MSAS – Assignment #2: Modeling

Matteo G. Juvara, matr. 249577

Exercise 1

To perform a preliminary analysis and evaluate a proposed satellite configuration, it is requested to model an active thermal control system and its associated mechanisms to assess the system's capability to maintain the temperatures within a certain range $[T_{min}; T_{max}]$. The satellite body is a rectangular cuboid (1.5 [m] height, 0.5 [m] side), with fixed solar panels (0.5 [m] side, 0.95 [m] length) located on the +Y-axis. The satellite flies with a fixed inertial attitude, with the satellite's Y-axis always directed towards the Sun. The satellite has two extendable radiators, which are located on the ±X-axis faces. The radiators are connected to the satellite body with electrically controlled hinges that can rotate along the Z-axis direction. The satellite with its radiators in open and closed configurations is represented in Figure 1.

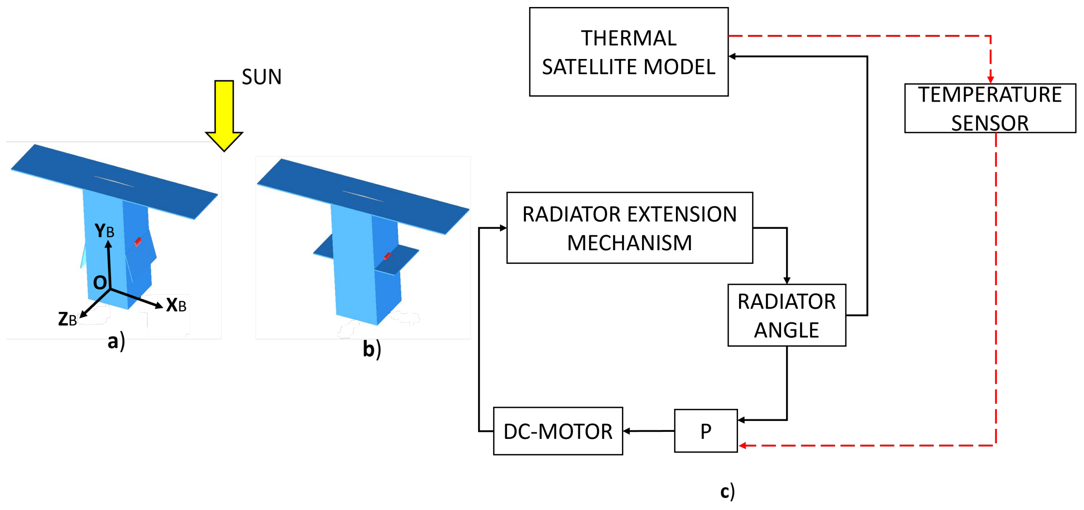

Figure 1: Satellite configuration: **a)** Closed radiator; **b)** Open radiator; **c)** Thermal control logic overview

The extension mechanism is based on a DC motor which gives the needed torque to move the radiator to the desired position. In Figure 2 the physical model of the mechanism is shown. The angle θ represents the rotation of the radiator with respect to the Z-axis and is measured counter-clockwise. For a rotation angle $\theta = 0$ deg, the radiators are in fully open configuration, for $\theta = -0.4\pi$ the radiators are in stowed (closed) configuration. The main parameters of the electro-mechanical model are reported in Table 1.

Parameter	Symbol	Value	Units
Resistance	R	0.1	Ω
Inductance	L	0.001	H
Motor constant	k_m	0.3	Nm/A
Radiator mass	m_r	0.2	kg
Radiator length	L_r	0.5	m

Table 1: Characteristics of the thermal control mechanism.

The radiator has a lower emissivity side, facing deep space in the closed configuration, and a higher emissivity side which is hidden when the radiator is in the stowed position. It is assumed

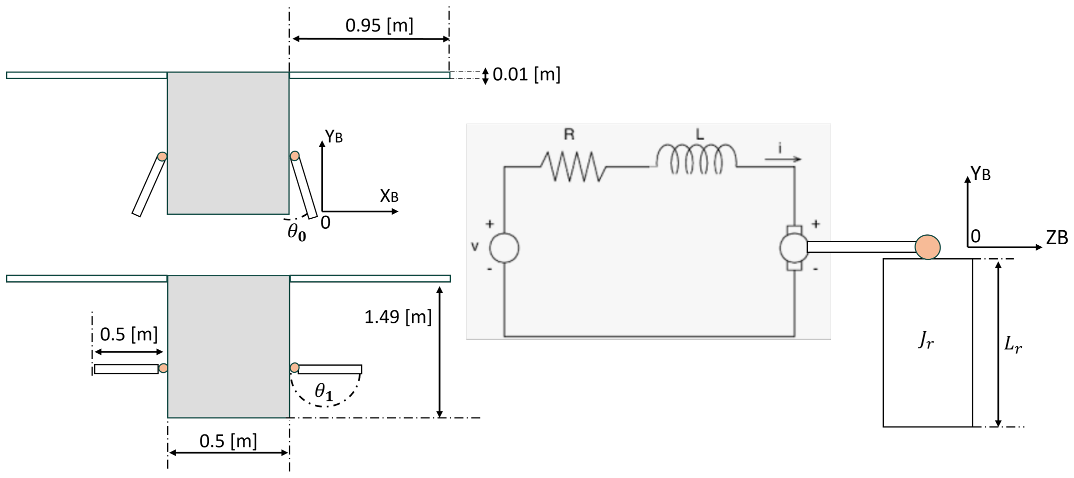

Figure 2: Schematics of the radiator extension mechanism

that radiative heat exchange among the satellite surfaces (re-radiation) is negligible and that no other thermal radiation sources (e.g., planets) are present. As a result, the thermal control of the satellite is regulated mainly by the radiator angle θ, as shown in Figure 1 c). By opening and closing the radiator, the effective area of the high- and low-emissivity sides exposed to deep space increases and decreases respectively. This results in a linear variation of the emissivity ϵ as a function of θ that can be expressed as in Eq. (1):

$$\epsilon(\theta) = \epsilon_{min} + \left[\frac{\epsilon_{max} - \epsilon_{min}}{0.4 \cdot \pi}\right](\theta(t) + 0.4 \cdot \pi) \tag{1}$$

When the radiator is in a closed configuration with $\theta(t) = -0.4\pi$ [rad] the emissivity is at its minimum $\epsilon_{min} = 0.01$. In fully open configuration instead, when $\theta(t) = 0$ [rad], the radiator emissivity reaches its maximum value $\epsilon_{max} = 0.98$. Using a lumped nodes thermal modeling, the satellite can be divided into 5 components (the two solar panels, the main body, and the two radiators). The radiators and solar panels are connected to the main body node by a conductive path, while all nodes are thermally radiating towards deep space (see Figure 3).

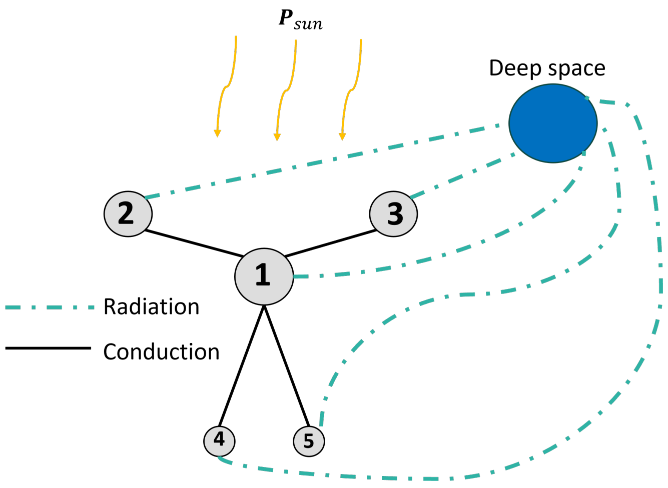

Figure 3: The thermal lumped nodes model

The solar radiation from the Sun is acting only on the two solar panel nodes and on the top face of the main-body node (the radiators are always in shadow). The temperature of deep space is constant and equal to T_{ds}= 3 [K]. The characteristics of the thermal network and its components are given in Table 2.

Parameter	**Symbol**	**Value**	**Unit**
Sun Power	P_{Sun}	1350	W/m^2
Heat capacity	C_1	1.5×10^5	J/K
Heat capacity	C_2, C_3	1187.5	J/K
Heat capacity	C_4, C_5	30	J/K
Thermal conductance	G_{12}, G_{13}, G_{14}, G_{15}	10	W/K
Absorptivity	α_1	0.6	-
Absorptivity	α_2, α_3	0.78	-
Emissivity	ϵ_1	0.45	-
Emissivity	ϵ_2, ϵ_3	0.75	-

Table 2: Thermal parameters of the system.

Part 1: causal modeling (9 points)

Considering the following main constitutive equations of the system:

$$C_i \frac{dT_i}{dt} = \sum_i^n (Q_{in}^i - Q_{out}^i) \tag{2}$$

$$V_{in} = k_p \cdot (T_1 - T_1^{ref}) \tag{3}$$

$$J_r \ddot{\theta} = \tau_{in} \tag{4}$$

where Eq. (2) is the conservation of energy at a thermal node; Eq. (3) is a proportional control law in which V_{in} is the input voltage applied on the DC-motor, and T_1^{ref} is the reference temperature of the main-body; and Eq. (4) is the simplified rotational radiator dynamics where the input torque is linked to current flowing into the motor through the relation: $\tau_{in}= k_m \cdot i$. Assuming that: the upper limit $\hat{T}_{max}$ and lower limit temperature $\hat{T}_{min}$ allowed for **node 1** are 300 [K] and 290 [K] respectively, the motor control is capable of maintaining the angle θ within the physical limits, all the thermal nodes start with a temperature of 298.15 [K], and that the radiator is closed at the beginning of the simulation ($\theta(0) = -0.4\pi$):

1. Formulate the full system of non-linear ODEs making explicit the state variables of the whole simulation.
2. Considering a simulation time of at least 50 [h], define the value for the proportional gain k_p such that: the temperature is kept around the target $T_1^{ref} = 294.15$ [K], and within $t = 10$ [h] the maximum temperature oscillations are less than 0.1% of T_1^{ref}. Plot the resulting temperature evolution over time on all thermal nodes.
3. Discuss the results and the ode used in Matlab for the integration.

