

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Project “Mars Lander Control” – Draft Paper

Course: Modeling and Control of Aerospace Vehicles (MCAV)

M.Sc. in Aerospace Engineering

Professor: Bruno Guerreiro

November 2023

Group 5

José Duarte Dias Corvo, student number 67118

Vasco Rafael da Ponte Luís Nunes, student number 67304

Abstract

“This project aims at designing and implementing a distributed formation control system, that allows several drones to follow a leader drone, maintaining the distance between them at a constant desired set-point. The project will guide you through the modeling of the system, the design of a constrained predictive controller, and also to the design, implementation and experimental validation of a distributed constrained model-based predictive control policy for a drone formation.” – Bruno Guerreiro, 12th October 2023

Index

1. Introduction	5
2. Draft Paper Subjects and Goals	6
3. Development of the goals imposed.....	8
3.1. Goal number 1	8
3.2. Goal number 2	9
3.3. Goal number 3	10
3.4. Goal number 4	10
3.5. Goal number 5	12
3.6. Goal number 6	15
4. Other Comments	17
5. Webgraphy	18
6. Attachments.....	19

List of Figures

Figure 1 – Perseverance Rover on Mars by NASA Mars Exploration	5
Figure 2 – Perseverance deployment phases by NASA Mars Exploration	6
Figure 3 – Coded variables of the nonlinear model of the lander	8
Figure 4 – Slide number 64 of “Modeling of Aerospace Vehicles”	8
Figure 5 – Coding of the net force of one of the retrorockets	9
Figure 6 – Coding of the moment vector of the 4 net forces of the retrorockets	9
Figure 7 – Slide number 66 of “Modeling of Aerospace Vehicles”	9
Figure 8 – Mars gravity by NSSDCA	10
Figure 9 – Mars air drag force by NSSDCA	10
Figure 10 – Mars atmosphere parameters in code	10
Figure 11 – The simulation parameters for the lander’s approach to Mars surface...	11
Figure 12 – The code for the approach to Mars surface	11
Figure 13 – The simulation obtained for the approach to Mars in MATLAB R2021a.	12
Figure 14 – Definition of all matrices needed for the linearisation of the model.....	14
Figure 15 – Coding line to develop the system.....	14
Figure 16 – Slide number 21 of “State-space Analysis and Control”	15
Figure 17 – The code developed to verify Jordan normal form, controlability, observability and stability	15
Figure 18 – Verification of the Jordan normal form of the linearized model.....	15
Figure 19 – Verification of the stability of the linearized model.....	16
Figure 20 – Verification of the controlability of the linearized model.....	16
Figure 21 – Verification of the observability of the linearized model.....	16
Figure 22 – The link to run the nonlinear model from the linearized model	17
Figure 23 – Exercise 2.5 by Bruno Guerreiro	17
Figure 24 – Exercise 3.2 by Bruno Guerreiro	17

1. Introduction

In aerospace missions that require landings (on the Moon or in other destinations, such as the planet Mars), the soft landing of a spacecraft is still a big challenge in terms of control and systems engineering. One of the cases of the application of this challenge was on the landing of the Perseverance Rover on planet Mars on February 2021 by NASA.

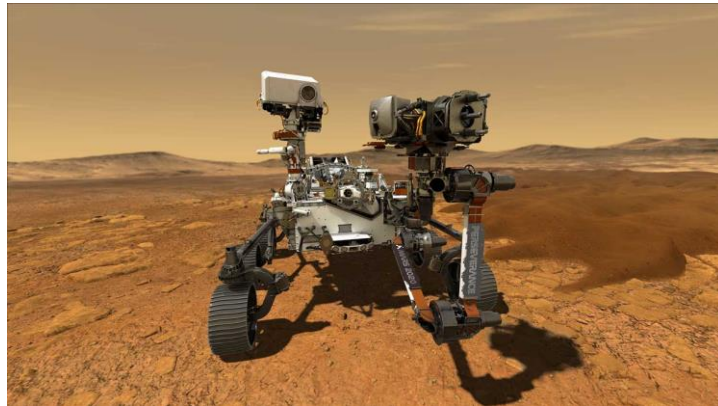


Figure 1 – Perseverance Rover on Mars by NASA Mars Exploration

This project aims at exploring the dynamic model of a lander like the one that deployed the Perseverance Rover and explore possible control strategies for achieving a soft and precise deployment, as well as the flyaway maneuvers to land in another safe site. In this draft paper, we will deal with the nonlinear modeling of the vehicle and the linearization for the relevant modes of operation.

