



**HAL**  
open science

## A Matlab®/Simulink® non-linear simulator for orbital spacecraft rendezvous applications.

Paulo Ricardo Arantes Gilz

► **To cite this version:**

Paulo Ricardo Arantes Gilz. A Matlab®/Simulink® non-linear simulator for orbital spacecraft rendezvous applications.. 2016. hal-01413328

**HAL Id: hal-01413328**

**<https://hal.science/hal-01413328>**

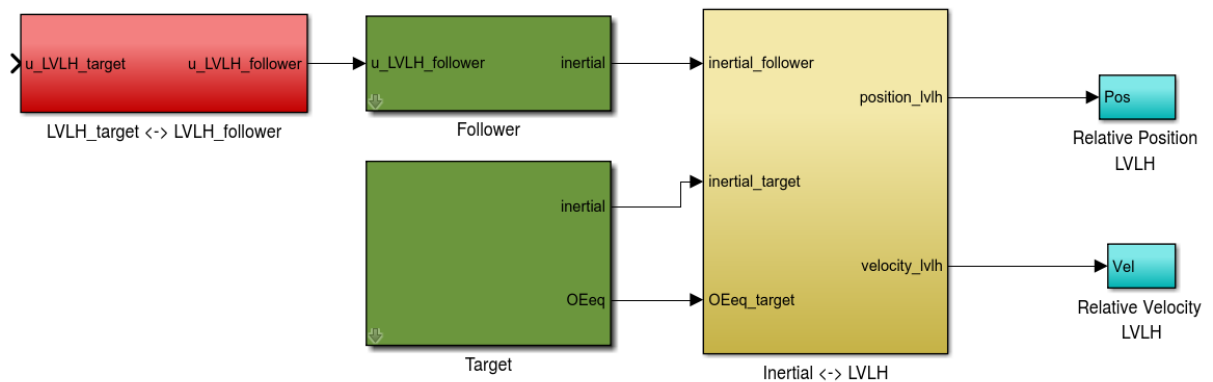
Submitted on 9 Dec 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A Matlab<sup>®</sup>/Simulink<sup>®</sup> non-linear simulator for orbital spacecraft rendezvous applications

Paulo R. Arantes Gilz  
LAAS-CNRS  
7, av. du Colonel Roche,  
31077 Toulouse, France,  
prarante@laas.fr



# Contents

<b>1</b>	<b>Brief introduction</b>	<b>3</b>
<b>2</b>	<b>Models and equations</b>	<b>3</b>
2.1	State representations . . . . .	3
2.2	Coordinate system changes . . . . .	4
2.2.1	Classical orbital elements state to equinoctial orbital elements state . . .	4
2.2.2	Equinoctial orbital elements state to inertial state . . . . .	4
2.2.3	RSW state to LVLH state / LVLH state to RSW state . . . . .	5
2.2.4	Computing the RSW basis in function of the inertial state . . . . .	5
2.2.5	Computing the relative LVLH state . . . . .	5
2.3	Dynamical equations . . . . .	6
2.4	$J_2$ disturbance . . . . .	7
2.5	Atmospheric drag disturbance . . . . .	7
2.6	Atmospheric density model . . . . .	7
2.7	Simulation parameters . . . . .	8
<b>3</b>	<b>Understanding the role of each block</b>	<b>9</b>
<b>4</b>	<b>How to use it?</b>	<b>12</b>

# 1 Brief introduction

This document details the models and equations (sec. 2) used in a Matlab<sup>®</sup>/Simulink<sup>®</sup> non-linear simulator for orbital spacecraft rendezvous applications and describe the role of each of its Simulink<sup>®</sup> blocks (sec. 3). It also explains how to perform the simulation of a given rendezvous scenario in practice (sec. 4).

