Polytechnic University of Catalonia
The School of Industrial, Aerospace and Audiovisual Engineering of Terrassa

D02 Tank Numerical Simulation

Thermal instabilities analysis into the ALBA's D02 tank and detailed description of the one-dimensional transient simulation algorithm.

ALBA Synchrotron

Author: Francesc Hernández Garcia
Co-Authors: Marcos Quispe, Jordi Iglesias
Date: June 2021

Abstract

The water-cooling system of the ALBA accelerator has experienced thermal instability problems over the last years, which affect the control of the entrance temperature in its four main rings: Service Area, Booster, Storage and Experimental Hall. Initial studies carried out by an infrastructure work team have determined that the main source of this problem is an anomaly in the cold-water supply due to an external Co-generation Plant (ST4), on which ALBA depends to thermally regulate the Accelerator and the Air Conditioning systems.

In this article, the effect of the thermal instabilities in the hydraulic refrigeration circuit of the ALBA Synchrotron is studied at the D02 water tank. The analysis is performed by the use of a one-dimensional node-centered numerical simulation, carried out employing a self-made matlab code based on actual historical data.

The heat transfer through the walls of the tank is quantified taking into account the thermal instabilities, and compared to the heat exchange behaviour in a steady state. The main objective of the paper is to study the necessity of insulating the tank, by the design of an adequate insulator which would reduce the heat loses in the tank. A transient simulation algorithm is presented and performed in order to solve the conduction heat transfer in the walls of the tank, solving the convection at their inner and outer sides.

Contents

List of Figures

List of Tables

1. Introduction

1.1. The ALBA Synchrotron

ALBA is a third-generation synchrotron light source facility located in Cerdanyola del Vallès, Catalonia, with more than eleven years of operation and eight operating beam-lines. It was constructed by CELLS[1], who is currently operating the synchrotron, and started its operation in March 2010.

Figure 1: The Alba Synchrotron.

Over the last years, the water-cooling thermohydraulic system of ALBA has been under thermal instability problems, which affect the control of the entrance temperature in its four main rings: SA, BO, SR and EA. In this article, this issues will be analyzed in one of the most important tanks of the hydraulic system, due to the necessity of understanding the transient behaviour of the thermal anomalies in the tank.

This is an utmost magnitude problem, insofar there is a direct linkage between the electron beam stability and the thermal stability of the water-cooling circuit. Moreover, this behaviour restricts ALBA's expansion capacity in terms of the machine current: ALBA operates at 250mA, although its design current is 400mA.

The problem could affect the growth expansion plans of ALBA II [1], which is an upgrade project to become a fourth-generation light source facility announced in 2020. The so-called 4th generation synchrotron facilities, compared to those of the 3rd generation, produce a brighter and more coherent photon beam, which means a higher thermal load in the machine. Thus, the thermohydraulic cooling system is going to be taken to its maximum working capacities.

[1]Spanish acronym of "Consorcio para la Construcción, Equipamiento y Explotación del Laboratorio de Luz de Sincrotrón, the Consortium for the Exploitation of the Synchrotron Light Laboratory".

1.2. Thermohydraulic circuit

In order to introduce the thermal issue, a simplified scheme of the thermohydraulic circuit of ALBA is explained.

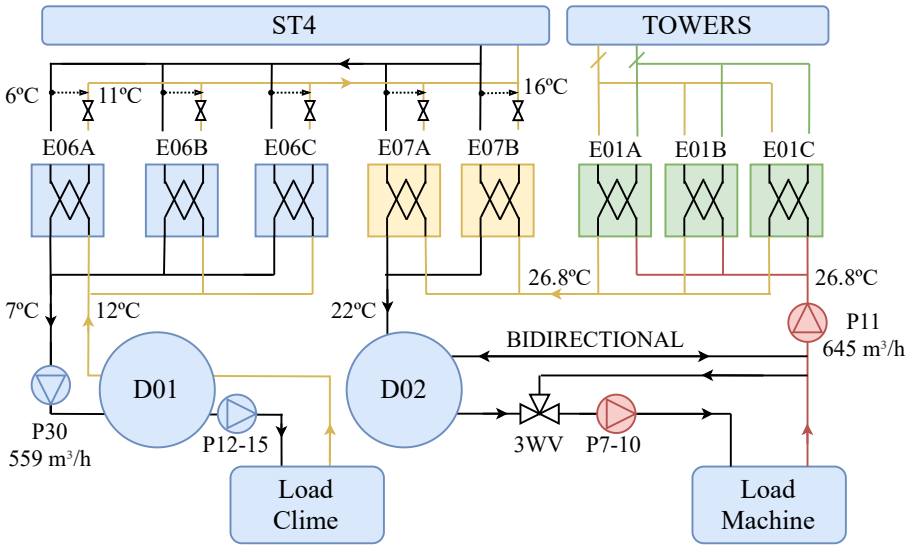


Figure 2: Alba's thermohydraulic circuit scheme.

The circuit begins with the hot deionized water returning from the machine headed to the P11 pump, which should move 645 m^3/h at 26.8 ºC, after a filtering process. The flow proceeds through the E01 heat exchangers, which would exchange heat with the Towers' water. This heat exchanging is restricted and E01A & E01C closed. All the flow circulates through E01B and steers to heat exchangers E07, where each exchanger takes 156 m^3/h and yields 1815 kW of heat to ST4 water, aiming to decrease the secondary flow temperature to the 22 ºC of the D02 tank.

The circuit ends with the 3-way valve and the P7-P10 pumps, which allow the impulsion of the water from D02 to the machine. The bidirectional tube must always maintain an outgoing flow from the tank, otherwise, a catastrophic situation would be generated in the circuit.

The climate load uses water from the D01 tank, employing the pumps P12-P15. D01's water is cooled with the E06 heat exchangers, each one taking 187 m^3/h and yielding 1080 kW of heat to ST4 primary flow. Once cooled, the flow returns to D01 circulating through the P30 pumps, which move 559 m^3/h. Two-way mixing valves and differential pressure valves are located in the primary flow circuit for both E06 & E07 exchangers.

1.3. Anomalies background

The primary entrance temperature in the E07 heat exchangers is unstable, as the co-generation plant ST4 has an irregular behavior, altering the tank temperature as consequence. It is important to understand the temperature evolution in the D02 tank in order to find out which behaviour should the Synchrotron have at these moments of instabilities.

The ST4 anomalies appear during changes in its operating modes, which last 9 hours per week as an average, decreasing the external mass flow and/or increasing its temperature. Otherwise, the flow circulating through the ALBA's circuit is about 400 m^3/h, under the design specified value of 650 m^3/h due to a cavitation phenomenon that appeared in the P11 pump. This second anomaly, in addition to the ST4 one, are strongly altering the tank thermal stability.

2. Thermal instabilities and tank description

2.1. Tank description

The D02 tank is a high capacity vertical tank able to storage 40000 liters of deionized water. It is made of stainless steel, with an empty mass of 3100kg. Its external diameter is 2950 mm, and it has a thickness of 6 mm. The main uniform body of the tank is 4775 mm, and it makes a total height of 5985 mm adding the top and bottom covers. The water gets inside the tank from an upper pipe of DN350, and goes out from a bottom pipe of DN350. There is a drawing of the tank at the Appendix I. The chosen properties for the stainless steel are:

$\lambda\ [W/mK]$	17	Thermal conductivity
$C_p\ [J/KgK]$	480	Specific heat
$\rho\ [kg/m^3]$	7500	Density

Table 1: Stainless steel specifications.

As the temperature variation of the walls of the tank is going to be between 20ºC and 25ºC, these properties have been considered constant. Despite this, the software will is able to accept temperature dependencies if needed. Finally, the height of the tank is set at 5.38 meters for the simulation, as it is the mean value between the height of the uniform body and total height.

One of the objectives of this article is to study the possibility of installing an insulator. Regarding the nomenclature, the inner wall of the tank will be called *inner wall*, the outer wall of the tank will be called *intermediate wall*, and the outer wall of the designed insulator will be called *outer wall*.

2.2. Thermal instabilities description

Historical records from the ALBA's server allowed us to plot the following tendencies for each ring:

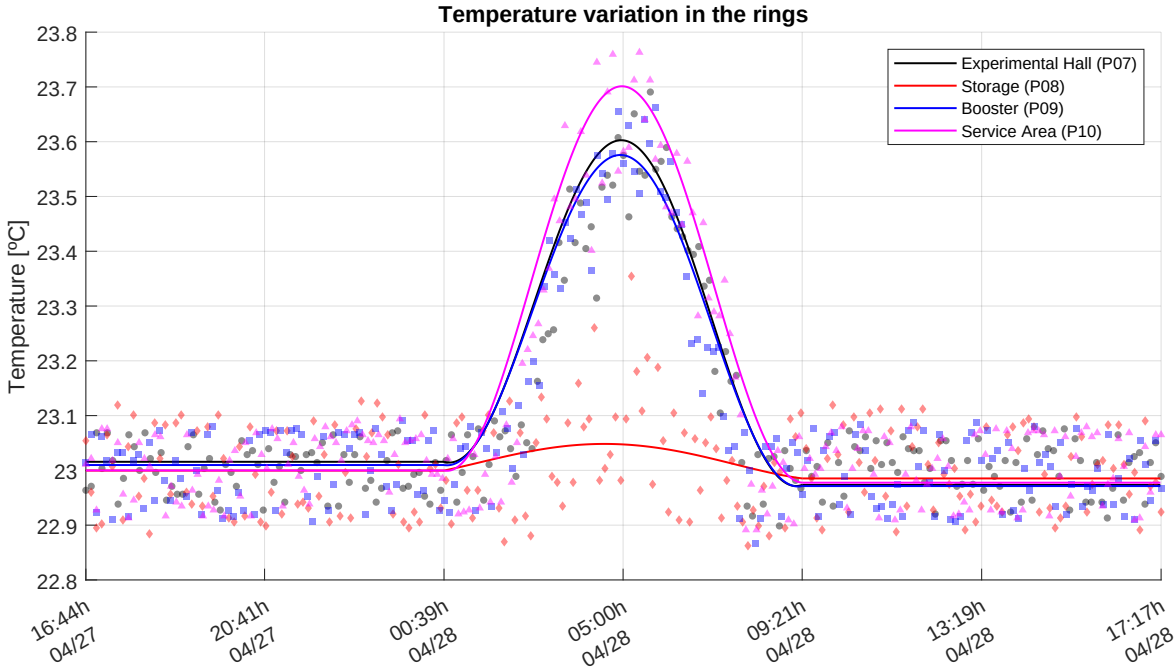


Figure 3: Temperature peak in the four main rings of 28/04/2021.

