April 14, 2018

Project Rocksanne 2-Alpha
Phase B

Documentation Simulator
REV. 0

SKYWARD EXPERIMENTAL ROCKETRY

Mission Analysis
Integrated Project Team

Skyward Experimental Rocketry
Politecnico di Milano

Author: Akram Ben Dhia, Matteo Pozzoli, Luca Mingozzi
Editor: Mauro De Francesco

Abstract

The purpose of this document is to describe the physics underlying the mechanics behind the simulator as well as how these physical phenomena are modeled and later transcribed into code. As such, the structure of the simulator will be discussed in detail with the help of figures and tables. Another aim is to introduce the basics needed to run the simulator correctly and properly understand the results displayed.

The simulator was created in order to predict the behavior of a sounding rocket flight during all its flight phases. All calculations (trajectory, forces, acceleration, etc.) will be used to roughly dimensionalize the rocket and its structure within the preliminary mission analysis.

Website:
http://www.skywarder.eu

E-mail:
mauro.defrancesco@skywarder.eu
akram.bendhia@skywarder.eu
matteo.pozzoli@skywarder.eu
luca.mingozzi@skywarder.eu

Contents

List of Figures

List of Tables

Change Log

Date	Revision	Editor	Changes
Data	0	Akram Ben Dhia, Matteo Pozzoli, Luca Mingozzi	First Version.

Nomenclature

$C_{D_{para}}$ Drag coefficient for parachute

$\mathcal{D}$ Aerodynamic drag

$\mathbf{f}$ External force acting on the rocket

g (Local) gravitatonal acceleration

$\mathbf{I}$ Inertia tensor

$\mathbf{I}_e$ Rocket inertia tensor at burnout

$\mathbf{I}_f$ Rocket inertia tensor at ignition

IPT Integrated Project Team

$\mathbf{k}$ Rotational momentum

$\mathcal{L}$ Roll aerodynamic moment

$\mathcal{M}$ Pitch aerodynamic moment

$\mathcal{N}$ Yaw aerodynamic moment

m Rocket mass

m_0 Full rocket mass at ignition

m_{para} Mass of parachute

m_s Rocket mass at burnout ($t = t_b$) — structural mass

Ω Attitude quaternion matrix (matrix that relates $\dot{\mathbf{q}}$ to $\mathbf{q}$)

$\mathbf{p}$ Rocket momentum

p x-component of angular rate (in body frame)

$\mathbf{q}$ attitude quaternion

q y-component of angular rate (in body frame)

$\hat{\mathbf{q}}$ Vectorial part of a quaternion: q_1, q_2, q_3

$\mathbf{r}$ Position vector to rocket center of mass

r z-component of angular rate (in body frame)

$\mathbf{R}_{A\rightarrow B}$ Rotational tensor that rotates a vector from frame A to frame B

ρ density

$\boldsymbol{\tau}$ External moment acting on the rocket

u x-component of vehicle velocity

$\tilde{\mathbf{u}}$ Effective speed (with wind effects)

v y-component of vehicle velocity

$\boldsymbol{\omega}$ Angular velocity vector

$\mathcal{W}$ Weight

w z-component of vehicle velocity

x x-axis coordinate

x_{cg} Center of mass position from nosetip

x_{cg} Center of mass position from nosetip

x_{cp} Center of pressure position from center of mass

x_{cp} Center of pressure position from center of mass

y y-axis coordinate

z z-axis coordinate

Notation summary

Vectors

$$\mathbf{a} \text{ or } \mathcal{A}$$

Vectors components

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \text{ or } \begin{bmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \end{bmatrix}$$

Tensors

$$\mathbf{A}$$

Tensors components

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

Column of a matrix with index i

$$\operatorname{col}(\mathbf{A})_i = \begin{bmatrix} A_{1i} \\ A_{2i} \\ A_{3i} \end{bmatrix}$$

Cross Product

$$\mathbf{a} \times \mathbf{b} = \mathbf{a}_\times \mathbf{b} = \underbrace{\begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}}_{\mathbf{a}_\times} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{1}$$

Chapter 1

6 D.o.F. Model of an ascending Rocket

Chapter 2

6 D.o.F. Model of an ascending Rocket

2.1 Hypotheses and assumptions

For the development of the rocket flight model few assumptions have been made (as also reported in [1] for a plane rigid model):

Flat Earth

The altitude and the length of the flight path are relatively small in comparison with the Earth typical lenghts so the curvature has been neglected.

Earth's surface-centered frame as inertial frame

The acceleration given by the most important Earth's movement, rotation, has been neglected. This assumption has the meaning of considering the reference frame placed on the surface of the Earth as inertial or, equivalently, the Coriolis accelerations are not considered.

Axisymmetric Rocket

The body-centered frame, that is the center of mass of the rocket, has one axis (the x-axis) that gives mass symmetry, the other two axes are taken to complete the principal reference frame [1]. This is also verified with the CAD calculation of the mass properties (see [2] for more details).

2.2 Reference Frames

The path flown by the rocket could be requested in different reference frames. In particular the trajectory, given in the form of spatial coordinates x, y, z, is requested in the inertial frame. The velocities u, v, w could be given in the inertial frame or in the body frame; so a distinction and explanation of the different reference frames used in the dynamics model is now exposed.

[1] In a principal reference frame the inertia tensor $\mathbf{I}$ is a diagonal matrix $\begin{bmatrix} I_x & & \\ & I_y & \\ & & I_z \end{bmatrix}$ that doesn't change with attitude variation.

2.2.1 North-East-Down Frame (NED)

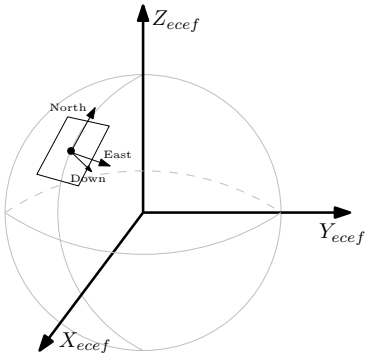


Figure 2.1: NED frame

The inertial frame on the Earth surface is intended as a North-East-Down frame. The x-axis is pointing North, the y-axis is pointing East, the z-axis is pointing downward to the Earth's center. The spatial coordinates describing the trajectory are given in this reference frame.

