Sensitivity Analysis of Parameter Influence on Thermal Design in UPMSat 3

Study Case 3

MSc in Space Systems

Author: Inés Arauzo Andrés

Tutor: Ignacio Torralbo Gimeno
Javier Piqueras Carreño

Madrid, 9 de marzo de 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the context of thermal testing, the effectiveness of sensor placement plays an important role in ensuring, not only the accuracy and reliability of test results but also the significance of them. The motivation to improve thermal testing is highlighted by the substantial costs —both in terms of finances and time— associated with thermal testing procedures, which makes even more necessary to get useful results.

Traditional approaches to sensor positioning often fall short in defining thermal experiments due to the complexity of representing the physical processes and properties in a thermal mathematical model. Thus, an adequate parameter selection is key when trying to reduce a thermal mathematical model. This might seem obvious at first sight, but complex thermal mathematical models can have up to thousands of parameters and surpass the million of nodes, so the job of selecting the parameters that capture the physics of the problem for every charge case must not be underestimated.

Throughout this study case we have developed an efficient tool to identify the input parameters that have a significant influence on any of the possible outcomes. This tool has been used on 2 different thermal models in order to get a proper validation:

- A real FPGA model with 8 nodes that is used to test the method in a real, yet simple scenario.
- The UPMSat-3, a real satellite that is currently being developed by the Universidad Politécnica de Madrid. This model is used to test the method in a real scenario.

The results of this work can be applied, in general, to many fields, however the work has been focused on spacecraft and space instruments thermal modeling and testing, in the context of the Space sector. The examples used along the document are centered on space hardware

and for that reason, in the next section, the spacecraft thermal control topic is introduced with a brief summary of the particularities of the space environment

1.1 Spacecraft Thermal Control

The thermal design of a spacecraft is primarily influenced by the conditions it encounters during its mission in space. From a thermal perspective, the space environment is characterized by a vacuum, Solar radiation (both direct and reflected by nearby planets, known as albedo), and the infrared emission of celestial bodies.

The main driver of a spacecraft thermal design is the in-flight environment where it needs to operate. From a thermal point of view, the space environment is characterized by the vacuum, the incoming Solar radiation, both direct and reflected by a nearby planet (albedo), and the infrared emission of the planet. Because a spacecraft operates in a vacuum, the only possible thermal interaction between the spacecraft and its environment its through radiation. On an Earth orbit, solar irradiance is the main heat load, with a mean value of 1366 W/m2 and a seasonal variation of ±1.7% due to the eccentricity of the orbit of the Earth around the Sun. The solar irradiance value scales with the square of the distance to the Sun, and its spectrum can be modeled, from a thermal point of view, as a black body at some 5762 K, where 99 % of the spectral emissive power of the Sun lies in the range 0.15 to 10 µm wavelength.

1.1.1 Thermal mathematical modelling

When creating a thermal model for an analysis of something space-related, the most common method is the lumped parameters [?], [?], [?], which consist of discretizing the physical system to a finite set of nodes, each of those representing an isotherm tiny volume with some properties associated to itself (with the thermal capacity of the material being among them).

The nodes are interconnected between themselves by the lineal (conductive and convective -if possible-) $G_{L_{ij}}$ and radiative $G_{R_{ij}}$ thermal conductances. The thermal charges are summed up in Q_i (with i being the number of the node); within this term, the solar charge, the planetary albedo, the electrical dissipation of the payloads or the infrared earth emission are represented among others. Now, using the equation of energy balance,

$$C_i \frac{dT_i}{dt} = Q_i + \sum_{j=1}^{n} G_{Lij} \left(T_j - T_i\right) + \sum_{j=1}^{n} G_{Rij} \sigma \left(T_j^4 - T_i^4\right) \tag{1.1}$$

we get a system of ordinary differential equations that can be solved through numerical methods.

The values of the thermal conductances and capacitances are not always known, as they are usually a function of physical parameters such as geometry, pressure, torque, or surface finishing. There are several ways of calculating the lineal conductances [**?**], [**?**], but when using reduced thermal models (where the geometry has been really simplified) it is usually better to take these G_L as parameters. As for the radiative conductances, G_R, they come from the Geometrical Mathematical Model, the GMM, that is, when defining the geometry of the thermal model, the geometry is also detailed in order to get the external thermal charges and the view factors, which, with the different surface coatings give the corresponding G_R. While the calculation -or at least the estimation- of the G_L can be done analytically and/or experimentally, the G_R are usually obtained numerically through a Monte Carlo analysis [**?**], as it is quite complex even for relatively simple geometries. Most of the G_L and G_R are calculated through software, however, they usually come from parameters that have been approximated, which is why, in simple models these conductances are often taken as parameters.

1.1.2 Analaysis cases

According to the ECSS, when analyzing the thermal requirements of a space mission, one must evaluate the stationary cold and hot cases first, as they are the most extreme cases, and, if those are properly defined, the temperatures the system might reach will be enveloped by them. This makes the information of these cases really useful.

Furthermore, the fact that these cases are stationary, eliminates the time variance term in **??**, which means getting the thermal capacitances and its associated uncertainty out of the equation.

This is why this project only focuses on the stationary cases -beginning with the hot and cold charge cases- each defined by a set of boundary conditions BC_K.

Chapter 2

Problem definition

As presented on **??**, parameter selection is a key step when creating and/or using models in almost any field, more so, in the space sector where everything has an extra cost associated. However, when talking more specifically about the thermal control for space missions, there are not so many techniques to extract the parameters that define the physics of the problem with most accuracy and observability. This is due to the fact that it has usually been overlooked, never optimizing the amount of information extracted from testing and mission monitoring. Thus, among this chapter the developed method is presented, along with the requirements the system must fulfil in order to use it.

2.1 Algorithm definition

A scheme of the method is developed on **??**. The method has been implemented on Python [**?**] using the Pycanha package [**?**], a thermal analysis tool developed by Dr.Piqueras that joins a Python front-end with a C+ background in order to get a user-friendly, yet fast thermal solver. All the functions and features created along this project have been implemented as libraries of this package, and can now be obtained open source on GitHub. Apart from pycanha, other commercial libraries such as NumPy [**?**] and SciPy [**?**] -For mathematical purposes- and Matplotlib [**?**] have been used. Throughout the following sections each step of the algorithm is briefly explained.

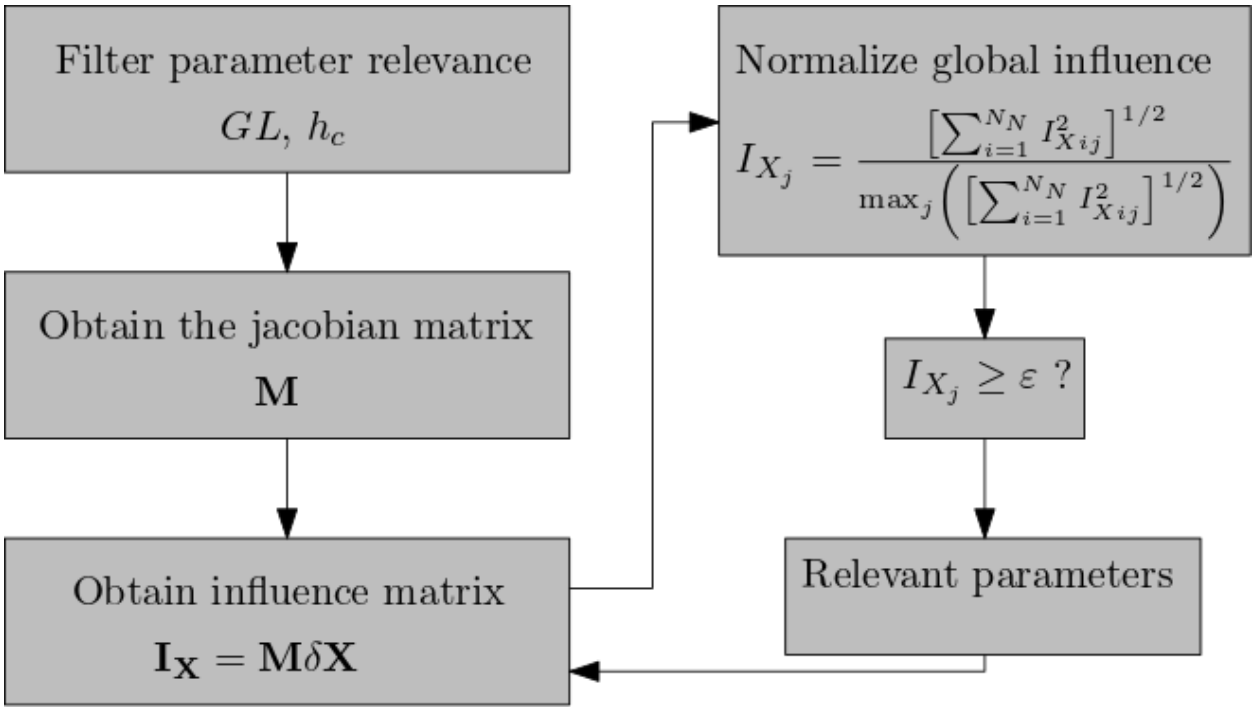

Figure 2.1: Flow diagram of the method algorithm

2.2 Data Read

The first step of the method consist of the acquisition of data, whether it comes from an external model or if it has been developed directly with Pycanha. This second option is way easier, as the data read would be as easy as reading the TMM object, and all the parameters, associated formulas, geometrical definition etc. would be known. However, as pycanha is still on development it is quite difficult to create any kind of complex system, due to the lack of a graphic interface to create the geometry.

In order to make the package more usable with regard to bigger projects, the compatibility with the most extended spacecraft thermal analysis tool, ESATAN-TMS [**?**], is provided within the package. The files that pycanha uses to replicate the model created in esatan are the .TMD and some of the .data files, which must be exported separately.

The .TMD files are binary coded files that result from each charge case, and they contain all the necessary information to run a TMM. These files give directly the lineal and radiative coupling matrices K and R, the thermal capacities C and the boundary conditions Q, and with that the system of equations presented on **??**, can be solved.

However, this just gives the numerical result, eliminating all the parameters and formulas

that are associated to the calculations of these matrices, which would not be useful to make any kind of analysis, so they have to be overwritten. Thus, after reading the .TMD, the .CONSTANTS.data and .LOCALS.data are read first and then, once the TMM has the parameters, the formulas are read in .COUPLINGS.data.

To summarize:

- Create an empty TMM object.
- Read the .TMD file with all the numerical data.
- Read the parameters files.
- Overwrite de conductances gotten from the .TMD with the formulas associated to the arameters.

2.3 Manual filter

Once all the data has been read, the engineer has to choose an initial set of parameters to begin the analysis with. This task must be done manually as it is not a matter of influence or sensitivity but the kind of parameters one wants to study. It takes not only an important engineering knowledge but also a great understanding of the model itself to be able to select an optimum set of parameters.