The coding resolution of this project is in the format of a GitHub repository after working on the software *MATLAB R2021a*.

2. Draft Paper Subjects and Goals

In this part of the project, we will address the full nonlinear of the lander in 3 dimensions, considering the “powered descent” landing phase, as showed in the Perseverance deployment phases.

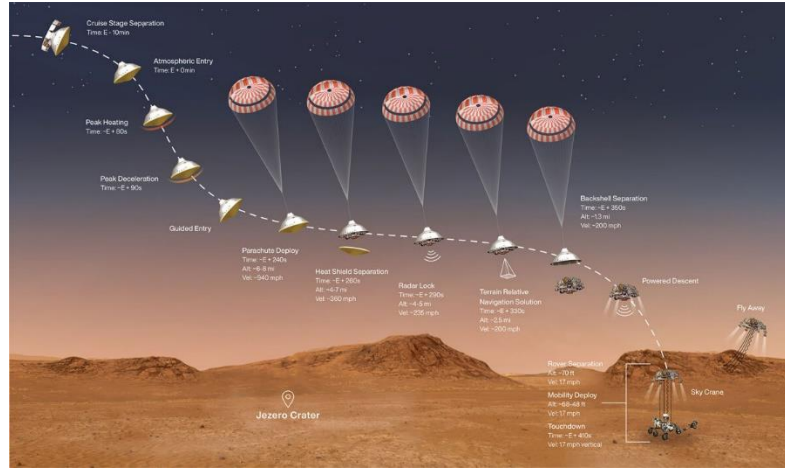


Figure 2 – Perseverance deployment phases by NASA Mars Exploration

In this landing phase we will use a NED (North-East-Down) local tangent plane centered at the Mars Jezero crater (Latitude 18.38°N, Longitude 77.58°E) as the inertial frame and we will assume the lander as a rigid-body in 3D space with the relevant external forces acting on the center of mass of the whole set of the lander and four retrorockets that power the lander (for simplicity we assume the body frame is located at the center of mass).

The thrust is given to the lander by four throttleable retrorockets capable of both deceleration and hovering. We will assume for now that the mass of the lander is constant.

For this project, it was considered the following state-space variables for the lander:

- The lander 3D position (origin of the body frame cm_t and the center of mass $p_{z_{cm}}$) relative to the inertial frame jlr is $\mathbf{p} = {}^{jlr}\mathbf{p}_{cm_t} \in \mathbb{R}^3$ is defined by the position of each retrorocket;
- The velocity of the lander described in the body frame is $\mathbf{v} \in \mathbb{R}^3$;
- The altitude of the body frame relative to the Euler angles is defined by the formula $\mathbf{R} = \text{Euler2R}(\mathbf{lbd})$;
- The relative angular velocity expressed in the body frame is $\mathbf{omg} \in \mathbb{R}^3$.

Regarding the forces acting on the lander, we consider the Mars gravity acceleration ($\mathbf{g}_{marte} = 3.73 [\text{m/s}^2]$), the air drag ($\mathbf{f}_a = 4\pi * r_{o_{marte}} * v^2$, with $r_{o_{marte}} = 0.02 [\text{kg/m}^3]$) and the four forces of propulsion of the retrorockets ($\mathbf{f}_p = [Fp1 \ Fp2 \ Fp3 \ Fp4]$), with these retrorockets forces being the only variables that we will be able to influence in order to control the trajectory of the lander. The sum of these forces ($\mathbf{T}_m = Fp1 + Fp2 + Fp3 + Fp4$) is the thrust force of the lander on the exact moment.

Regarding the retrorockets, we assume that they have a fixed outward inclination relative to the z axis of 20° or $\frac{\pi}{9} \text{ rad}$ (in the yz -plane), avoiding damage to the lander below. Each retrorocket can vary its thrust from null to its maximum.

The goals for this draft paper are the following:

- **Goal number 1:** Obtain the rigid-body nonlinear model of the lander, considering the state variables above.
- **Goal number 2:** obtain the net force and moment vector expressions at the vehicle center of mass (origin of the body frame) as a function of the individual forces generated by each of the four retrorockets.
- **Goal number 3:** Obtain the Mars gravity and air drag force vectors described in the body frame.
- **Goal number 4:** Obtain the final nonlinear model of the lander and simulate the model in *MATLAB*, considering initial resting state vector and small constant thrust, equal to all retrorockets.
- **Goal number 5:** Obtain the linearized models around the above equilibrium points/trajectories, noting that there will be a **lateral+longitudinal** model pair for each retrorocket, considering the kinematics of the Euler angles **zyx**.
- **Goal number 6:** Analyze each linearizes model regarding their controllability, observability and stability.