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{NED}} \tag{2.1}$$

2.2.2 Body Frame

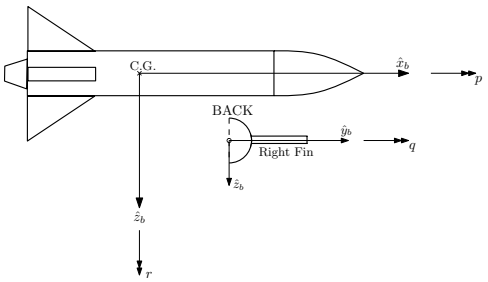


Figure 2.2: Body Frame

The body frame is centered in the center of mass of the rocket, aligned with the mass symmetry axes (that, as stated in sec. section §2.1, coincide with symmetry axes). The linear velocities could be expressed in this frame

$$\mathbf{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}_{\text{B}} \tag{2.2}$$

and the angular rates are also expressed in this reference frame

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}_B \tag{2.3}$$

2.3 Rotations

Since some forces are easier to be written in NED frame and others in body frame, rotations from a frame to the other are needed. The rotations are achieved through the use of **unit quaternions** as reported in [3] , [4] or [5].
This parametrization has been chosen to avoid the gimbal lock that typically occurs when the attitude is near the vertical flight[2], flight condition that is typical for a rocket flight.

2.4 Dynamics Equations

The equation written in the body frame, that is not an inertial frame, as referred in 2.2.2.

2.4.1 Mass Properties

The mass rocket mass properties $(m(t), \mathbf{I}(t))$ are functions of time: they change during the thrusted flight. The model used is a linear model (so the rates of change for the mass properties are constant:

$$\begin{cases} \dot{m} = & \text{constant} = -\frac{m_0 - m(t_b)}{t_b} \\ \dot{\mathbf{I}} = & \text{constant} = -\frac{\mathbf{I_f} - \mathbf{I_e}}{t_b} \end{cases} \tag{2.4}$$

where $\mathbf{I}_f$ and $\mathbf{I}_e$ are respectively the inertia tensor for the wet configuration (with all the propellant embarked) and dry configuration (all the propellant burnt), calculated with CAD softwares or experimentally.

Traslational Equation

Using Poisson's relation for the kinematics in a non-inertial frame, the momentum equation for the rocket would be:

$$\left(\frac{d\mathbf{p}}{dt}\right)_{NED} = \left(\frac{d\mathbf{p}}{dt}\right)_B + \boldsymbol{\omega}_B \times \mathbf{p}_B = \sum_i \mathbf{f}_i \tag{2.5}$$

being

$$\mathbf{p}_B = \begin{bmatrix} mu \\ mv \\ mw \end{bmatrix}$$

and

$$\boldsymbol{\omega}_B \times \mathbf{p}_B = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} mu \\ mv \\ mw \end{bmatrix}$$

[2] e.g. Tait-Bryan 'ZYX' sequence (yaw, pitch, roll) where the gimbal lock occurs for pitch angle = 90^O

$$\begin{cases} m(\dot{u} + qw - rv) + \dot{m}u = & f_x \\ m(\dot{v} + ru - pw) + \dot{m}v = & f_y \\ m(\dot{w} + pv - qu) + \dot{m}w = & f_z \end{cases} \tag{2.6}$$

f_x, f_y, f_z are the external forces acting on the rocket during flight.

$$\begin{cases} f_x = & \mathcal{T} - \mathcal{D}_x + \mathcal{W}_x \\ f_y = & -\mathcal{D}_y + \mathcal{W}_y \\ f_z = & -\mathcal{D}_z + \mathcal{W}_z \end{cases} \tag{2.7}$$

$\mathcal{D}_x$, $\mathcal{D}_y$, $\mathcal{D}_z$ are the components of aerodynamic drag in body axes (see 2.4.3 for details), $\mathcal{W}_x$, $\mathcal{W}_y$, $\mathcal{W}_z$ are the components of weight in body axes[3].

Rotational Equation

Using Poisson's relation for the kinematics in a non-inertial frame, the rotational momentum equation for the rocket would be:

$$\left(\frac{d\mathbf{k}}{dt}\right)_{NED} = \left(\frac{d\mathbf{k}}{dt}\right)_B + \boldsymbol{\omega}_B \times \mathbf{k}_B = \sum_i \boldsymbol{\tau}_i \tag{2.8}$$

being

$$\mathbf{k}_B = \mathbf{I}\boldsymbol{\omega}_B = \begin{bmatrix} pI_x \\ qI_y \\ rI_z \end{bmatrix}$$

In an expanded form we get:

$$\begin{cases} \dot{p}I_x + p\dot{I}_x + qr(I_z - I_y) = & m_x \\ \dot{q}I_y + q\dot{I}_y + pr(I_x - I_z) = & m_y \\ \dot{r}I_z + r\dot{I}_z + pq(I_y - I_x) = & m_z \end{cases} \tag{2.9}$$

m_x, m_y, m_z are the external moments acting on the rocket during flight (thrust misalignement is neglected), $\dot{I}_i$ is the i-th rate of change of inertia tensor, calculated as:

$$\dot{\mathbf{I}} = \frac{\mathbf{I}_f - \mathbf{I}_e}{t_b} = \begin{bmatrix} \dot{I}_x & & \\ & \dot{I}_y & \\ & & \dot{I}_z \end{bmatrix} \tag{2.10}$$

$$\begin{cases} m_x = & \mathcal{L} \\ m_y = & \mathcal{M} \\ m_z = & \mathcal{N} \end{cases} \tag{2.11}$$

$\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$ are respectively the roll, pitch and yaw aerodynamic moment acting on the rocket (see **??** for details).

[3] $\begin{bmatrix} \mathcal{W}_x \\ \mathcal{W}_y \\ \mathcal{W}_z \end{bmatrix}_B = \mathbf{R}_{NED \to B} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}_{NED}$

2.4.2 Kinematics Relations

The attitude expressed through a quaternion $\mathbf{q}$ is related to angular rates of the rocket. The expression that relates the two quantities is (as reported in [5, 3]) is:

$$\dot{\mathbf{q}} = \frac{1}{2} \underbrace{\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}}_{\Omega} \mathbf{q} \tag{2.12}$$

2.4.3 Aerodynamics model

The model for the aerodynamic actions acting on the rocket is taken from [6] and herebelow reported in summary[4].

$$\begin{cases} \mathcal{D}_x &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|^2\,SC_A \\ \mathcal{D}_y &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|^2\,SC_{Y\beta}\,\beta \\ \mathcal{D}_z &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|^2\,SC_{N\alpha}\,\alpha \\ \mathcal{L} &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|\,SC\left(|\tilde{\mathbf{u}}|\,C_l + \frac{C_{lp}\,p\,C}{2}\right) \\ \mathcal{M} &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|\,SC\left(|\tilde{\mathbf{u}}|\,C_{m\alpha}\,\alpha + \frac{(C_{m\dot{\alpha}}+C_{mq})\,q\,C}{2}\right) \\ N &= \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|\,SC\left(|\tilde{\mathbf{u}}|\,C_{n\beta}\,\beta + \frac{(C_{nr}\,r+C_{np}\,p)\,C}{2}\right) \end{cases} \tag{2.13}$$

The $|\tilde{\mathbf{u}}|$ is the effective velocity (wind effects[5] included)

$$\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u}_{\text{wind}} \tag{2.14}$$

The Aerodynamic coefficients are calculated using the Missile Datcom software version 1997 (see [7] for a complete documentation), sample input file is attached at the end of the report (B.1) and the complete set of coefficients is loaded in the files `for006_155.mat` and `for006_145.mat` distributed together with the script associated to this work. Also the classic Barrowman method and Extended Barrowman Method have been investigated, see [8].