It will be further explained on latter chapters, but these choices are way easier to make if the model is developed by the same person who makes the analysis, as the parametrization will be purposefully done in a way that simplifies this task; this is the case for the 2 first cases that are analyzed, where due to this (and to the simpliccity of the models), this step can be skipped. However, in the third case, the UPMSat3 model is way more complex than the other two example cases and furthermore, it has been developed by a different person who had no idea of the purpose of this parametrization, which highlights the importance of this step.

However, for the proposed analysis, the materials and coatings do not change, and thus, neither does the thermo-optical properties, the densities or the conductivities associated to them. The radiative conductances were not even parametrized in first place, but as the thermo-optical properties do not change and neither does the geometry, they will not be part of this initial set. Other parameters such as TIME or PERIOD do not variate either so can also be excluded. Furthermore, as previously mentioned, the study does not take into

account the time variation, it is stationary, and thus the thermal capacities, have been also excluded.

Taking this filter into account, the final list must mainly comprehend lineal conductances G_L, or alternatively the h_c.

2.4 Jacobian and influence matrix calculation

The algorithm uses the influence matrix to find which parameters have most influence on which nodes. In order to do that, it is necessary to calculate the Jacobian or Sensitivity matrix $\boldsymbol{M_t}$, defined by the derivatives of the temperature T_i at each node with respect to each parameter p_j,

$$\boldsymbol{M_t} = \frac{\partial T_i}{\partial p_j} = \begin{bmatrix} \frac{\partial T_1}{\partial p_1} & \frac{\partial T_1}{\partial p_2} & \cdots & \frac{\partial T_1}{\partial p_{N_P}} \\ \cdots & \cdots & & \cdots \\ \frac{\partial T_{N_N}}{\partial p_1} & \frac{\partial T_{N_N}}{\partial p_2} & \cdots & \frac{\partial T_{N_N}}{\partial p_{N_P}} \end{bmatrix}. \tag{2.1}$$

For really simple models, the Jacobian can be obtained analytically, but in general there is no analytical solution, and it must be computed numerically. Thus, the approach has been to approximate these derivatives by central finite differences[1]. The central difference approximation for the partial derivative $\frac{\partial T_i}{\partial p_j}$ is given by:

$$\frac{\partial T_i}{\partial p_j} \approx \frac{T_i(p_j + \Delta p_j) - T_i(p_j - \Delta p_j)}{2\Delta p_j} \tag{2.2}$$

Here, Δp_j is a small perturbation applied to the parameter p_j. The central difference method is chosen because it provides a more accurate approximation of the derivative compared to forward or backward differences, especially when dealing with numerical computations.

To determine the appropriate perturbation Δp_j for each parameter p_j,the following expression is used:

$$\Delta p_j = p_j \cdot \sqrt{\varepsilon} \tag{2.3}$$

where ε is the machine epsilon for floating-point arithmetic, meaning the upper bound on the relative error due to rounding in floating-point calculations. In Python, this can be obtained using `epsilon = np.finfo(float).eps`

The perturbation Δp_j is chosen as a small fraction of the parameter p_j itself, scaled by

[1]The options of forward and backward differences have also been implemented

the square root of the machine epsilon. This ensures that the perturbation is neither too large (which could lead to inaccurate approximations due to nonlinear effects) nor too small (which could result in numerical inaccuracies due to round-off errors).

To summarize, the computation of the Jacobian matrix $\boldsymbol{M_t}$ using central differences can be written as:

$$(\boldsymbol{M_t})_{ij} = \frac{T_i(p_j + \Delta p_j) - T_i(p_j - \Delta p_j)}{2\Delta p_j}, \tag{2.4}$$

with

$$\Delta p_j = p_j \cdot \sqrt{\varepsilon}, \tag{2.5}$$

ensuring a balanced trade-off between accuracy and numerical stability in the computation of the derivatives.

However, in order to choose the most adequate parameters to determine the reduced model, the matrix of influence $\mathbf{I_X}$ is defined below:

$$\mathbf{I_X} = \begin{bmatrix} \frac{\partial T_1}{\partial p_1}\delta p_1 & \frac{\partial T_1}{\partial p_2}\delta p_2 & \dots & \frac{\partial T_1}{\partial p_{N_P}}\delta p_{N_P} \\ \dots & \dots & & \dots \\ \frac{\partial T_{N_N}}{\partial p_1}\delta p_1 & \frac{\partial T_{N_N}}{\partial p_2}\delta p_2 & \dots & \frac{\partial T_{N_N}}{\partial p_{N_P}}\delta p_{N_P} \end{bmatrix} = \mathbf{M}\delta\boldsymbol{P} \tag{2.6}$$

where $\mathbf{M_t}$ is the jacobian or sensibility matrix defined in the previous section, and $\delta\boldsymbol{P}$ is a vector containing the allowable variation of each parameter within the design. In the influence matrix $\mathbf{I_X}$ each column represents the temperature variation of the nodes that would be generated by a deviation on the parameter δp_i. Therefore, the elements of this matrix have dimensions of temperature, showing the effect of every parameter in the model, which would not be possible using the jacobian matrix directly.

Preliminarly, the parameter deviation $\delta\boldsymbol{P}$ was taken as a 10% deviation of the parameters, as at first the real allowable deviation was not available. It is interesting to see the change in the results between both of these deviations (see Chapter 5) as it shows the importance of using the influence matrix instead of just the jacobian. Representations of influence matrices will be seen and analyzed in latter chapters, for example in **??**

2.5 Normalization and importance filter

Each influence matrix shows the effect of varying each one of the parameters in terms of temperature, giving an easy visible relation of the importance of each parameter and where it can be measured. However, in order to choose which parameters to continue with, the influence is normalized as follows:

$$I_{X_j} = \frac{\left[\sum_{i=1}^{N_N} I_{Xij}^2\right]^{1/2}}{max_j\left(\left[\sum_{i=1}^{N_N} I_{Xij}^2\right]^{1/2}\right)}, \tag{2.7}$$

where I_{Xij} are the values of the influence matrix $\boldsymbol{I_X}$. This way it is easier to see the impact each parameter has on the model, and then choose the most optimal ones to run the tests.

With this in mind, a second filter is set so that if the normalized global influence of the parameters is below a threshold ε_{gi}, they are eliminated. This process should retain the most important parameters, eliminating first the ones that are not useful for further analysis, and now the ones we can not measure adequately, leaving now one last step to ensure that all the parameters are linearly independent.

2.6 Linear dependence filter

With the influence matrix properly calculated, one last filter is applied. Within this step, the intention is to eliminate the parameters that are linearly dependent among themselves. In order to see this dependence, the Pearson correlation coefficient matrix is used. These coefficients measure the linear correlation between two variables, returning a value between -1 and +1. The closest the value of the coefficient is to +1 or -1, the more linearly dependent those values are, either directly (+1) or inversely (-1).

For a dataset with p variables (in this case the parameters, disposed as columns of the influence matrix), the Pearson correlation coefficient matrix $\mathbf{R}$ is a $p \times p$ matrix where each element r_{ij} is the Pearson correlation coefficient between the i-th and j-th variables.

Given a data matrix $\mathbf{M}$ of size $n \times p$, where n is the number of observations (in this case nodes), the Pearson correlation coefficient matrix can be computed as follows:

First, the data matrix $\mathbf{M}$ is standardized to have zero mean and unit variance:

$$\mathbf{Z} = \frac{\boldsymbol{I_X} - \mu}{\sigma},$$

where μ is a vector of means and σ is a vector of standard deviations for each parameter.

Then, the Pearson correlation coefficient matrix $\mathbf{R}$ is calculated as:

$$\mathbf{R} = \frac{1}{n-1}\mathbf{Z}^\top \mathbf{Z},$$

which, in matrix notation, can be written as:

$$\mathbf{R} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1p} \\ r_{21} & 1 & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \cdots & 1 \end{bmatrix},$$

where each r_{ij} is computed using the standardized data matrix $\mathbf{Z}$.

The Pearson correlation coefficient r_{ij} between the i-th and j-th variables can also be expressed in terms of the dot product of their standardized values:

$$r_{ij} = \frac{\sum_{k=1}^{n}(z_{ik}z_{jk})}{n-1},$$

where z_{ik} and z_{jk} are the k-th observations of the i-th and j-th standardized variables, respectively.

Taking the absolute value of this matrix $\mid \mathbf{R} \mid$, we have a diagonal and symmetrical matrix with values going from 0 to +1, (with the main diagonal filled with ones) where the closest a value is to 1, the more linearly related are the corresponding parameters. Thus, a correlation threshold ε_{LD} is set around 0.997, and whenever any pair of parameters is higher than the threshold, one parameter of the pair is eliminated.

After that, the columns corresponding to those parameters p_{LD} are eliminated from the Influence matrix, resulting in a reduced matrix I_{Xli}, standing for linearly independent.

2.7 FPGA model

An FPGA, or Field-Programmable Gate Array, is a type of electronic device that can be programmed by the user after it has been manufactured. This means that an FPGA can be configured to perform a wide variety of tasks, making it very versatile and very useful for space missions. However, they usually have a simple and very similar disposition of components, which makes them easy to thermally model.

In this case, the thermal model disposition is based on the one in [**?**], and it has been represented on **??**

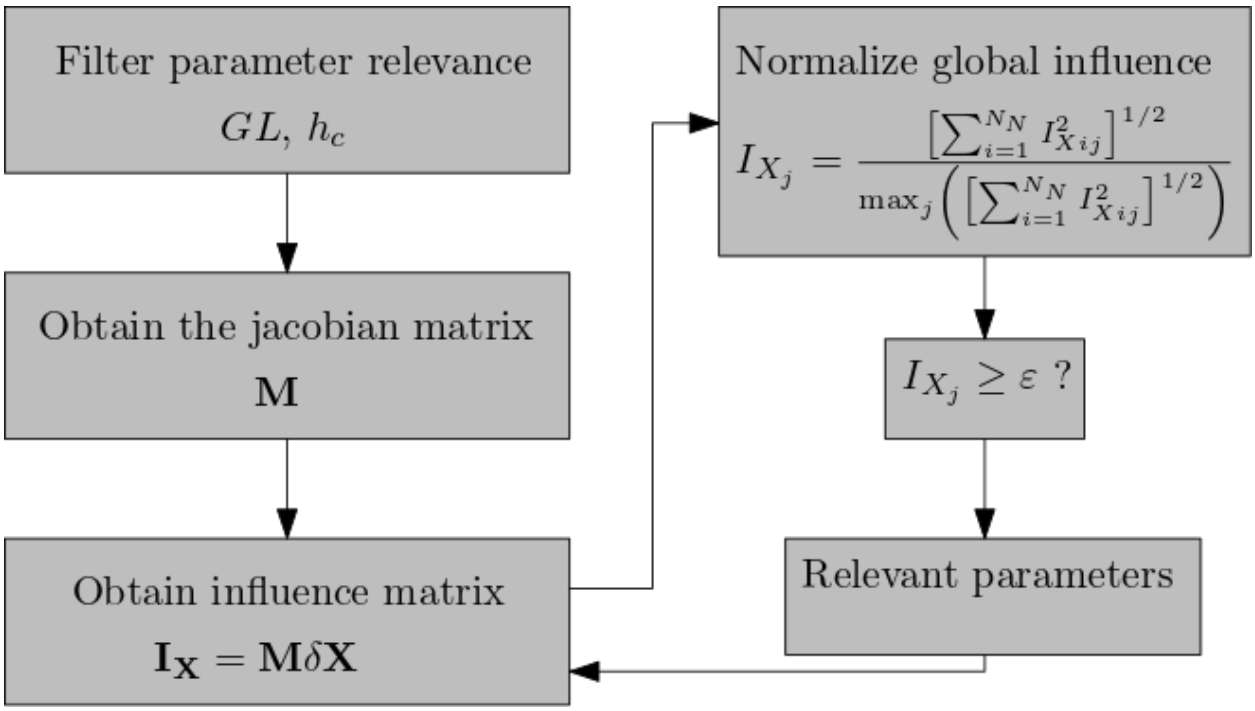

Figure 2.2

Here, in the following table, the nodes are named in order to explain and clarify the model and the dissipation of each of them is also defined,

Table 2.1: Node definition for the FPGA model.