3. Development of the goals imposed

In order to obtain the goals proposed by coding in *MATLAB R2021a*, the group based its research on the examples provided by Professor Bruno Guerreiro on the 2023 version of FCT/UNL Moodle.

3.1. Goal number 1

Obtain the rigid-body nonlinear model of the lander, considering the state variables above.

```
p_dot=R*v;
lbd_dot = Euler2Q(lbd)*omg;
v_dot= -skew(omg)*v + g_marte*R'*zI - R*D*R'*v - 1/mt*(T_m+fa);
om_dot = -inv(jlr)*skew(omg)*jlr*omg + inv(jlr)*np;
```

Figure 3 – Coded variables of the nonlinear model of the lander

On this code:

- “*p_dot*” is the linear motion of the lander;
- “*lbd_dot*” is the inversion of the Euler angles attitude into the kinematics of the lander;
- “*v_dot*” is the instant velocity of the lander;
- “*om_dot*” is the instant angular velocity of the lander.

This code was based on the lecture slides for this course by Bruno Guerreiro, to be more precise in the PowerPoint “Modeling of Aerospace Vehicles”.


Multirotor Model

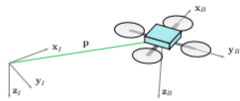
- Rigid-body kinematics and dynamics.
- External forces:
 - ▶ Gravity force in B , $\mathbf{f}_g = m\mathbf{g}\mathbf{R}^T\mathbf{z}_I$.
 - ▶ Body and rotor aerodynamic drag force:

$$\mathbf{f}_a = -\mathbf{R}\mathbf{D}\mathbf{R}^T\mathbf{v}$$
 - \mathbf{D} is a diagonal constant drag coefficient matrix.
 - Air velocity in I neglected.
 - Aerodynamic moment neglected.
 - ▶ Two or more rotor pairs, generating \mathbf{f}_p and \mathbf{n}_p .
- Multirotor dynamics:

$$\dot{\mathbf{v}} = -\mathbf{S}(\boldsymbol{\omega})\mathbf{v} + g\mathbf{R}^T\mathbf{z}_I - \mathbf{R}\mathbf{D}\mathbf{R}^T\mathbf{v} + \frac{1}{m}\mathbf{f}_p$$

$$\dot{\boldsymbol{\omega}} = -\mathbf{J}^{-1}\mathbf{S}(\boldsymbol{\omega})\mathbf{J}\boldsymbol{\omega} + \mathbf{J}^{-1}\mathbf{n}_p$$





NOVA B. Guerreiro, MCAV 23/24
State-space Models
Rigid-body dynamics
Vehicle Models

Figure 4 – Slide number 64 of “Modeling of Aerospace Vehicles”

3.2. Goal number 2

Obtain the net force and moment vector expressions at the vehicle center of mass (origin of the body frame) as a function of the individual forces generated by each of the 4 retrorockets.

```
p1=[0 pz_cm -y1/2
      -pz_cm 0 x1/2
      y1/2 -x1/2 0];
Fp1=T1*[sin(20*pi/180); 0; cos(20*pi/180)];
%np1;
np1=p1*Fp1;
```

Figure 5 – Coding of the net force of one of the retrorockets

```
fp=[Fp1 Fp2 Fp3 Fp4];
T_m=Fp1+Fp2+Fp3+Fp4;
np=[np1; np2; np3; np4];
```

Figure 6 – Coding of the moment vector of the 4 net forces of the retrorockets

On this code:

- “ p_i ” is the position of each retrorocket;
- “ Fp_i ” is the propulsion force of each retrorocket;
- “ np_i ” is the net force of each retrorocket;
- “ np ” is the moment vector of the lander.

This code was also based on the PowerPoint “Modeling of Aerospace Vehicles” by Bruno Guerreiro.

Quadrotor propulsion: total force and moment

■ Rotor constant positions in body frame (arm length l):

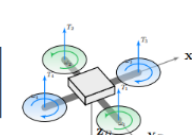
$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ l \\ 0 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ -l \\ 0 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} l \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} -l \\ 0 \\ 0 \end{bmatrix}$$

■ Total propulsion force:

$$\mathbf{f}_p = \sum_i \mathbf{f}_i = \begin{bmatrix} 0 \\ 0 \\ -\sum_i T_i \end{bmatrix} = c_T \begin{bmatrix} 0 \\ 0 \\ -\sum_i \omega_i^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -T_1 - T_2 - T_3 - T_4 \end{bmatrix}$$