Part 2: acausal modeling (6 points)

Using the Modelica standard library, reproduce in Dymola the physical model of the thermal-electro-mechanical system described above. You can build your **own model block** in Modelica to implement specific signals such as $\epsilon(\theta)$ or the control strategy. Please note that when you create a new model block (see in Table 3) you can insert input/output variables and your equations (see as an example the **Text View** of the block: *Modelica/Blocks/Math/Add*). Then, simulate it in OpenModelica comparing the results with the ones obtained using the causal model.

Physical element	*Modelica library*
Rigid-body	Modelica/Mechanics/MultiBody/Parts
Revolute joint	Modelica/Mechanics/MultiBody/Parts
Fixed-translational element	Modelica/Mechanics/MultiBody/Parts
DC-motor	Modelica/Electrical/Machines/BasicMachines/DCMachines
Thermal node	Modelica/Thermal/HeatTransfer/Components
Thermal resistances	Modelica/Thermal/HeatTransfer/Components
Logical switch	Modelica/Blocks/Logical/Switch
Hysteresis	Modelica/Blocks/Logical/Switch
Temperature sensor	Modelica/Thermal/HeatTransfer/Sensors
Your model block	File/New/Block

Table 3: Modelica libraries.

(15 points)

Part 1

Point 1

The complete linear system can be retrieved from the main constitutive equations, in particular Eq. (2), Eq. (3), and Eq. (4). The spacecraft is divided into five distinct thermal nodes, as seen in Fig. 3. The nodes operate under the following assumptions:

- **Body (Node 1):** The top surface of the main body is assumed to contribute only to the solar irradiation. The radiative heat exchange with the deep space, on the other hand, considers all the other faces of the main body. As the radiators never fully close, their shadowing effects are assumed negligible. These effects vary with the angular position of the radiators and would require a more complex and dynamic computation of the exposed area;
- **Solar Panels (Nodes 2 & 3):** The top surface of the panels are considered to contribute only to the solar irradiation, while all the other faces of the panels are considered to contribute to the radiative heat exchange.
- **Radiators (Nodes 4 & 5):** The radiators only contribute to the radiative heat exchange with the deep space. Their thickness can be assumed negligible, thus only the top and bottom surfaces are considered. Their emissivity depends on their angular position, θ, as reported in Eq. (1).

The heat powers involved in the model are the solar irradiation, the radiative heat exchange with the deep space and the conductive power between the nodes. The equations and the surfaces used for each node are expressed below.

$$
\begin{aligned}
Q_i^{sun} &= P_{sun} \cdot \alpha_i \cdot A_i^{sun} \\
Q_i^{rad} &= \sigma \cdot \epsilon_i \cdot A_i^{rad} \cdot (T_i^4 - T_{ds}^4) \\
Q_{1i}^{cond} &= G_{1i} \cdot (T_i - T_1)
\end{aligned}
$$

Nodes	A_{sun} [m^2]	A_{rad} [m^2]
1	0.25	3.24
2	0.475	0.499
3	0.475	0.499
4	0	0.5
5	0	0.5

Table 4: Thermal Nodes Areas

The DC motor is governed by a system of two ODEs which describe the electrical and mechanical behaviours of the motors:

$$
\begin{cases}
{}^{di}\!/_{dt} = {}^{1}\!/_{L} \cdot (V_{in} - R \cdot i - EMF) \\
\ddot{\theta} = {}^{\tau_{in}}\!/_{J}
\end{cases}
$$

Where V_{in} is the input voltage of the motor given by the proportional control law (Eq. (3)), EMF is the back electromotive force of the motor ($EMF = k_m \cdot \dot{\theta}$), τ_{in} is the torque generated by the motor ($\tau_{in} = k_m \cdot i$), J is the radiator inertia moment ($J = {}^{1}\!/_{3} \cdot m_r \cdot L_r^2$).

The full system of non-linear ODEs is composed of eight equations, one for each state variable.

$$\begin{cases} \dfrac{dT_1}{dt} = \dfrac{1}{C_1} \cdot (\sum_{i=1}^{4} Q_{1i}^{cond} + Q_1^{sun} - Q_1^{rad}) \\ \dfrac{dT_2}{dt} = \dfrac{1}{C_2} \cdot (-Q_{21}^{cond} + Q_2^{sun} - Q_2^{rad}) \\ \dfrac{dT_3}{dt} = \dfrac{1}{C_3} \cdot (-Q_{31}^{cond} + Q_3^{sun} - Q_3^{rad}) \\ \dfrac{dT_4}{dt} = \dfrac{1}{C_4} \cdot (-Q_{41}^{cond} - Q_4^{rad}) \\ \dfrac{dT_5}{dt} = \dfrac{1}{C_5} \cdot (-Q_{51}^{cond} - Q_5^{rad}) \\ \dfrac{di}{dt} = \dfrac{1}{L} \cdot (V_{in} - R \cdot i - EMF) \\ \dfrac{d\theta}{dt} = \dot{\theta} \\ \dfrac{d\dot{\theta}}{dt} = \dfrac{k_m}{J} \cdot i \end{cases}$$

The state variable of the systems are the temperature of each of the five thermal nodes, the current of the DC motor, the angular position and velocity of the radiators with respect to the Z-axis, as shown in Fig. 1.

Points 2 and 3

To simulate the model, the function `RHS1` was implemented. This function computes the equations shown in the previous subsection and applies a control on the angular position θ of the radiators, ensuring that it stays between the given bounds.

In particular the control acts in two distinct conditions. The lower limit is obtained when the temperature of the main body is lower than the reference temperature ($T_1 < T_{ref}$) and the radiator's angular position tries to exceed it's inferior bound ($\theta \leq \theta_{min}$). On the other hand, the upper limit is obtained when the temperature of the main body is higher than the reference temperature ($T_1 > T_{ref}$) and the radiator's angular position tries to exceed it's upper bound ($\theta \geq \theta_{min}$). While these conditions are true, the input voltage is set to zero ($V_{in} = 0$) to halt the movement of the radiators outside it's bounds. In all other cases the input voltage is controlled by the proportional law expressed in Eq. (3).

The simulation is performed using the solver `ode15s`, as the model exhibits a moderate stiffness due to the combination of the slow dynamics of the thermal system and the fats dynamics of the DC motor. The absolute and relative tolerances are set to `1e-12` to ensure a smooth and precise solution. The initial conditions of the state variables are reported in Table 5.

T_1 [K]	T_2 [K]	T_3 [K]	T_4 [K]	T_5 [K]	i [A]	θ [rad]	$\dot{\theta}$ [rad]
298.15	298.15	298.15	298.15	298.15	0	$-0.4 \cdot \pi$	0

Table 5: Initial Conditions of the State Variables

The proportional constant k_p was chosen through a trial and error process to satisfy the mission requirements of keeping the main body temperature T_1 within 0.1% of the reference temperature T_{ref}. The adopted value is $k_p = 0.0002$, which satisfies all system requirements and minimizes the movement of the radiators at the same time. It must be noted that using this coefficient the voltage imposed to the circuit reaches values in the order of 10^{-5} and lower. If these voltages aren't high enough for the voltage supply, the value of k_p must be increased, leading to higher voltages and at the same time more frequent movement of the radiators.

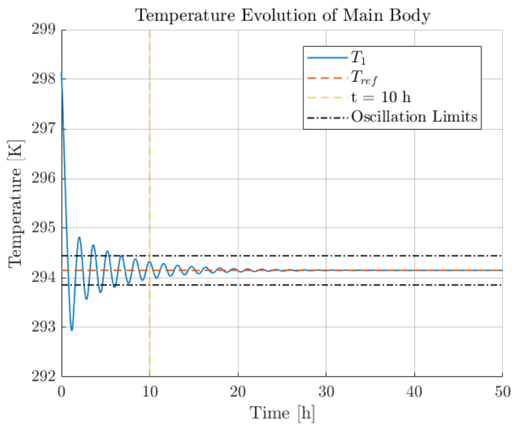

Figure 4: Temperature of the main body

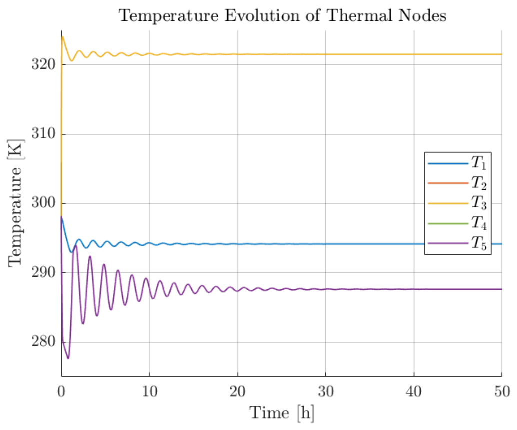

Figure 5: Temperature of the thermal nodes

As seen in Fig. 4, the temperature of the main body remains inside the required boundaries within the time-frame requested. The temperature of all the thermal nodes initially shift rapidly from the initial temperature closer to their equilibrium temperature. The solar panel have a higher equilibrium temperature as they have the highest area exposed to the solar irradiation, while the radiators, which aren't exposed to the Sun, have a lower equilibrium temperature. It must be noted that the radiator temperatures exhibit larger oscillations with respect to other thermal nodes due to the change in emissivity, which modulates their radiative cooling capacity.