Node Number	**Structure Part**	$\boldsymbol{Q_I}$ [W]
1	Board	3
2	Junction	0
3	Case	0
4	Heat sink	0
5	Strap joint	0
6	Structure A	0
7	PCB frame	0
9	Environment	0

As can be seen in **??**, the radiation is considered at two nodes only; in the Board one, since the temperature difference with the environment reaches its peak here, and in node 4, as

the main purpose of the heat sink is to radiate heat to the environment. The rest of the structure is radiatively ignored due to the small size of the visible areas and the simplicity of the model. Apart from this, the environment temperature was set to 293.15 K.

Having been above-mentioned, this model was directly developed with the pycanha package, with the main purpose of using this method, which is why the parameters that define the model are all G_Ls. Thus, the first step of manually selecting these conductances can be skipped. The parameters are presented in **??** below:

Table 2.2: Parameter definition for the FPGA model.

Parameter	**Description**	**Value**[W]
GL_{12}	Conductance between the board and the junction	0.5
GL_{23}	Conductance between the junction and the case	2
GL_{34}	Conductance between the case and the Heat sink	0.2
GL_{45}	Conductance between the Heat sink and the thermal strap joint	0.15
GL_{56}	Conductance between the thermal strap joint and the structure	0.15
GL_{17}	Conductance between the Board and the Board frame	0.4
GL_{76}	Conductance between the Board frame and the Structure	0.4

Again, as the initial list does not compel any extra parameter (optical properties, conductivities, thermal capacities etc.) the initial list of parameters is the one in **??**. Thus, we can proceed to calculate the jacobian matrix.

The Jacobian matrix result is presented in

$$\boldsymbol{J} = \begin{bmatrix} 7.63 \cdot 10^{-6} & -2.37 \cdot 10^{0} & -1.36 \cdot 10^{0} & -1.18 \cdot 10^{0} & -1.58 \cdot 10^{1} & -8.39 \cdot 10^{-4} & -3.24 \cdot 10^{-4} \\ 0.00 \cdot 10^{0} & 2.64 \cdot 10^{0} & -1.00 \cdot 10^{0} & -1.66 \cdot 10^{0} & -2.21 \cdot 10^{1} & -1.22 \cdot 10^{-3} & -4.67 \cdot 10^{-4} \\ 0.00 \cdot 10^{0} & 2.41 \cdot 10^{0} & -9.13 \cdot 10^{-1} & 1.20 \cdot 10^{0} & -2.37 \cdot 10^{1} & -1.30 \cdot 10^{-3} & -4.96 \cdot 10^{-4} \\ 0.00 \cdot 10^{0} & 4.11 \cdot 10^{-2} & -1.56 \cdot 10^{-2} & 2.06 \cdot 10^{-2} & 2.74 \cdot 10^{-1} & -2.16 \cdot 10^{-3} & -8.30 \cdot 10^{-4} \\ 7.63 \cdot 10^{-6} & 2.05 \cdot 10^{-2} & -7.79 \cdot 10^{-3} & 1.03 \cdot 10^{-2} & 1.37 \cdot 10^{-1} & 1.71 \cdot 10^{-1} & -6.51 \cdot 10^{-2} \\ 7.63 \cdot 10^{-6} & 9.54 \cdot 10^{-6} & 3.81 \cdot 10^{-6} & 5.72 \cdot 10^{-5} & 5.09 \cdot 10^{-5} & 2.54 \cdot 10^{-5} & 0.00 \cdot 10^{0} \\ 7.63 \cdot 10^{-6} & -1.18 \cdot 10^{0} & 4.49 \cdot 10^{-1} & -5.92 \cdot 10^{-1} & -7.89 \cdot 10^{0} & -4.07 \cdot 10^{-4} & -1.62 \cdot 10^{-4} \end{bmatrix}, \tag{2.8}$$

this matrix does not really throw us much information, but following the algorithm, the Jacobian matrix is used to obtain the influence matrix. In this case, the deviation used to compute this matrix was taken from [**?**], and it can be seen on **??**

Table 2.3: Parameter deviation for the FPGA parameters

Parameter number	Name	Deviation
1	GL_{12}	0.05
2	GL_{23}	0.04
3	GL_{34}	0.20
4	GL_{45}	0.02
5	GL_{56}	0.01
6	GL_{17}	0.01
7	GL_{76}	0.04

The resulting influence matrix is:

$$\boldsymbol{I_X} = \begin{bmatrix} 2.7249 \cdot 10^{-1} & 1.7781 \cdot 10^{-1} & 3.8147 \cdot 10^{-7} & 2.4882 \cdot 10^{-1} & 3.5786 \cdot 10^{-2} & 2.7247 \cdot 10^{-1} & 2.7247 \cdot 10^{-1} \\ 2.0065 \cdot 10^{-1} & 3.0645 \cdot 10^{-1} & 7.6294 \cdot 10^{-7} & 1.6752 \cdot 10^{-1} & 1.3073 \cdot 10^{-1} & 2.0064 \cdot 10^{-1} & 2.0064 \cdot 10^{-1} \\ 1.8270 \cdot 10^{-1} & 2.7902 \cdot 10^{-1} & 7.6294 \cdot 10^{-7} & 2.0678 \cdot 10^{-1} & 1.7235 \cdot 10^{-1} & 1.8268 \cdot 10^{-1} & 1.8268 \cdot 10^{-1} \\ 3.1155 \cdot 10^{-3} & 4.7585 \cdot 10^{-3} & 7.6294 \cdot 10^{-7} & 3.5278 \cdot 10^{-3} & 7.2231 \cdot 10^{-3} & 3.0830 \cdot 10^{-3} & 3.0823 \cdot 10^{-3} \\ 1.5579 \cdot 10^{-3} & 2.3796 \cdot 10^{-3} & 7.6294 \cdot 10^{-7} & 1.7647 \cdot 10^{-3} & 3.6118 \cdot 10^{-3} & 4.1271 \cdot 10^{-3} & 1.0448 \cdot 10^{-3} \\ 3.8147 \cdot 10^{-7} & 0 \cdot 10^{0} & 3.8147 \cdot 10^{-7} & 7.6294 \cdot 10^{-7} & 3.8147 \cdot 10^{-7} & 0 \cdot 10^{0} & 3.8147 \cdot 10^{-7} \\ 3.6228 \cdot 10^{-1} & 3.1494 \cdot 10^{-1} & 4.5207 \cdot 10^{-1} & 3.5044 \cdot 10^{-1} & 2.4393 \cdot 10^{-1} & 3.6227 \cdot 10^{-1} & 3.6227 \cdot 10^{-1} \end{bmatrix}, \tag{2.9}$$

which is graphically presented in **??**

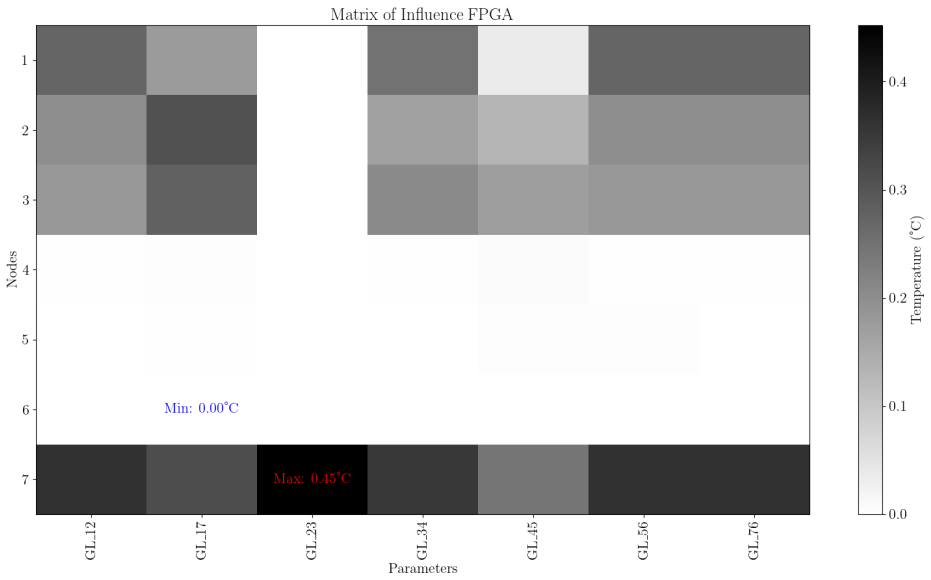

Figure 2.3: Influence matrix of the FPGA

In **??** it can be seen that the columns corresponding to parameters GL_12, GL_56 and 76 look quite similar, due to the fact that nodes 5 and 7 are unnecessary, as they are not dissipating any heat, nor radiatively nor conductively.

The next step is to see the normalized influence of the parameters, which can be seen in **??**. Setting the influence threshold on $\varepsilon_I = 0.1$, in this case all the parameters are way aboveit, possibly due to the fact that being such a simple model it is difficult for any parameter to not be representative.