■ Total propulsion moment:

$$\mathbf{n}_p = \sum_i (\mathbf{n}_i + \mathbf{p}_i \times \mathbf{f}_i) = \begin{bmatrix} l(T_2 - T_1) \\ l(T_3 - T_4) \\ Q_1 + Q_2 + Q_3 + Q_4 \end{bmatrix} = \begin{bmatrix} l(T_2 - T_1) \\ l(T_3 - T_4) \\ \frac{c_Q}{c_T}(-T_1 - T_2 + T_3 + T_4) \end{bmatrix}$$



Nova B. Guerreiro, MCAV 23/24
State-space Models
Rigid-body dynamics
Vehicle Models

Figure 7 – Slide number 66 of “Modeling of Aerospace Vehicles”

3.3. Goal number 3

Obtain the Mars gravity and air drag force vectors described at the body frame.

To obtain the Mars gravity and air drag force vectors described at the body frame, we checked the Mars fact sheet by NSSDCA to obtain the following variables:

- Mars gravity is equal to 3.73 m/s^2 ;
- Mars air drag force is $\approx 0.020 \text{ kg/m}^3$.

Surface gravity (mean) (m/s ²)	3.73
--	------

Figure 8 – Mars gravity by NSSDCA

Surface density: $\sim 0.020 \text{ kg/m}^3$
--

Figure 9 – Mars air drag force by NSSDCA

```
g_marte=3.73;%m/s
ro_marte=0.020;%kg/m^3
```

Figure 10 – Mars atmosphere parameters in code

After this research, we coded the following parameters:

- “*g_marte*” is the Mars gravity variable.
- “*ro_marte*” is the Mars air drag force variable.

3.4. Goal number 4

Obtain the final nonlinear model of the Lander and simulate the model in MATLAB, considering initial resting state vector and small constant thrust, equal to all retrorockets.

With the goal of obtaining the final nonlinear model of the set (**lander+rover**) as it reaches Mars surface, there was a need to define the parameters for this occurrence. The parameters defined were the following:

- “*zI*” is the gravity constant that defines its direction;
- “*nx*” and “*ny*” are the parameters of the simulation in order of *x* and *y*, respectively;
- “*x0*” is the matrix of zeros in the axis *x*;
- “*Dt*” is the velocity of iterations;
- “*t*” is length of time from entry of the atmosphere until the rover’s release;
- “*T*” is equilibrium thrust, with “*mt*” being the total mass of the set;
- “*u_NL*” is the nonlinear vector of the set;
- “*Nsim*” is the vector that defines the simulation duration (depends on “*t*”);
- “*x*” and “*y*” are the axis used to define these parameters.

```
%Modelo não linear

zI = [0;0;1];

% Parametros de simulação
nx = 12;
ny = 4;
x0 = zeros(nx,1);
Dt = 0.1;
t = 0:Dt:60;
T=mt*g_marte;
u_NL=[T;0;0;0]*ones(size(t));

C = [ eye(3) , zeros(3) , zeros(3) , zeros(3)
      zeros(1,3), zeros(1,3) , zI' , zeros(1,3) ];

% simulação do sistema não-linear
Nsim = length(t);
x = zeros(nx,Nsim);
y = zeros(ny,Nsim);
x(:,1) = x0;
```

Figure 11 – The simulation parameters for the lander's approach to Mars surface

For the parameters that were defined, the approach to Mars went with a separate code.

```
for k = 1:Nsim
    % prepare variables:
    p=x(1:3,k)
    v= x(4:6,k);
    lbd = x(7:9,k);
    omg = x(10:12,k);
    R = Euler2R(lbd);
    T = u_NL(1,k);
    np = u_NL(2:4,k);
```

Figure 12 – The code for the approach to Mars surface

With the code for the approach provided, it was possible to simulate the approach.