These thermal instabilities tend to appear during the first hours in the morning, increasing the temperature of the main four rings of ALBA: Service Area (SA), Booster (BO), Storage (SR) and Experimental Hall (EA). The plot represent the thermal instabilities of the morning of the 28th of April of 2021. The continuous lines correspond to polynomial regressions of the curves, while the scatter points correspond to the actual experimental data. As it is possible to see, there is a peak about 5AM of different amplitudes for each ring. More concretely, the mean amplitudes for the EA, SR, BO, SA rings are 0.59 $°C$, 0.05 $°C$, 0.57 $°C$, 0.70 $°C$, respectively.

The last instabilities affect the water of the tank in the following way:

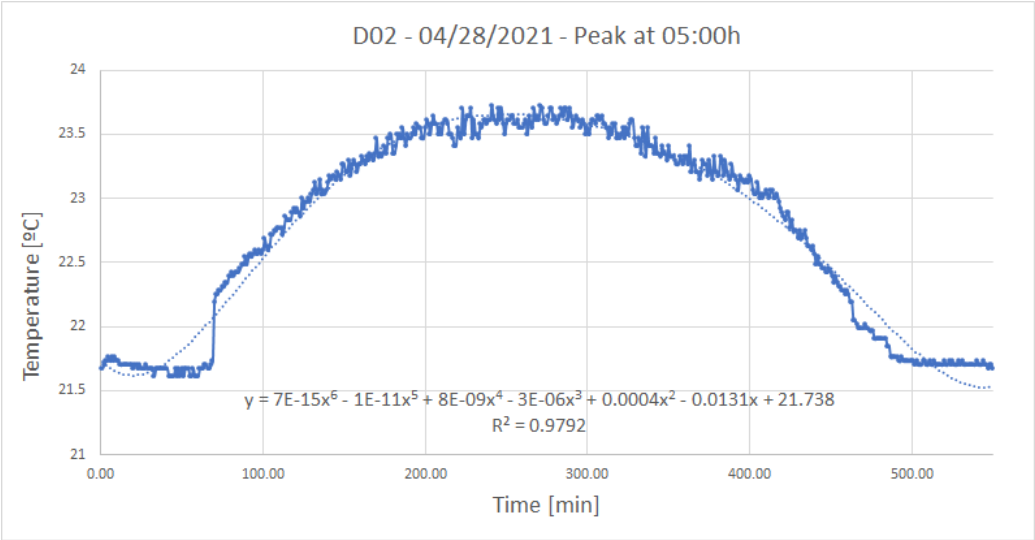


Figure 4: Tank temperature variation.

As it is possible to see, the tank temperature increase can be approached by a 6th grade equation with a good correlation coefficient of $R^2 = 0.9792$. This yields:

$$T = 7 \cdot 10^{-15} t^6 - 1 \cdot 10^{-11} t^5 + 8 \cdot 10^{-9} t^4 - 3 \cdot 10^{-6} t^3 + 0.0004 t^2 - 0.0131 t + 21.738 \qquad (2.1)$$

Where the time shall be introduced in minutes. The simulation time is going to start at the 19th minute in order to avoid the left minimum of the polynomial regression, and will last 525 minutes, as it is the needed time to complete the oscillation. Thus, the equation shall be modified by changing t for $(t/60 + 19)$. This allows us to introduce the time in seconds and moves the curve to the left. Finally, the temperature needs to be converted from Celsius degrees to Kelvin.

3. Discrete conduction heat transfer equations

3.1. Transient state discretization coefficients

In this section, the conduction equations employed in solving the conduction heat exchange through the walls of the tank are presented. In regards of the energy balance equation in a discrete control volume, where the internal energy equals the heat exchange:

$$\dot{U} = \sum \dot{Q} \tag{3.1}$$

$$\frac{\partial}{\partial t} \int_{C\forall} u \cdot \rho \cdot d\forall = \sum \dot{Q} \tag{3.2}$$

Where ρ is the density of the material and u is the internal energy per unit of mass. Developing the expression with a mean internal energy:

$$\bar{u}_p = \frac{1}{\rho_p \forall_p} \int_{C\forall} u \rho d\forall \tag{3.3}$$

$$\rho_p \forall_p \frac{\partial \bar{u}_p}{\partial t} = \sum \dot{Q} \tag{3.4}$$

Where the subscript p is referred to a node centered inside the control volume. Employing $d\bar{u} = \bar{c_p} dT$ and numerically integrating between the present and following instants, t^n and t^{n+1}:

$$\rho_p \forall_p \int_{t^n}^{t^{n+1}} \frac{\partial \bar{u}_p}{\partial t} dt = \int_{t^n}^{t^{n+1}} \sum \dot{Q} dt \tag{3.5}$$

$$\rho_p \forall_p \bar{c_p}_p \left(T_P^{n+1} - T_P^n\right) = \left[\beta \sum \dot{Q}_p^{n+1} + (1-\beta) \sum \dot{Q}_p^n\right] \Delta t \tag{3.6}$$

Regrouping:

$$\frac{\rho_p \forall_p \bar{c_p}_p}{\Delta t} T_P^{n+1} = \beta \sum \dot{Q}_p^{n+1} + (1-\beta) \sum \dot{Q}_p^n + \frac{\rho_p \forall_p \bar{c_p}_p}{\Delta t} T_P^n \tag{3.7}$$

Where $\sum \dot{Q}_p^n$ and $\sum \dot{Q}_p^{n+1}$ represent the total heat summation at the present and following instants, respectively, where the first is known at the moment of the calculation.

The parameter β is a numerical number which defines the numerical integration method as follows:

- $\beta = 0 \quad \longrightarrow$ Explicit integration method: 1st order approach.
- $\beta = 0.5 \quad \longrightarrow$ Crank-Nicolson integration method: 2nd order approach.
- $\beta = 1 \quad \longrightarrow$ Implicit integration method: 1st order approach.

Considering a centered node P surrounded by a left node W and a right node E, it is possible to obtain the summation of heat at the control volume of P, considering one-dimensional conduction heat flux between the nodes with the Fourier's law, and convective heat flow when necessary (for first and last node, at the inner and outer wall). Once the summations of heat at n and $n+1$ are calculated, they are collocated at the equation *(2.7)*, which is modified in order to obtain:

$$a_P T_P^{n+1} = a_E T_E^{n+1} + a_W T_W^{n+1} + b_P \tag{3.8}$$

Where the a_P, a_E, a_W and b_P are the discretization coefficients which multiply the nodal temperatures.

3.1.1. First node discretization coefficients

For a node-centered mesh, the first node ($i = 1$) is located at the inner side of the tank wall, at his face. Thus, the volume of the control volume of the node is non-existent ($\forall_p = 0$) due to its position at the left face of the control volume of the second node.

Performing the summation of heat at the instant t^{n+1}:

$$\sum \dot{Q}^{n+1} = \alpha_{int}^{n+1} \cdot \left(T_{int}^{n+1} - T_P^{n+1}\right) S_P - \lambda_e^{n+1} \frac{T_P^{n+1} - T_E^{n+1}}{d_{PE}} S_P \tag{3.9}$$

The parameter α_{int} is the convective coefficient of the water inside the tank evaluated at the new instant. The coefficient λ is the conduction coefficient of the material at the new instant, evaluated in the face of the subscript (east or west) by the use of an harmonic mean. For example, λ_e is calculated as:

$$\lambda_e = \frac{d_{PE}}{\frac{d_{Pe}}{\lambda_P} + \frac{d_{Ee}}{\lambda_E}} \tag{3.10}$$

Where d_{PE} is the distance between nodes P and E, and d_{Pe} is the distance between the node P and the right east face of its control volume. For this particular case, $\lambda_e = \lambda_E$. The surface $S_P = 2\pi R_{el}(1)H$ is the lateral area of the node, where H is the height of the tank, and $R_{el}(i)$ is a vector which contains the faces of the control volumes, defined as:

$$R_{el}(i) = R_{int} + \frac{R_{ext} - R_{int}}{N} \cdot (i - 1) \qquad \forall\, i \in [1 : N + 1] \tag{3.11}$$

Where N is the number of control volumes of the geometry discretization and it is defined as a numerical parameter. Introducing the equation *(2.9)* into the equation *(2.7)* and regrouping, it is possible to extract the following discretization coefficients:

$$\begin{cases} a_P = \beta \cdot \left[\alpha_{int}^{n+1} S_P + \lambda_e^{n+1} \frac{S_P}{d_{PE}}\right] \\ a_E = \beta \cdot \lambda_e^{n+1} \frac{S_P}{d_{PE}} \\ a_W = 0 \\ b_P = \beta \cdot \alpha_{int}^{n+1} T_{int}^{n+1} S_P + (1 - \beta) \sum \dot{Q}^n \end{cases} \tag{3.12}$$

3.1.2. Intermediate nodes discretization coefficients

For an arbitrary node (i) positioned inside the wall, the process is analogous to the previous case. Performing the summation of heat at the new instant:

$$\sum \dot{Q}^{n+1} = \lambda_w^{n+1} \frac{T_W^{n+1} - T_P^{n+1}}{d_{PW}} S_W - \lambda_e^{n+1} \frac{T_P^{n+1} - T_E^{n+1}}{d_{PE}} S_E + \dot{q}_{\forall p}^{n+1} \forall_p \tag{3.13}$$

Unlike the previous case, there is no convective heat inside the wall and it is necessary to take into account the conduction heat yielded by both west and east nodes. In regard to the surfaces, $S_W = 2\pi R_{el}(i-1)H$ and $S_E = 2\pi R_{el}(i)H$, and the volume of the discrete volume equals $\forall_p = \pi \left(R_{el}(i)^2 - R_{el}(i-1)^2\right) H$. The parameter $\dot{q}_{\forall p}$ is the internal heat generated inside the material, if existing, in $[W/m^3]$. It can be a function of the time and position and it is going to be used in the verification of the code.