[4] Rewritten in a way that avoids the presence of $|\tilde{\mathbf{u}}|$ at the denominator that is a issuing situation in the first phases of simulation when $|\tilde{\mathbf{u}}| \sim \mathbf{0}$

[5] Wind constant in altitude $\frac{\partial \mathbf{u}_{\text{wind}}}{\partial z} = \mathbf{0}$.

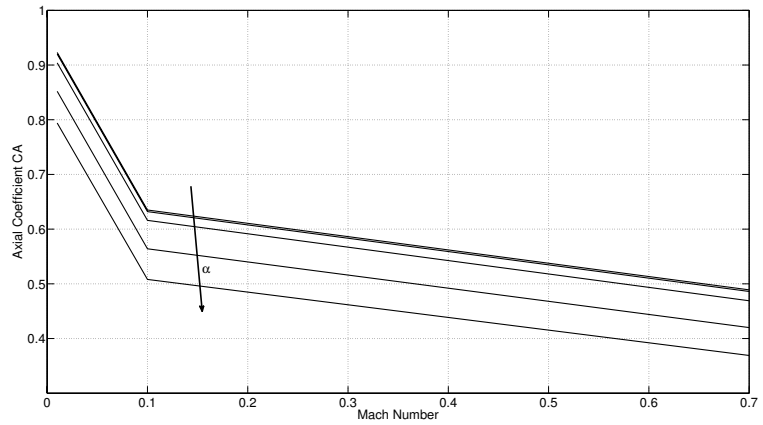


Figure 2.3: Axial Coefficient calculated with Missile Datcom

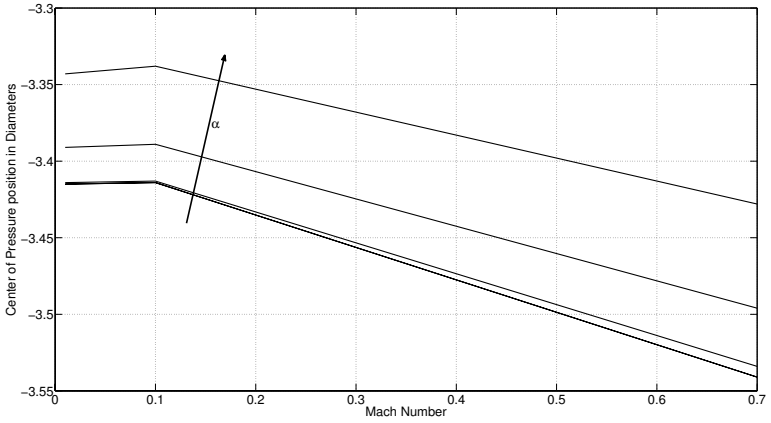


Figure 2.4: Center of Pressure position x_{cp} from center of mass x_{cg} expressed in calibers (static margin)

2.4.4 Final ODE System

The final ODE system counts 15 differential equations and it's reported here below

$$
\begin{cases}
\dot{x} &= u_{NED} \\
\dot{y} &= v_{NED} \\
\dot{z} &= w_{NED} \\
\dot{u} &= \frac{\mathcal{T}-\mathcal{D}_x+\mathcal{W}_x-m(qw-rv)}{m} \\
\dot{v} &= \frac{\mathcal{D}_y+\mathcal{W}_y-m(ru-pw)}{m} \\
\dot{w} &= \frac{\mathcal{D}_z+\mathcal{W}_z-m(pv-qu)}{m} \\
\dot{p} &= \frac{\mathcal{L}-qr(I_z-I_y)-p\dot{I}_x}{I_x} \\
\dot{q} &= \frac{\mathcal{M}-pr(I_x-I_z)-q\dot{I}_y}{I_y} \\
\dot{r} &= \frac{\mathcal{N}-pq(I_y-I_x)-r\dot{I}_z}{I_z} \\
\dot{q}_0 &= -q_1 p - q_2 q - q_3 r \\
\dot{q}_1 &= q_0 p + q_3 q - q_2 r \\
\dot{q}_2 &= -q_3 p + q_0 q + q_1 r \\
\dot{q}_3 &= q_2 p - q_1 q + q_0 r \\
\dot{m} &= -\frac{m_0-m(t_b)}{t_b} \\
\dot{\mathbf{I}} &= -\frac{\mathbf{I_f}-\mathbf{I_e}}{t_b}
\end{cases}
\tag{2.15}
$$

note that the velocity for the spatial coordinates are in NED frame, so the velocity vector must be rotated:

$$\mathbf{u}_{NED} = \mathbf{R}_{B\to NED}\mathbf{u}_B$$

Chapter 2

3 D.o.F Model for Parachute descent

Chapter 3

3 D.o.F Model for Parachute descent

At the apogee the rocket ejects the first smaller parachute, the *drogue parachute*, that has the target to slow down the rocket without too much dispersion on the landing spot. Reached the main parachute opening altitude h_{main}, main parachute is opened, the velocity is reduced and the rocket lands softly to the ground. For better explanation on the parachute deployment procedure check [9].

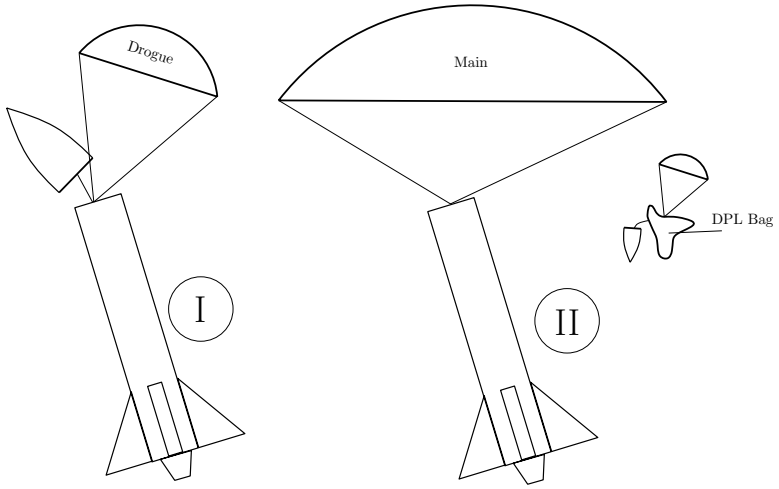


Figure 3.1: Parachute Deployment Procedure

3.1 Dynamics equations

The model used for the parachute is quite simpler, compared to the rocket one. The descending rocket body is modeled as a point with mass $\tilde{m} = m_s + m_{para}$ which has a reference surface S_{para} and drag coefficient $C_{D_{para}}$.