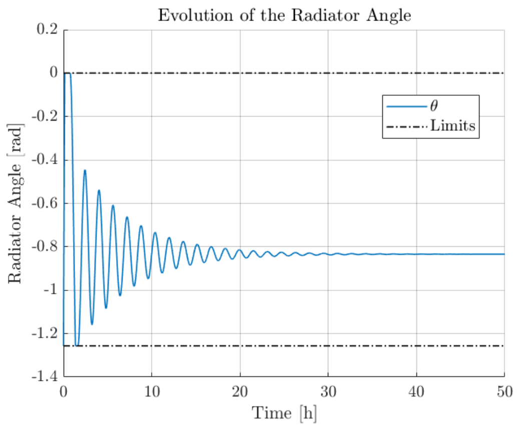

Figure 6: Radiator angular position

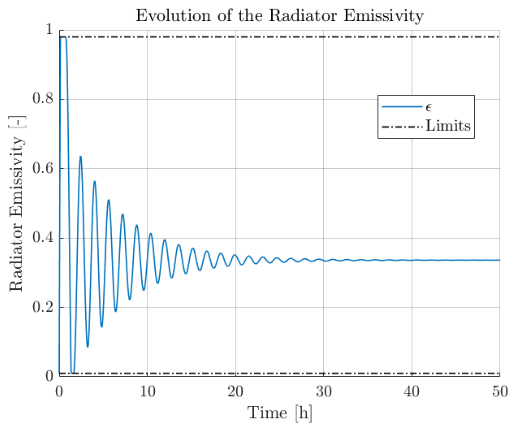

Figure 7: Radiator emissivity

The radiators initially open completely and remain in this position until the main body temperature drops sufficiently close to the reference temperature. At this point the radiators start oscillating until they reach a fixed position of around $\theta = -0.265 \cdot \pi$ rad. As seen if Fig. 7, the behaviour of the emissivity of radiators mimics closely their angular position. The emissivity oscillates initially until reaching an equilibrium at around $\epsilon = 0.336$.

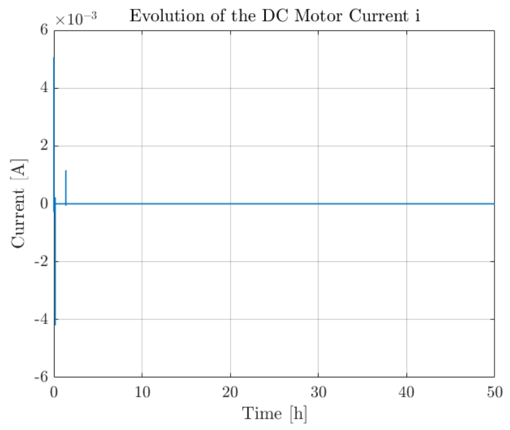

Figure 8: DC motor current evolution

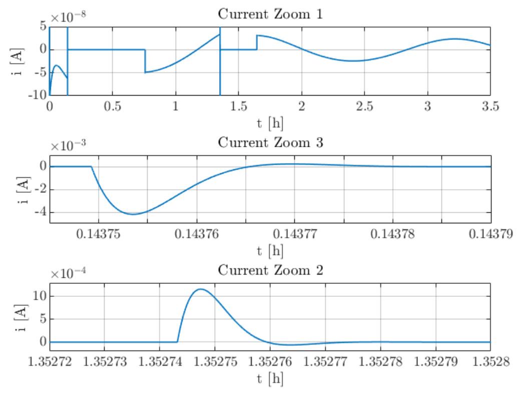

Figure 9: Current zooms

Initially the current in the DC motor current exhibits an irregular behaviour as the radiators reach saturation, imposing the current to $i = 0$ A when the radiators remain fixed in a boundary position. This is behaviour is paired with sharp current peaks, as seen in the second and third subplots of Fig. 9, which are transients of the motor while it switches on and off. Finally, the current assumes an oscillating behaviour until it flatlines at $i = 0$ A, as no current is generated when the radiators are fixed.

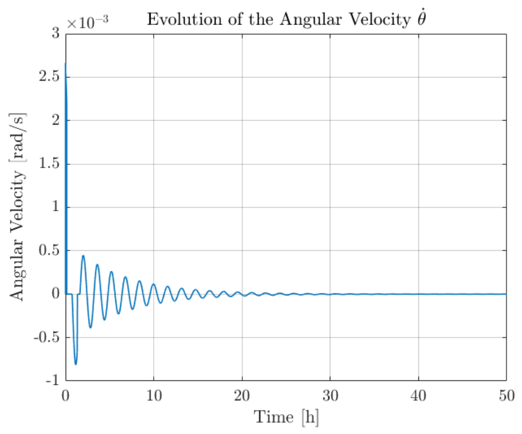

Figure 10: Radiator angular velocity

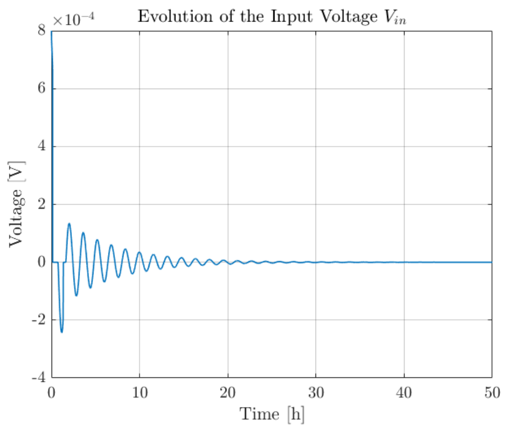

Figure 11: DC motor voltage evolution

As seen in Fig. 10 and Fig. 11, the behaviour of the radiator angular velocity, $\dot{\theta}$, and the DC motor input voltage, V_{in}, have an almost identical behaviour. This is due to the fact that the input voltage directly drives the motor responsible for radiator's angular movement.

The final values of the state variables after 50 hours are reported in Table 6.

T_1 [K]	T_2 [K]	T_3 [K]	T_4 [K]	T_5 [K]	i [A]	θ [rad]	$\dot{\theta}$ [rad]
294.15	321.50	321.50	287.62	287.62	$-8.1 \cdot 10^{-12}$	$-0.265 \cdot \pi$	$5.8 \cdot 10^{-8}$

Table 6: Final Values of the State Variables

Part 2

To implement an acausal model, the system was divided into two main parts: the thermal and electro-mechanical subsystems. The `Modelica Library` components, together with a slightly modified block, were utilized to build these models. The thermal and electro-mechanical subsystems were developed as separate model, exploiting a greater flexibility in handling signals and defining equations. These two subsystems were then integrated to perform a full system simulation. The additional custom component used in the model is defined as `VariableRadiation` and is very similar to `BodyRadiation` block, with the difference that it accepts as input the emissivity used, effectively allowing for a variable emissivity to be implemented.

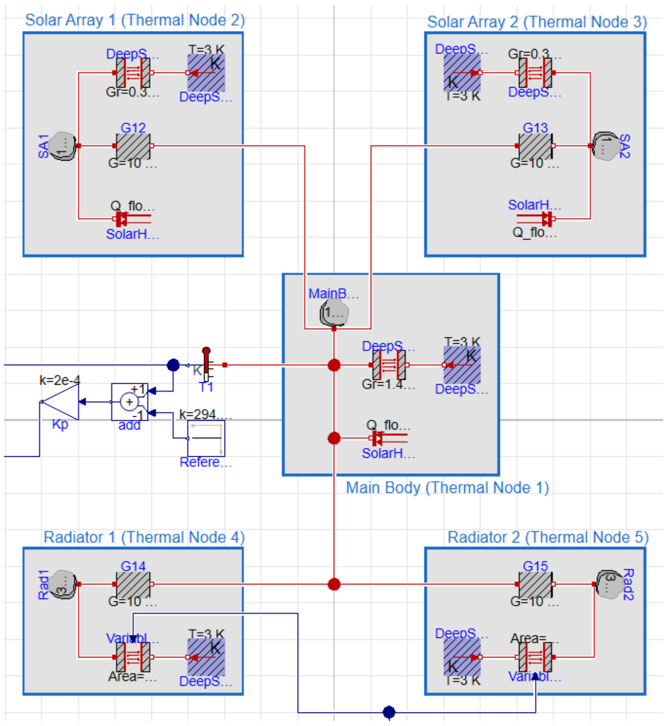
Solar Array 1 (Thermal Node 2)
Solar Array 2 (Thermal Node 3)
Main Body (Thermal Node 1)
Radiator 1 (Thermal Node 4)
Radiator 2 (Thermal Node 5)
SA1
SA2
G12
G13
G14
G15
MainB...
T1
Kp
k=2e-4
add
k=294...
Refere...
Rad1
Rad2
T=3 K
DeepS...
SolarH...
Q_flo...
Variabl...
Area=...

Figure 12: Thermal Model

As seen in Fig. 12, the model is clearly divided in the five thermal nodes. The main heat sources are the same seen in Part 1 and have been modeled in the thermal subsystem through the use of the following blocks:

- For the solar irradiation of the main body and the panels a `FixedHeatFlow` source block was used, which used as input the heat computed through the following formula:

$$Q_i^{sun} = P_{sun} \cdot \alpha_i \cdot A_i^{sun} \tag{5}$$

- For the radiative heat exchange with the deep space a `BodyRadiation` block was used connected to a `FixedTemperature` source block, set at 3 K, the deep space temperature. The `BodyRadiation` used as input the net radiation conductance G_r, which is defined as:

$$Gr = A_{rad} \cdot \epsilon \tag{6}$$

  It must be noted that for the radiators the custom block `VariableRadiation` was used, which took as an external output also the emissivity of the surface and used as an internal output only the radiative area.

- For the conductive heat a simple `ThermalConductor` block was place between one thermal node and the other. This block used as input only the conductance coefficient between the two nodes.