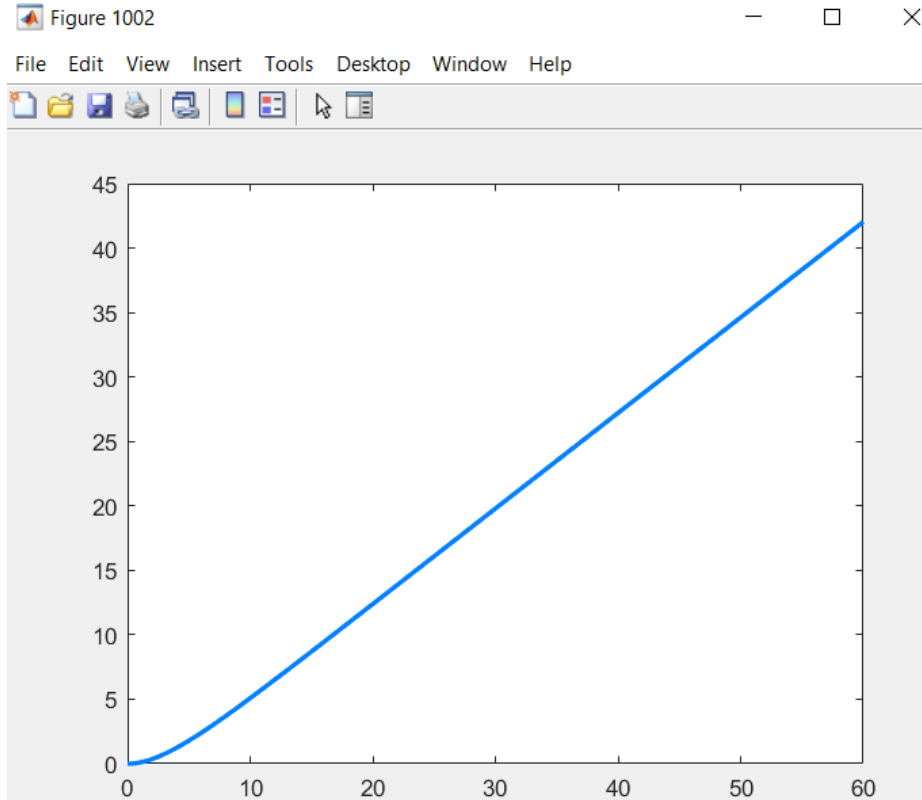


Figure 13 – The simulation obtained for the approach to Mars in MATLAB R2021a

3.5. Goal number 5

Obtain the linearized models around the above equilibrium points/trajectories, noting that there will be a **lateral+longitudinal** model pair for each retrorocket, considering the kinematics of the Euler angles **zyx**.

With the intention develop the linearized models around the equilibrium trajectories, there's a need to determine the following equations:

- $\dot{x}(t) = Ax(t) + Bu(t);$
- $y(t) = Cx(t) + Du(t).$

To determine these equations, we need to develop the matrixes **A**, **B**, **C** and **D**.

$$\bullet \quad A = \begin{bmatrix} \frac{\partial f_1}{\partial p} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial \lambda} & \frac{\partial f_1}{\partial \omega} \\ \frac{\partial f_2}{\partial p} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial \lambda} & \frac{\partial f_2}{\partial \omega} \\ \frac{\partial f_3}{\partial p} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial \lambda} & \frac{\partial f_3}{\partial \omega} \\ \frac{\partial f_4}{\partial p} & \frac{\partial f_4}{\partial v} & \frac{\partial f_4}{\partial \lambda} & \frac{\partial f_4}{\partial \omega} \end{bmatrix} = \begin{bmatrix} 0_{3 \times 3} & I_3 & A_1 & 0_{3 \times 3} \\ 0_{3 \times 3} & A_2 & A_3 & A_4 \\ 0_{3 \times 3} & 0_{3 \times 3} & A_5 & A_6 \\ 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} & A_7 \end{bmatrix};$$