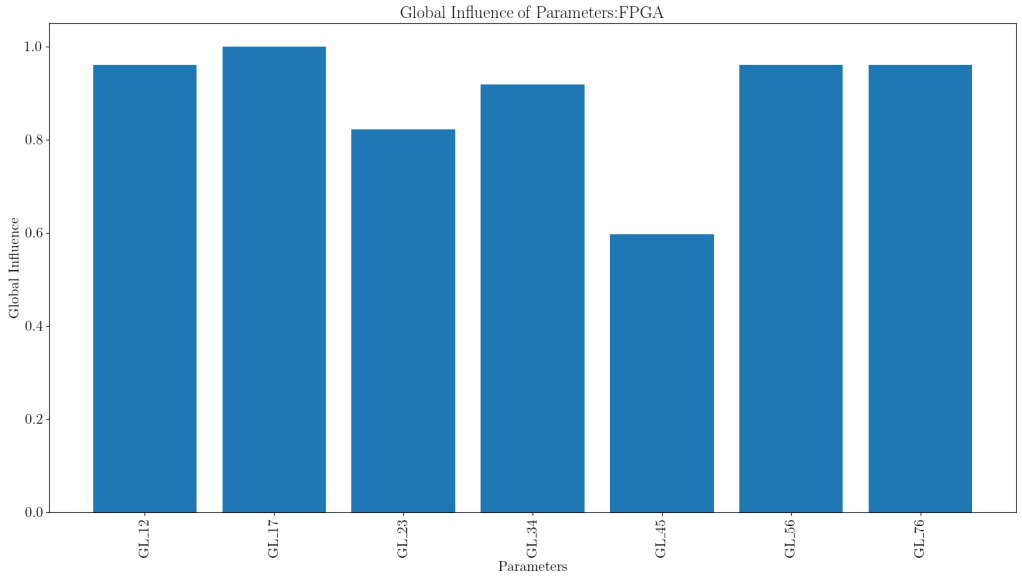

Figure 2.4: Normalized global influence of the parameters of the FPGA

Finally, following the procedure from **??**, one can calculate the Pearson Correlation Coefficients Matrix, getting the following matrix:

$$\boldsymbol{R} = \begin{bmatrix} 1.0 & -6.9058 \cdot 10^{-1} & 2.1059 \cdot 10^{-1} & -2.0384 \cdot 10^{-1} & 2.8079 \cdot 10^{-1} & 3.8368 \cdot 10^{-1} & -3.7221 \cdot 10^{-1} \\ -6.9058 \cdot 10^{-1} & 1.0 & -2.1383 \cdot 10^{-1} & 2.4005 \cdot 10^{-1} & -4.5252 \cdot 10^{-1} & -4.6117 \cdot 10^{-2} & 3.8611 \cdot 10^{-2} \\ 2.1059 \cdot 10^{-1} & -2.1383 \cdot 10^{-1} & 1.0 & 2.8671 \cdot 10^{-1} & 7.9104 \cdot 10^{-1} & 2.2341 \cdot 10^{-1} & -2.1606 \cdot 10^{-1} \\ -2.0384 \cdot 10^{-1} & 2.4005 \cdot 10^{-1} & 2.8671 \cdot 10^{-1} & 1.0 & 1.9556 \cdot 10^{-1} & 1.3120 \cdot 10^{-1} & -1.3233 \cdot 10^{-1} \\ 2.8079 \cdot 10^{-1} & -4.5252 \cdot 10^{-1} & 7.9104 \cdot 10^{-1} & 1.9556 \cdot 10^{-1} & 1.0 & 3.4234 \cdot 10^{-1} & -3.3551 \cdot 10^{-1} \\ 3.8368 \cdot 10^{-1} & -4.6117 \cdot 10^{-2} & 2.2341 \cdot 10^{-1} & 1.3120 \cdot 10^{-1} & 3.4234 \cdot 10^{-1} & 1.0 & -9.9971 \cdot 10^{-1} \\ -3.7221 \cdot 10^{-1} & 3.8611 \cdot 10^{-2} & -2.1606 \cdot 10^{-1} & -1.3233 \cdot 10^{-1} & -3.3551 \cdot 10^{-1} & -9.9971 \cdot 10^{-1} & 1.0 \end{bmatrix}, \tag{2.10}$$

where it can be appreciated that the values of r_{57}, r_{12} and r_{75} are above the ε_{LD}, thus making these parameters linearly dependent. We have chosen to retain only GL_{12}, due to the physical importance of that parameter.

Thus, the parameters that are chosen to study the FPGA model are:

Table 2.4: Final Parameters chosen for the FPGA model

Parameter number	**Name**	**Value**
1	GL_{12}	0.5
2	GL_{23}	0.4
3	GL_{34}	2.0
4	GL_{45}	0.2
6	GL_{17}	0.1

Take into account that even though this is a really simple model and this process here looks unnecessary, as there are just 9 initial parameters, this is just an explanation; in the next chapter we will be analyzing a real complex model and the importance of following a standard procedure will come clear.

Chapter 3

Application to the UPMSat3

3.1 Introduction

The UPMSat-3 is a university satellite project led by the "Instituto Universitario de Microgravedad Ignacio Da Riva" (IDR/UPM), a research institute from the Universidad Politécnica de Madrid (UPM). It is being developed in collaboration with the Real-Time Systems research group from UPM (STRAST, Sistemas de Tiempo Real y Arquitectura de Servicios Telemáticos). The UPMSat-3 is characterized as a 12U CubeSat with 0.2 × 0.2 × 0.34 cm dimensions. The satellite is scheduled for launch in mid-2024 aboard the Spectrum launcher from ISAR Aerospace.

The primary mission objective is to serve as an in-orbit demonstrator and qualification of different payloads and technologies in space, such as the demonstrators of the UC3M (Universidad Carlos III de Madrid), or the instruments from the company Hydra Space Systems.

This satellite is a predecessor of another two missions, the UPMSat-2 and the UPMSat, two larger satellites also developed mainly by the IDR/UPM. These satellites were caracterized as small-sats, as they were around 0.5 x0.5x 0.6 m each and rounded the 50 kg. However, the mission they were made to accomplish was almost the same: scientific in-orbit demonstrators that to qualify different payloads in a sun-synchronous LEO orbit.

3.2 Thermal Design of the UPMSat3

The thermal design of the UPMSat3 has also been developed in the IDR/UPM and a thermal model has been created to simulate the thermal behavior of the satellite.

3.2.1 Geometrical and thermal mathematical model

The thermal model is composed of 3 trays, each one with different instruments and payloads. The GMM is shown in Figure **??** while the interior of the model can be seen in Figure **??**.

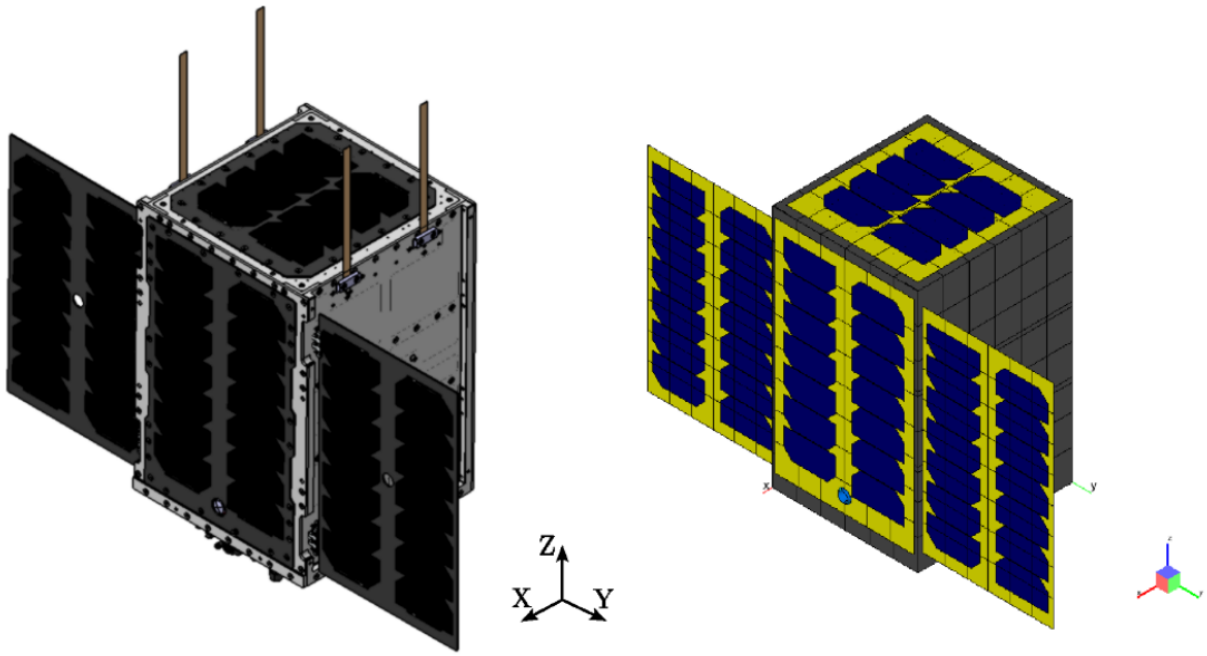

Figure 3.1: 3D and ESATAN-TMS models.

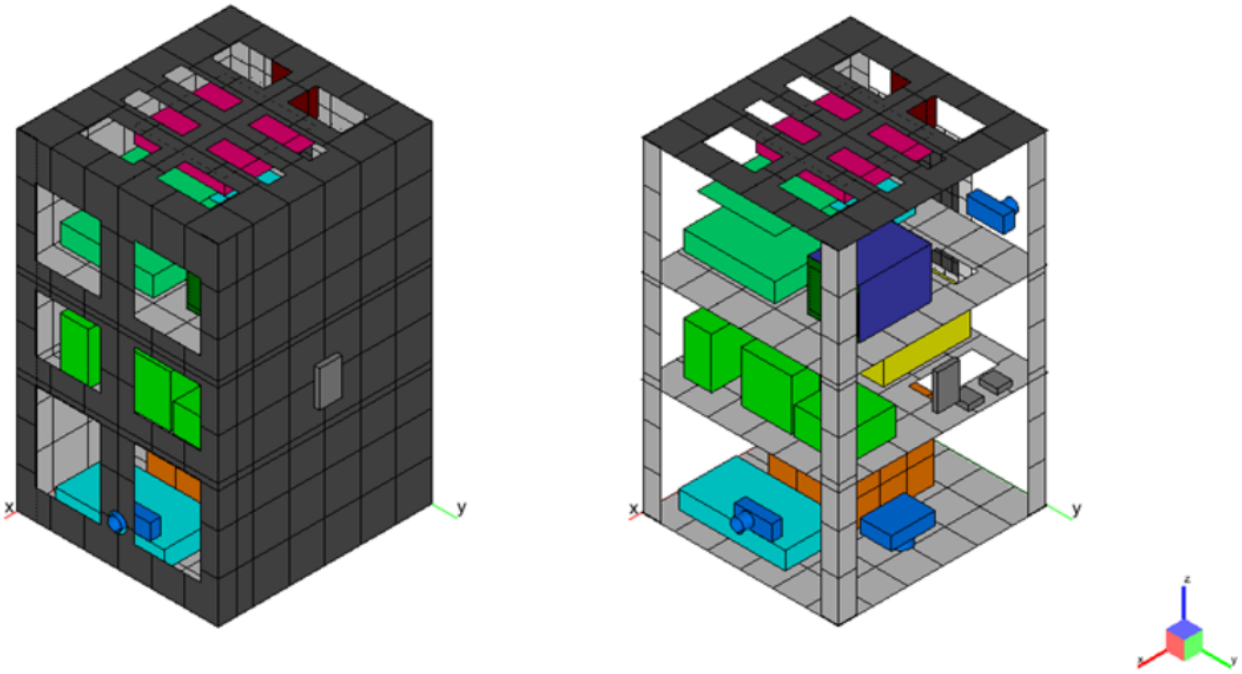

Figure 3.2: Structure, instruments and payloads.