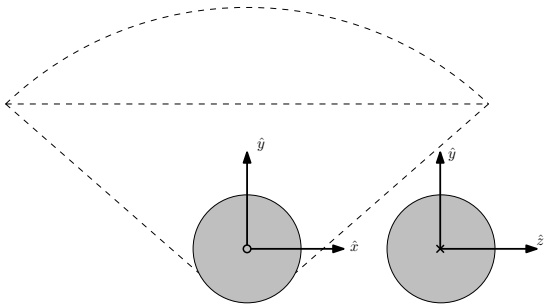


Figure 3.2: Rocket descent with parachute model

The equations describing the descending trajectory of the body rocket+parachute are only **translational**; recalling 2.5 and since $\boldsymbol{\omega} = \mathbf{0}$ for this model, the ODE system is:

$$\left(\frac{d\mathbf{p}}{dt}\right)_{NED} = \left(\frac{d\mathbf{p}}{dt}\right)_{B} = \sum_{i} \mathbf{f}_i \tag{3.1}$$

$$\begin{cases} \tilde{m}\dot{u} &= -\mathcal{D}_x \\ \tilde{m}\dot{v} &= -\mathcal{D}_y \\ \tilde{m}\dot{w} &= -\mathcal{D}_z + \tilde{m}g \end{cases} \tag{3.2}$$

3.1.1 Aerodynamics Model

The model adopted in this phase for the aerodynamics characteristics is simple. The aerodynamic coefficient for the parachute and its surface are assumed constant. Drag of linkers and strings is neglected.

$$\mathcal{D} = \frac{1}{2}\rho\,|\tilde{\mathbf{u}}|^2\, S_{para}\, C_{D_{para}} \tag{3.3}$$

$$\begin{cases} \mathcal{D}_x &= \mathcal{D}\frac{u}{|\tilde{\mathbf{u}}|} \\ \mathcal{D}_y &= \mathcal{D}\frac{v}{|\tilde{\mathbf{u}}|} \\ \mathcal{D}_z &= \mathcal{D}\frac{w}{|\tilde{\mathbf{u}}|} \end{cases} \tag{3.4}$$

3.1.2 Final ODE System

The final ODE system for rocket descent with parachute counts 6 ordinary differential equations

$$\begin{cases} \dot{x} &= u \\ \dot{y} &= v \\ \dot{z} &= w \\ \tilde{m}\dot{u} &= -\mathcal{D}_x \\ \tilde{m}\dot{v} &= -\mathcal{D}_y \\ \tilde{m}\dot{w} &= -\mathcal{D}_z + \tilde{m}g \end{cases} \tag{3.5}$$

this model here is used both for the drogue and the main parachutes descents.

Chapter 3

Model limitations

The simulator is considered to be finished for the ascend phase. However, some problems do occur in the physical model as far as the descend phase is concerned. The main issue is the lack of degrees of freedom during all parachutes phase. In fact, only three are considered for the parachute and the rocket is modeled as a single rigid body. In order to model the body in a more realistic manner, twelve degrees of freedom are needed: six for the rocket and six for the parachute. This enables an accurate prediction of the the descend phase's dynamics as well as the landing point of the rocket.

Another factor of influence on the landing zone prediction is the relative inaccuracy of the wind model (particularly changes of wind with altitude and prediction of the upper level wind). Moreover, the aerodynamic model should be validated, and aerodynamic coefficients should be tested in wind gallery or validated.

The last apparent limitation is the transient phase from the apogee to the opening of the drogue (including the stabilization of the two bodies-system). This transition occurs again at the opening of the main parachute and eventually the Rogallo wing.

Chapter 4

Main Input

The simulator contains two main input files: a (.m) file called config and a (.mat) file containing two matrices . With the information contained within these files, the program has all the needed data concerning the rocket and the simulation environment.

- **config.m**

  The configuration file is a MATLAB (.m) file located in the simulator root folder. It sets the simulation parameters of (type of simulation, wind, location, etc.) and the rocket parameters (engine, geometry, mass distribution, etc.).

- **The matrices**

  The matrices are two MATLAB matrices describing two states respectively: one with the parameters of the rocket full of fuel (rocketname-full.mat) and another with the parameters of the empty rocket (rocketname-empty.mat). Each matrix contains the flight parameters (altitude, Mach numbers, Angle of Attack,...) and all aerodynamic parameters during the flight phases previously defined in Chapter 2. These matrices were generated by transforming the for006.dat file (The .dat file is an output of the NASA software missile datcom 1 that compiles all the aerodynamic parameters of the rocket.) using a python script called parser.

Chapter 5

Simulator structure

The simulator is composed of 22 .m files in Matlab and these are divided in 4 scripts and 18 functions as follows:

- **Scripts:**
  - ascend_plot.m
  - config.m
  - standard_plot.m
  - start_simulation.m
- **Functions:**
  - ascend.m
  - ballistic_descent.m
  - descent_parachute.m
  - event_apogee.m
  - event_drg2_opening.m
  - event_landing.m
  - event_main_opening.m
  - interp4_easy.m
  - parfor_progress.m
  - std_run_ballistic.m
  - std_run.m
  - stoch_run_bal_p.m
  - stoch_run_bal.m
  - stoch_run_p.m
  - stoch_run.m
  - wind_const_generator.m
  - wind_matlab_generator.m
  - wind_vert_generator.m

Every single script listed will be explained in detail in the following parts of this chapter.