Parameter	Node 1	Node 2	Node 3	Node 4	Node 5
Q^{sun} [W]	202.5	500.175	500.175	-	-
G_r [m^2]	1.458	0.37425	0.37425	-	-
G_i [W/K]	-	10	10	10	10

Table 7: Input parameters of the thermal model

The parameters used as input for the thermal model as reported in Table 7

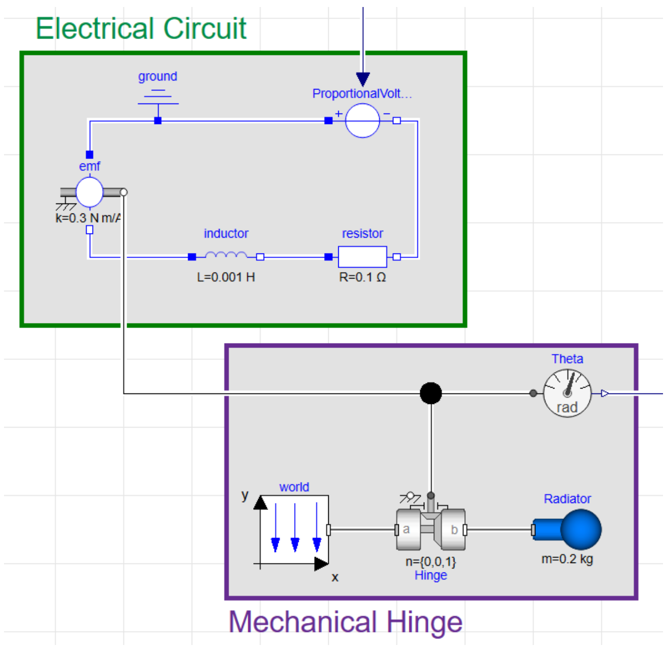

Figure 13: Electro-mechanical model

In Fig. 13 the electro-mechanical model is highlighted. The electrical circuit uses a `SignalVoltage` source block to generate a voltage based on a value output from the control logic. The voltage is then connected to a RL circuit which powers a DC motor, here modeled with a `RotationalEMF` block. This block creates a mechanical torque, which is then imposed to a `Joint.Revolute` block which mimics a hinge with attached a `Body` block with mass and inertia of the radiators. The angular position of the radiators is then measured and output to the emissivity calculation for the radiators and the control logic.

Fig. 14 shows the control logic implemented in the model, which is the same explained in Part 1. It receives as input the angular position of the radiators as well as the temperature of the main body and outputs a signal voltage to the electric circuit. On the other hand, Fig. 15 shows the computation of the radiators emissivity, based on Eq. (1). It takes as input only the angular position of the radiators and outputs the emissivity of the radiators.

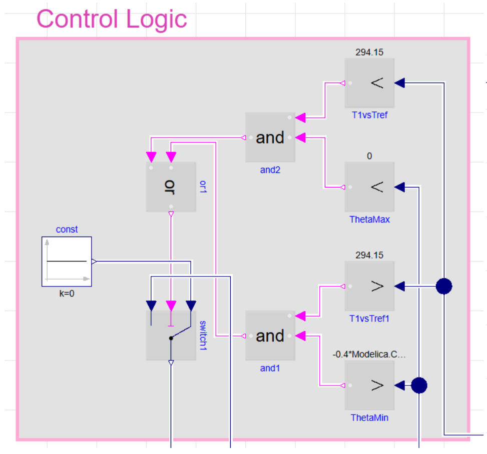

Figure 14: Control Logic

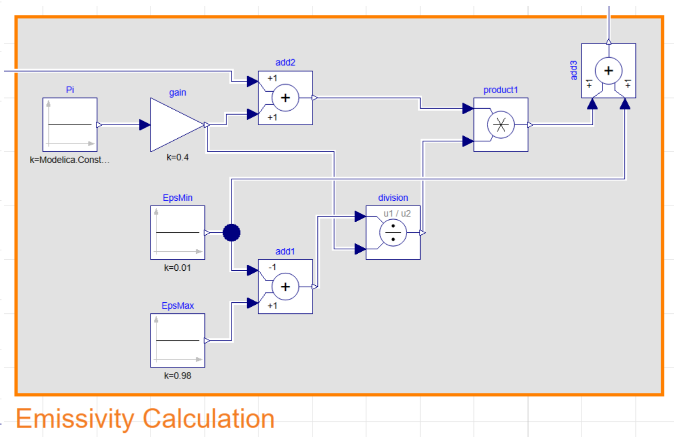

Figure 15: Emissivity Calculation

The simulation is performed using the DASSL solver, which ensures accuracy and stability for stiff problems such as the system at hand. The tolerance is set to 10^{-9} and the length of each interval is set to 1 s, matching what was done for the causal modeling on Matlab for consistency. The results are shown below.