The simulator here discussed is a modified version of the one developed by Mounir Kara-Zaitri during his PhD thesis at the LAAS-CNRS. The implemented modifications were performed in order to obtain a dedicated tool for simulating and developing control algorithms for the orbital spacecraft rendezvous in the case where the leader spacecraft is passive and the control applied on follower spacecraft is originally computed on the leader LVLH frame.

For a given orbital rendezvous scenario, the output of the simulator is the evolution of the relative position and velocity between the two spacecrafts, obtained by simulating the evolution of their trajectories via the integration of the Gauss equations for the orbital motion under the disturbances provoked by the Earth's flatness (the  $J_2$ -effect) and the atmospheric drag.

## 2 Models and equations

Hereafter we introduce some variables, state representations and coordinate system changes that will be further employed in the equations modelling the evolution of the spacecrafts trajectories and the disturbances provoked by the atmospheric drag and the Earth's flatness.

**Remark:** the following references are strongly recommended to help the reader to understand the nomenclature and notations used in the sequel:

- Mounir's PhD thesis: [http://thesesups.ups-tlse.fr/1026/1/Kara%2DZaitri\\_Mounir.pdf](http://thesesups.ups-tlse.fr/1026/1/Kara%2DZaitri_Mounir.pdf)
- Gerogia's PhD thesis: <http://thesesups.ups-tlse.fr/2105/1/2013TOU30170.pdf>

### 2.1 State representations

- The classical orbital elements state:

$$X_{OE_c} = [a, e, i, \Omega, \omega, \nu]^t \quad (2.1)$$

where:

- $a$  is the semi-major axis of the orbit;
- $e$  is the eccentricity the orbit;
- $i$  is the inclination;
- $\Omega$  is the longitude of ascending node;
- $\omega$  is the argument of perigee;

–  $\nu$  is the true anomaly;

- The equinoctial orbital elements state:

$$X_{OE_{eq}} = [p, f, g, h, k, L]^t \quad (2.2)$$

- The inertial state:

$$X_I = [x_I, y_I, z_I, \dot{x}_I, \dot{y}_I, \dot{z}_I]^t \quad (2.3)$$

representing the position and the velocity in the Earth's inertial frame.

- The LVLH state:

$$X_{LVLH} = [x_{LVLH}, y_{LVLH}, z_{LVLH}, \dot{x}_{LVLH}, \dot{y}_{LVLH}, \dot{z}_{LVLH}]^t \quad (2.4)$$

representing the position and the velocity in the LVLH frame.

- The RSW state:

$$X_{RSW} = [x_{RSW}, y_{RSW}, z_{RSW}, \dot{x}_{RSW}, \dot{y}_{RSW}, \dot{z}_{RSW}]^t \quad (2.5)$$

representing the position and the velocity in the RSW frame.

## 2.2 Coordinate system changes

### 2.2.1 Classical orbital elements state to equinoctial orbital elements state

$$\begin{aligned} p &= a(1 - e^2) \\ f &= e \cos(\Omega + \omega) \\ g &= e \sin(\Omega + \omega) \\ h &= \tan(i/2) \cos \Omega \\ k &= \tan(i/2) \sin \Omega \\ L &= \Omega + \omega + \nu \end{aligned} \quad (2.6)$$

### 2.2.2 Equinoctial orbital elements state to inertial state

$$\begin{aligned} x_I &= \frac{r}{s^2} (\cos L + \alpha^2 \cos L + 2hk \sin L) \\ y_I &= \frac{r}{s^2} (\sin L - \alpha^2 \sin L + 2hk \cos L) \\ z_I &= \frac{2r}{s^2} (h \sin L - k \cos L) \\ \dot{x}_I &= -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2f hk + \alpha^2 g) \\ \dot{y}_I &= -\frac{1}{s^2} \sqrt{\frac{\mu}{p}} (-\cos L + \alpha^2 \cos L - 2hk \sin L - f + 2ghk + \alpha^2 f) \\ \dot{z}_I &= \frac{2}{s^2} \sqrt{\frac{\mu}{p}} (h \cos L + k \sin L + fh + gk) \end{aligned} \quad (2.7)$$

where

$$\begin{aligned}
\mu & : \text{ Earth's gravitational constant} \\
s^2 & = 1 + h^2 + k^2 \\
w & = 1 + f \cos L + g \sin L \\
r & = p/w \\
\alpha^2 & = h^2 - k^2
\end{aligned} \tag{2.8}$$