The geometry of the model is defined by 1151 nodes (geometrical and non geometrical). The properties that define the model have been fully parametrized, defining each bulk material by its thermal conductivity, specific heat and density, and each surface material by its emissivity and absorptivity. These properties can be seen in **??**, and they have been taken from [**?**], where the distribution of the materials and/or coatings can be found. For the current analysis this will not be necessary, as the parameters we are focusing in are the lineal conductances.

3.2.2 Mission and charge cases

With respect to the mission and environment the satellite will suffer, the UPMSat 3 will describe a sun-synchronous orbit whose parameters are presented in **??**. A representation of this orbit in ESATAN-TMS can be seen in **??**.

Table 3.1: Orbital parameters UPMSat-3 mission

Orbit parameters	**Value**
Semimajor axis [km]	6928
Eccentricity	0.001
Inclination [deg]	97.405
RAAN [deg]	79.812
Argument of the Perigee [deg]	0.0
True anomaly [deg]	0.0
Primary pointing vector (Zenit)	$[0.985, 0.0, -0.174]$
Secondary pointing vector (Normal to orbit)	$[0.174, 0.0, 0.985]$

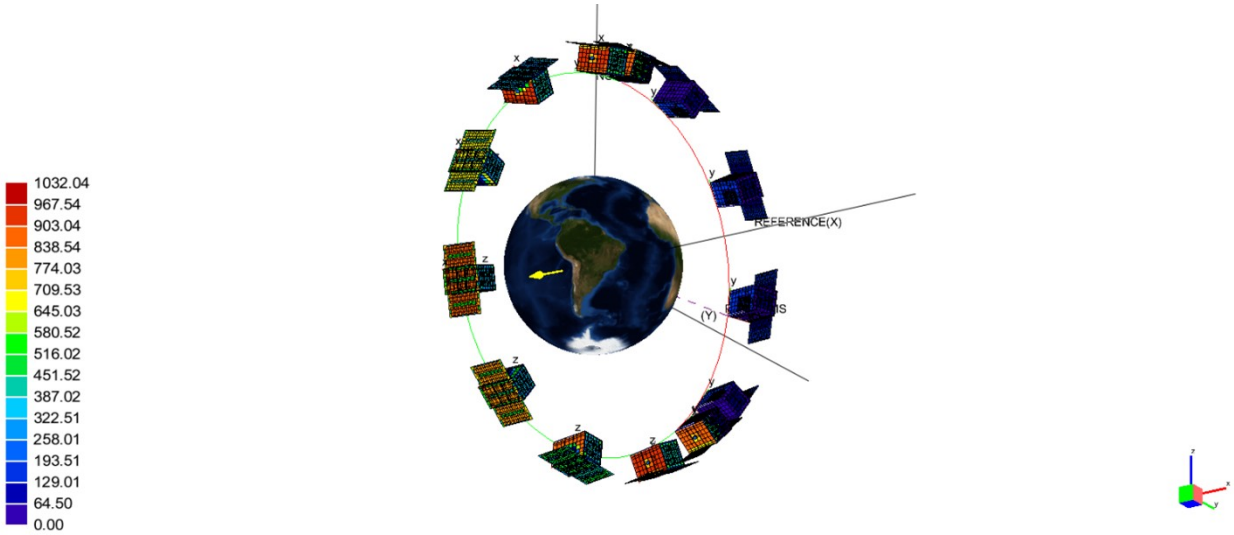

Figure 3.3: Orbit representation of the UPMSat-3

Besides, the charge cases defined for the model are exposed in **??**, where the dissipation of the payloads are established.

Table 3.2: Dissipation of the instruments for each charge case

Element	Dissipation [W]				
	Safe	**Nominal**	**Payload 1**	**Payload 2**	**Latency**
Tray A					
Cubeseuse-Sun	0.11	0.11	0.11	0.11	0.00
Module Bat-pw	0.50	0.50	0.50	0.50	1.00
Startraker	0.15	0.15	0.15	0.15	0.00
UC3M 2	0.00	0.00	0.00	0.00	0.00
Ebox_UC3M	0.00	0.00	7.00	0.00	0.00
Tray B					
ADCS	2.28	2.28	2.28	2.28	0.00
Shriakutap 1	0.37	0.37	0.37	0.37	0.00
Shriakudian 2	0.37	0.37	0.37	0.37	0.00
Shrinkwian 3	0.37	0.37	0.37	0.37	0.00
6U deployable SA 1	TBD	TBD	TBD	TBD	0.00
6U deployable SA 2	TBD	TBD	TBD	TBD	0.00
Magnetometer 1	0.07	0.07	0.07	0.07	0.00
Magnetometer 2	0.07	0.07	0.07	0.07	0.00
Tray C					
UC3M 1	0.00	0.00	0.00	0.00	0.00
BF_UC3M	0.00	0.00	1.00	0.00	0.00
A3200	3.25	3.25	3.25	3.25	0.00
IDR-COMS	0.00	0.00	0.00	2.50	0.00
Cubesense-Nadir	0.11	0.11	0.11	0.11	0.00
Total	7.65	7.65	15.65	10.15	1.00

However, one must realize that this orbit represents a dynamic state, not a steady one, and the current method only works for steady states (at least for now). To solve this issue, 4 steady state cases were launched:

- Two flight steady cases, in which the temperatures are taken at one point of the orbit as if it was steady:

    1. STEADY FLIGHT HOT: The satellite is in a PAYLOAD 1 mode, with all the instruments on and dissipating the most power, 15.76 W, at the hottest point of the orbit (with the panels facing the Sun).

2. STEADY FLIGHT COLD: The satellite is in a LATENCY mode, with all the instruments off except for the battery module, that is consuming 1 W. This point is calculated at the coldest point of the orbit, the middle of the eclipse.

- Two TVAC steady cases, in which the temperatures are calculated as in an enclosed environment (simulating a thermal vacuum chamber):

1. STEADY TVAC HOT: The satellite is in a PAYLOAD 1 mode, with all the instruments on and dissipating the most power, 15.76 W, with the enclosure at 298 K.
2. STEADY TVAC COLD: The satellite is in a LATENCY mode, with all the instruments off except for the battery module, that is consuming 1 W, with the enclosure at 273 K.

Note that, as the model is the same for every charge case, the initial step of manually filtering the parameters can be done for all of them. However, the following steps must be done separately, because the derivatives have to be calculated with the different datasets of temperatures.

3.3 Validation of pycanha solvers

As aforementioned, pycanha is a software on development phase, so before implementing the method on each charge case, the solvers used to calculate the jacobian matrices and the temperatures have to be validated with the ones obtained with ESATAN-TMS. The results of the comparison can be seen in **??**. The results are quite similar, with a maximum difference of 10^{-2}, which is a good result for the first validation, as both the ESATAN and the pycanha solvers had a tolerance of this same order.

Table 3.3: Validation of the pycanha steady solvers

Cold TVAC	**Hot TVAC**	**Cold Flight**	**Hot Flight**
$6 \cdot 10^{-3}$	$4 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$

3.4 Application of the method to the UPMSat-3 model

3.4.1 Data read

With the TMMs for every charge case generated, the parameters can be read. In this step the value of the parameter does not matter, so reading the parameters for one of the cases is enough to select which parameters to exclude, and then apply the same filter for every charge case. The initial parameter set read can be seen in **??**, where all the parameters are defined

Table 3.4: Initial parameter set for the UPMSat3 model.

Number	Parameter	Value	Description
1	AB_TA	$8.82135 \cdot 10^{-2}$	GL between angle beams and Tray A
2	AB_TB	$5.28 \cdot 10^{-2}$	GL between angle beams and Tray A
3	AB_TC	$5.28 \cdot 10^{-2}$	GL between angle beams and Tray A
4	AB_TD	$7.062 \cdot 10^{-2}$	GL between angle beams and Tray A
5	Cp_Al_6061	$8.96 \cdot 10^{2}$	Specific heat of Al6061
6	Cp_Al_6082	$9.35 \cdot 10^{2}$	Specific heat of Al 6082
7	Cp_Al_7075_T651	$9.6 \cdot 10^{2}$	Specific heat of Al7075
8	Cp_DELRIM_300	$2.88 \cdot 10^{3}$	Specific heat of DELRIM
9	Cp_PCB_mat	$5 \cdot 10^{2}$	Specific heat of the PCB material
10	Cp_SolCell_Ga001	$3.5 \cdot 10^{2}$	Specific heat of the solar cells
11	Cp_Solar_pane001	$7.11 \cdot 10^{2}$	Specific heat of solar panel substrate
12	Dens_Al_6061	$2.7 \cdot 10^{3}$	Specific heat of Al6061
13	Dens_Al_6082	$2.7 \cdot 10^{3}$	Specific heat of Al 6082
14	Dens_Al_7075_T651	$2.81 \cdot 10^{3}$	Specific heat of Al 7075
15	Dens_DELRIM_300	$1.38 \cdot 10^{3}$	Specific heat of DELRIM
16	Dens_PCB_mat	$1.7 \cdot 10^{3}$	Specific heat of the PCB material
17	Dens_SolCell_Ga001	$5.32 \cdot 10^{3}$	Specific heat of the solar cells
18	Dens_Solar_pane001	$1.55 \cdot 10^{3}$	Specific heat of solar panel substrate
19	Frame_corner	$1.8234 \cdot 10^{-1}$	GL between the radiator frame and the panel
20	Frame_mid	$8.995 \cdot 10^{-2}$	GL between the radiator frame and the panel
21	GL_BF	$1.6 \cdot 10^{-1}$	GL between the butterfly and the frame
22	GL_THB	$2.74 \cdot 10^{-1}$	GL between the THB and the tray
23	HY_CPY_RX	$1.24 \cdot 10^{-1}$	GL between the hydra instruments and their tray
24	HY_PCB12_STR_BOT	$2.8 \cdot 10^{-2}$	GL between the hydra instruments and their tray
25	HY_PCB1_STR_TOP	$6 \cdot 10^{-2}$	GL between the hydra instruments and their tray
26	HY_PCB2_STR_TOP	$1.04 \cdot 10^{-1}$	GL between the hydra instruments and their tray
27	HY_STR_TX	$3.714 \cdot 10^{-1}$	GL between the hydra instruments and their tray
28	HY_TX_CPU	$7.97 \cdot 10^{-1}$	GL between the hydra instruments and their tray
29	Hc_bat_eq	$3.24 \cdot 10^{1}$	h_c between the battery module and the tray
30	Hc_general	$3 \cdot 10^{2}$	h_c assumed for every non defined conduction
31	Hc_solar_cells	$1 \cdot 10^{3}$	h_c between solar cells and substrate
32	Node_to_instrument	$2 \cdot 10^{0}$	h_c assumed between the non geometric nodes and its node

Table 3.5: Initial parameter set for the UPMSat3 model.