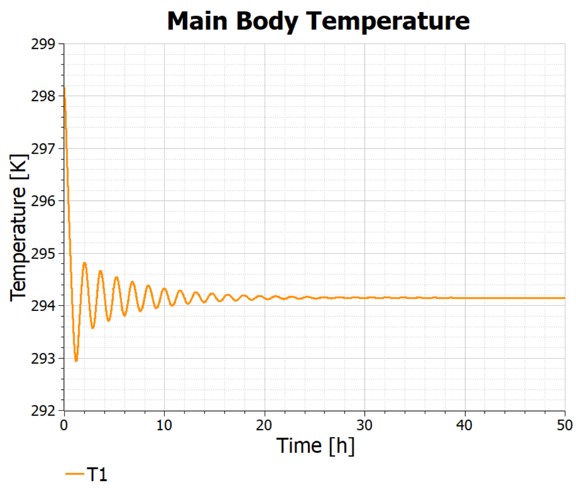

Figure 16: Temperature of the main body

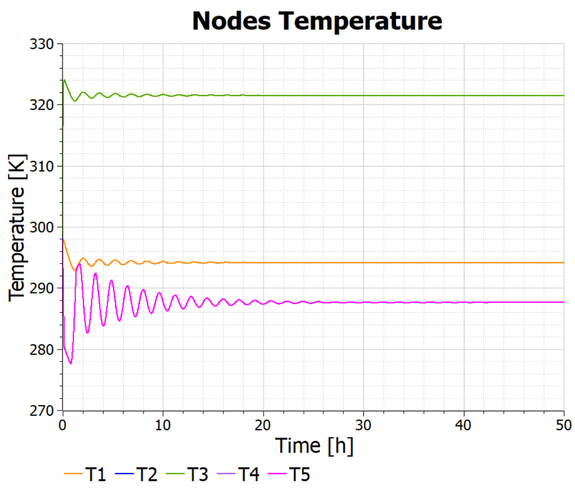

Figure 17: Temperature of the thermal nodes

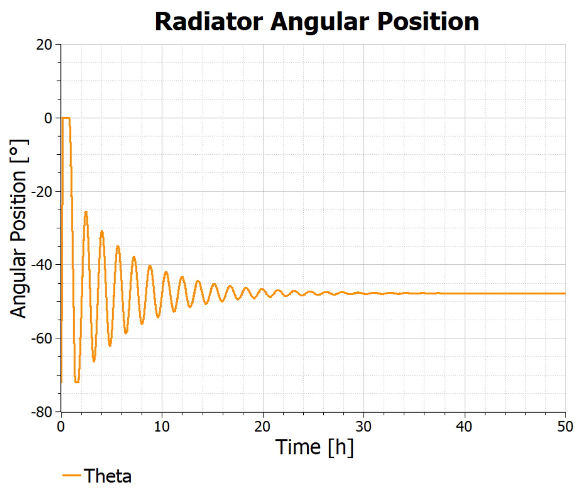

Figure 18: Angular position of the radiators

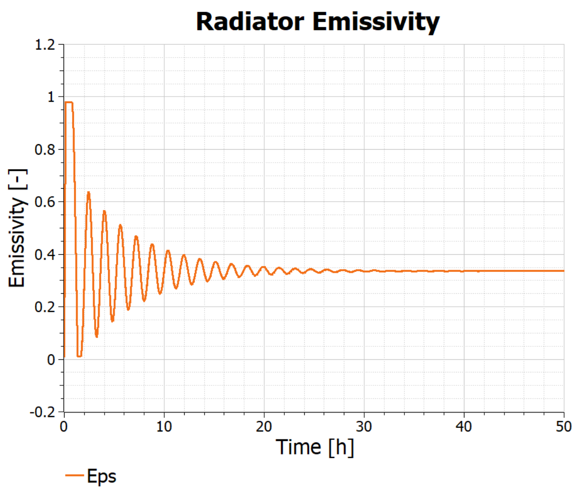

Figure 19: Emissivity of the radiators

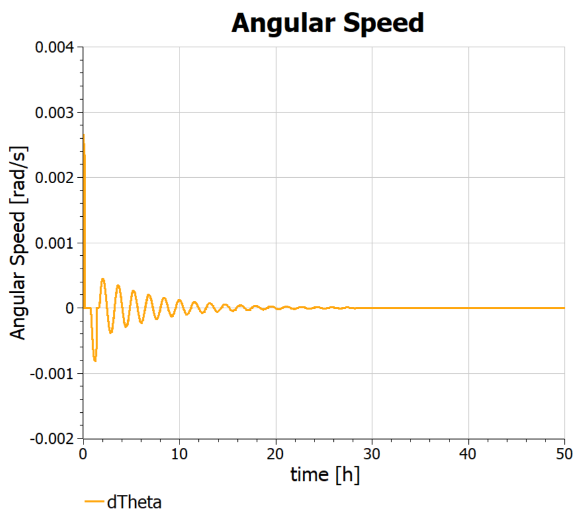

Figure 20: Angular speed of the radiators

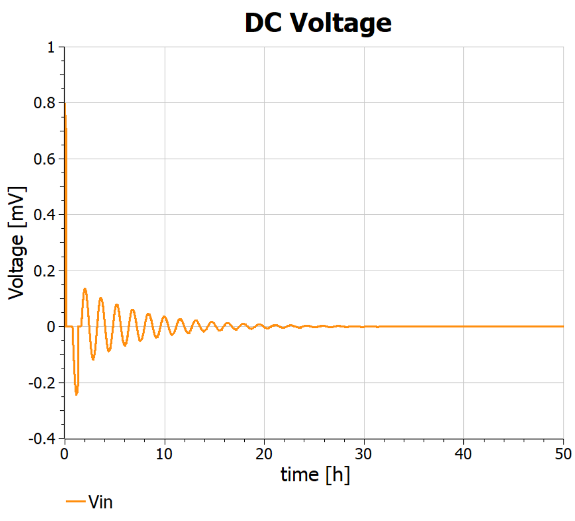

Figure 21: DC motor voltage

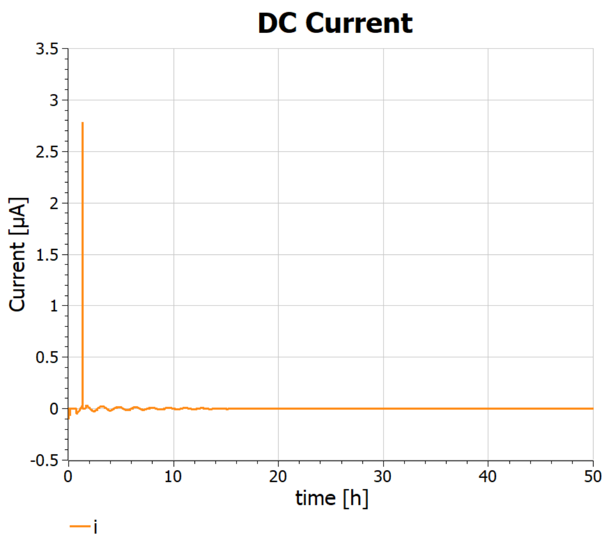

Figure 22: DC motor current

The graphs of the state variables and the voltage of the motor match the values computed through causal modeling seen in Part 1. We can observe the same pattern of the current of the DC motor in Fig. 22. The same irregular patterns are present in the beginning due to the control activating and also the current peaks due to the transients are present, even though not all of the ones seen in the causal modeling are present. The final values of the acausal model simulation are shown in Table 8, of which the proximity to the parameters seen in Part 1 can be appreciated.

T_1 [K]	T_2 [K]	T_3 [K]	T_4 [K]	T_5 [K]	i [A]	θ [rad]	$\dot{\theta}$ [rad]
294.15	321.50	321.50	287.62	287.62	$-8.1 \cdot 10^{-12}$	$-0.265 \cdot \pi$	$5.8 \cdot 10^{-8}$

Table 8: Final Values of the State Variables

A time comparison is performed between the causal and acausal simulations. To ensure statistical relevance, the simulations are repeated 10 times, balancing sufficient statistical reliability with low computational effort. The mean computational time is computed and shown in Table 9.

	Mean computational time [s]
Matlab	0.4165
OpenModelica	2.137

Table 9: Computational time comparison

The results highlight the higher efficiency of the causal model in Matlab, with a significantly lower mean computational time. In contrast, the acausal approach in OpenModelica may provide more physical accuracy for complex systems.

Exercise 2

A simplified conceptual map of the Attitude Control System (ACS) of a satellite is depicted in Figure 23. For the accuracy of some measurements that the spacecraft has to obtain, the probe has to fly in an orbit very close to the Earth's surface. The spacecraft is therefore subjected to a high atmospheric drag. The ACS has to compensate for the deceleration caused by this drag continuously. In general, ACSs are made of 3 sensors in orthogonal directions and 3 thrusters to detect and compensate for linear and angular accelerations on the probe in each direction, but for the sake of this project, only compensation in the tangential direction will be considered.

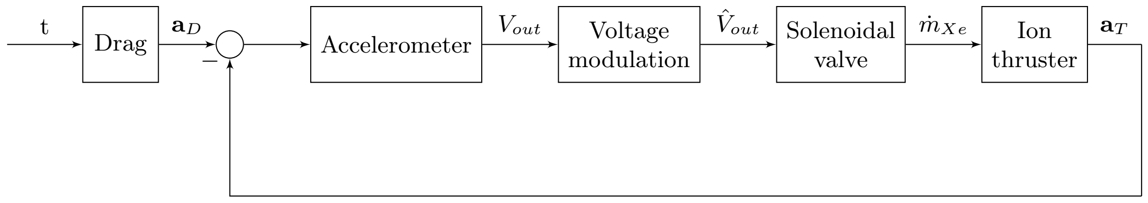

Figure 23: Block diagram of the ACS.

The ACS is modeled with several modules: an **accelerometer** measuring the accelerations acting on the spacecraft, a **voltage modulation block** controlling the output voltage, the **control valve** modifying the aperture of the thruster valve, and the **ion thruster**.

The accelerometer, kept aligned with the direction of velocity, presents inside a seismic mass that is subject to an external acceleration $\mathbf{a_D}$ and $\mathbf{a_T}$ given only by the drag D and the thrust T, respectively. The mass is in between the stators of a condenser and, moving for the acceleration given by these two forces changes the output voltage V_{out} of the circuit of the accelerometer. The dynamics of the seismic mass itself can be modeled as a mass-spring-damper system as follows

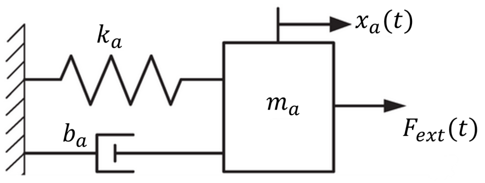

Figure 24: Model of the accelerometer.

$$\dot{x}_a = v_a \tag{7}$$

$$F_{ext}(t) = \frac{T-D}{M_{SC}} m_a = m_a \dot{v}_a + b_a v_a + k_a x_a \tag{8}$$

where M_{SC} is the mass of the spacecraft, m_a is the seismic mass value, and b_a and k_a are respectively the damper and the spring terms present in the accelerometer. For the causal modeling, consider the V_{out} as directly proportional to the velocity of the seismic mass, as $V_{out} = K_{acc} v_a$ [1].

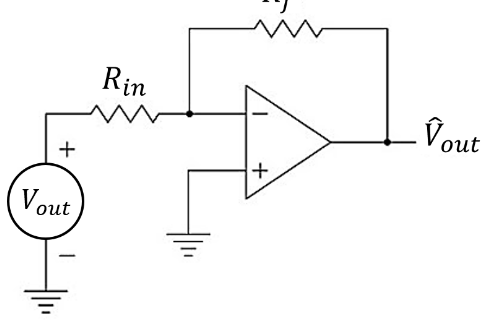

Figure 25: Inverting operational amplifier.

The output of the accelerometer V_{out} needs to be modulated into $\hat{V}_{out}$ to adjust the control for the solenoidal valve. This modulation is performed through an operational amplifier, displayed in Figure 25. The operational amplifier in inverting configuration modifies V_{out} into $\hat{V}_{out}$ accordingly to:

$$\hat{V}_{out} = -\frac{R_f}{R_{in}} V_{out} \tag{9}$$

where R_f and R_{in} are the two resistances modifying the input voltage.

The voltage $\hat{V}_{out}$ is itself the input of the solenoidal valve, shown in Figure 26, and creates a current I commanding the flow control valve. The current passes through a circuit modeled with only a solenoid with a variable inductance $L(x_v)$. As the current variably flows, a magnetic field is generated. A resulting force f_v acts on the spool, which as a result moves.

The armature-spool arrangement is then modeled again as a lumped parameter spring-mass-damper system

$$\dot{x}_v = v_v \tag{10}$$

$$m_v\dot{v}_v = -k_v x_v - b_v v_v + \frac{1}{2}I^2\frac{dL}{dx_v} \tag{11}$$

$$\dot{I} = \frac{1}{L(x_v)}\hat{V}_{out} \tag{12}$$

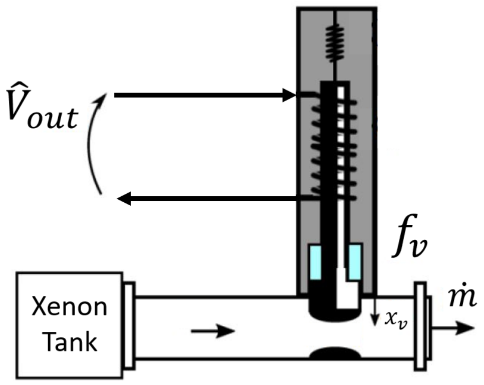

Figure 26: Model of the solenoidal valve.

where x_v and v_v represent the position and velocity of the spool, and b_v and k_v are respectively the damper and the spring terms, and m_v is the mass of the spool. The dynamics of the current I is generated by $\hat{V}_{out}$ through the variable inductance $L(x_v) = \frac{1}{\alpha+\beta x_v}$, depending on the position of the valve itself.

As the spool moves, the area $A_v(x_v)$ of the duct through which a Xenon flow is passing changes. The resulting mass flow rate $\dot{m}_{Xe}$ of Xenon is ejected into the ion thruster, shown in Figure 27. In particular, the area of the valve can be modeled as

$$A_v(x_v) = A_0 + \ell(x_{v,\max} - x_v) \tag{13}$$

In particular, the orifice area $A_v(x_v)$ has been modeled as linearly varying with x_v. The tool controlling the area is a flap of width ℓ and maximum extension $x_{v,\max} = \ell$, for simplicity. The value A_0 is the minimum area of the orifice, such that:

$$0 \leq x_v \leq x_{v,\max}$$

$$\underbrace{A_0 = A_v(x_{v,\max})}_{\text{minimum area}} \leq A_v \leq \underbrace{A_v(0) = A_0 + \ell^2}_{\text{maximum area}} \tag{14}$$

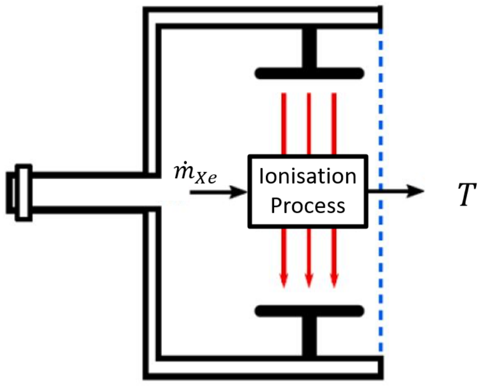

Figure 27: Model of the ion thruster.