Inserting the equation *(2.13)* into the equation *(2.7)*, the following discretization coefficients are obtained:

$$\begin{cases} a_P = \beta \cdot \left[\lambda_w^{n+1} \frac{S_W}{d_{PW}} + \lambda_e^{n+1} \frac{S_E}{d_{PE}}\right] + \frac{\rho_p \forall_p \bar{c_p}_p}{\Delta t} \\ a_E = \beta \cdot \lambda_e^{n+1} \frac{S_P}{d_{PE}} \\ a_W = \beta \cdot \lambda_w^{n+1} \frac{S_W}{d_{PW}} \\ b_P = \frac{\rho_p \forall_p \bar{c_p}_p}{\Delta t} T_P^n + (1-\beta) \sum \dot{Q}^n + \beta \cdot \dot{q}_{\forall p}^{n+1} \forall_p \end{cases} \tag{3.14}$$

It is necessary to note that the density and specific heat are considered constants in temperature and time, although the code is able to consider these dependencies.

3.1.3. Last node discretization coefficients

The last node ($i = N+1$) is located at the right side of the last control volume, thus, its volume is zero ($V_p = 0$). In this case, it is only necessary to consider the convective heat between the outer side of the wall of the tank with the external medium, and the conduction heat with the left node (west). The summation of heat at the new instant yields:

$$\sum \dot{Q}^{n+1} = \lambda_w^{n+1} \frac{T_W^{n+1} - T_P^{n+1}}{d_{PW}} S_P - \alpha_{ext}^{n+1} \cdot \left(T_P^{n+1} - T_{ext}^{n+1}\right) S_P \tag{3.15}$$

The surface equals $S_P = 2\pi R_{el}(i = N+1) \cdot H$ and T_{ext} is the external temperature of the air. The last can be a function of the time, and the free convective coefficient has to be calculated using an empirical approach. Introducing the equation *(2.15)* into the equation *(2.7)*, the obtained discretization coefficients are:

$$\begin{cases} a_P = \beta \cdot \left[\alpha_{ext}^{n+1} S_P + \lambda_w^{n+1} \frac{S_P}{d_{PW}}\right] \\ a_E = 0 \\ a_W = \beta \cdot \lambda_w^{n+1} \frac{S_P}{d_{PW}} \\ b_P = \beta \cdot \alpha_{ext}^{n+1} T_{ext}^{n+1} S_P + (1-\beta) \sum \dot{Q}^n \end{cases} \tag{3.16}$$

3.2. Internal convective coefficient

The internal convective coefficient α_{int} has been calculated using empirical approaches for a forced convection inside circular ducts. The mean Nusselt number ($\bar{Nu}$) is calculated as follows:

$$\bar{Nu} = C Re^m Pr^n K \tag{3.17}$$

where C is a constant and K is the dimensionless correction function in the Nusselt equation. The dimensionless parameters Re (Reynolds' number) and Pr (Prandtl number) are calculated as follows:

$$Re = \frac{\rho \bar{v} D}{\mu}, \tag{3.18}$$

$$Pr = \frac{\mu c_p}{\lambda} \tag{3.19}$$

The density ρ, the specific heat c_p, the conductive coefficient λ and the dynamic viscosity μ of the internal fluid (water) are calculated by means of the following empirical relations:

$$\rho_{H_2O}(T) = 847.2 + 1.298 \cdot T - 2.657 \cdot 10^{-3} \cdot T^2 \left[\frac{kg}{m^3}\right] \tag{3.20}$$

$$c_{p_{H_2O}}(T) = 5648.8 - 9.140 \cdot T + 14.21 \cdot 10^{-3} \cdot T^2 \left[\frac{J}{kgK}\right] \tag{3.21}$$

$$\lambda_{H_2O}(T) = -1.176 + 7.915 \cdot 10^{-3} \cdot T + 1.486 \cdot 10^{-5} \cdot T^2 - 1.317 \cdot 10^{-7} \cdot T^3 + 2.476 \cdot 10^{-10} \cdot T^4 - 1.556 \cdot 10^{-13} \cdot T^5 \left[\frac{W}{mK}\right] \tag{3.22}$$

$$\mu_{H_2O}(T) = 0.9149 - 1.2563 \cdot 10^{-2} \cdot T + 6.9182 \cdot 10^{-5} \cdot T^2 - 1.9067 \cdot 10^{-7} \cdot T^3 + 2.6275 \cdot 10^{-10} \cdot T^4 - 1.4474 \cdot 10^{-13} \cdot T^5 \left[\frac{kg}{ms}\right] \tag{3.23}$$

The temperature T corresponds to the internal mean temperature of the tank T_{int}, which is a known function in time. This set of equations is valid at any instant for temperatures between 0^{o}C and 80^{o}C.

The mean velocity $\bar{v}$ is calculated employing the mass conservation equation with the diameter of the tank, due to the direction of the flow from the bottom to the top section of the tank. The volumetric flow circulating through D02 is known, as it is the same that the P11 pump is moving, and it is introduced as an input in the simulation. The mass conservation equation yields:

$$\bar{v} = \frac{Q}{\pi \frac{D^2}{4}} \tag{3.24}$$

For the range of possible values of the volumetric flow involved in the circuit (from 380 m^3/h to 650 m^3/h), the Reynolds' number is always greater than 2500 at the working temperatures. Thus, the fluid inside the tank is going to have a turbulent behaviour for all the possible scenarios. The mean Nusselt number is calculated using the equation *(3.17)*, where the involved constants are calculated as follows:

C	m	n	K	Operating conditions
0.027	0.8	0.33	$\left(\frac{\mu}{\mu_w}\right)^{0.14}$	Turbulent high viscous liquids with $0.6 < \mathrm{Pr} < 100$

Table 2: Nusselt number empirical approach for a forced convection.

Where the range of values of the Prantdl number has been checked and confirmed in the simulation. The parameter μ_w is the dynamic viscosity of the water at the temperature of the inner wall of the tank. This temperature is unknown at the beginning of the calculation. Thus, it is necessary to develop an iterative algorithm in order to calculate α_{int} and the temperatures through the wall. Finally, the mean internal convective coefficient is calculated as:

$$\alpha_{int} = \lambda \frac{\bar{Nu}}{D} \tag{3.25}$$

3.3. External convective coefficient

The external convective coefficient α_{ext} has been calculated using empirical approaches for a natural convection in a vertical cylinder of large diameter. The mean Nusselt number yields:

$$\bar{Nu} = CR_a^n K \tag{3.26}$$

where C and K are constants and R_a is the Rayleigh number calculated as the product of the Grashof (Gr) and Prandtl numbers.

$$R_a = Gr \cdot Pr = \frac{g\beta\rho^2|T_w - T_f|H^3}{\mu^2} \cdot \frac{\mu c_p}{\lambda} \tag{3.27}$$

The parameter β is the thermal expansion coefficient of air, g is the gravity constant and H is the height of the D02 tank. T_w is the temperature of the external wall (unknown at the moment of the calculation, which yields an iterative process), and T_f is the temperature of the fluid (known at any instant). The fluid properties of the air (assuming semi-perfect gas hypothesis) are calculated with the following set of equations:

$$\rho_{air}(T) = \frac{P}{287 \cdot T} \left[\frac{kg}{m^3}\right] \tag{3.28}$$

$$\begin{aligned} c_{p_{air}}(T) = 1034.09 - 2.849 \cdot 10^{-1} \cdot T + 7.817 \cdot 10^{-4} \cdot T^2 \\ -4.971 \cdot 10^{-7} \cdot T^3 + 1.077 \cdot 10^{-10} \cdot T^4 \left[\frac{J}{kgK}\right] \end{aligned} \tag{3.29}$$

$$\lambda_{air}(T) = \frac{2.648 \cdot 10^{-3} \cdot \sqrt{T}}{1 + (245.4/T) \cdot 10^{-12/T}} \left[\frac{W}{mK}\right] \tag{3.30}$$

$$\mu_{air}(T) = \frac{1.458 \cdot 10^{-6} \cdot T^{1.5}}{T + 110.40} \left[\frac{kg}{ms}\right] \tag{3.31}$$

$$\beta_{air}(T) = \frac{1}{T} \left[K^{-1}\right] \tag{3.32}$$

The equations above are evaluated at the film temperature $T_m = (T_w + T_f)/2$, where P is the pressure in the Synchrotron elevation (evaluated with the ISA atmosphere). If the Rayleigh number is greater than 10^9, the flow is considered turbulent. In the range of operation of the tank, the Rayleigh number takes values between 10^{10} and 10^{11}. Thus, the natural convection involves air with a turbulent behaviour. The involved constants in the equation *(3.26)* are calculated by means of the following table:

C	n	K	Operating conditions
0.0246	2/5	$\left[\frac{Pr^{1/6}}{1+0.494 \cdot Pr^{2/3}}\right]^{2/5}$	Turbulent flow

Table 3: Nusselt number empirical approach for a natural convection.