Number	Parameter	Value	Description
33	OBCTTC_ADCS_EXP	$1.14 \cdot 10^{-1}$	GL between the OBC module and the ADCS
34	PID_heater	$9 \cdot 10^{-1}$	Dissipation of the PID heater
35	Rad_Frame_Y	$2.59 \cdot 10^{-3}$	GL between the radiator and its frame
36	Rad_Frame_Z	$3.36 \cdot 10^{-3}$	GL between the radiator and its frame
37	SP_HINGE	$3 \cdot 10^{-1}$	GL between the solar panel and its hinge
38	Stracker_spacer	$1 \cdot 10^{-2}$	GL between the startracker and its spacer
39	TAD_CS	$3.2831 \cdot 10^{-2}$	GL between the TAD and the CS
40	TA_LPXY	$1.1564 \cdot 10^{-1}$	GL between Tray A and the lateral panels
41	TA_UC3M2_1	$1.3025 \cdot 10^{-2}$	GL between Tray A and the UC3M instrument
42	TA_UC3M2_2	$2.1 \cdot 10^{-2}$	GL between Tray A and the UC3M instrument
43	TB_ADCS	$8.85 \cdot 10^{-1}$	GL between Tray B and the ADCS
44	TB_LPX	$9.96 \cdot 10^{-2}$	GL between Tray B and the lateral panels
45	TB_LPY	$1.08 \cdot 10^{-1}$	GL between Tray B and the lateral panels
46	TB_RW12	$3.62655 \cdot 10^{-2}$	GL between Tray B and the RW12 instrument
47	TB_RW3	$9.89393 \cdot 10^{-2}$	GL between Tray B and the RW3 instrument
48	TC_LPX	$9.96 \cdot 10^{-2}$	GL between Tray C and the lateral panels
49	TC_LPY	$1.08 \cdot 10^{-1}$	GL between Tray C and the lateral panels
50	TC_OBCTTC	$1.1 \cdot 10^{-2}$	GL between Tray C and the OBC
51	TC_OBCTTC_S	$1.11 \cdot 10^{-1}$	GL between Tray C and the OBC
52	TD_LPX	$1.33215 \cdot 10^{-1}$	GL between Tray D and the lateral panels
53	TD_LPY	$1.4445 \cdot 10^{-1}$	GL between Tray D and the lateral panels
54	TIMECT	0.0	Duration of the simulation
55	UC3M1_LPXMIN	$5.2 \cdot 10^{-2}$	GL between the UC3M instrument and the lateral panels
56	WIRE_24AWG	$1.1 \cdot 10^{-3}$	Dissipation of the wires
57	k1_Solar_pane001	$5.0 \cdot 10^{0}$	Conductivity of the solar panels
58	k2_Solar_pane001	$5.0 \cdot 10^{0}$	Conductivity of the solar panels
59	k3_Solar_pane001	$5.0 \cdot 10^{0}$	Conductivity of the solar panels
60	k_Al_6061	$1.67 \cdot 10^{2}$	Conductivity of Al 6061
61	k_Al_6082	$1.70 \cdot 10^{2}$	Conductivity of Al 6082
62	k_Al_7075_T651	$1.30 \cdot 10^{2}$	Conductivity of Al 7075
63	k_DELRIM_300	$2.1 \cdot 10^{-1}$	Conductivity of DELRIM
64	k_DELRIN	$2.5 \cdot 10^{-1}$	Conductivity of DELRIN
65	k_PCB_mat	$4.5 \cdot 10^{-1}$	Conductivity of PCB material

3.4.2 Manual filter

Once the parameters have been read, the thermal conductivites, the densities and the specific heats are excluded from the list, as well as other parameters that resul unnecessary for the current analysis, such as the period of the orbit or the dissipation of the PID.

Table 3.6: Parameter set for the UPMSat3 after the manual filter.

Number	Parameter	Value
1	AB_TA	0.0882135
2	AB_TB	0.0528
3	AB_TC	0.0528
4	AB_TD	0.07062
5	Frame_corner	0.18234
6	Frame_mid	0.08995
7	GL_BF	0.16
8	GL_THB	0.27445565624
9	HY_CPY_RX	0.124346
10	HY_PCB12_STR_BOT	0.028
11	HY_PCB1_STR_TOP	0.06
12	HY_PCB2_STR_TOP	0.104
13	HY_STR_TX	0.371014
14	HY_TX_CPU	0.79768
15	Hc_bat_eq	32.4
16	Hc_general	300.0
17	Hc_solar_cells	1000.0
18	Node_to_instrument	2.0
19	OBCTTC_ADCS_EXP	0.114

Table 3.7: Parameter set for the UPMSat3 after the manual filter.

Number	Parameter	Value
20	Rad_Frame_Y	0.00259
21	Rad_Frame_Z	0.00336
22	SP_HINGE	0.3
23	Stracker_spacer	0.01
24	TAD_CS	0.032831
25	TA_LPXY	0.11564
26	TA_UC3M2_1	0.013025
27	TA_UC3M2_2	0.021
28	TB_ADCS	0.885
39	TB_LPX	0.0996
30	TB_LPY	0.108
31	TB_RW12	0.0362655
32	TB_RW3	0.0989393
33	TC_LPX	0.0996
34	TC_LPY	0.108
35	TC_OBCTTC	0.011
36	TC_OBCTTC_S	0.111
37	TD_LPX	0.133215
38	TD_LPY	0.14445
49	UC3M1_LPXMIN	0.052
40	WIRE_24AWG	0.0011

Note that the list of parameters only retains the GLs and h_cs, being set of linear conductances in different units.

3.4.3 Jacobian and influence matrix calculation

With a reduced set of parameters, the Jacobian is calculated for every charge case. These are not represented, neither shown mathematically because these matrices are size 1151x40, which is absurd as it would not fit. Then, using the parameters' deviation shown in **??**, the influence matrices for each charge case are calculated. However, the influence matrices show very interesting results, so they have been represented using a reduction matrix. This matrix is used to join the nodes that belong to a certain part of the structure and make its average, so instead of having 1151 rows corresponidng to each node, we have 41 that correspond to

some part of the structure.

The different influence matrices can be seen in Figures **??**, **??**, **??** and **??**

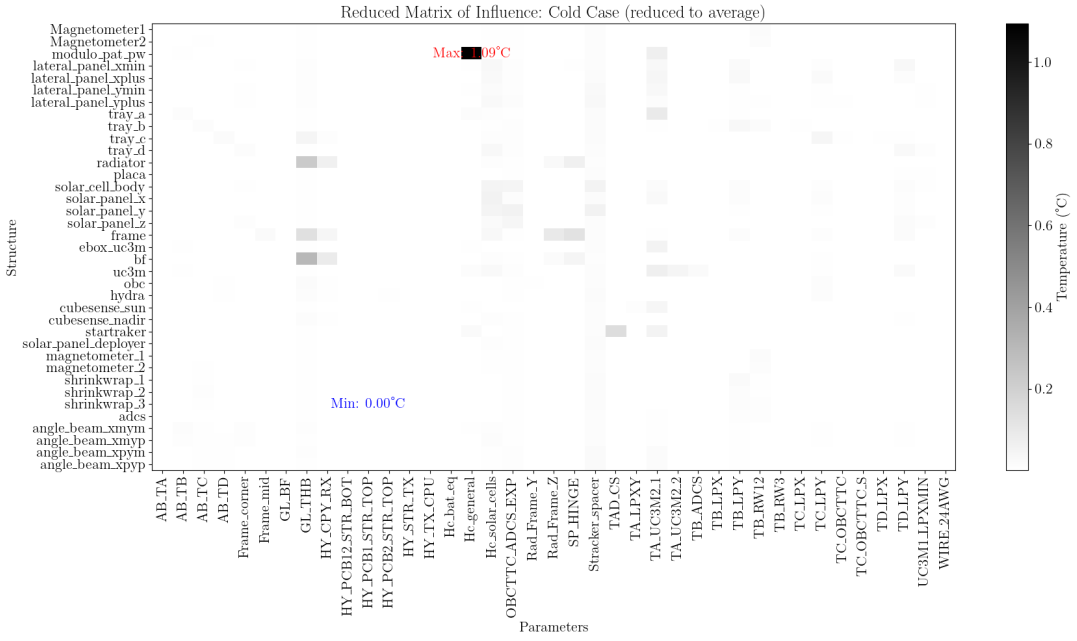

Figure 3.4: FLIGHT Cold case Influence matrix

The distribution of temperature variation represented in **??** makes perfect sense: the temperature change in most parts of the structure is close to 0, except for the area close to the battery module, where the a change in the parameter hc_general would have a significant impact on the temperature. This is because the battery module is the only part of the structure that is dissipating power in this case. Other parts that are impacted by a change in the parameters are the radiator and its frame, which are impacted by the change in the conductance of the thermal strap, the radiator and the frame, which is normal due to the conection with the battery.

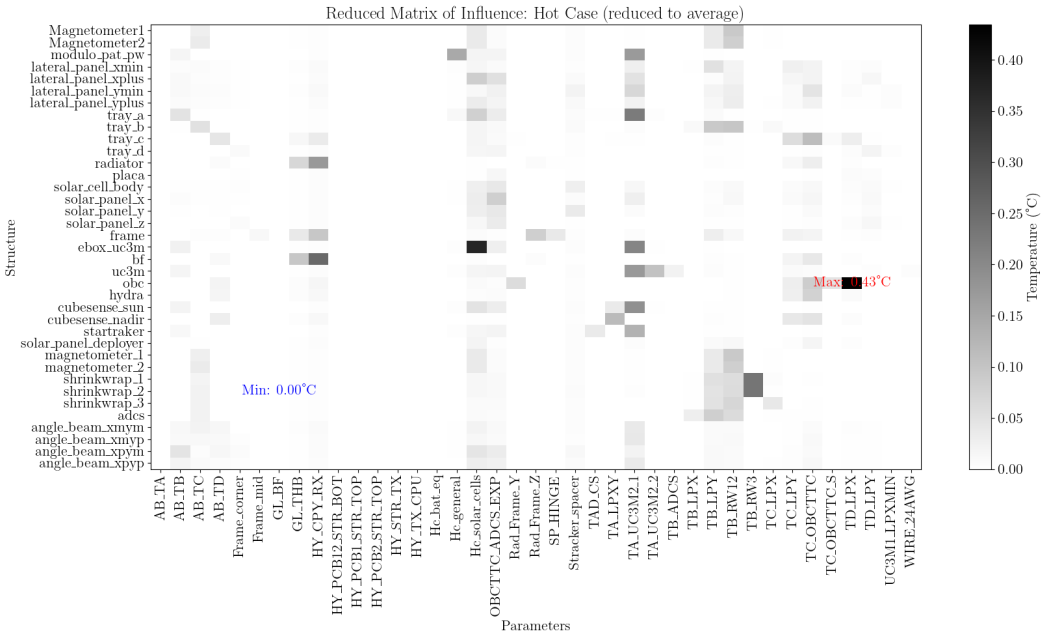

Figure 3.5: FLIGHT Hot case Influence matrix

Now, in **??**, the temperature change is more distributed, as the power is dissipated in more parts of the structure. The battery module is still a highly impacted part, although definitely not the most significant one. Here the nodes correponding to the Ebox of the UC3M, the OBC, the buterfly and the UC3M instruments would be the most impacted by a change in the parameters. Seeing it the other way round, in this case the hc_general parameters looses part of the huge impact it had in the cold case, as the power is dissipated in more parts of the structure. Now, parameters like the conductance between different instruments or between the tray and some of these instruments gain a lot of importanmce. This means it would be a lot easier to make measurements of these parameters in the hot case than in the cold one. However, see that the highest variation in this hot case is about 60% the variation of the cold case, so even though in the hot case more parameters can be measured, parameters as the hc_general will be better measured in the cold case.