The ion thruster comprises an ionization chamber that utilizes a strong magnetic field, ionizing the Xenon flux. The generated ions are moved towards the acceleration grid, which accelerates the ions using a strong electric field, producing a thrust on the spacecraft as the final effect. The flux of Xenon $\dot{m}_{Xe}$ can be expressed as follows

$$\dot{m}_{Xe} = A_v(x_v)\sqrt{k\rho_{Xe}p_T(\frac{2}{k+1})^{\frac{k+1}{k-1}}} \tag{15}$$

where $\rho_{Xe} = \dfrac{p_T}{\bar{R}T_T}$, k and $\bar{R}$ being respectively Xenon specific heat ratio and gas constant. In the tank, the gas is at a total pressure of p_T and temperature of T_T. The Xenon flow enters the thruster and is here ionized into ions with mass m_i and charge q. After the acceleration imposed by the acceleration grid with a voltage ΔV, the ions exit from the nozzle with a velocity v_{exit}, producing the thrust T, as follows

$$T = \dot{m}_{Xe} v_{ext} = \dot{m}_{Xe} \sqrt{\frac{2q\Delta V}{m_i}} \tag{16}$$

Table 10 reports the values for the parameters to simulate the ACS system, with symbol and unit of measure. The drag $D(t)$ can be modeled as a function of time only, as it follows

$$D(t) = 2.2 - \cos(\omega_s t) + 1.2 \sin(\omega_o t) \cos(\omega_o t) \tag{17}$$

where t enters in seconds, and $D(t)$ is in mN.

Component	**Parameter**	**Symbol**	**Value**	**Unit**
Accelerometer	Spacecraft mass	M_{SC}	300	kg
	Seismic mass	m_a	0.32	kg
	Accelerometer damper	b_a	$[1.5 \cdot 10^3 – 2 \cdot 10^4]$	Ns/m
	Accelerometer spring	k_a	$[5 \cdot 10^{-5} – 3 \cdot 10^{-3}]$	N/m
	Acc. proportional coefficient	K_{acc}	1	Vs/m
Amplifier	Inverting resistance	R_{in}	[0.1–10]	Ω
	Feedback resistance	R_f	$[1 \cdot 10^4 – 8 \cdot 10^4]$	Ω
Solenoidal Valve	Spool mass	m_v	0.1	kg
	Valve spring	k_v	$1 \cdot 10^3$	N/m
	Valve damper	b_v	$1 \cdot 10^3$	Ns/m
	Solenoid constant	α	$2.1 \cdot 10^{-2}$	1/H
	Solenoid gain	β	-60	1/Hm
	Minimum area	A_0	$4.7 \cdot 10^{-12}$	m^2
	Maximum extension	$x_{v,\max}$	$1 \cdot 10^{-5}$	m
Thruster	Heat ratio	k	1.66	-
	Tank pressure	p_T	$2 \cdot 10^5$	Pa
	Tank temperature	T_T	240	K
	Gas constant	$\bar{R}$	63.32754	J/kg K
	Charge	q	$1.6 \cdot 10^{-19}$	C
	Voltage	ΔV	2000	V
	Ion mass	m_i	$2.188 \cdot 10^{-25}$	kg
Drag	Secular pulsation	ω_s	$1.658226 \cdot 10^{-6}$	rad/s
	Orbital pulsation	ω_o	$1.160758 \cdot 10^{-3}$	rad/s

Table 10: Parameters for the simulation of the ACS.

Part 1: causal modeling (9 points)

Reproduce in Matlab the physical model of the ACS, in particular

1. Formulate the full system of non-linear ODEs making explicit the state variables of the whole simulation. Tune the values of the parameters not explicitly stated in Table 10 to have the compensating thrust T the most similar to the disturbing drag D.

2. Choose the appropriate initial conditions and simulate for three orbital periods T_o, where $T_o = 2\pi/\omega_o$.

3. Discuss the selection of the ODE integration scheme.

Part 2: acausal modeling (6 points)

Reproduce in Simscape the physical model of the ACS. Choose the appropriate initial conditions and parameters, simulate for three orbital periods T_o, and compare the results with those obtained in the prior point.

Note: when modeling the solenoid valve in Simscape, please consider the nominal values shown in Table 11.

Parameter	**Value**	**Unit**
Pull-in forces [F_1 - F_2]	[9000 - 12]	N
Stroke [x_1 - x_2]	[0 - 0.1]	mm
Maximum Stroke	0.1	mm
Rated voltage	0.6	mV
Rated current	0.1	A
Contact stiffness	0.12×10^6	N/m
Contact damping	10^4	Ns/m

Table 11: Solenoid parameters in Simscape.

(15 points)

Part 1

Point 1

The system of non-linear ODEs for this problem is composed of: the dynamics of the accelerometer, modeled as a spring-mass-damper; the governing equations of the solenoidal valve, modeled as an electric circuit with variable inductance; the armature-spool mechanism of the solenoidal valve, modeled one again as a spring-mass-damper. The state variables of the problem are the accelerometer displacement x_a, the accelerometer velocity v_a, the valve displacement x_v, the valve velocity v_v, and the valve current i. The five equations of the problem, one for each state variable, are shown below:

$$\begin{cases} \dfrac{dx_a}{dt} = v_a \\ \dfrac{dv_a}{dt} = \dfrac{T-D}{m_{sc}} - \dfrac{b_a}{m_a} v_a - \dfrac{k_a}{m_a} x_a \\ \dfrac{dx_v}{dt} = v_v \\ \dfrac{dv_v}{dt} = \dfrac{1}{2}\dfrac{dL}{dx_v} \cdot i^2 - \dfrac{b_v}{m_v} v_v - \dfrac{k_v}{m_v} x_v \\ \dfrac{di}{dt} = \dfrac{1}{L(x_v)} \cdot \hat{V}_{out} \end{cases}$$

Where $\hat{V}_{out}$ is the voltage computed using Eq. (9), having as input $V_{out} = K_{acc} \cdot v_a$. The drag is computed using Eq. (17), while the thrust of the spacecraft was obtained using Eq. (13), Eq. (15), and Eq. (16) .The Matlab function `RHS2` was developed to compute the derivatives of the state variables at each time step. The variable inductance $L(x_v)$ and it's derivative ${}^{dL}/_{dx_v}$ can be computed using the equations reported below:

$$L(x_v) = \frac{1}{\alpha + \beta \cdot x_v} \tag{18}$$

$$\frac{dL(x_v)}{dx_v} = \frac{-\beta}{(\alpha + \beta \cdot x_v)^2} \tag{19}$$

In order to tune the parameters b_a, k_a, R_{in}, and R_f the Matlab function `ga` is exploited. This function employs the genetic algorithm to find the minimum of the provided cost function. This system was chosen to explore all possible combinations between the four variables and optimize them to solve the non-linear equation system. The options shown in Table 12 are utilized to balance the solution's accuracy and the computational effort. In particular, parallelization is highly suggested, if possible, to significantly reduce the computational time required for multiple simulations.

Population Size	Max Generations	UseParallel	Tolernaces
50	150	`true`	`1e-6`

Table 12: `ga` function parameters

The function `costFunction` solves the system of equations and computes the cost function as the integral of the absolute difference between the drag and the thrust in time, `trapz(t, abs(T-D))`. This method was chosen because of its simple implementation and compatibility with the genetic algorithm. The genetic algorithm is then performed, keeping a constant generation seed to have reproducibility and setting the `ode15s` tolerances to `1e-6`. The parameters shown in Table 14 are obtained.

b_a $[Nm/s]$	k_a $[N/s]$	R_{in} $[\Omega]$	R_f $[\Omega]$
1500.34	$1.56 \cdot 10^{-3}$	0.1794	$7.48 \cdot 10^{4}$

Table 13: Optimized parameters

Points 2 and 3

Due to the different temporal scales between the drag oscillations and the system dynamics, as well as the equations' non-linearity and the difference in the parameters' order of magnitude, the problem can be considered to be stiff. In order to solve the system Matlab's `ode15s` solver is chosen due to its accuracy for long simulation periods and its robustness in handling stiff problems. The tolerances are set to `1e-6` inside the cost function to decrease computational effort, while are set to `1e-9` in the main code to have more accurate results. The initial conditions are set to zero for every variable except for the valve displacement, which uses $x_v = x_{v,max}$ to simulate the valve closed at the beginning.

x_a [m]	v_a [m]	x_v [m]	v_v [m]	i [A]
0	0	$x_{v,max}$	0	0

Table 14: Initial Conditions

After an initial transient shown in Fig. 29, the thrust successfully aligns with the drag throughout the simulation, as seen in Fig. 28, demonstrating the effectiveness of the ACS in compensating the atmospheric drag and ensuring stability.