Finally, the mean external convective coefficient is calculated as:

$$\alpha_{ext} = \lambda \frac{\bar{Nu}}{H} \tag{3.33}$$

3.4. Steady state discretization coefficients

To finish up, the discretization coefficients for the starting steady state (when t = 0 sec) are presented below. As the case is a particularization of the coefficients of the transient state and the summation of heat is equivalent, the equations are not going to be developed again.

3.4.1. First node discretization coefficients

The discretization coefficients for the first node yield:

$$\begin{cases} a_P = \alpha_{int} S_P + \lambda_e \frac{S_P}{d_{PE}} \\ a_E = \lambda_e \frac{S_P}{d_{PE}} \\ a_W = 0 \\ b_P = \alpha_{int} T_{int} S_P \end{cases} \tag{3.34}$$

3.4.2. Intermediate node discretization coefficients

The discretization coefficients for the intermediate nodes yield:

$$\begin{cases} a_P = \lambda_w \frac{S_W}{d_{PW}} + \lambda_e \frac{S_E}{d_{PE}} \\ a_E = \lambda_e \frac{S_P}{d_{PE}} \\ a_W = \lambda_w \frac{S_W}{d_{PW}} \\ b_P = \dot{q}_{\forall p} \forall_p \end{cases} \tag{3.35}$$

3.4.3. Last node discretization coefficients

Finally, the discretization coefficients for the last node yield:

$$\begin{cases} a_P = \alpha_{ext} S_P + \lambda_w \frac{S_P}{d_{PW}} \\ a_E = 0 \\ a_W = \lambda_w \frac{S_P}{d_{PW}} \\ b_P = \alpha_{ext} T_{ext} S_P \end{cases} \tag{3.36}$$

4. Algorithm and verification

4.1. TDMA solver

The algorithm used to solve the equation system defined by equations *(3.8)* is the TDMA[2] due to the tri-diagonal nature of the equation system. This algorithm solves all the temperatures in a direct manner, thus, the iterations will by necessary to solve the non-constant physical parameters and not for the resolution of the system, reducing the computing time.

We want to solve the following equation system:

$$a_P[i]T_P^{n+1}[i] = a_E[i]T_E^{n+1}[i] + a_W[i]T_W^{n+1}[i] + b_P[i] \tag{4.1}$$

Defining the coefficients $P[i]$ and $R[i]$ as:

$$P[i] = \frac{a_E[i]}{a_P[i] - a_W[i]P[i-1]}, \tag{4.2}$$

$$R[i] = \frac{b_P[i] + a_W[i]R[i-1]}{a_P[i] - a_W[i]P[i-1]} \tag{4.3}$$

It is possible to demonstrate that the temperatures can be calculated as:

$$T[i] = P[i]T[i+1] + R[i] \tag{4.4}$$

Once the coefficients *(4.2)* and *(4.3)* are calculated from the first to the last node, *(4.4)* must be solved from the last to the first node. This procedure is done for both steady and transient states temperature resolution.

4.2. Global Algoritm

A scheme of the global algorithm used to solve the problem is presented below.

1. **Input parameters:** definition of the input parameters by the user.
   a) Geometry parameters such as the internal, intermediate and outer radii of the tank, and height of the tank.
   b) Tank and isolation materials definition.
   c) External and internal temperatures evolution functions.
   d) Internal heat load of the walls function definition (if needed).
   e) Physical parameters of the air and water, defined in *(3.2) and (3.3)*.
   f) Numerical parameters such as the number of nodes, increment of time per iteration (Δt), convergence tolerance (δ), start temperature for the steady state, numerical scheme (β) and maximum time criterion. The code also gives the possibility of saving the temperature distribution at any instant, with a parameter called *save_time*.
   g) Specific parameters of the problem such as the volumetric flow circulating inside the tank.
2. **Previous calculation:** calculus previous to the iterative method.
   a) Geometry discretization, taking into account the equation *(3.11)* for the faces of the control volumes, and node-centering the nodes.
   b) Determination of the material associated to each node, creating a vector called T_{mat} with the number of each material.
   c) Calculations such as the flow circulating area of the tank, velocity of the flow with the mass conservation equation, and initial physical parameters of the air and water.

[2]Tri-Diagonal Matrix Algorithm.

d) Initialization of the wall temperature $T^* = T_{start}$, doing the first temperature assignation before the iterations for the steady state solution. Initialize also the time vector as $t(1) = 0$.

3. **Steady state solution**: iterative algorithm to solve the steady state temperatures, defined as the temperatures at t = 0 sec, first set of values of T^n.
   a) Evaluation of the internal and external convective coefficients as defined in the section *(3.2)* and *(3.3)*, with the temperatures of the fluids evaluated at $t = 0$ *sec*.
   b) Evaluation of the thermal conductivity coefficient of each control volume face at the temperatures T^*, by means of the harmonic mean.
   c) Calculation of the discretization coefficients as defined in section *(3.4)*, evaluating the heat internal load ($q_{\forall_p}$) at $t = 0$ *sec*.
   d) Apply the TDMA solver as defined in section *(4.1)* to obtain the vector T.
   e) Is $max|T - T^*| < \delta$?
      i. If it is not, save $T^* = T$ and go to *3a)* for a new iteration.
      ii. If it is, stop the iterations. Save vector $T^n = T$.

4. **Transient state solution**: iterative algorithm to solve the evolution of the temperatures in time. The solution yields T^{n+1}, and a criterion of maximum time is applied in order to go forward in time.
   a) Initialization of the vector $T^* = T^n$. This represents the initial temperature for the iterations over T^{n+1}. Calculation of $t^{n+1} = t^n + \Delta t$, where $t^n = 0$ *sec* in the first iteration.
   b) Evaluation of the internal and external convective coefficients as defined in the section *(3.2)* and *(3.3)*, with the temperatures of the fluids evaluated at t^n and t^{n+1}. This yields both convective coefficients evaluated at the present and next instant.
   c) Evaluation of the thermal conductivity coefficient of each control volume face both in T^n and T^*, by means of the harmonic mean. This yields the conduction coefficients evaluated at the present and next instant.
   d) Calculation of the discretization coefficients as defined in section *(3.1)*, evaluating the heat internal load at t^n (for the calculation of $\sum \dot{Q}^n$) and t^{n+1} (for the evaluation of b_P).
   e) Apply the TDMA solver as defined in section *(4.1)* to obtain the vector T^{n+1}.
   f) Is $max|T^{n+1} - T^*| < \delta$?
      i. If it is not, save $T^* = T^{n+1}$ and go to *4b)* for a new iteration.
      ii. If it is, evaluate the maximum simulation time criterion: is $t < t_{max}$?
         A. If it is not, save $T^n = T^{n+1}$ and go to *4a)*. Save variables of interest for futures plots.
         B. If it is, stop the iterations.

5. **Final calculations and results plotting:** visual representation of the results.
   a) First of all, the temperature distribution through the wall is represented for the initial, final, and saved instants.
   b) The temperature evolution of the inner wall, intermediate wall and outer wall is represented against time. These vectors are saved inside the transient state solver, in *4f)*.
   c) Represent the evolution of convective heat released by the water against time. This vector is also calculated in *4f)* as the conduction heat of the first layer of the inner wall.
   d) Integrate the heat evolution against time to obtain the energy loss of the water inside the tank. This integral is done with the trapezoidal method. The obtained value will be used as control parameter for a future convergence analysis.
   e) Plot of the evolution of the inner and exterior temperatures of the water and air, respectively.
   f) Calculation of the heat delivered in the inner and outer wall for the steady state, in order to compare them and verify if the code is working correctly.

4.3. MMS CODE VERIFICATION

In order to verify the unsteady solver, the MMS (Method of Manufactured Solutions) has been performed. This method consists on performing the simulation with an internal heat load calculated from applying an invented temperature distribution to the conduction differential equation in cylindrical coordinates. If the temperature distribution resulting from the simulation is equivalent to the one from the analytical analysis, the code works correctly. The differential conduction heat transfer equation, for one-dimension in cylindrical coordinates, yields:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(\lambda r \frac{\partial T}{\partial r}\right) + \dot{q}_{\forall} \tag{4.5}$$