### 2.2.3 RSW state to LVLH state / LVLH state to RSW state

$$\begin{aligned}
x_{\text{LVLH}} & = y_{\text{RSW}} & \dot{x}_{\text{LVLH}} & = \dot{y}_{\text{RSW}} \\
y_{\text{LVLH}} & = z_{\text{RSW}} & \text{and} & & \dot{y}_{\text{LVLH}} & = \dot{z}_{\text{RSW}} \\
z_{\text{LVLH}} & = -x_{\text{RSW}} & & & \dot{z}_{\text{LVLH}} & = -\dot{x}_{\text{RSW}}
\end{aligned} \tag{2.9}$$

### 2.2.4 Computing the RSW basis in function of the inertial state

The following computations must be performed in order to express the RSW orthonormal basis  $(\vec{R}, \vec{S}, \vec{W})$  in the Earth's inertial frame in function of the inertial state  $X_I = [x_1, y_1, z_1, \dot{x}_1, \dot{y}_1, \dot{z}_1]^t$ :

$$\begin{aligned}
\vec{R} & = \frac{1}{\sqrt{x_1^2 + y_1^2 + z_1^2}}(x_1, y_1, z_1) & \vec{T} & = \frac{1}{\sqrt{\dot{x}_1^2 + \dot{y}_1^2 + \dot{z}_1^2}}(\dot{x}_1, \dot{y}_1, \dot{z}_1) \\
\vec{W} & = \frac{1}{\|\vec{U}\|}\vec{U} & \text{where} & & \vec{U} & = -\vec{T} \times \vec{R} \\
\vec{S} & = \frac{1}{\|\vec{V}\|}\vec{V} & & & \vec{V} & = \vec{U} \times \vec{R}
\end{aligned} \tag{2.10}$$

### 2.2.5 Computing the relative LVLH state

Hereafter we show how to compute the relative state between two spacecrafts in the LVLH frame of the leader spacecraft  $(\bar{X}_{\text{LVLH}}^l)$  in function of the inertial states of each spacecraft  $(X_I^f$  and  $X_I^l)$  and the equinoctial orbital elements of the leader spacecraft  $(X_{O_{Eq}}^l)$ :

First we define the inertial relative state  $\bar{X}_I = [\bar{x}_1, \bar{y}_1, \bar{z}_1, \dot{\bar{x}}_1, \dot{\bar{y}}_1, \dot{\bar{z}}_1]^t$  as:

$$\bar{X}_I = X_I^f - X_I^l \tag{2.11}$$

and the inertial relative position and velocity vectors  $\vec{p}_1$  and  $\vec{v}_1$ :

$$\vec{p}_1 = (\bar{x}_1, \bar{y}_1, \bar{z}_1) \quad \text{and} \quad \vec{v}_1 = (\dot{\bar{x}}_1, \dot{\bar{y}}_1, \dot{\bar{z}}_1) \tag{2.12}$$

After that, the RSW orthonormal base of the leader spacecraft  $(\vec{R}^l, \vec{S}^l, \vec{W}^l)$  is computed in function of the inertial state of the leader satellite  $(X_I^l)$  via (2.10). The relative state in the