In order to have an idea about the path of the simulator, a schematic flowchart is presented:

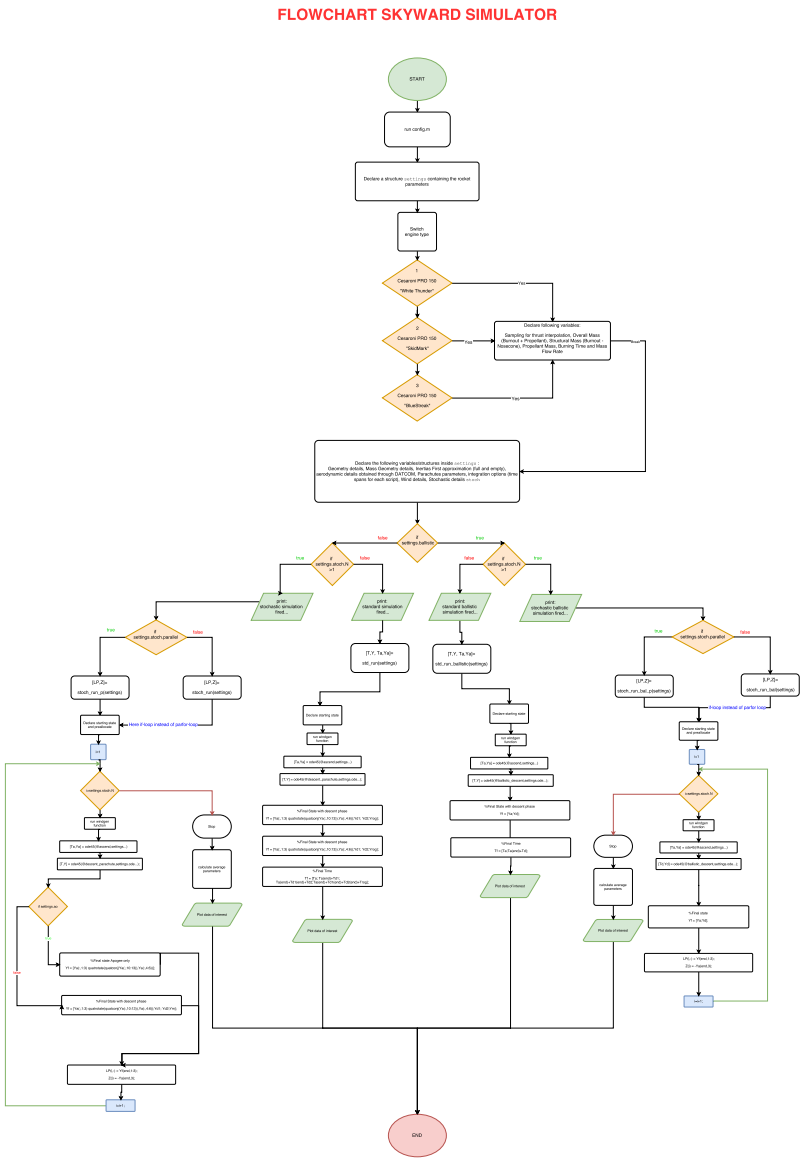


Figure 5.1: Flowchart

CONFIG.M

It contains all possible settings of the simulator and is the only file you have to edit beforehand in order to set all parameters, input and output for the program.
You can edit the following features within:

- the rocket start position
- the motor settings
- geometry, mass and aerodynamic details of the rocket chosen
- the parachutes options
- integration options
- The wind generator
- type of simulation
- Plot settings

START_SIMULATION.M

This script is the MAIN of the simulator and the one you execute in order to start the simulator. The steps are roughly outlined below:

1. It runs the config.m file to load the needed data in the MATLAB workspace.
2. According to the selected type of simulation, the corresponding functions are executed.
3. State vectors of position, velocities and accelerations during all flight phases are extracted from the matrix (Y).
4. The maximum values of all three vectors are computed.
5. Temperature of nose cone and Mach number are computed.
6. Series of ‘fprintf’ display the relevant data collected during the simulation onscreen.
7. If the condition of the plots are set to true, two scripts are then executed in order to plot the data.

STANDARD_PLOT.M

The script plots the following figures:

- Altitude vs time
- Velocities vs time
- Acceleration vs time
- 3D Trajectory (East [m]and North [m] with respect to the altitude [m])
- Angular rates vs time
- Mach number vs time
- Stagnation temperature vs time

ASCEND_PLOT.M

The script plots the following figures:

- Thrust vs time
- Angle Alpha vs time
- Angle Beta vs time
- Aerodynamic Coefficients vs time
- Axial Force vs time
- Mach number vs time
- Stability margin vs time

STD_RUN.M

INPUTS		Settings => set in the config.m file
OUTPUTS	**Tf**	Time steps of all flight phases -return to main with "T"
	Yf	Matrix of state with position and velocities -return to main with "Y"
	Ta	Time steps for ascend only -return to main with "Ta"
	Ya	Matrix of state for ascend only -return to main with "Ya"
FUNCTION EXPLANATION	This function calls back a series of function in order to calculate the different flight phases, in this case: (wind generation, ascend, drogue1,drogue2,rogallo). Then it assembles all the output of the series of functions in the main output of this function(Tf,Yf,ecc...) and plots the data directly in the function if the setting default plot is on true.	
EXPLANATION OF PARTICULAR LINE	**37-**	Call of the function for constant wind generator
	44-	Call of "ode45" function for the ascend
	51-71	Call of "ode45" function for the descend phase
	-	For "ode45" function see matlab documentation

STD_RUN_BALLISTIC.M

INPUTS		Settings => set in the config.m file
OUTPUTS	**Tf**	Time steps of all flight phases -return to main with "T"
	Yf	Matrix of state with position and velocities -return to main with "Y"
	Ta	Time steps for ascend only -return to main with "Ta"
	Ya	Matrix of state for ascend only -return to main with "Ya"
FUNCTION EXPLANATION	The function runs a ballistic simulation for the phase of descent with no parachute. In order to calculate the different phases, the function ascend and ballistic descend are used. Then the function assembles the matrices of times and states for the output and plots the data directly in the function if the user wishes so.	

EXPLANATION OF PARTICULAR LINE	**50-**	Call of the ode45 function for the ballistic descent
	56-58	Assemble of the matrices by column

STOCH_RUN.M/STOCH_RUN_P.M[1]

INPUTS		Settings => set in the config.m file
OUTPUTS	**LP**	Vector of position of the different landing points calculated
	Z	Vector of altitude at the apogee for each iteration
FUNCTION EXPLANATION	This function calculates with a 'for' loop each time a standard simulation as seen in "std_run.m" with different wind intensity and direction. The different landing zone and altitude are saved in the outputs. Then there is a part to display the data of interest and for the plotting of data (this plot is different from the one for non stochastic function).	
EXPLANATION OF PARTICULAR LINE	**32-**	For loop for calculating the number of simulations set in the config file
	36-79	Same path as std_run.m
	82-84	Creation of the output vector

[1] The file with _p.m use a "parfor" loop instead a simple for to solve the problem in parallel.

STOCH_RUN_BAL.M/STOCH_RUN_BAL_P.M2[2]

INPUTS		Settings => set in the config.m file
OUTPUTS	**LP**	Vector of the different landing points calculated
	Z	Vector of altitude at the apogee for each iteration
FUNCTION EXPLANATION	This function calculated with a 'for' loop each time a ballistic simulation similar to "std_run_ballistic.m" with different wind intensity and direction. The different landing zones and altitudes are saved in the outputs. And for the rest, it is the same as in "stoch_run.m".	
EXPLANATION OF PARTICULAR LINE		

[2] The file with _p.m use a "parfor" loop instead a simple for to solve the problem in parallel.