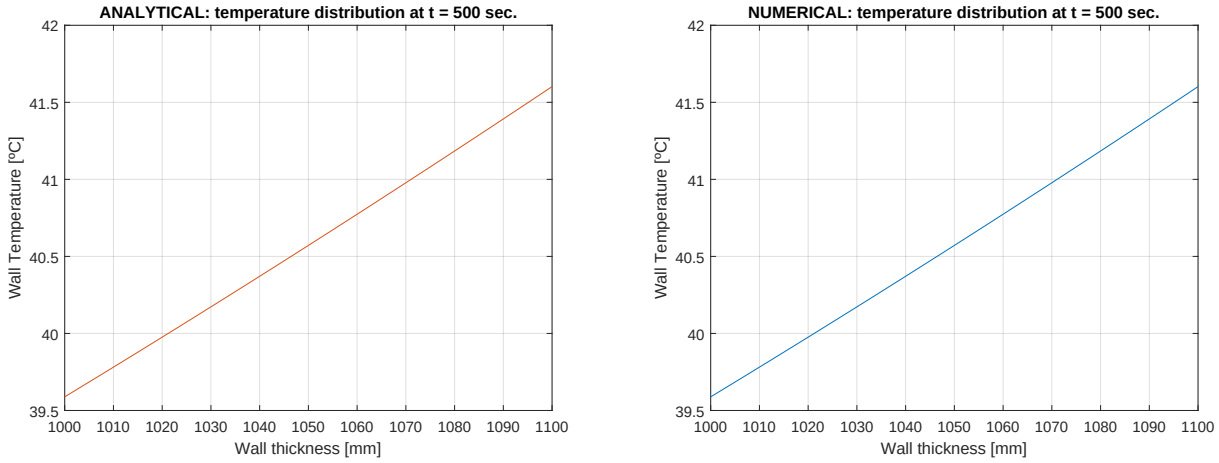
Starting from a random temperature distribution like:

$$T(r,t) = 303.15 + 20r^2 sin(\omega t) \tag{4.6}$$

Introducing *(4.6)* into *(4.5)* and isolating $\dot{q}_{\forall}$, it is possible to obtain:

$$\dot{q}_{\forall} = 20\rho c_p \omega r^2 \cos(\omega t) - 80\lambda \sin(\omega t) \tag{4.7}$$

The simulation is performed for only one cylindrical solid wall with the thermophysical properties defined in *table 1*, with an initial temperature of 303.15 K, $\omega = 10^{-3}$ rad/s, 5000 elements, and $\Delta t = 1$ sec. The steady solution is avoided because the initial temperature distribution shall match with the equation *(4.6)* evaluated at $t = 0$ sec. The wall radius is 2 meters, the height is 10 meters, and the simulation maximum time is 600 seconds. Finally, the temperature distribution of the wall is represented below at $t = 500$ sec for $1.0\ m < r < 1.1\ m$.



(a) Analytical temperature distribution.

(b) Numerical temperature distribution.

Figure 5: Temperature distribution comparison for the MMS verification.

As it is possible to see, the numerical temperature distribution matches with the analytical one with an error less than 0.01%. Thus, it is possible to conclude that the solver for the unsteady solution is working correctly.

4.4. Steady state energy balance

In order to verify the steady state solver from which the calculation is started, it is necessary to verify if a global balance of energy is achieved. This calculation is performed in a function for every simulation, in order to verify if the code has worked correctly.

In the steady state, the heat delivered by the inner wall must be equal to the heat delivered by the outer wall ($\dot{Q}_{in} = \dot{Q}_{out}$). This calculus is carried out taking into account the conduction heat transfer at the first and last layer of the wall, respectively.

$$\dot{Q}_{in} = \lambda_e \frac{T(1) - T(2)}{d_{PE}} S_E \tag{4.8}$$

$$\dot{Q}_{out} = \lambda_w \frac{T(end-1) - T(end)}{d_{PW}} S_W \tag{4.9}$$

It has no-sense to talk about specific values, as the simulation will give different values for each entrance parameter. In despite of this, without the effect of any insulator, the heat delivered equals 6.5138 W for the steady state for each inner and outer wall.

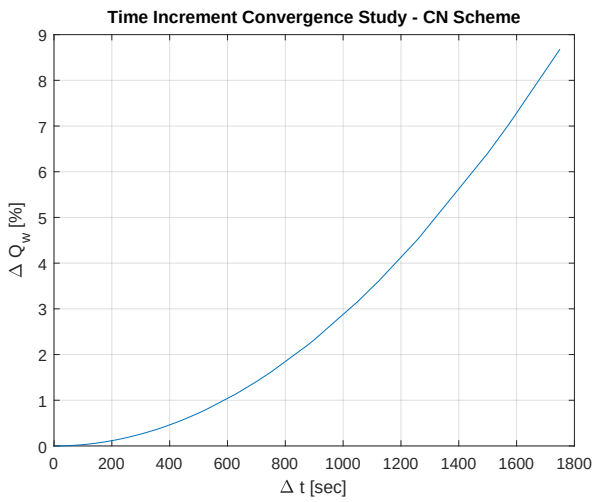
This is an important conclusion, as it is saying that the steady solver is working correctly, and the D02 tank delivers 6.5138 W of heat when there is no thermal instability.

5. Numerical parameters evaluation

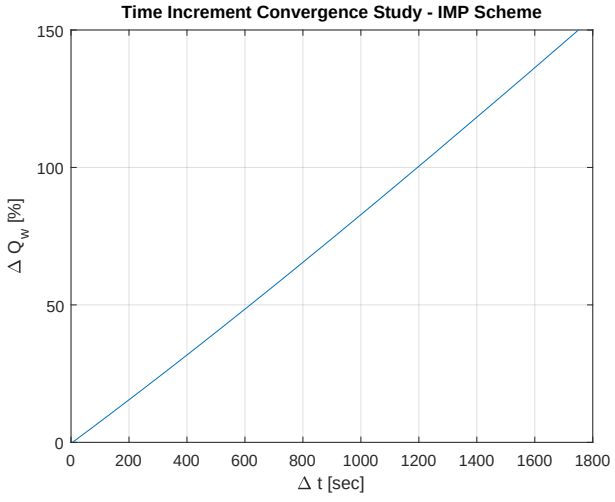
5.1. Numerical scheme and time increment

As the simulation is done in a large period of time (almost 9 hours) and the computing time of the computer is actually limiting, it is convenient to perform an analysis of the Δt used in the discretization of the time vector.

The Crank-Nicolson and Implicit numerical schemes are considered for this study, as the Explicit scheme is misplaced due to its non-unconditional stable nature with Δt and large period of simulation time. By performing the simulation at different Δt and saving a control parameter, such as the global heat loss of the water, it is possible to obtain the following plots:



(a) Crank-Nicolson scheme.



(b) Implicit scheme.

Figure 6: Relative variation of the parameter of control against the time increment.

The relative variation of the control parameter is calculated like a relative error with the value at $\Delta t = 1\ sec$ as reference. As expected, the Implicit scheme error has a 1st order dependency with the time increment, while the Crank-Nicolson scheme has a 2nd order dependency. Both of them are unconditional stable, so the election of one of the schemes must be guided by the computing time and the relative error.

As the error of the Implicit scheme is much higher than the error of the Crank-Nicolson scheme, the best election for this simulation is a Crank-Nicolson numerical scheme ($\beta = 0.5$). The chosen time increment is $\Delta t = 60\ sec$, since the computing time is short enough and almost gives no-error. Despite this, any value for the time increment less than $\Delta t \leq\ 1000\ sec$ would give an error less than 3%. So, from now on, every result will be done under the last statements.

5.2. Number of elements convergence

The convergence of the solver with the number of elements is studied in this section. In order to simplify the calculus, both tank and insulator walls have been simulated with the same number of elements. It is important to notice that for the realization of this calculus, the insulator thickness has been approached by higher values than the ones obtained in the following sections, making it a conservative approach. The chosen control parameter is the same as before, the total heat delivered by the convection of the inner water.

Performing the simulation at different numbers of elements (for each wall), the convergence curve yields:

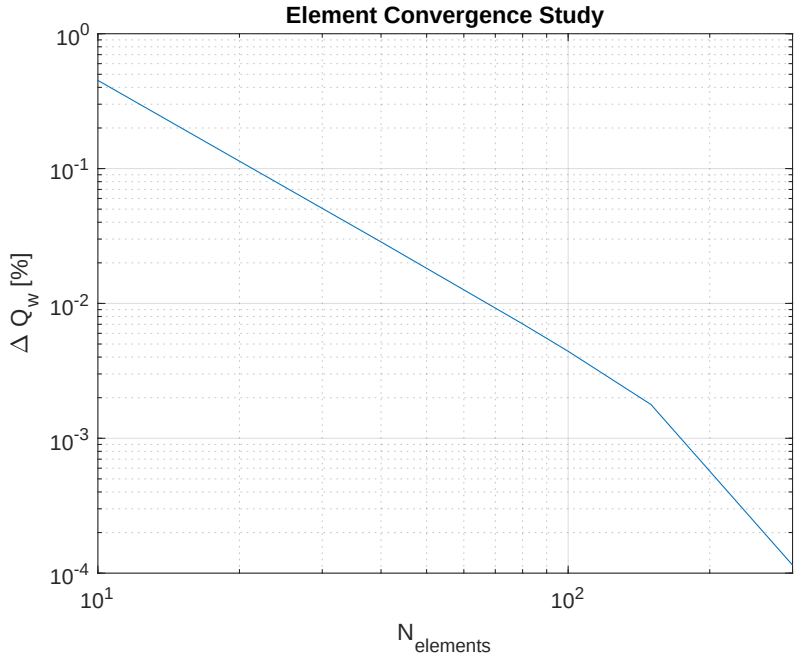


Figure 7: Convergence of the solver with the number of elements.

Note that the relative variation of the control parameter is calculated like a relative error with the value at $N_{element} = 1000$ as reference. The curve has been represented in a logarithmic scale for both axis. As it is possible to see, the number of elements required to obtain good results is not high, due to the thickness of the walls. One hundred elements for each wall will give an error of less than 0.01% and a reasonably good computational effort. Thus, the number of elements chosen is $N_{ins} = 100$ for the insulator wall, and $N_{wall} = 20$ for the tank's wall due to its smaller thickness. This will not increase the error significantly, although it will significantly decrease the computing effort.

6. Insulator design

In this section, proper insulation is to be designed from the thermophysical parameters to the thickness.

6.1. Thermophysical properties

Best insulation materials should have the lowest thermal conductivity, good price, and must adapt easy to the shape of the body. The durability of the insulator has to be taken into account in the election. A good insulator has a higher specific heat because it takes time to absorb more heat before it actually heats up to transfer the heat.