LVLH frame of the leader spacecraft is then computed as follows:

$$\bar{X}_{\text{LVLH}}^l = \begin{bmatrix} \bar{x}_{\text{LVLH}}^l \\ \bar{y}_{\text{LVLH}}^l \\ \bar{z}_{\text{LVLH}}^l \\ \dot{\bar{x}}_{\text{LVLH}}^l \\ \dot{\bar{y}}_{\text{LVLH}}^l \\ \dot{\bar{z}}_{\text{LVLH}}^l \end{bmatrix} = \begin{bmatrix} \langle \vec{S}^l, \vec{p}_1 \rangle \\ \langle \vec{W}^l, \vec{p}_1 \rangle \\ -\langle \vec{R}^l, \vec{p}_1 \rangle \\ \langle \vec{S}^l, \vec{v}_1 \rangle - \dot{\nu}^l \langle \vec{R}^l, \vec{p}_1 \rangle \\ \langle \vec{W}^l, \vec{v}_1 \rangle + \\ -\langle \vec{R}^l, \vec{v}_1 \rangle - \dot{\nu}^l \langle \vec{S}^l, \vec{p}_1 \rangle \end{bmatrix} \quad (2.13)$$

where  $\langle \cdot, \cdot \rangle$  is the dot product in  $\mathbb{R}^3$  and

$$\dot{\nu}^l = \sqrt{\mu p^l} \left( \frac{1 + f^l \cos L^l + g^l \sin L^l}{p^l} \right)^2 \quad (2.14)$$

**Remark:** the terms  $-\dot{\nu}^l \langle \vec{R}^l, \vec{p}_1 \rangle$  and  $-\dot{\nu}^l \langle \vec{S}^l, \vec{p}_1 \rangle$  are due to the relative velocity between frames.

### 2.3 Dynamical equations

The dynamics of the spacecrafts movement is modelled by a non-classical form of the Gauss planetary equations. We choose this different representation because the classical one presents singularities for low eccentricity ( $e$ ) and inclination ( $i$ ) values. However, by rewriting these equations in function of the equinoctial orbital elements, these singularities are eliminated.

Given that, the dynamics of the equinoctial orbital elements is given by:

$$\frac{dX_{OE_c}}{dt} = A\vec{u} + B \quad (2.15)$$

where  $\vec{u} = \vec{u}_c + \vec{u}_{J_2} + \vec{u}_d$  is the combined acceleration induced by a possible control law and the disturbances ( $J_2$ -effect and atmospheric drag) expressed in the RSW coordinate system and:

$$A = \sqrt{\frac{p}{\mu}} \begin{bmatrix} 0 & \frac{2p}{w} & 0 \\ \sin L & \frac{(w+1)\cos L + f}{w} & -\frac{g(h \sin L - k \cos L)}{w} \\ -\cos L & \frac{(w+1)\sin L + g}{w} & \frac{f(h \sin L - k \cos L)}{w} \\ 0 & 0 & \frac{s^2 \cos L}{2w} \\ 0 & 0 & \frac{s^2 \sin L}{2w} \\ 0 & 0 & \frac{h \sin L - k \cos L}{w} \end{bmatrix} \quad (2.16)$$

$$B = \left[ 0, 0, 0, 0, 0, \frac{w^2}{p^2} \sqrt{\mu p} \right]^t \quad (2.17)$$

## 2.4 $J_2$ disturbance

The effect of the Earth's flatness is modelled as a external disturbance that provokes an acceleration on the satellite. This acceleration is given in the RSW coordinates system by:

$$\vec{u}_{J_2} = -\frac{3\mu J_2 R_e^2}{2r^4} \begin{bmatrix} 1 - \frac{12(h \sin L - k \cos L)^2}{(1 + h^2 + k^2)^2} \\ \frac{8(h \sin L - k \cos L)(h \cos L + k \sin L)}{(1 + h^2 + k^2)^2} \\ \frac{4(h \sin L - k \cos L)(1 - h^2 - k^2)}{(1 + h^2 + k^2)^2} \end{bmatrix} \quad (2.18)$$

where  $J_2$  is the second degree term in Earth's gravity potential and  $R_e$  is the Earth's radius.