ASCEND.M

INPUT		Settings => set in the config.m file
	t	time
	Y	State variable
	uw	Wind velocity (1st component)
	vw	Wind velocity (2nd component)
	ww	Wind velocity (3rd component)
OUTPUT	**dY**	Increment in the state
SCRIPT EXPLANATION	The function does an interpolation using the full and empty configurations. This linear interpolation between the two conditions computes the value of the aerodynamics coefficients at a certain time. The function then saves some persistent variables which will be valuable in the plotting.	
EXPLANATION OF PARTICULAR LINE		

BALLISTIC_DESCENT.M

INPUT		Settings => set in the config.m file
	t	time
	Y	State variable
	uw	Wind velocity (1st component)
	vw	Wind velocity (2nd component)
	ww	Wind velocity (3rd component)
OUTPUT	**dY**	Increment in the state
SCRIPT EXPLANATION	The function does an interpolation using the empty and DATCOM configurations. This linear interpolation between the two conditions computes the value of the aerodynamics coefficients at a certain time. The function then computes the final derivative state assembling state.	
EXPLANATION OF PARTICULAR LINE		

DESCENT_PARACHUTE.M

INPUT		Settings => set in the config.m file
	t	time
	Y	State variable
	uw	Wind velocity (1st component)
	vw	Wind velocity (2nd component)
	ww	Wind velocity (3rd component)
OUTPUT	**dY**	Increment in the state
SCRIPT EXPLANATION	The function sets the aerodynamic coefficients following the parachute deployed in the simulation. The parachutes are approximated as rectangular surfaces with the normal vector perpendicular to the relative velocity.	
EXPLANATION OF PARTICULAR LINE		

WIND_CONST_GENERATOR.M

INPUTS	**AzMin**	Minimum angle of azimuth from north set in the config file
	AzMax	Maximum angle of azimuth from north set in the config file
	ElMin	Minimum elevation set in the config file
	ElMax	Maximum elevation set in the config file
	MagMin	Maximum magnitude set in the config file
	MagMax	Maximum magnitude set in the config file
OUTPUTS	**uw**	First horizontal component of the wind in NED
	vw	Second horizontal component of the wind in NED
	ww	Vertical component of the wind in NED
FUNCTION EXPLANATION	This function evaluate a random value in the interval set in the config file and pass by the input of the function for the angle of azimuth, elevation, magnitude. Then calculated the cosine matrix with the function "angle2dcm" and extract from the matrix the three component of the wind in NED axis	
EXPLANATION OF PARTICULAR LINE	**19-21**	Evaluate random value for the angle and magnitude
	24-	For "angle2dcm" see the matlab documentation
	25-27	Extract from matrix 'R' the component of wind

WIND_MATLAB_GENERATOR.M

INPUTS		Settings => set in the config.m file
	z	Altitude of the rocket
	t	Time
	Q	Matrix of rotation with quaternions
OUTPUTS	**uw**	First horizontal component of the wind in NED
	vw	Second horizontal component of the wind in NED
	ww	Vertical component of the wind in NED
FUNCTION EXPLANATION	This function is used for evaluate the two horizontal component of wind according to hwm07 with the function "atmoshwm" that need the position, the day of the year and time of the day. The vertical wind is considered constant and set in config file. If matrix Q was given as input the function rotate the component in body frame instead,as standard, in NED frame.	
EXPLANATION OF PARTICULAR LINE	**24-**	For "atmoshwm" see the matlab documentation
	26-	Set the vertical wind
	29-	Control the number of input in this case if is present 'Q' as input
	31-	Rotate the wind from NED frame to body FRAME

WIND_VERT_GENERATOR.M

INPUTS	**MagMin**	Min magnitude of the wind set in the config file
	MagMax	Max magnitude of the wind set in the config file
OUTPUTS	**ww**	Vertical component of the wind
FUNCTION EXPLANATION	This function generate from an interval of magnitudes gave by the user a random magnitude for the vertical wind. The value is positive for random values > 0.5 and negative for random values < 0.5 .	
EXPLANATION OF PARTICULAR LINE		

EVENT_*****.M[3]

INPUTS	**Value** **Isterminal** **Direction**	Variables used in ode45 function.
OUTPUTS	**t** **Y** **settings** **varagin**	Variables containing position, velocities, angular rates, time, ecc… of the rocket in a particular point of the launch.
FUNCTION EXPLANATION	Event.m functions stop an integration (with ode45 function) to allow the boot of another integration. For example event_ascend.m apogee stops ode45 integration when vz=0. Then simulator runs the integration about the descent phases.	

[3] This scheme is valid for all files called event_something.

interp4_easy.m

INPUTS	**(X1...X4)**	Column vectors of the variables the function depends on
	(x1...x4)	Single values for each variable on which the interpolation should be done.
OUTPUTS	**V**	[NxMxLxI] matrix containing the interpolated values.
FUNCTION EXPLANATION	This function interpolates with nearest-neighbor method a R4->R function F(x1...x4), given a [NxMxLxI] Matrix that discretizes the function.	
EXPLANATION OF PARTICULAR LINE		

Chapter 6

Main Output

The simulator processes input (through tweaking the config.m script) to give back data of importance to the subsequent analysis and study of the features that are to be improved upon.

The amount as well as the type of outputs change according to the type of simulator chosen during input compilation phase. In fact, there is:

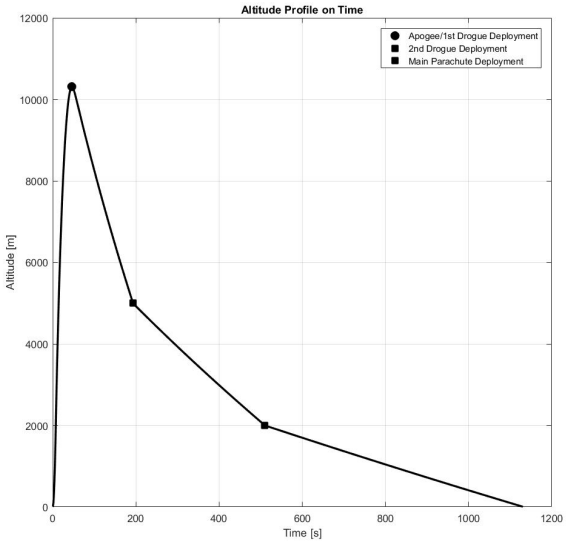
- **Ballistic simulation:** simulation without drogues opening.
- **Last drogue failure simulation:** simulation without opening the second drogue.
- **Apogee only simulation:** this simulation stops when the rocket reaches the apogee, so the output is missing some data usually found in the other simulations like e.g. opening of drogues or landing.
- **Stochastic simulation:** this simulation gives a number of output parameters according to the chosen number of stochastic simulations.