A wide range of insulation materials is available. For this simulation, a polyurethane foam insulator is chosen because it matches with the necessities. The properties of this insulator are summarized below.

$\lambda\ [W/mK]$	0.049	Thermal conductivity
$C_p\ [J/KgK]$	2500	Specific heat
$\rho\ [kg/m^3]$	30	Density

Table 4: Polyurethane foam insulator specifications.

As it is possible to see, both density and thermal conductivity are low while the specific heat is relatively high.

6.2. Analytical minimum radius for the steady state

Regarding an analytical solution for the wall at the steady state, the heat per unit of height yields:

$$\dot{q}^* = \frac{\pi\left(T_{int} - T_{ext}\right)}{\frac{1}{2\alpha_{int}r_{int}} + \frac{1}{2\alpha_{ext}r_{ins}} + \frac{1}{2\lambda_{wall}}\ln\left(\frac{r_{ext}}{r_{int}}\right) + \frac{1}{2\lambda_{ins}}\ln\left(\frac{r_{ins}}{r_{ext}}\right)} \tag{6.1}$$

The last equation takes into account the convection heat inside the tank, the conduction through the wall of the tank and the wall of the insulator, and the convection on the outer side. In regard to the nomenclature, r_{int} represents the inner radius, r_{ext} the external radius of the tank without insulator, and r_{ins} the external radius with insulator.

The denominator of the last equation is redefined as the thermal resistance:

$$R = \frac{1}{2\pi\alpha_{int}r_{int}} + \frac{1}{2\pi\alpha_{ext}r_{ins}} + \frac{1}{2\pi\lambda_{wall}}\ln\left(\frac{r_{ext}}{r_{int}}\right) + \frac{1}{2\pi\lambda_{ins}}\ln\left(\frac{r_{ins}}{r_{ext}}\right) \tag{6.2}$$

Which can be minimized by differentiating it over the insulator radius to obtain the critical radius:

$$\frac{dR}{dr_{ins}} = 0 \quad \longrightarrow \quad r_{cr} = \frac{\lambda_{ins}}{\alpha_{ext}} \tag{6.3}$$

As the external convective heat transfer coefficient of air takes a mean value of $\alpha = 0.097\ W/m^2K$ for the steady state, the critical radius takes a value of:

$$r_{cr} = 0.51\ m$$

As it is possible to see, the critical radius is minor than the internal radius of the tank. This means that every possible thickness for the insulator will produce an increase in the insulating effect, working properly in the steady state.

6.3. Numerical radius for the transient state

In this section, an optimal thickness for the insulator will be determined taking into account the transient behaviour caused by the thermal instabilities. A simulation is performed for several different values of the external radius of the insulator, saving the total heat delivered by the internal water as control parameter. The result yield:

Figure 8: Relative heat delivered by the water against the insulator thickness.

The delivered heat is expressed in a relative way. It is important to remember that Q is calculated as the integral of $\dot{Q}$ over the thermal instabilities. Thus, it is calculated in *Joules*. There is a clear minimum in:

t_{ins} $[mm]$	48	Insulator wall thickness
ΔQ [%]	-17	Delivered heat relative variation

Table 5: Optimal insulator parameters.

These are the optimal parameters for the insulator, as they minimize the heat loss of the water when the thermal instabilities happen.

The next table compares the effect of installing the insulator in both steady and transient regimes.

Performance comparison	Steady $\dot{Q}$ $[W]$	Unsteady Q $[J]$
Current performance	6.5138	617823
Performance with insulator	5.8657	512604
Relative variation [%]	-10	-17

Table 6: Effect of installing the insulator.

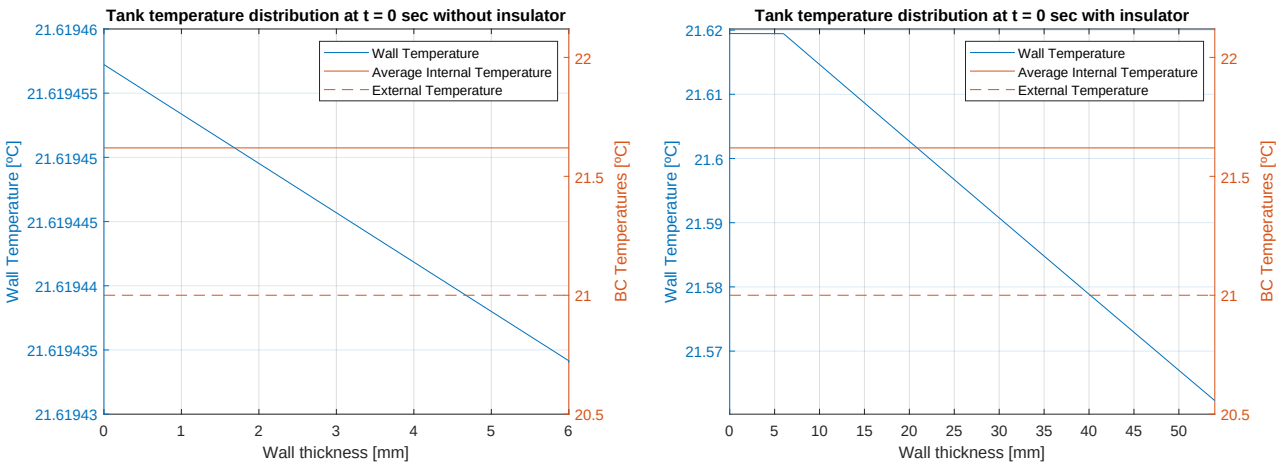
As it is possible to see, the insulator reduces in 10% and 17% the delivered heat in the steady and unsteady regimes, respectively. This performance could be improved for the steady regime, by increasing the thickness of the insulator as seen in *section 6.2*, but this would have a counterproductive effect on the performance of the insulator when the thermal instabilities happen. Moreover, it would affect on the installation price.

7. Simulation results

In this section, the results of the simulations will be introduced for both steady and transient states, taking into account the effect of the insulator.

7.1. Initial temperature distribution

The temperature distribution of the wall in the steady state, before the thermal instabilities, is represented in the plots below.



(a) Non-insulation temperature distribution.

(b) Temperature distribution with insulation.

Figure 9: Temperature distribution in the walls with and without insulation.

On the one hand, it is possible to conclude that the temperature gradient through the wall of the tank without insulator is almost nonexistent due to its order of magnitude. On the other hand, including an insulation would make this temperature gradient bigger, as the temperature would drop 0.05ºC in the insulator wall due its small conductive heat transfer coefficient.

7.2. Heat flux and temperatures evolution

The time evolution of the temperatures of the three walls (internal wall of the tank, external wall of the tank, and external wall of the insulation) is represented in this section. Moreover, the evolution of the heat power flux is also represented, in order to understand the behaviour of the heat transfer during the thermal instabilities.

For the case without the insulation:

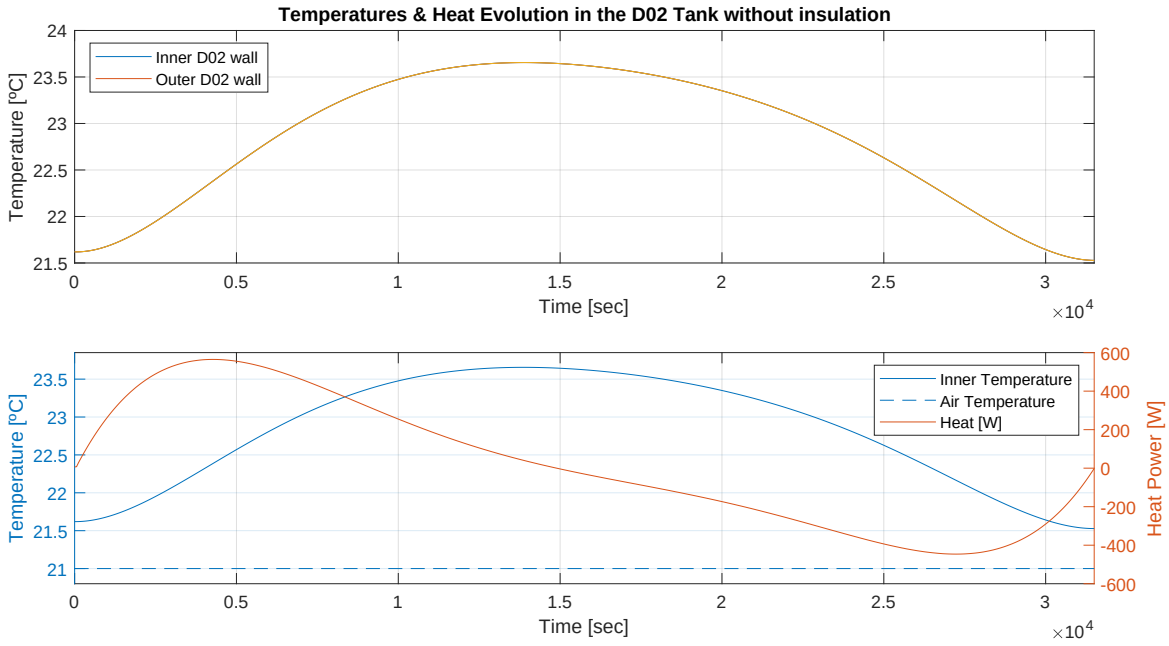


Figure 10: Heat flux and temperature evolution against time for the non-insulation case.