## 2.5 Atmospheric drag disturbance

The effect of the atmospheric drag is also modelled as a external disturbance that provokes an acceleration on the satellite. This acceleration is given in the RSW coordinates system by:

$$\vec{u}_d = -\frac{\rho S C_d \mu}{2m} \frac{\mu}{p} \sqrt{1 + 2(g \sin L + f \cos L) + f^2 + g^2} \begin{bmatrix} f \sin L - g \cos L \\ 1 + f \cos L + g \sin L \\ 0 \end{bmatrix} \quad (2.19)$$

where  $\rho$  is the atmospheric density and  $m$ ,  $S$  and  $C_d$  are respectively the mass, the cross sectional area and the drag coefficient of the spacecraft. In this equation,  $\rho$  is not a constant, and the model used for it is discussed in the next subsection.

## 2.6 Atmospheric density model

The atmospheric density model initially used by Mounir in the first version of the simulator was given in function of the distance between the satellite and the center of the Earth by the following equation:

$$\rho(x_1, y_1, z_1) = \rho_{LH} \exp \left( \frac{R_e + 400000 - \sqrt{x_1^2 + y_1^2 + z_1^2}}{46830} \right) \quad (2.20)$$

where  $\rho_{LH}$  is a constant that depends of the solar activity ( $2.2644 \times 10^{-12}$  for low and  $3.5475 \times 10^{-11}$  for high solar activity). The author himself admitted in his work that this model was an



approximation and should be modified in order to obtain a better verisimilitude.

The main problem noticed by using this model during simulations is that as the distances between the satellite and the Earth decreases,  $\rho$  grows exponentially and the intensity of the drag disturbance increases too much.

In order to avoid this behaviour, the new simulator uses the following equation instead:

$$\rho(x_1, y_1, z_1) = \min \left\{ \rho_{LH} \exp \left( \frac{R_e + 400000 - \sqrt{x_1^2 + y_1^2 + z_1^2}}{46830} \right), \rho_{LH} \right\} \quad (2.21)$$

## 2.7 Simulation parameters

By default, the integration of the differential equation (2.15) is performed by a simple first-order Euler scheme with fixed-step in this case, in order to control the simulation, the user only need to inform the initial and final time and the integration step in order to perform:

$$t^* = t_0$$

While ( $t^* < t_f$ ) do

$$X_{OE_{eq}}(t^* + \text{step}) = X_{OE_{eq}}(t^*) + (\text{step}) \left. \frac{dX_{OE_{eq}}}{dt} \right|_{t^*}$$

$$t^* = t^* + \text{step}$$

Loop

**Remark:** evidently another integration method may be chosen in Simulink<sup>®</sup>, with it being up to the user to set the new required parameters.

### 3 Understanding the role of each block

- Blocks follower and target:

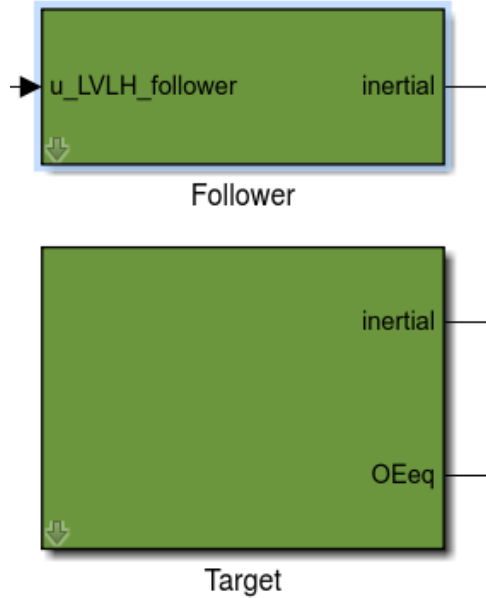


Figure 1: These blocks compute the inertial state of the spacecrafts by integrating equation (2.15) from  $t_0$  to  $t_f$  and making the conversion between orbital elements and inertial state.