While starting the simulation with the same configuration, there will always be a slight difference between two or more output parameters; this is due to mainly to the wind being randomly generated. As such, the conditions change for each simulation.

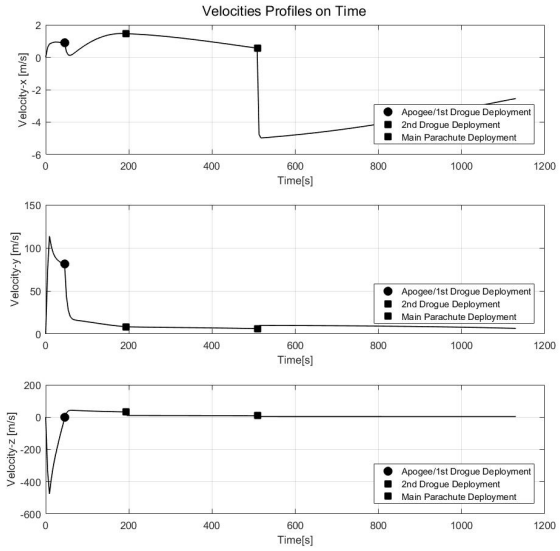
The generated output are mostly in the form of graphs and some data displayed in the command window at the end of each simulation:

- **Data:**
  - Maximum altitude reached (Apogee)
  - Maximum speed reached
  - Maximum Mach number reached (It can differ from the Mach number at maximum speed because the Mach number depends on air temperature and density. If these values are constant, then the maximum speed coincides with the maximum Mach number.)
  - Maximum acceleration reached (The speed at launch pad exit is also displayed.)

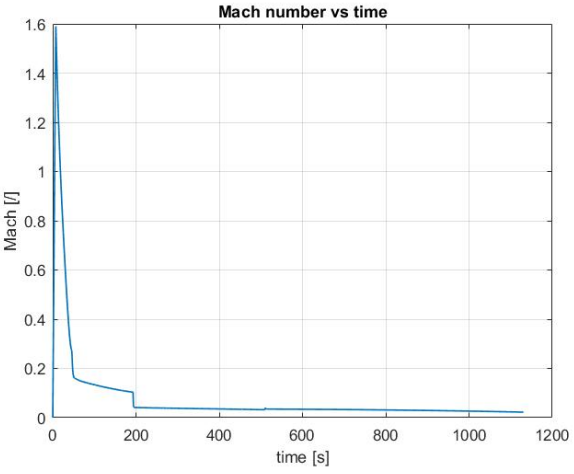
Altitude profile



Velocity profile



Mach Number profile



Acceleration profile

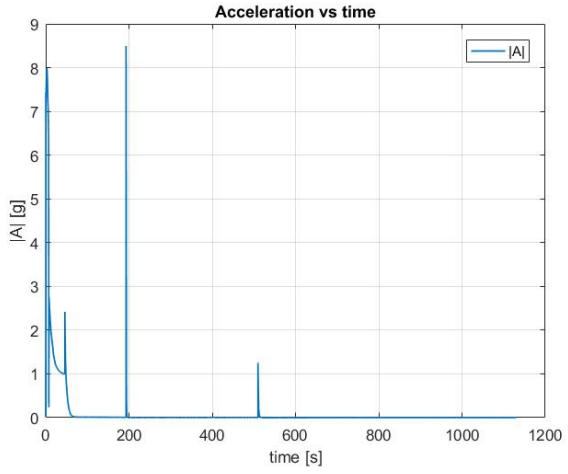


Table 6.1: Overview of the displayed data

- **Graphs:**
  - Angles vs time: displays trends of angles as a function of time (from 0 to Apogee), according to rocket axis (alpha, beta and roll) and according to Earth axes, where the x-axis coincides with the North and the y-axis coincides with the East (pitch and yaw angles).
  - Angular rate vs time: displays trends of pitch, yaw and roll rates.
  - Drag Coefficient vs time.
  - Altitude vs time (to the landing).
  - Axial forces vs time displays trends (to the reach of apogee) of: Drag, Thrust and sum of all axial forces as a function of time.
  - Module of acceleration vs time (to the landing).
  - Velocities vs time: displays trends of the velocity vector components along the three rocket axes (velocity x,y and z) as a function of time. Also plotted is a the module of velocity vector with respect to its z component.
  - Mach number vs time (to the reach of apogee).
  - Temperature profile vs time: displays trends of surrounding environment and rocket temperatures as a function of time.
  - Stability Margin vs time (to the reach of apogee).
  - Trajectory: displays the position along the trajectory, taking the launch point as the graph origin. The simulator plots according to the horizon axes. A complete trajectory in the form of 3D graph (altitude-North-East) and a planar displacement (North-East) are thus obtained.