For the case with the insulation:

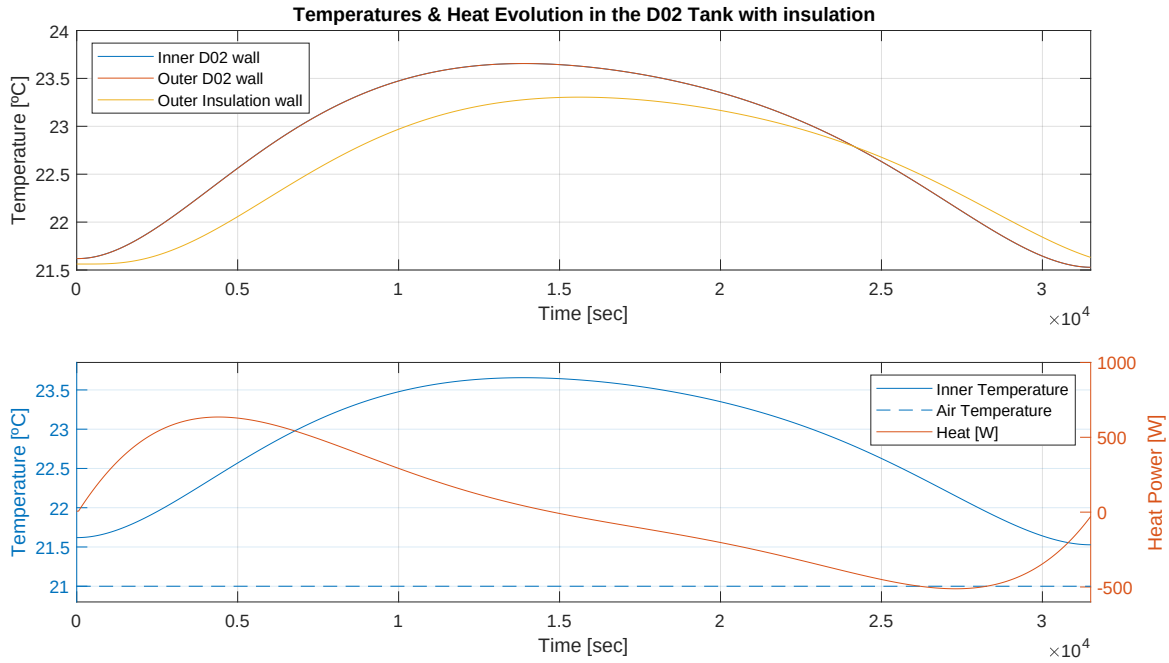


Figure 11: Heat flux and temperature evolution against time for case with insulation.

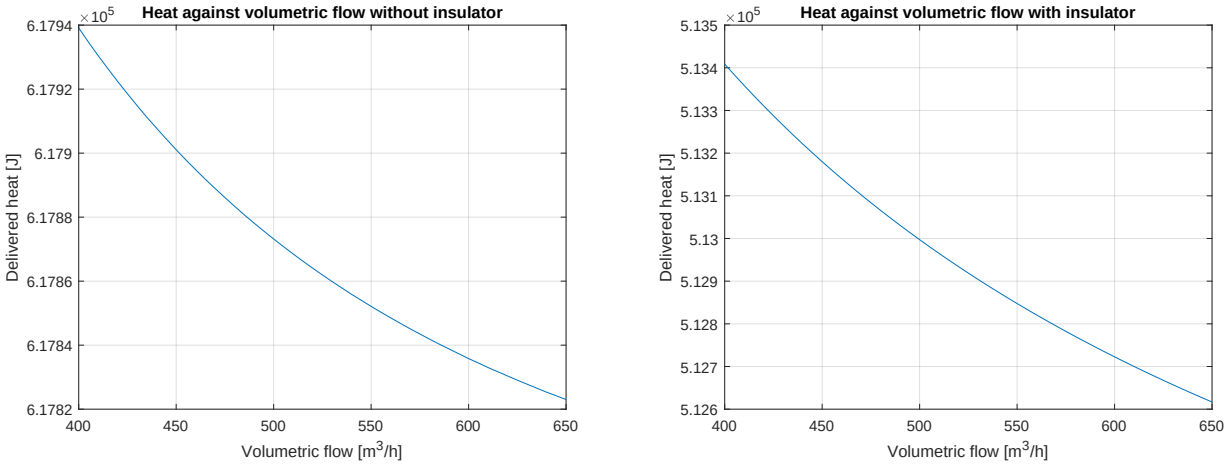
Is possible to conclude that the temperatures of the tank's wall follow the same tendency as the temperature of the water inside the tank, for both cases. The delivered heat is positive (abandons the water) while the temperature of the water inside the tank increases, until it reaches its maximum. After that, the heat flow turns negative, as the inner wall gets cooler faster than the outer wall.

The insulation prevents the heat to abandon the water, and the area under the heat power curve is -17% less for the case with insulation. Regarding to the temperature of the insulator, it follows a different tendency than the temperature of the D02 wall, due to its thermophysical properties. As the c_p value is higher for the insulation, it gets warmer slower than the walls of the tank, making the temperature gradient greater in the insulation zone when the inner temperature increases. The counter effect happens when the temperature decreases, and the D02 wall gets cooler faster than the insulation wall.

Finally, it is important to note that the external air temperature is set at 21^{o}C, due to the acclimatization of the ALBA's installation where the D02 tank is located.

7.3. Effect of the volumetric flow

As the P11 pump is under cavitation phenomena, the volumetric flow that circulates within the circuit is being reduced from 650 m^3/h to 400 m^3/h to avoid the cavitation. In this section, the effect of this reduction is going to be studied inside the tank for the transient behaviour. The following plot is obtained modifying the volumetric flow that circulates through the circuit and saving the water delivered heat as control parameter.



(a) Non-insulation case.

(b) Insulation case.

Figure 12: Delivered heat against volumetric flow.

As it is possible to see, the heat delivered by the water during the thermal instability increases when the volumetric flow is reduced. In despite of this, the variation is minimal, as it increases 0.02% for the non-insulation case and 0.15% for the insulation case. Thus, it is possible to conclude that the reduction of the mass flow is not injuring in a relevant manner the heat transfer when the thermal instabilities occur.

8. Conclusions

Concluding, the effect of the thermal instabilities into the D02 tank has been studied by means of a self-made one-dimensional matlab simulation. The code solves numerically in a transient manner the combined problem, taking into account the convection inside the tank, the conduction through its walls, and the external natural convection. The following conclusions have been extracted:

- The effect of the thermal instabilities in the temperature of the four rings and tank is studied with actual data. By doing this, the temperature evolution of the water inside the tank is described in a mathematical model using statistical methods.
- The employed algorithm has been presented schematically, and the code has been verified, concluding that each part of it works correctly.
- The solver has been studied numerically, concluding that the best combination of parameters is 20 elements for the inner wall, 100 elements for the insulator wall, a Crank-Nicolson numerical scheme, and a time increment of 60 seconds.
- The studied insulator is the polyurethane foam, which optimal situation for the problem is found for an insulator thickness of 48 mm.
- The heat flux delivered by the water in the steady state, without taking into account any thermal instability, is about 6.51 W. Installing the proposed insulation would decrease this value by 10%. Any insulation thickness greater than the proposed would increase the insulating effect in the steady state, although it would disserve it when the instabilities occur.
- About 618 kJ of energy are lost by the water when the thermal instabilities occur. The proposed insulation would decrease this number by 17%, which is the optimal situation.
- It is concluded that the temperature gradient through the wall of the tank is really tiny. Installing the insulator would increase this feature, as expected due to its small conductive heat transfer coefficient.
- The evolution of the heat flux delivered by the water is compared to the evolution of the temperature of the internal fluid against time, with and without the insulation, in order to understand the behaviour of the heat transfer when the thermal instabilities occur.
- Finally, the reduction of the volumetric flow due to the cavitation phenomena appeared in the P11 pump is not affecting the heat transfer in a relevant manner, when the thermal instabilities occur.

To sum up, a powerful tool has been designed in the development of the calculus algorithm employed to solve this problem. ALBA has got a new tool, which could be used to study related issues.

References

[1] Alba initiates its upgrade process to become a 4th generation facility. https://www.albasynchrotron.es/en/media/news/alba-initiates-its-upgrade-process-to-become-a-4th-generation-facility. (Accessed on 05/25/2021).

[2] Alba main page. https://www.cells.es/en. (Accessed on 05/25/2021).

[3] Engineering materials specifications and characteristics tables and charts - ferrous and non-ferrous | engineers edge. https://www.engineersedge.com/manufacturing_menu.shtml. (Accessed on 06/22/2021).

[4] Thermal insulation materials, technical characteristics and selection criteria. http://www.fao.org/3/y5013e/y5013e08.htm. (Accessed on 06/22/2021).

[5] Bergadà. *Mecánica de fluidos : breve introducción teórica con problemas resueltos*. Iniciativa Digital Politècnica, Barcelona, 2017. ISBN: 9788498806885.