These green blocks contain the following blocks:

- Block  $J_2$  effect:

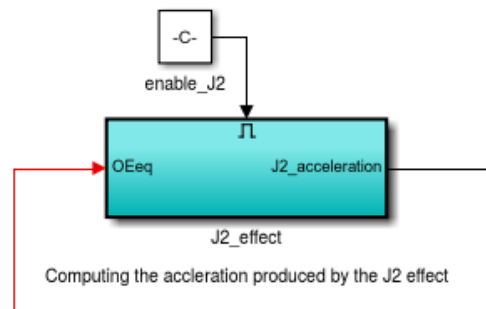


Figure 2: Computes the  $J_2$  acceleration in the RSW frame in function of the orbital elements using equation (2.18).

– Block drag effect:

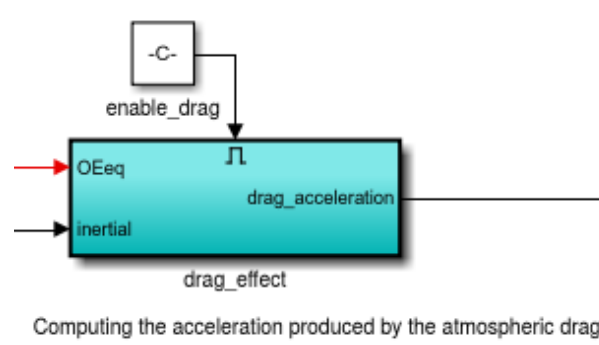


Figure 3: Computes the drag acceleration in the RSW frame in function of the orbital elements and the inertial state using equation (2.19).

– Block  $\frac{dX_{OE_{ed}}}{dt}$ :

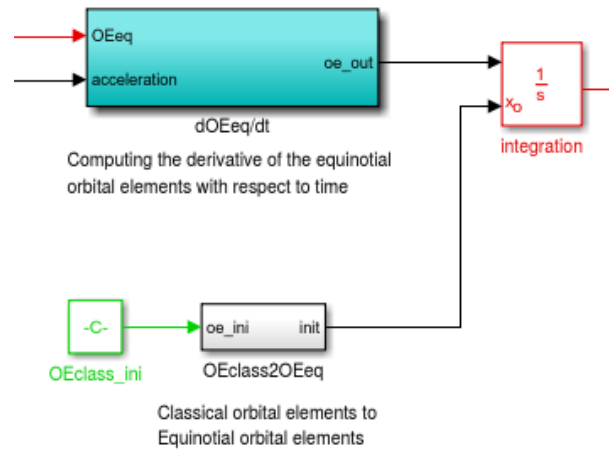


Figure 4: Computes the derivative of the orbital elements via equation (2.15). To initialize the integrator, performs the variable change (2.6).

– Block equinotial orbital elements to inertial:

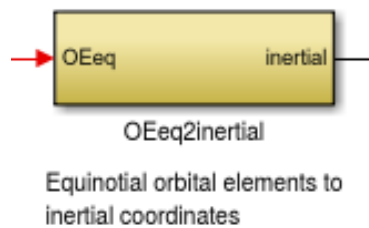


Figure 5: Performs the variable change (2.7).

- Block inertial to LVLH:

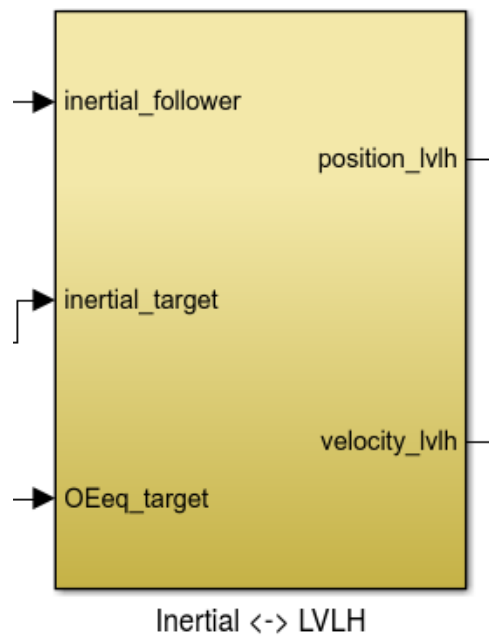


Figure 6: Computes the relative state between the satellites in the LVLH coordinates system of the leader spacecraft via the transformations presented in subsection 2.2.5.