$$\begin{aligned}
 \circ \quad A_1 &= \begin{bmatrix} \partial/\partial\phi \\ \partial/\partial\theta \\ \partial/\partial\psi \end{bmatrix} \times \begin{bmatrix} R_{11}v_x + R_{12}v_y + R_{13}v_z \\ R_{21}v_x + R_{22}v_y + R_{23}v_z \\ R_{31}v_x + R_{32}v_y + R_{33}v_z \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \\
 \circ \quad A_2 &= \begin{bmatrix} \partial/\partial v_x \\ \partial/\partial v_y \\ \partial/\partial v_z \end{bmatrix} \times \begin{bmatrix} -skew(\omega) - 2D(v_z + v_y) \\ -skew(\omega) - 2D(v_x + v_z) \\ -skew(\omega) - 2D(v_x + v_y) \end{bmatrix}^T = \begin{bmatrix} 0 & -2D & -2D \\ -2D & 0 & -2D \\ -2D & -2D & 0 \end{bmatrix}; \\
 \circ \quad A_3 &= skew(g_{mars} \times z_I); \\
 \circ \quad A_4 &= \begin{bmatrix} \partial/\partial\omega_x \\ \partial/\partial\omega_y \\ \partial/\partial\omega_z \end{bmatrix} \times \begin{bmatrix} v_z\omega_y - v_y\omega_z \\ v_x\omega_z - v_z\omega_x \\ v_y\omega_x - v_x\omega_y \end{bmatrix}^T = \begin{bmatrix} 0 & v_z & -v_y \\ -v_z & 0 & v_x \\ v_y & -v_x & 0 \end{bmatrix}; \\
 \circ \quad A_5 &= \begin{bmatrix} \partial/\partial\phi \\ \partial/\partial\theta \\ \partial/\partial\psi \end{bmatrix} \times \begin{bmatrix} \omega_x + \omega_y \sin(\phi) \tan(\theta) + \omega_z \cos(\phi) \tan(\theta) \\ \omega_y \cos(\phi) - \omega_z \sin(\phi) \\ (\omega_y \times \sin(\phi) / \cos(\theta)) + (\omega_z \times \cos(\phi) / \cos(\theta)) \end{bmatrix}^T = \\
 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \\
 \circ \quad A_6 &= Euler2Q(\lambda); \\
 \circ \quad A_7 &= \begin{bmatrix} \partial/\partial\omega_x \\ \partial/\partial\omega_y \\ \partial/\partial\omega_z \end{bmatrix} \times \begin{bmatrix} \omega_y\omega_z jlr \\ \omega_x\omega_z jlr \\ \omega_x\omega_y jlr \end{bmatrix}^T = \begin{bmatrix} 0 & \omega_z jlr & \omega_y jlr \\ \omega_z jlr & 0 & \omega_x jlr \\ \omega_y jlr & \omega_x jlr & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \\
 \bullet \quad B &= \begin{bmatrix} \partial f_1/\partial T & \partial f_1/\partial np \\ \partial f_2/\partial T & \partial f_2/\partial np \\ \partial f_3/\partial T & \partial f_3/\partial np \\ \partial f_4/\partial T & \partial f_4/\partial np \end{bmatrix} = \begin{bmatrix} 0_{3 \times 1} & 0_{3 \times 1} \\ \frac{1}{nt} z_1 & 0_{3 \times 1} \\ 0_{3 \times 1} & 0_{3 \times 1} \\ 0_{3 \times 1} & jlr^{-1} \end{bmatrix}; \\
 \bullet \quad C &= \begin{bmatrix} I_3 & 0_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{1 \times 3} & 0_{1 \times 3} & z_I^T & 0_{1 \times 3} \end{bmatrix}; \\
 \bullet \quad D &= 0_{4 \times 4} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.
 \end{aligned}$$

To obtain some of the values, it was used the pre-defined Euler angles, as showed in slide 46 of “Modeling of Aerospace Vehicles” by Bruno Guerreiro.

$$\bullet \quad Q(\lambda) = \begin{bmatrix} 1 & \sin(\phi) \tan(\theta) & \sin(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) / \cos(\theta) & \cos(\phi) / \cos(\theta) \end{bmatrix}.$$

```
%Linearização

v_x=v(1);
v_y=v(2);
v_z=v(3);
omg_x=omg(1);
omg_y=omg(2);
omg_z=omg(3);

%Para matriz A

A1=zeros(3);
A2=[0,-2*D,-2*D;
    -2*D,0,-2*D;
    -2*D,-2*D,0];
A3=skew(g_marte*zI);
A4=[0,v_z,-v_y;
    -v_z,0,v_x;
    v_y,-v_x,0];
A5=zeros(3);%Porque omg_x, omg_y, omg_z são iguais a zero
A6=Euler2Q(lbd);
A7=zeros(3);

A=[ zeros(3), eye(3) , A1 , zeros(3)
    zeros(3), A2      , A3, A4
    zeros(3), zeros(3) , A5 , A6
    zeros(3), zeros(3) , zeros(3) , A7 ];

%Para matriz B

B=[ zeros(3,1), zeros(3);
    (1/mt)*zI, zeros(3);
    zeros(3,1), zeros(3);
    zeros(3,1), inv(jlr)];

%Para matriz C
C = [ eye(3) , zeros(3) , zeros(3) , zeros(3)
      zeros(1,3), zeros(1,3) , zI' , zeros(1,3) ];

%Para matriz D
D = zeros(4);
```

Figure 14 – Definition of all matrices needed for the linearisation of the model

To obtain the final values, it was needed a coding line to enter the system.

```
sys = ss(A,B,C,D);
```

Figure 15 – Coding line to develop the system

3.6. Goal number 6

Analyse each linearized model regarding their controlability, observability and stability.

Having the linearized model developed, and to analyse the model regarding their controlability, their observability and their stability, it's important to obtain the Jordan normal form (as stated in slide 21 of "State-space Analysis and Control" for stability).