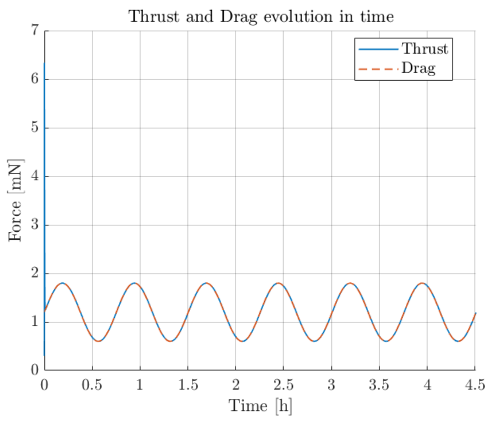

Figure 28: Thrust and drag behaviour

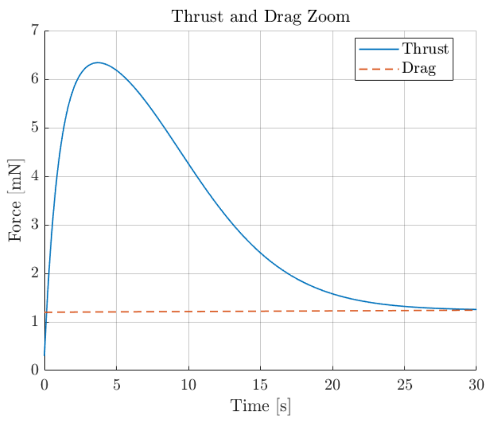

Figure 29: Zoom of the thrust

The accelerometer displacement x_a initially increases during the transient phase, as the net force between the drag and thrust causes a rapid rise in displacement. Subsequently, it oscillates following the periodic behaviour of the drag, as seen in Fig. 30, confirming the ACS system's response to dynamic variations. On the other hand, the accelerometer's velocity v_a, a significant initial peak is observed due to the transient response. After this phase, the velocity stabilizes at an almost zero value with small amplitude oscillations, as shown in Fig. 31.

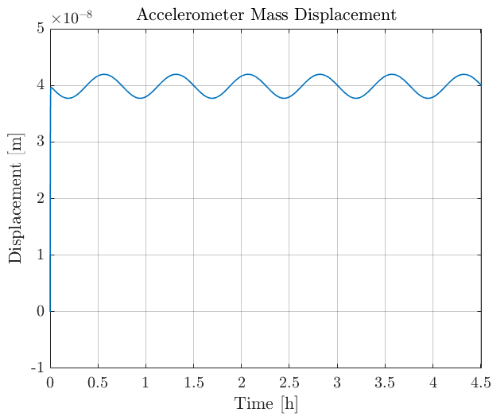

Figure 30: Accelerometer displacement

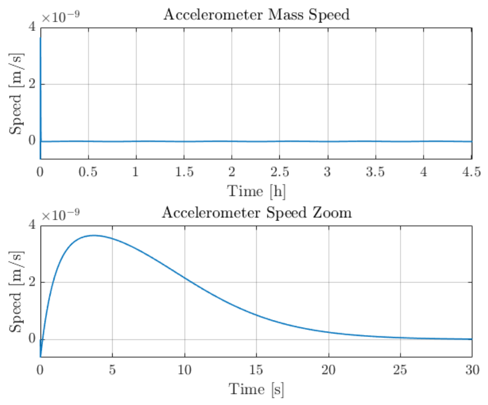

Figure 31: Accelerometer speed

Analogously, the solenoidal valve displacement initially decreases as the valve opens and then exhibits an oscillatory behaviour, as seen in Fig. 32. Consequently, the velocity shows a significant peak at the start of the simulation, followed by a stabilization close to zero, as shown in Fig. 33.

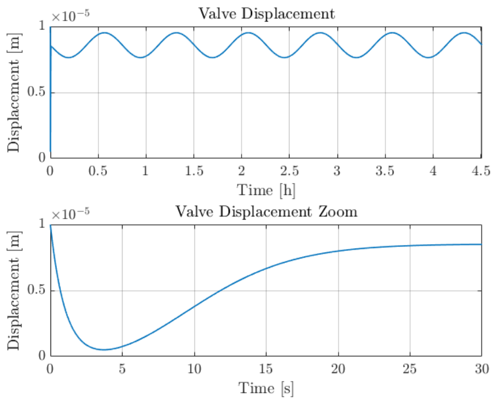

Figure 32: Valve displacement

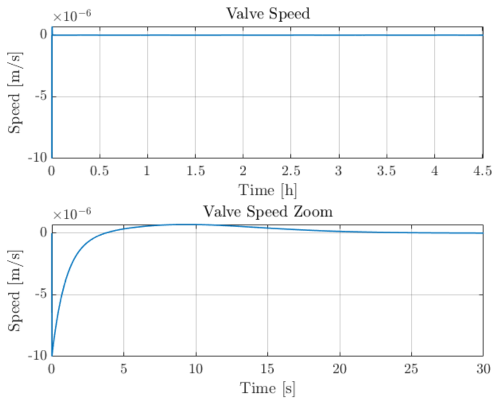

Figure 33: Valve speed

Fig. 34 shows the current passing in the solenoid, which exhibits a brief transient followed by regular oscillations synchronized with the modulated voltage. Finally, Fig. 35 shows how the solenoid valve orifice's area remains within the physical limitations expressed by Eq. (14). In particular, it must be noted that the behaviour of the area is opposite to that of the valve displacement, as expected looking at Eq. (13).

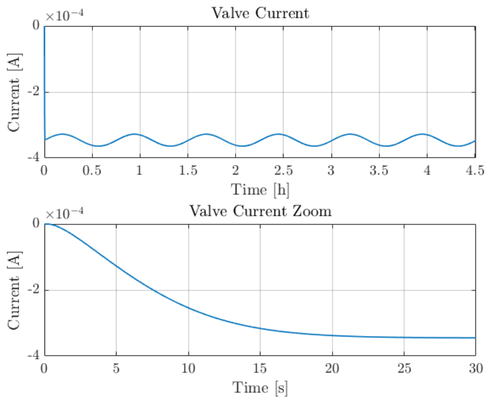

Figure 34: Valve current

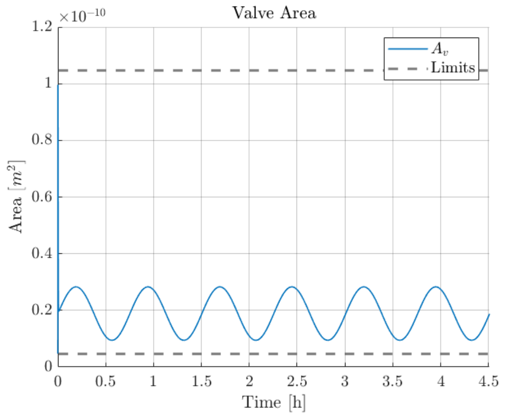

Figure 35: Orifice area

Part 2

The system is divided into its five main components, making it easier to build the acausal model in Simscape. The main components of the system are the accelerometer, the amplifier, the solenoidal valve, the thruster, and the block which computes the drag.

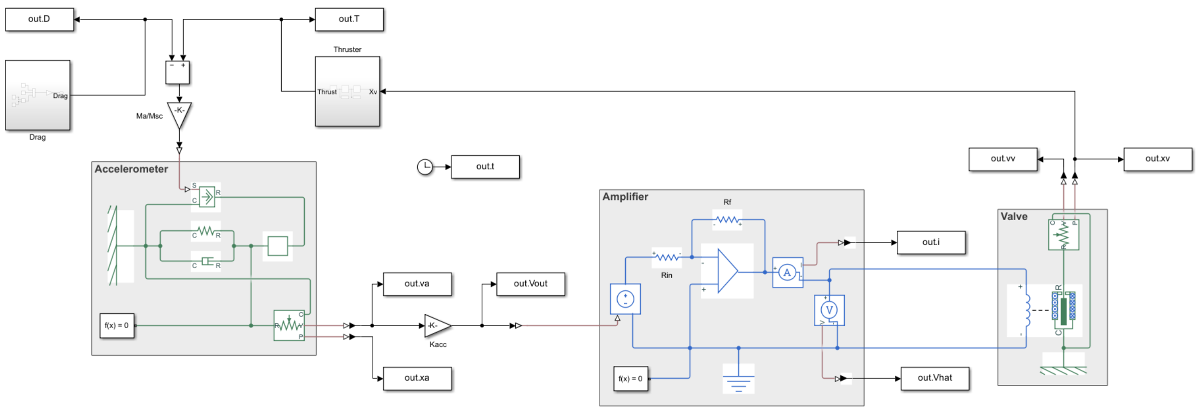

Figure 36: Acausal model in Simscape

The accelerometer is modeled using blocks from Simscape's Mechanical library. In particular the `Translational Spring`, `Translational Damper`, and `Mass` blocks were used to model the dynamics of the spring-mass-damper. A `Mechanical Translational Reference` block was used to ensure that the reference system is kept the same between all blocks and an `Ideal Force Source` block was used to transmit to the system the difference between the thrust and the drag, scaled by $^{m_a}/_{m_{sc}}$. An `Ideal Translational Motion Sensor` was used to measure the position x_a and the velocity v_a of the accelerometer mass, with its initial position set to zero. The velocity is then turned to voltage by multiplying it by the coefficient K_{acc}. The accelerometer block is shown in Fig. 37.

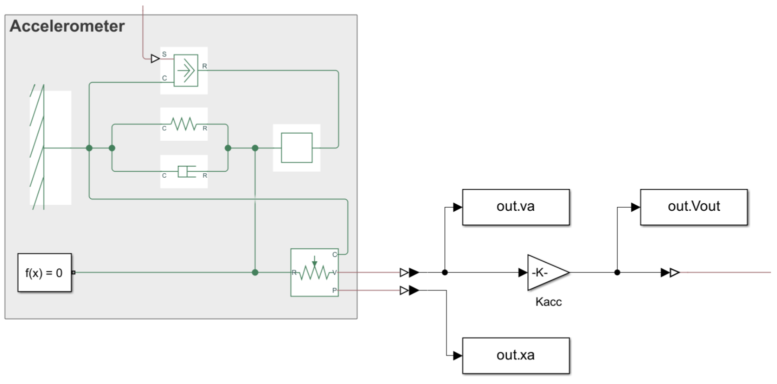

Figure 37: Accelerometer in Simscape

The amplifier and the thruster's valve are modeled through the use Simscape's Mechanical and Electrical libraries. In particular the voltage of the accelerometer is input in the system through the use of a `Controlled Voltage Source` and amplified with the use of two `Resistors` and a `Op-Amp` blocks, in the same configuration seen in Fig. 25. An `Electrical Reference` block was then used to ensure that the electrical circuit was grounded.