- Blocks position and velocity:

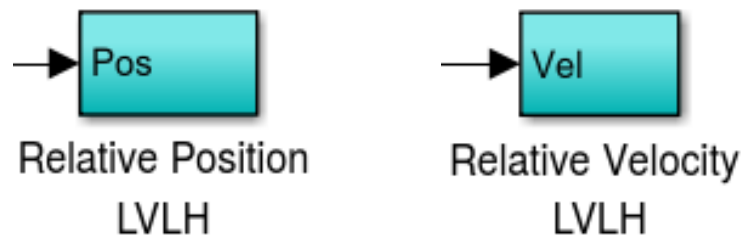


Figure 7: Writes the relative position and velocity computed in the LVLH frame to an array in the workspace of Matalab.




- Block LVLH leader to LVLH follower:



Figure 8: Change the coordinate frame of the input from leader LVLH to follower LVLH.

## 4 How to use it?

The file `Simulator.zip` contains three files that must be extracted to a same directory:

	<code>dx_oe.m</code>	3,8 ko	Texte
	<code>run_simulation.m</code>	2,0 ko	Texte
	<code>simulator.slx</code>	33,3 ko	Archive

- `simulator.slx`, the simulator itself;
- `run_simulation`, a script to be run in order to perform the simulation;
- `dx_oe.m`, an auxiliary function.

In order to run the simulator, the script `run_simulation.m` must be run in Matlab<sup>®</sup>. This file declares and assigns the variables that describe the rendezvous scenario and calls the simulator. Hereafter we explain step-by-step the meaning of each section of the script (**all values in SI base units**):

- The script starts by assigning the constants  $R_e$ ,  $\mu$ ,  $J_2$ :

```
% Constants
Re = 6378136.55;           % Earth radius
mu = 3.986004418e14;      % Earth gravitational constant
J2 = 1.0826268361960958e-3; % J2-effect constant
```

- After that, the classical orbital elements of the leader spacecraft are assigned and arranged in a vector `oe_l`:

```
% Leader classical orbital elements
a = 7011e3;                % Semi-major axis [m]
exc = 0.4;                 % Eccentricity
i = 0;                    % Inclination
Omega = 0;                 % Longitude of ascending node
omega = 0;                 % Argument of periapsis
nu0 = 0;                   % Initial true anomaly
oe_l = [a; exc; i; Omega; omega; nu0]; % Leader's orbital parameters (Leader)
```

- The classical orbital elements of the follower spacecraft are computed from the relative state between spacecrafts in the leader LVLH frame the auxiliary function `dx_oe` (in the case these values are known *a priori*, they can be directly assigned to the vector `oe_f`):

```
% Relative state in the leader LVLH reference frame
X_0 = [20; 10; 0; 0; 0; 0];

% Computing the follower classical orbital parameters
oe_f = oe_l + dx_oe(X_0,oe_l,mu); % Chaser's orbital parameters (Follower)
```

- The next section of the script is responsible for turning on/off the effect of the disturbances and choosing between high or low solar activity:

```
% Effects to be taken into account
J2_effect = 1;           % J2 effect (0 - off / 1 - on)
drag_effect = 1;       % atmospheric drag effect (0 - off / 1 - on)
solar_actvty = 1;     % Solar activity (0 - low / 1 - high)
```