Internal stability of continuous LTI systems

Definition ([2])

The continuous-time LTI system described above is said to be:

1. marginally stable iff all the eigenvalues of **A** have negative or zero real parts and all the Jordan blocks corresponding to eigenvalues with zero real part are 1×1 ;
2. asymptotically and exponentially stable iff **A** is Hurwitz (all the eigenvalues strictly negative real part); or
3. unstable iff at least one eigenvalue of **A** has positive real part or zero real part, but the corresponding Jordan block is larger than 1×1 .

Nova B. Guerreiro, MCAV 23/24
State-space Analysis
MIMO Modal Analysis
Specs and Control

Figure 16 – Slide number 21 of "State-space Analysis and Control"

To obtain these parameters, the code was developed around the Jordan normal form and the developed eigenvalues.

```
[Vj,Jor] = jordan(A),
% [Vj,Jor] = jordan(sym(A)),
[V,DL,W] = eig(A);
mode_obs = C*V,
mode_ctrl = W'*B,
```

Figure 17 – The code developed to verify Jordan normal form, controlability, observability and stability

Having succeeded, the linearized model obtained the following results.

Jor =											
0	1.0000	0	0	0	0	0	0	0	0	0	0
0	0	1.0000	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-1.2136	0	0	0	0	0	0	0	0
0	0	0	0	0.6068	0	0	0	0	0	0	0
0	0	0	0	0	0	1.0000	0	0	0	0	0
0	0	0	0	0	0	0	1.0000	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.6068	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1.0000
0	0	0	0	0	0	0	0	0	0	0	0

Figure 18 – Verification of the Jordan normal form of the linearized model

vj =

3.0735	2.5325	6.2604	-0.6956	-5.5648	6.1470	10.1302	16.6944	-16.6944	-1.6480	0	0
-3.0735	-7.5976	-10.4340	-0.6956	11.1296	-6.1470	-10.1302	-16.6944	16.6944	0	0	0
3.0735	2.5325	6.2604	-0.6956	-5.5648	0	0	0	0	0	0	0
0	3.0735	2.5325	0.8442	-3.3767	0	6.1470	10.1302	-10.1302	0	0	0
0	-3.0735	-7.5976	0.8442	6.7535	0	-6.1470	-10.1302	10.1302	0	0	0
0	3.0735	2.5325	0.8442	-3.3767	0	0	0	0	0	0	0
0	1.0000	0.1987	0	0	0	1.0000	0.1987	0	0	0	0
0	0	0	0	0	0	1.0000	0.1987	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1.0000	0
0	0	1.0000	0	0	0	0	1.0000	0	0	0	0
0	0	0	0	0	0	0	1.0000	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1.0000

Figure 19 – Verification of the stability of the linearized model

mode_ctrl =

1.0e-03 *

0	0.2189	0.1971	0
0	-0.2189	-0.1971	0
0	0.2189	-0.1971	0
-0.0223	-0.1015	-0.2091	0
-0.0729	-0.1508	0.1358	0
0.0431	-0.2527	0	0
0	-0.3096	0	0
0	0	-0.2787	0
0	0	0	-0.2431
0	0.3096	0	0
0	0	0.2787	0
0	0	0	0.2431

Figure 20 – Verification of the controllability of the linearized model

mode_obs =

1.0000	0	0	0.6980	0.3671	-0.0508	-0.5774	-0.5774	0	0.5774	0.5774	0
0	1.0000	0	-0.3490	0.3671	-0.5775	0.5774	0.5774	0	-0.5774	-0.5774	0
0	0	1.0000	-0.3490	0.3671	0.6283	-0.5774	0.5774	0	0.5774	-0.5774	0
0	0	0	0	0	0	0	0	1.0000	0	0	-1.0000

Figure 21 – Verification of the observability of the linearized model

4. Other Comments

While defining the simulation, it was used “*lbd_dot*” (instead of “*R_dot*”) to simplify the simulation.

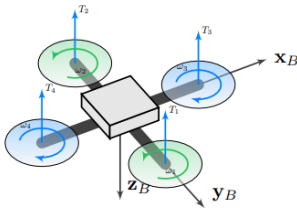
The nonlinear model and the linearized model are on different tabs to be user-friendly, having a link from the linearized model to run the nonlinear model.

[run Modelo_G5.m](#)

Figure 22 – The link to run the nonlinear model from the linearized model

To help the progress in this project, the group based some of its knowledge from the resolved coding exercises done by Bruno Guerreiro, more precisely the exercises 2.5 and 3.2.