The valve was simply modeled using a `Solenoid` block, with the parameters reported in Table 11. The position x_v and velocity v_v of the valve is measured through a `Ideal Translational`

`Motion Sensor`. The amplifier and the valve models are shown in Fig. 38.

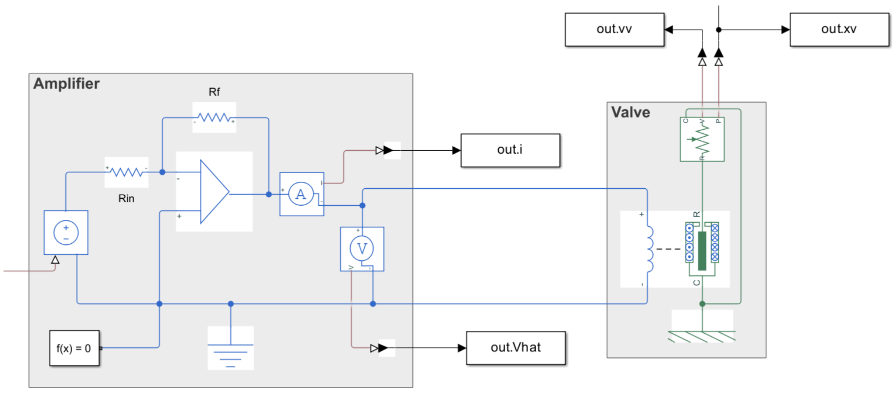

Figure 38: Amplifier and valve in Simscape

The thrust and drag of the system were calculated through the use of basic Simulink blocks and their difference was used as the force applied to the accelerometer.

The solver used to simulate the system employs the same configurations as the causal model, Matlab's `ode15s` solver with tolerances set to `1e-9`. The `Solver Configuration` blocks are implemented in the model to ensure that all blocks are set to their default configuration. The results of the simulation are reported in the figures below.

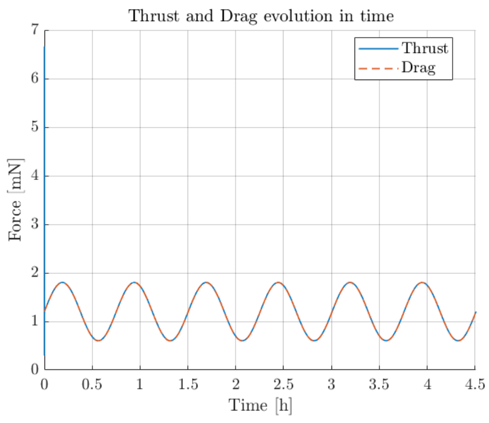

Figure 39: Thrust and drag behaviour

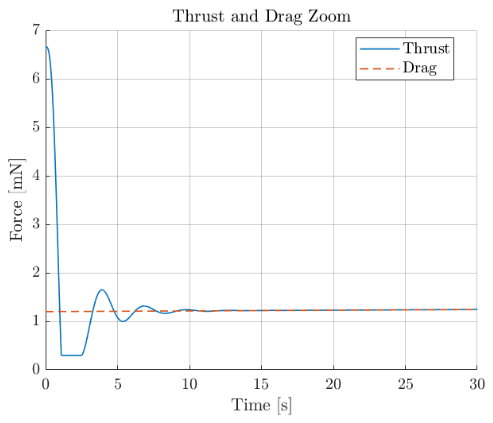

Figure 40: Zoom of the thrust

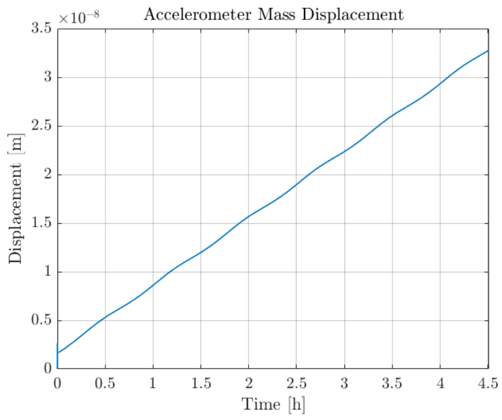

Figure 41: Accelerometer displacement

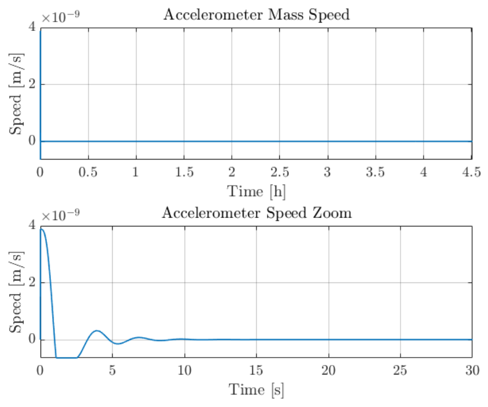

Figure 42: Accelerometer speed

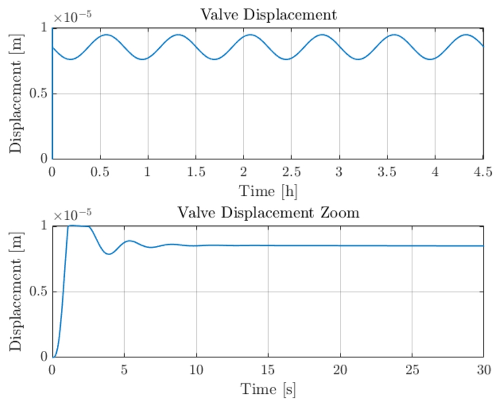

Figure 43: Valve displacement

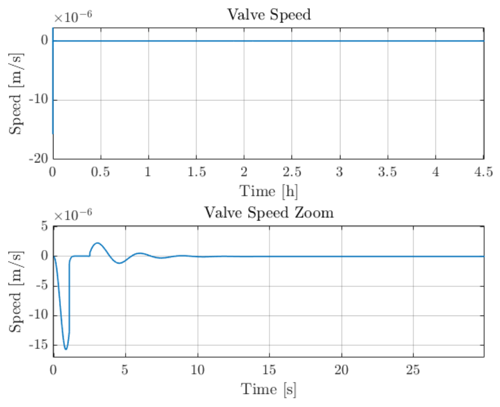

Figure 44: Valve speed

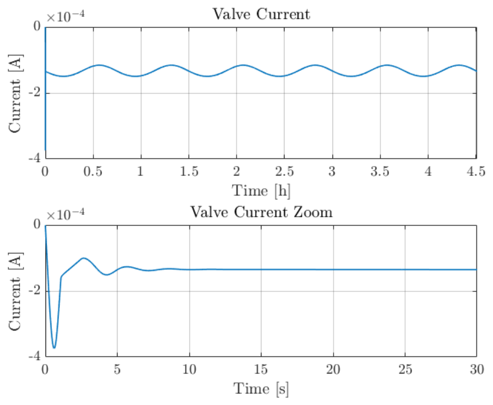

Figure 45: Valve current

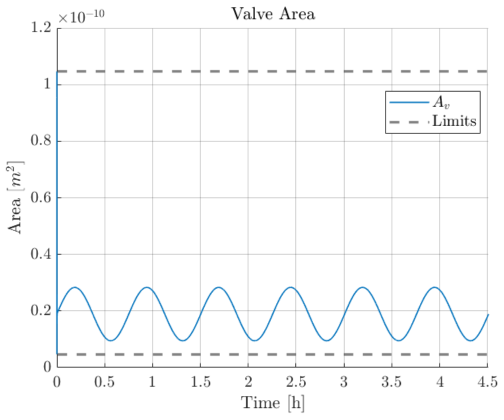

Figure 46: Orifice area

When the steady-state is achieved, the solutions from the acasual model are coherent with the casual model, confirming the validity of both implementations. However, noticeable discrepancies arise during the transient phases. These differences are likely due to a different approach in handling initial conditions between Matlab and Simscape. Indeed, while Matlab explicitly sets the initial states for the solver, Simscape imposes physical consistency through system constraints.

A significant discrepancy appears in the accelerometer displacement x_a and can be observed by confronting Fig. 41 and Fig. 30. In Simscape, a residual discrepancy between drag and thrust introduces a small net force, which leads to a small gradual displacement of the accelerometer.

	Mean Error [N]
Matlab	$4.602 \cdot 10^{-6}$
Simscape	$2.730 \cdot 10^{-6}$

Table 15: Mean error comparison

Table 15 compares the mean error of the ACS response to the drag between causal and acausal modeling. The mean error is computed as the absolute value of the difference between the thrust and the drag, removing the first thirty seconds to avoid discrepancies due to initial transients. It can be observed that the values of this error are comparable and this confirms the consistency of the results.

	Mean computational time [s]
Matlab	0.0104
Simulink	0.5277

Table 16: Computational time comparison

To ensure a reliable time comparison, the computational time of multiple simulations were computed for both the Matlab and Simscape models, through the use of Matlabs `tic` and `toc` functions. It was chosen to perform the simulation ten times as a good trade-off between statistical relevance and low computational effort. The Matlab model achieves a faster execution times due to its causal approach and its efficiency in handling ODEs. In contrast, the Simscape model requires significantly more computational effort as it solves a larger set of equations while ensuring physical consistency.

References

[1] R. Mukhiya, M. Garg, P. Gaikwad, et al. *Electrical equivalent modeling of MEMS differential capacitive accelerometer*. In: Microelectronics Journal. Vol. 99, pp. 104770, 2020. `https://doi.org/10.1016/j.mejo.2020.104770`