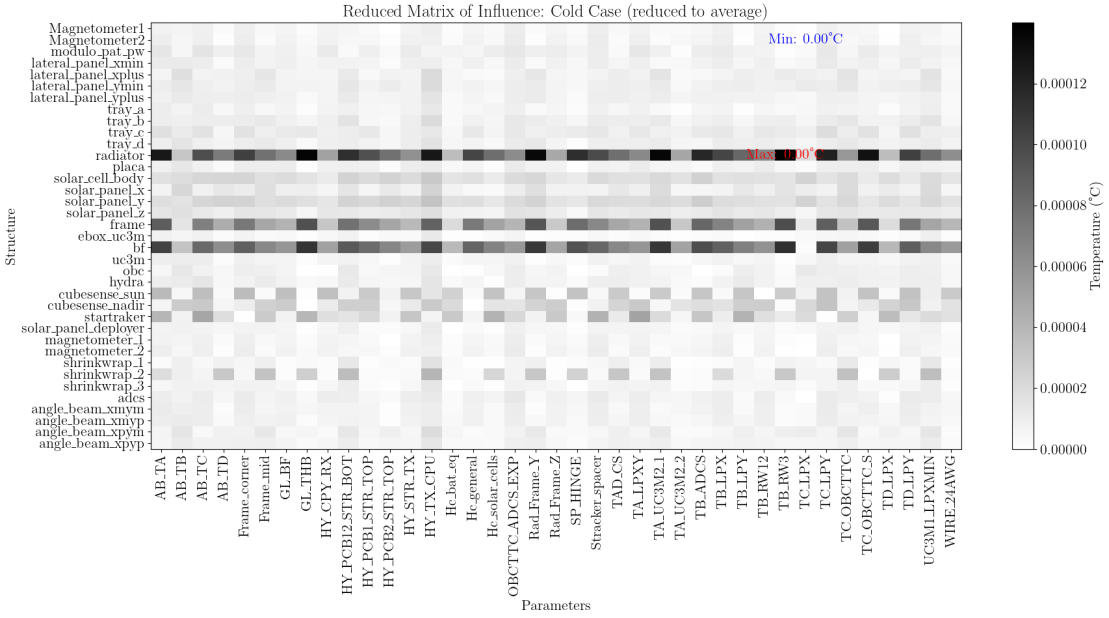

Figure 3.6: TVAC Cold case Influence matrix

As opposed to the FLIGHT cold case, the TVAC cold case is very evenly distributed. However, taking a look at the units, its easy to see that the temperature deviation is so low that it is not worth it to measure any of these parameters, as the temperature change is really low. This charge case is not useful for the current analysis and will not be further considered, but a new valid cold TVAC charge case will be presented as future work.

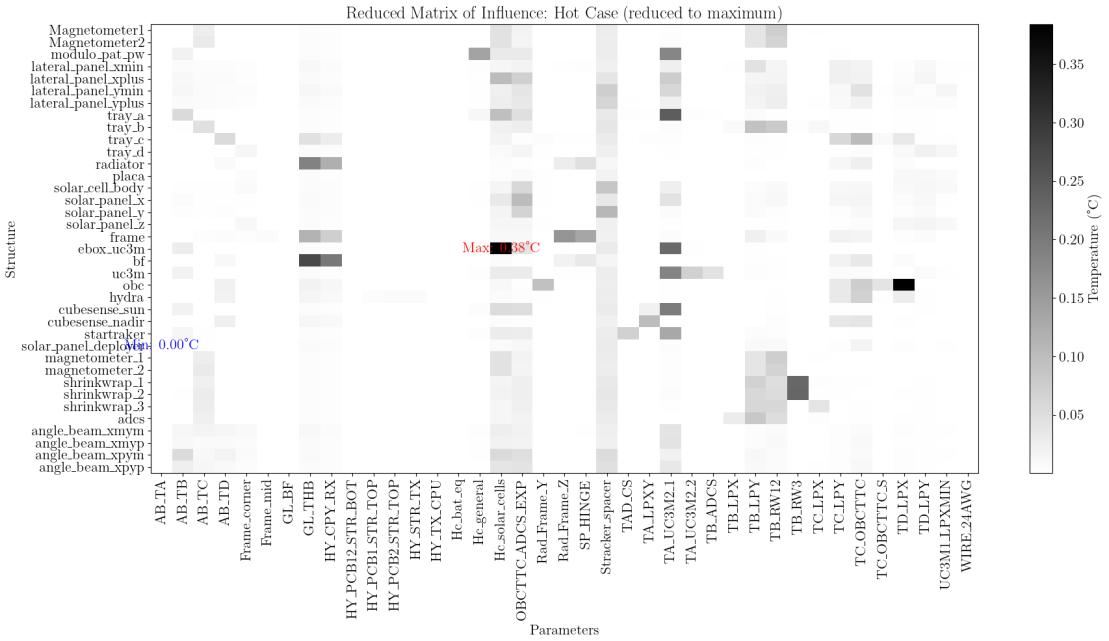

Figure 3.7: TVAC Hot case Influence matrix

As for the hot TVAC case, the similarities with the FLIGHT hot case are clear, which is to be expected. In this case, the maximums and minimums variate, but the distribution is quite similar, which is quite good as this means the test case is properly designed, and possibly the sensor optimal placement would be very similar in both cases.

3.4.4 Normalization of the influence matrices and normalization filter

In order to compare the influence matrices, they have to be normalized, which is done exposed in **??**.

The result of this normalization does not really provide any new information from the one of the non-normalized matrices, as the values are the same, but this visualization helps discern which parameters have no importance, which ones are important because they have an impact on lots of parts of the structure even though they dont have a peak at one specifically (which is way more difficult to see in the non-normalized matrix) and if these last more distributed parameters are more or less important than the ones that do have a peak. By doing this, and seeing the initial resul the normalization threshold, ε_{nf}, has been set to 0.05.

Note that in order to place sensors it will be easier if they actually have a peak, as in the node it peaks a variation of this parameter will be more noticeable. If a parameter is distributed, it will be more difficult to measure, as the variation will be more distributed, but it does not mean that parameter is less important.

The normalized matrices can be seen in Figures **??**, **??**, **??** and **??** below.

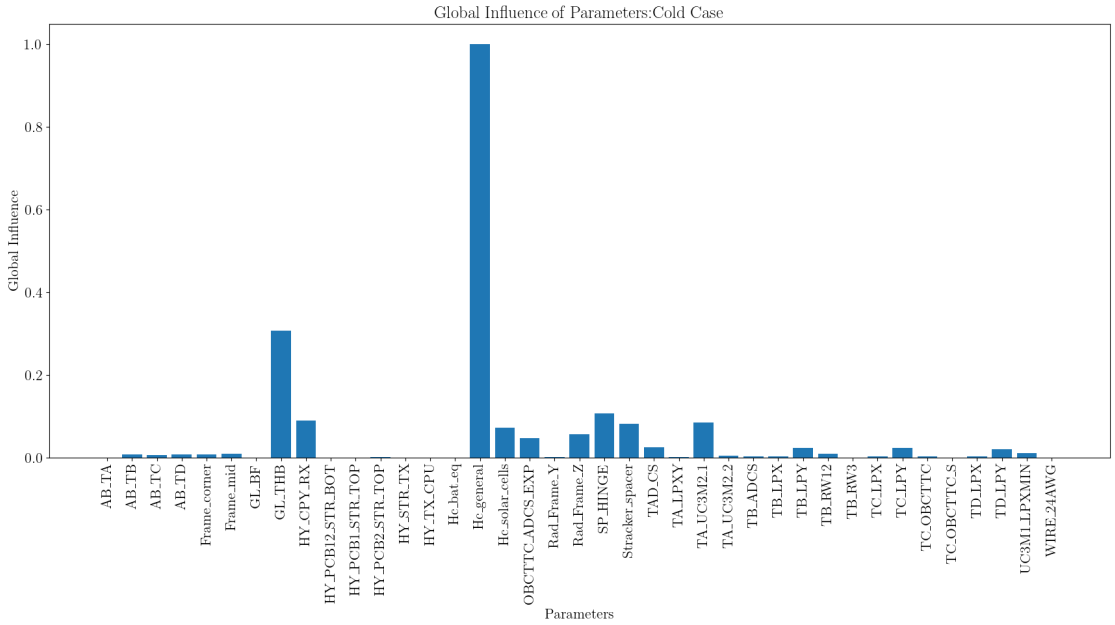

Figure 3.8: FLIGHT Cold case Normalized Influence matrix

As it was aforementioned, in the cold case the only parameters that have a significant impact are the ones that are connected to the battery module (hc_general), the radiator and its frame, and the thermal strap. The rest of the matrix of influence is really close to 0. The parameters that are retained after the filter can be seen in table **??**.

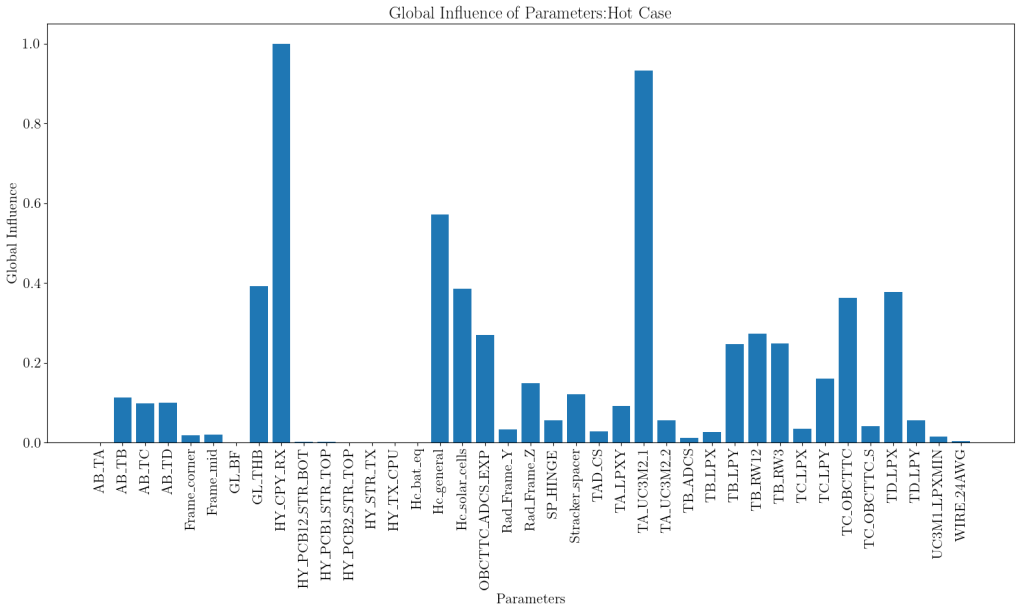

Figure 3.9: FLIGHT Hot case Normalized Influence matrix

With the same threshold, a lot more parameters are retained in the hot case, as the power is dissipated in more parts of the structure, as it was explained in the non-normalized matrix. However, it is important to see that all the parameters that were retained in the cold case are also retained in the hot case, so we would be able to use the same sensors in this case.