Exercise 2.5
[\[Go to Solution\]](#)
Consider a drone with the “+” rotor configuration and arm length l .



(a) Obtain the propulsion force vector as a function of the squared angular velocities of each rotor, assuming the relation between thrust and angular squared velocity of each rotor $T_i = c_T \omega_i^2$.

(b) Obtain the propulsion moment vector as a function of the squared angular velocities of each rotor, assuming the relation between torque and angular squared velocity of each rotor $Q_i = c_Q \omega_i^2$.

(c) Obtain the configuration matrix, considering as the input vector $\mathbf{u} = [T \quad \mathbf{n}_p^T]^T$ and the rotor angular velocities vector $\omega_r = [\omega_1^2 \quad \omega_2^2 \quad \omega_3^2 \quad \omega_4^2]^T$, where $T = \sum_i T_i$.

Figure 23 – Exercise 2.5 by Bruno Guerreiro

Exercise 3.2
[\[Go to Solution\]](#)
Consider the multirotor nonlinear model given by:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{R}\mathbf{v} \\ \dot{\mathbf{v}} &= -\mathbf{S}(\omega)\mathbf{v} + g\mathbf{R}^T(\lambda)\mathbf{z}_I - \mathbf{R}(\lambda)\mathbf{D}_1\mathbf{R}^T(\lambda)\mathbf{v} - \frac{1}{m}T\mathbf{z}_I \\ \dot{\lambda} &= \mathbf{Q}(\lambda)\omega \\ \dot{\omega} &= -\mathbf{J}^{-1}\mathbf{S}(\omega)\mathbf{J}\omega - \mathbf{R}(\lambda)\mathbf{D}_2\mathbf{R}^T(\lambda)\omega + \mathbf{J}^{-1}\mathbf{n}_p \end{aligned}$$

with $m = 1.37kg$, $\mathbf{J} = \text{diag}(J_{xx}, J_{yy}, J_{zz}) = \text{diag}(0.022, 0.011, 0.031)kgm^2$, $\mathbf{D}_1 = 2\mathbf{D}_2 = \beta\mathbf{I}_3$, and $\beta = 0.1$.

(a) Considering an LTI model, resulting from the linearization around hover ($\mathbf{v}_e = \omega_e = \lambda_e = 0$), given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned}$$

Show that, considering $\mathbf{x} = [\mathbf{p}^T \quad \mathbf{v}^T \quad \lambda^T \quad \omega^T]^T$, $\mathbf{u} = [T \quad \mathbf{n}_p^T]^T$, $\mathbf{y} = [\mathbf{p}^T \quad \psi]^T$, the linearized system matrices can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{I}_3 & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & 2\beta\mathbf{I}_3 & \mathbf{S}(g\mathbf{z}_I) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_3 \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \beta\mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 3} \\ -\frac{1}{m}\mathbf{z}_I & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 1} & \mathbf{J}^{-1} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{1 \times 3} & \mathbf{0}_{1 \times 3} & \mathbf{z}_I^T & \mathbf{0}_{1 \times 3} \end{bmatrix}, \quad \mathbf{D} = \mathbf{0}_{4 \times 4}$$

(b) Obtain the modes of the system, computing the modal controllability and observability degrees, and characterize the system in terms of stability.

(c) Separate the model in four degrees of freedom: x motion, y motion, z motion, and yaw motion.

(d) Simulate the nonlinear and linear models in Matlab/Octave.

Figure 24 – Exercise 3.2 by Bruno Guerreiro

5. Webgraphy

- Bruno Guerreiro, "Lecture slides for Modeling and Control of Aerospace Vehicles (MCAV)", on the 2023 version of the FCT/UNL Moodle
- Bruno Guerreiro, "MCAV Exercise book", on the 2023 version of the FCT/UNL Moodle
- Bruno Guerreiro, "Exercise book code", on the 2023 version of the FCT/UNL Moodle
- Mars Fact Sheet where we got (*g_{marte} and ro_{marte}*) by the NSSDCA - <https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html>

6. Attachments

Github repository by José Corvo: https://github.com/Jose-Corvo-Lv99/Projeto_MCAV_G5.git

- On the repository, the nonlinear model link is https://github.com/Jose-Corvo-Lv99/Projeto_MCAV_G5/blob/7d8f252cdee5709893b92f362047b0180fe74eaf/Modelo_G5.m
- On the repository, the linearized model link is https://github.com/Jose-Corvo-Lv99/Projeto_MCAV_G5/blob/7d8f252cdee5709893b92f362047b0180fe74eaf/Linear_G5.m