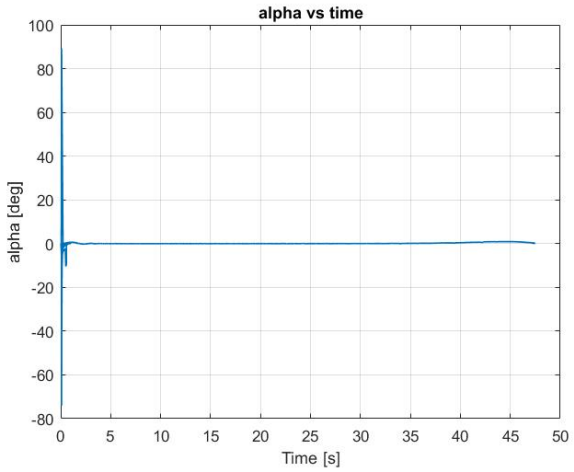
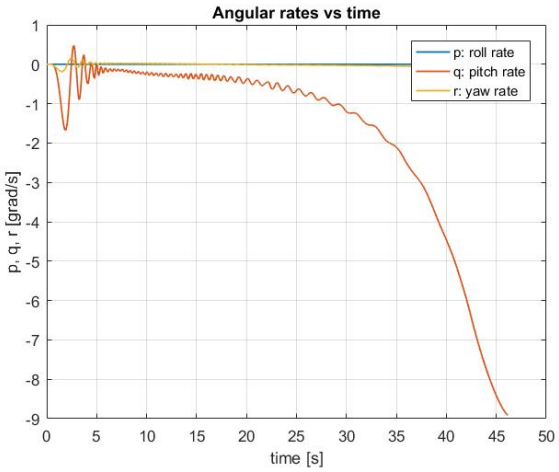
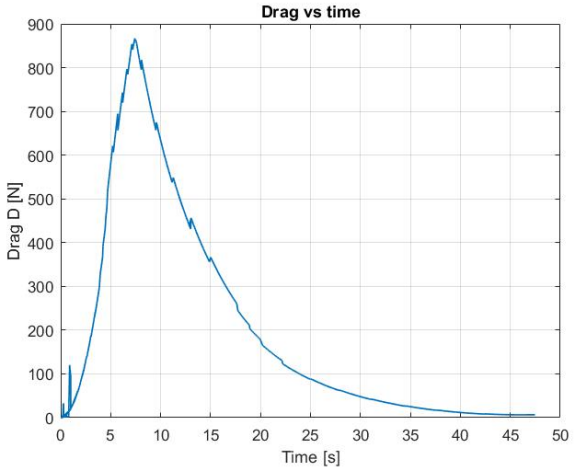
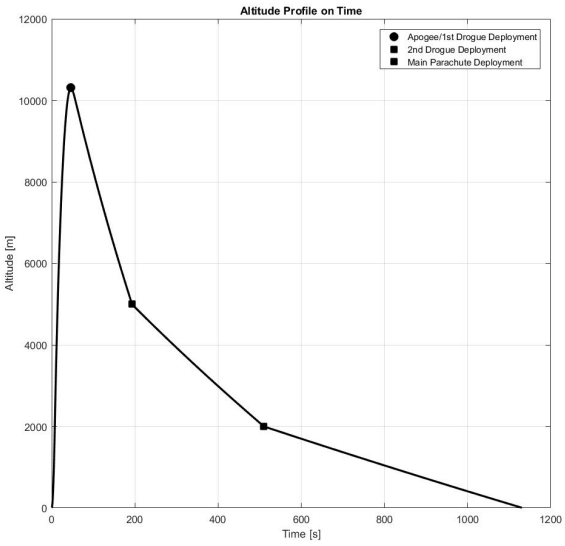
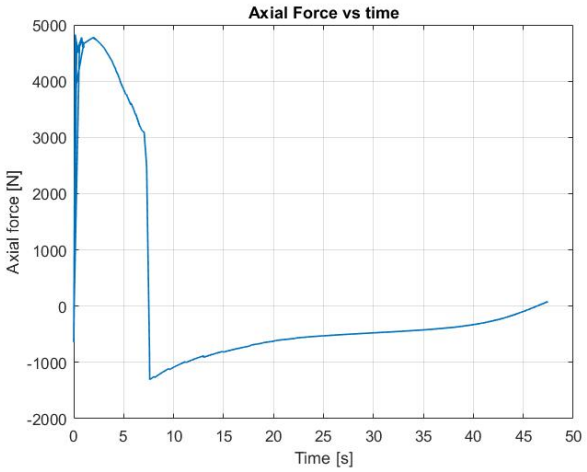
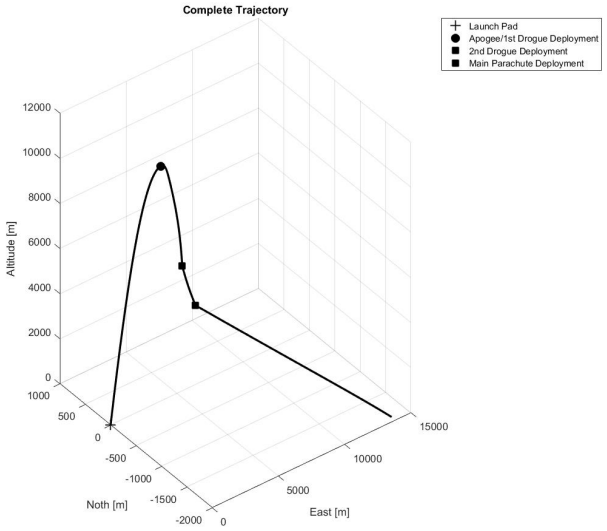
Angles vs time	Angular rate vs time
	
Drag Coefficient vs time	**Altitude vs time**
	

Table 6.2: Overview of the graphs (Part 1)

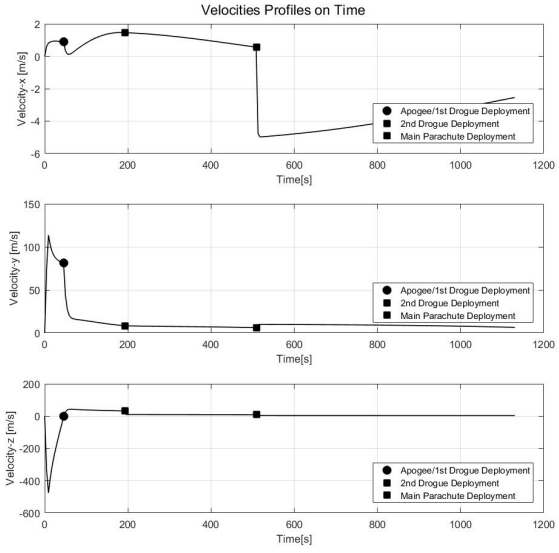
Axial forces vs time



Trajectory



Velocities vs time



Stability Margin vs time

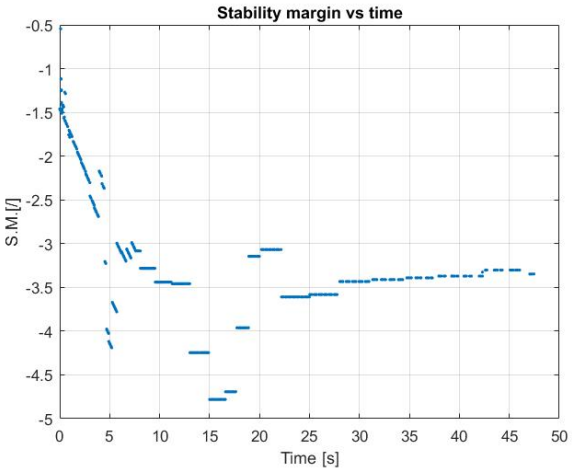


Table 6.3: Overview of the graphs (Part 2)

Chapter 7

Future Improvements

The simulator in this state describes with an acceptable accuracy the physics of the main phase (Ascent). However, some relevant updates are required in different areas.

Listed below are some possible improvements:

- Addition of a more precise wind model: This is especially relevant at high altitudes.
- Taking into account the Coriolis contribution.
- Introduction of more degrees of freedom (up to 12 if possible) in the descent phase (only 3 degrees as of now)
- Development of a suitable and practical GUI (Graphical User Interface).

Bibliography

[1] Document 1