Finally, the TVAC hot case retains one more parameter than the FLIGHT hot case, being all the retained parameters of the flight cases subsets of this TVAC hot case, which will be really valuable for further analysis of the tests.

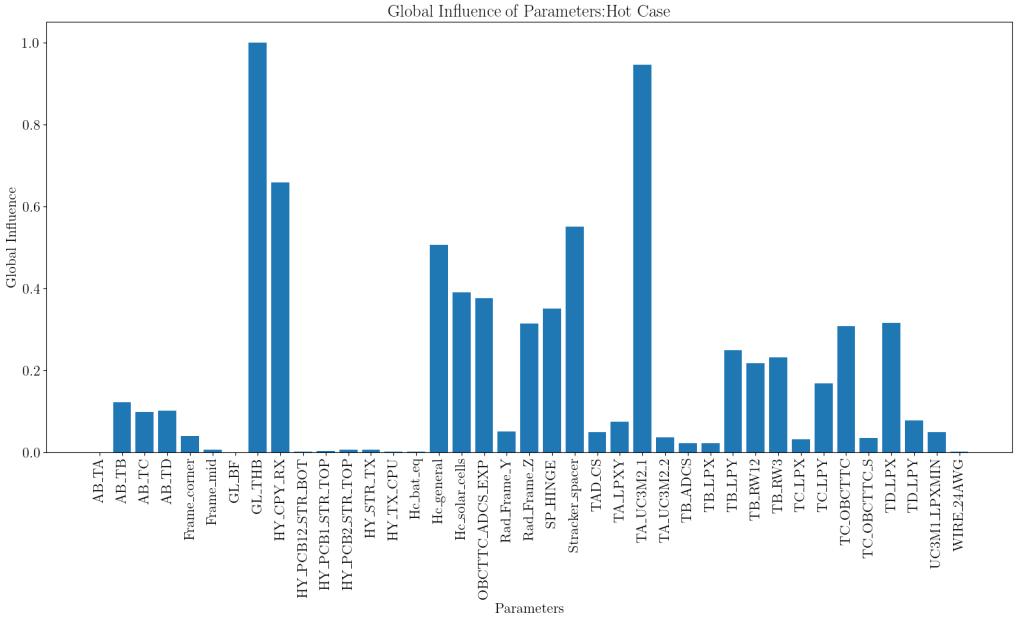

Figure 3.10: TVAC Hot case Normalized Influence matrix

Table 3.8: Retained parametersafter the normalization filter for each charge case.

Flight-Cold	**Flight-Hot**	**TVAC-Hot**
Hc_solar_cells	Hc_solar_cells	TB_RW3
Hc_general	Hc_general	AB_TD
TA_UC3M2_1	TA_UC3M2_1	TA_UC3M2_1
Stracker_spacer	Stracker_spacer	GL_THB
GL_THB	GL_THB	TC_LPY
SP_HINGE	TB_LPY	Hc_solar_cells
HY_CPY_RX	HY_CPY_RX	TD_LPX
	TB_RW12	TB_RW12
	TB_RW3	TC_OBCTTC
	TC_OBCTTC	TB_LPY
	AB_TB	OBCTTC_ADCS_EXP
	TD_LPX	Hc_general
	Rad_Frame_Z	Rad_Frame_Z
	AB_TD	Stracker_spacer
	OBCTTC_ADCS_EXP	SP_HINGE
		AB_TB
		HY_CPY_RX

3.4.5 Linear dependence filter

Finally, the linear dependence filter is applied to the retained parameters. The final set of parameters that are retained after this filter can be seen in **??**. See that in all the valid charge cases the only dependent parameter (among the ones that passed the previous threshold), was the HY_CPY_RX, which can be explained by the direct connection it had with the thermal strap, which, may have led to these two parameters to be dependent.

Table 3.9: Retained parameters after the normalization filter for each charge case.

Flight-Cold	**Flight-Hot**	**TVAC-Hot**
Hc_solar_cells	Hc_solar_cells	TB_RW3
Hc_general	Hc_general	AB_TD
TA_UC3M2_1	TA_UC3M2_1	TA_UC3M2_1
Stracker_spacer	Stracker_spacer	GL_THB
GL_THB	GL_THB	TC_LPY
SP_HINGE	TB_LPY	Hc_solar_cells
	TB_RW12	TB_RW12
	TB_RW3	TC_OBCTTC
	TC_OBCTTC	TB_LPY
	AB_TB	OBCTTC_ADCS_EXP
	TD_LPX	Hc_general
	Rad_Frame_Z	Rad_Frame_Z
	AB_TD	Stracker_spacer
	OBCTTC_ADCS_EXP	SP_HINGE
		AB_TB
		TD_LPX

Finally, an SVD decomposition has been done over the initial set of parameters and over the last one for the TVAC case, the resulting eigenvalues are presented below in REFERENCIA SVD, showing that the nodes coincide with 5% deviation. Thus, we can conclude that the physics of the problem can be represented with this reduced set of parameters.

Figure 3.11: Comparison of eigenvalues TVAC hot case reduced an original matrices

Chapter 4

Conclusions and future work

4.1 Conclusions

Among the present study case, a method to analyze the influence of each parameter and to select the most important ones has been presented. This method focuses on the computation of the Jacobian and the Influence matrices to later use them as tools to recognize the influence of the parameters on every part of the structure as well as to see how the parameters are related among themselves.

This method has been first applied to a simple model, looking for a better understanding of the process itself. After that it has been used in a real space demonstrator, not only obtaining some fascinating results, but also developing some test and flight cases to a real space mission and improving some features to the software in use.

With the implementation of the method to the UPMSat-3 the parameters that must be analyzed in order to be able to retain the physics of the problem are reduced a 56%, which, as will be mentioned in the future work section, is key to reduce the number of sensors necessary to test the satellite. Furthermore, this analysis has shown what parameters can be measured with each charge case, giving an automatic mathematical reasoning to the charge cases proposed.

In the same line, the method has been proven useful to see if the proposed charge cases are not valid or convenient, as the cold TVAC case proposed was discarded in the implementation of the method, otherwise it would have been presented as valid. This is quite interesting due to the costs (in terms of money and time) of doing a TVAC test; doing a TVAC test to find out it was useless is not only really expensive but can also delay the whole project.

It is also worth highlighting the improvements made to the pycanha package throughout the

implementation of the current method, such as:

- The implementation of Jacobian calculation for steady cases, in an analytical way as the one proposed in [**?**], and also numerically, through centered forward and backward differences, as well as the computation of influence matrices.
- Useful features like the read-label and the reduction matrix computation that will come in really handy for future projects.
- The correction of some errors, in the data read functions that were not properly defined and led to a misinterpretation of correct data.

4.2 Future work

The work presented in this study case, may be used as a starting point for future ones. In this context, some ideas have been proposed as points where this method could be improved, or a further research could be done:

1. Development and analysis of more charge cases to ensure that this parameter reduction retains the physics of the problem not only in the hot and cold extreme cases, but also at points in between those.
2. Definition of sensor positioning. Due to the scope of this project, the positioning of the sensors has only been mentioned, but an optimization of the thermocouples positioning would be a fascinating advance to this method.
3. Definition influence weights. The proposed method does not present the possibility of some parameters being more influential, maybe not in a physical way, but for engineering purposes.
4. Correlation between the TVAC cases and the flight ones. This relation has been envisioned in several occasions among this project, but an actual mathematical correlation is yet to be done.
5. Correlation with real test cases. Although no correlation has been done yet (not with computer models, neither with physical ones), correlating a simple model -like the FPGA- with a real TVAC test, would be very useful to validate the method.

Bibliography

[1] D. G. Gilmore, Spacecraft Thermal Control Handbook. Volume I, Vol. I, Aerospace Press, El Segundo, California, 2002.

[2] J. Meseguer, I. Pérez-Grande, A. Sanz-Andrés, Spacecraft Thermal Control, Woodhead Publishing, 2012.

[3] R. Karam, Satellite Thermal Control for Systems Engineers, American Institute of Aeronautics and Astronautics, Inc, 2012.

[4] I. Garmendia, E. Anglada, H. Vallejo, M. Seco, Accurate calculation of conductive conductances in complex geometries for spacecrafts thermal models, Advances in Space Research 57 (4) (2016) 1087–1097.
URL http://dx.doi.org/10.1016/j.asr.2015.12.027

[5] S. Appel, R. Patrício, H. P. de Koning, O. Pin, Automatic linear conductor generation solution for lumped parameter models, in: 34th International Conference on Environmental Systems, 2004.

[6] P. P. Almazan, Accuracy of monte carlo ray-tracing thermal radiation calculations: A practical discussion, in: Sixth European Symposium on Space Environmental Control Systems, 1997, pp. 579–591.

[7] P. S. Foundation, Python Documentation, accessed: 2024-05-22 (2024).
URL https://docs.python.org/3/

[8] N. Developers, NumPy Documentation, accessed: 2024-05-22 (2024).
URL https://numpy.org/doc/stable/

[9] S. Community, SciPy Documentation, accessed: 2024-05-22 (2024).
URL https://docs.scipy.org/doc/scipy/

[10] M. D. Team, Matplotlib Documentation, accessed: 2024-05-22 (2024).
URL https://matplotlib.org/stable/contents.html

[11] I. Aero, ESATAN-TMS Official Site, accessed: 2024-05-22 (2024).
URL https://www.esatan-tms.com/

[12] AMD, XQR Kintex UltraScale Documentation, accessed: 2024-06-05 (2024).
URL https://docs.amd.com/v/u/en-US/ds882-xqr-kintex-ultrascale

[13] I. de Microgravedad Ignacio Da Riva, UPMSAT-3 Thermal Control Subsystem, 2024, accessed: 2024-05-22.

[14] 3d printing price calculator - original prusa 3d printers [online].

Appendices

Anexo A

Título del anexo

Aquí puedes meter la información que no sea imprescindible en el cuerpo del trabajo pero si que interese que esté en el documento.