e6 -0.6805699e5
0.1181257e5
                       0.9625966e6  0.2123127e6 -0.2622793e6  0.1337130e6 -
0.2329995e5];
            else
                c=[ 0.4041761e2 -0.1305719e3  0.1466809e3 -0.7120296e2
0.1269605e2
                      -0.8127720e3  0.2273565e4 -0.2577261e4  0.1259045e4 -
0.2254978e3
                       0.5130043e4 -0.1501308e5  0.1717142e5 -0.8441698e4
0.1518796e4
                      -0.1600170e5  0.4770469e5 -0.5473492e5  0.2699668e5 -
0.4870306e4
                       0.2384718e5 -0.7199064e5  0.8284653e5 -0.4098358e5
0.7411926e4
                      -0.1363104e5  0.4153499e5 -0.4793581e5  0.2377854e5 -
0.4310233e4];
            end
        else
            c=[-0.4021335e2 -0.1326983e3  0.3778864e3 -0.2808660e3
0.6513531e2
```

```
                  0.4255789e3  0.3528126e3 -0.2077888e4  0.1726543e4 -
0.4191477e3
                 -0.1821662e4  0.7905357e3  0.3934271e4 -0.3969334e4
0.1027991e4
                  0.3070231e4 -0.2941540e4 -0.3276639e4  0.4420217e4 -
0.1230778e4
                 -0.2196848e4  0.2585118e4  0.1382776e4 -0.2533006e4
0.7451387e3
                  0.5494959e3 -0.6604225e3 -0.3328077e3  0.6335703e3 -
0.1879812e3];
        end
    else
        c=[ 0.1281061e2 -0.3389179e3  0.6861935e3 -0.4667627e3  0.1029662e3
              0.2024251e3  0.1668302e3 -0.1147876e4  0.9918940e3 -0.2430215e3
             -0.5750743e3  0.8259823e3  0.2329832e3 -0.6503359e3  0.1997989e3
              0.5106207e3 -0.1032012e4  0.4851874e3  0.8214097e2 -0.6527048e2
             -0.1898953e3  0.4347501e3 -0.2986011e3  0.5423180e2  0.5039459e1
              0.2569577e2 -0.6282710e2  0.4971077e2 -0.1404385e2
0.8450500e0];
    end
end
end
```

4. expAtm function

```
%This function calculates density through an exponential model
%rho=expAtm(H)
%H is the geopotential height in km
%rho is the density in kg/m^3
%rho0,h0,SH are updated each 10 km (source: Wertz, 1978)

function rho=expAtm(H)
[rho0,h0,SH]=getSH(H);
rho=rho0*exp(-(H-h0)/SH);    %kg/m^3
end
```

5. getSH function

```
%[rho0,h0,SH]=getSH(H) this function gives coefficient for an exponential
atmosphere model
%works below 100 km
function [rho0,h0,SH]=getSH(H)

%input data from table 8-4 of Fundamentals of Astrodynamics, D.A. Vallado
h0=[0,25,30,40,50,60,70,80,90];
rho0=[1.225,3.899e-2,1.774e-2,3.972e-3,1.057e-3,3.206e-4,8.770e-5,1.905e-
5,3.396e-6];
SH=[7.249,6.349,6.682,7.554,8.382,7.714,6.549,5.799,5.382];

n=fix(H/10);

if H>100 || H<0
    error('H must be in the range 0km < H < 100 km; current value H=%.2f
km\n',H)
end

if H<30
    n=2;
    if H<25
        n=1;
    end
end

if H==100
    n=9;
end

h0=h0(n);

rho0=rho0(n);
SH=SH(n);
end
```

RINGRAZIAMENTI

Giunto ad un piccolo traguardo dei miei studi universitari, desidero ringraziare tutti coloro che in questi tre anni intesi hanno contribuito alla mia crescita personale ed accademica.

In primo luogo voglio ringraziare la mia famiglia, per sostenermi in ogni scelta e per non farmi mai mancare il vostro affetto, se non fosse per voi non sarei arrivato fin qui. Ringrazio in particolare Giulia, per avermi insegnato che la pazienza non è mai troppa.

Ringrazio il professor Alfredo Locarini per avermi dato la possibilità di svolgere questo lavoro sotto la sua supervisione e per la disponibilità dimostrata nei miei confronti in svariate occasioni.

Ringrazio gli amici di sempre, per tutte le esperienze vissute insieme e per ricordarmi che ogni tanto la vita va presa alla leggera. Grazie per aiutarmi a sgombrare la testa nei periodi più stressanti.

Ringrazio Giacomo, Paolo e Raoul per l'ironia e la goliardia che hanno portato alla nascita dei *Quattro Ingegneri*. La vostra compagnia ha fatto sì che questi tre anni fatti di treni regionali, situazioni improbabili e talvolta qualche lezione siano davvero volati. La vita universitaria senza di voi sarà certamente più monotona, anche se abbiamo ancora tanto da fare insieme.

Un grazie va a tutte le persone conosciute a Forlì, in particolare a Chiara, Filippo e Melania, diventati ben più che semplici compagni di corso.