- This section assigns the values of the mass, the drag coefficient and the cross sectional surface of both spacecrafts:

```
% Spacecrafts properties
m_f = 20000 ;           % Follower mass
m_l = 462949 ;         % Target mass
d_s_f = 50 ;           % Follower drag surface
d_s_l = 1703 ;         % Target drag surface
d_c_f = 2.274 ;        % Follower drag coefficient
d_c_l = 3 ;            % Target drag coefficient
```

- Setting the initial and final simulation times and the integration step:

```
% Simulation parameters
t0 = 0;                % Initial time
tf = 5000;             % Final time
step = 1;              % Integration step
```

- Opening and launching the simulator:

```
open('simulator.slx');
sim('simulator.slx');
```

- After the end of the simulation, two vectors `R1v1h` and `V1v1h` are produced, containing respectively the relative positions and velocities in the leader LVLH frame. The following script plots the obtained relative trajectory (presented in Fig. 9):

```
% Plotting the obtained trajectory
plot3(R1v1h(:,1),R1v1h(:,2),R1v1h(:,3),'-b');
grid on;
title('Relative trajectory');
xlabel('x'); ylabel('y'); zlabel('z');
```

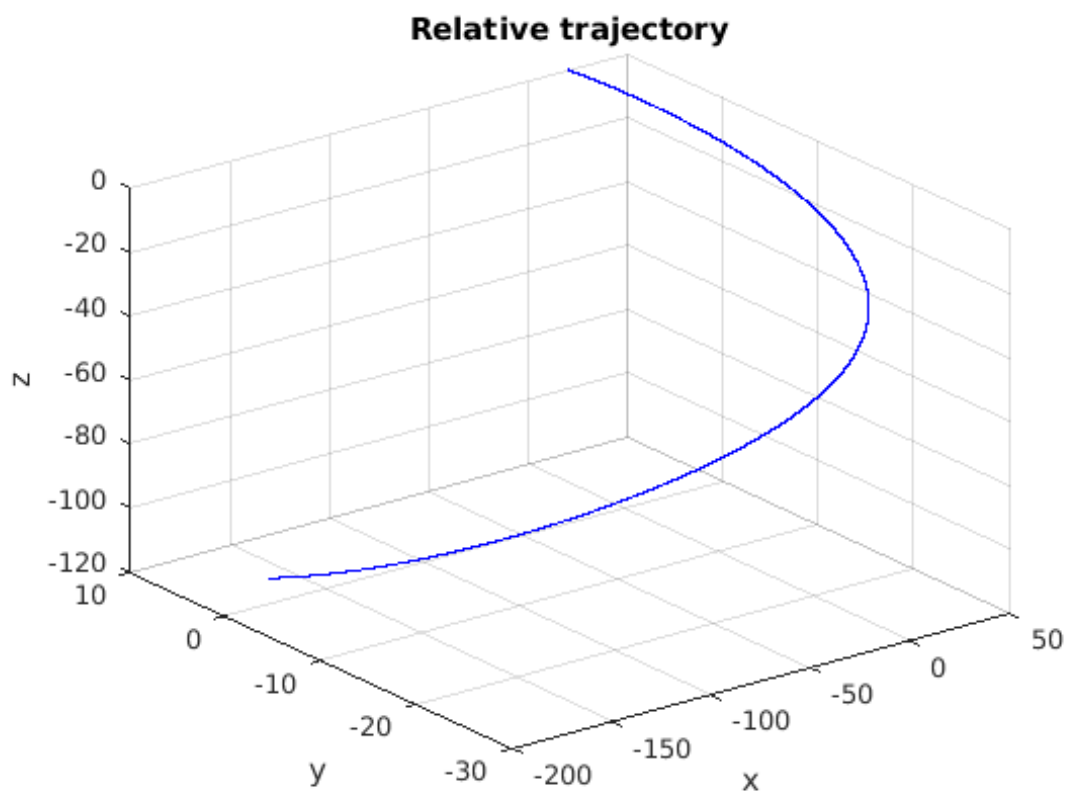


Figure 9: Relative trajectory obtained after running the simulation