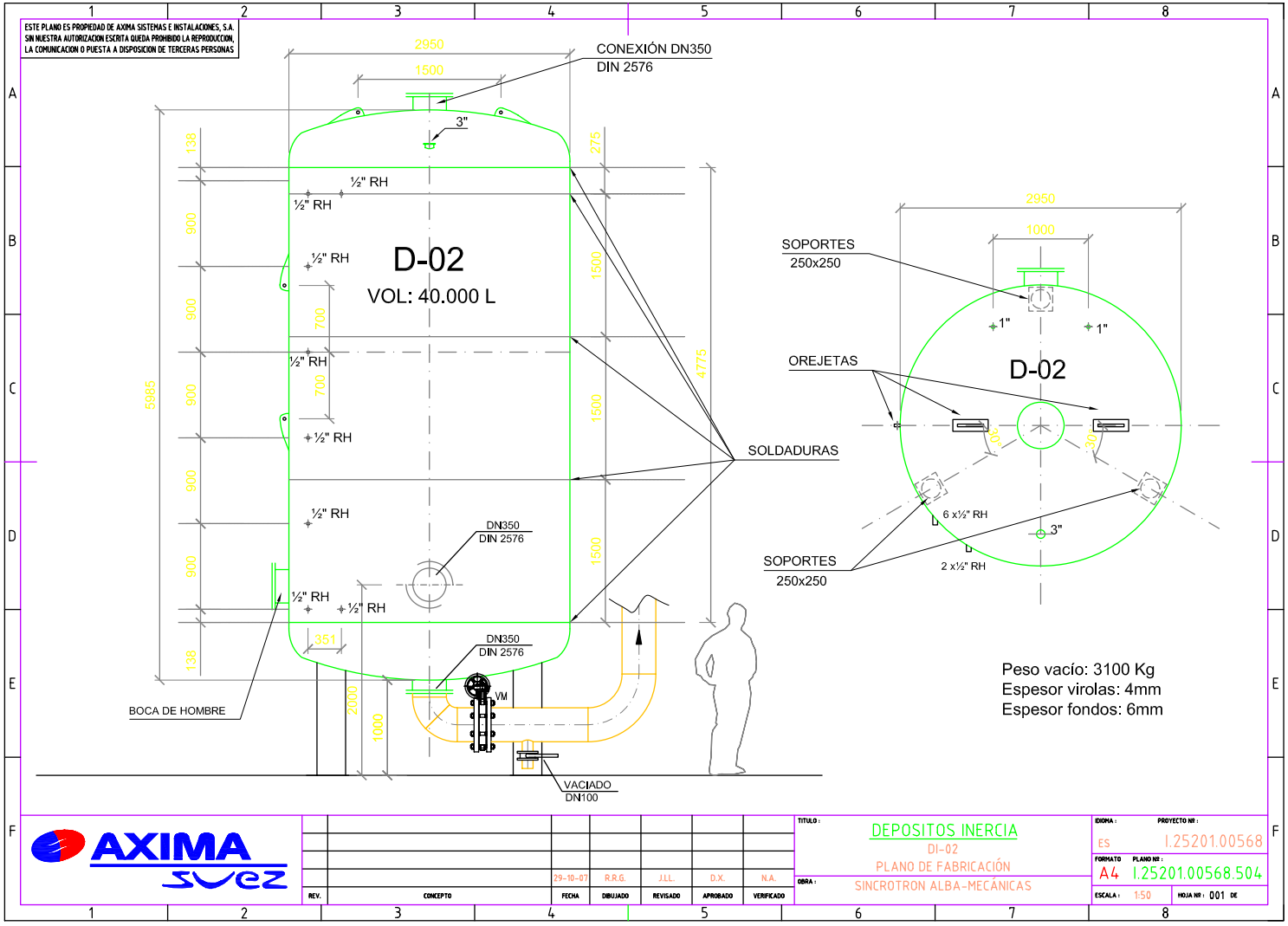
[6] Frank Incropera. *Fundamentos de transferencia de calor*. Prentice-Hall, México, 1999. ISBN: 9701701704.

[7] Irving Shames. *Mechanics of fluids*. McGraw-Hill, New York, 1982. ISBN: 0070563853.

[8] Michael Spearpoint, Dennis Pau, Charles Fleischmann, and K.Y. Li. Thermophysical properties of polyurethane foams and their melts. *Fire and Materials*, 06 2014.

A. Appendix

A.1. Appendix I: Drawing of the D02 tank



A.2. Appendix II: Main code (main.m)

```
%-------------------------------------------------------------------------
%           CONDUCTION HEAT TRANSFER SOLVER - CILINDRICAL TANK
%      ONE-DIMENSIONAL UNSTEADY: Implicit, Explicit & Crank-Nicolson
%          METODH: NODE CENTERED DISCRETIZATION, FINITE VOLUMES
%                       ALGORITHM: ITERATIVE TDMA
%              AUTHOR: Francesc H. G. (fran.14hg@gmail.com)
%-------------------------------------------------------------------------
%% MAIN
clc, clear all, close all

%% -----------------------------------------------------------------------
%                               1. INPUT
%           User definition of the different solver's inputs
%-------------------------------------------------------------------------

% 1.1. Volumetric Flow
Q = 650;                    % [m^3/h]

% 1.2. Geometry
% Internal tank wall, external tank wall, external insulation wall
R = [2.944, 2.950, 2.998];  % Radius [m]
H = 5.38;                   % TANK height [m]

% 1.3. Numerical data
N = [20, 100];              % Number of elements of tank and insulation walls
tolerance = 1e-6;           % Convergence tolerance
Tstart = 20;                % Start temperature for first/steady solution [deg]
delta_t = 60;               % Time increment for unsteady solutions [sec]
time_max = 525*60;          % Maximum time for the unsteady simulation [sec]
save_time = 300*60;         % Save temperature distribution at this instant [sec]
beta = 0.5;                 % Heat integration method [0 (EXP), 0.5 (CN), 1 (IMP)]

% 1.4. MODIFY THE MATERIALS FUNCTION (materials.m)
% 1.5. MODIFY INTERNAL//EXTERNAL TEMPERATURE FUNCTION (BC_temperatures.m)
% 1.6. MODIFY INTERNAL VOLUMETRIC HEAT GENERATION FUNCTION (internal_heat.m)
% 1.7. MODIFY FLUID PARAMETERS FUNCTION IF NOT WATER (fluid_parameters.m)

%% -----------------------------------------------------------------------
%                             2. CALCULUS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 2.1. Geometry discretization
[r_node,r_element] = node_distribution(R,N);

% 2.2. TDMA based SOLVER for steady solution (time=0)
[T_initial,iteration,Tmat,time] = ...
    steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q);

% 2.3. TDMA based SOLVER for unsteady solutions
[T_wall_D02_int,T_wall_D02_ext,T_insulation_ext,T_instant,T_final,Qf,time] = ...
    unsteady_solver(T_initial,N,H,R,r_node,r_element, ...
    Tmat,time,delta_t,time_max,beta,tolerance,save_time,Q);

%% -----------------------------------------------------------------------
%                             3. RESULTS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 3.1. Initial Temperature plot
instant_plot (T_initial,r_node,0);

```

```
% 3.2. Saved Instant Temperature plot
instant_plot (T_instant,r_node,save_time);

% 3.3. Final Temperature plot
instant_plot (T_final,r_node,time(end));

% 3.4. Boundaries temperatures vs time plot
time_plot (T_wall_D02_int,T_wall_D02_ext,T_insulation_ext,Qf,time);

% 3.5. Global heat balance ( + if outgoing, - if ingoing )
Qw = trapz(time,Qf); % [J]

% 3.6. Steady verification
[Q_in_st,Q_out_st] = steady_verification(T_initial,Tmat,r_node,r_element,H);
```

A.3. Appendix III: Geometry discretization function (node_distribution.m)

```
function [r_node, r_element] = node_distribution(R,N)

% Node distribution
r_n(1,1) = R(1); r_el = R(1);
for j = 1:length(N)
    delta_r = (R(j+1)-R(j))/N(j);
    if j == 1
        for i = 2:(N(j)+1)
            r_n(i,j) = R(j) + delta_r/2 + (i-2)*delta_r;
            r_el(i,j) = R(1) + (i-1)*delta_r;
            ii = i;
        end
    else
        for i = 1:N(j)
            r_n(i,j) = R(j) + delta_r/2 + (i-1)*delta_r;
            r_el(i,j) = R(j) + i*delta_r;
            ii = i;
        end
    end
    jj = j;
end

r_n(ii+1,jj) = R(end);
r_node = nonzeros(r_n);
r_element = nonzeros(r_el);

end
```

A.4. Appendix IV: TDMA function (TDMA.m)

```
function [T] = TDMA(COEFF)

P(:,1) = zeros(size(COEFF,1),1);
R(:,1) = zeros(size(COEFF,1),1);

P(1,1) = COEFF(1,2) / COEFF(1,1);
R(1,1) = COEFF(1,4) / COEFF(1,1);

for i = 2:size(COEFF,1)
   P(i,1) = COEFF(i,2) / (COEFF(i,1)-COEFF(i,3)*P(i-1,1));
   R(i,1) = (COEFF(i,4) + COEFF(i,3)*R(i-1,1)) / ...
       (COEFF(i,1)-COEFF(i,3)*P(i-1,1));
end

T(size(COEFF,1),1) = R(size(COEFF,1),1);

for i = size(COEFF,1)-1:-1:1
    T(i) = P(i,1)*T(i+1,1) + R(i);
end

end
```

A.5. Appendix V: Fluid parameters function (fluid_parameters.m)

```
function [rho,Cp,mu,lambda] = fluid_parameters(T)

% Water (from 273 ¬ 573 K)
rho = 847.2 + 1.298*T - 2.657*10^(-3)*T^2;
Cp = 5648.8 - 9.140*T + 14.21*10^(-3)*T^2;
if T<353
    mu = 0.9149 - 1.2563*10^(-2)*T + 6.9182*10^(-5)*T^2 - ...
    1.9067*10^(-7)*T^3 + 2.6275*10^(-10)*T^4 - 1.4474*10^(-13)*T^5;
elseif T ≥ 353
    mu = 3.7471 * 10^(-2) - 3.5636 * 10^(-4)*T + 1.3725 * 10^(-6)*T^2 - ...
        2.6566 * 10^(-9)*T^3 + 2.5766 * 10^(-12)*T^4 - 1 * 10^(-15)*T^5;
end
lambda = -1.176 +7.915*10^(-3)*T + 1.486*10^(-5)*T^2- 1.317*10^(-7)*T^3 + ...
    2.476*10^(-10)*T^4 - 1.556 * 10^(-13)*T^5;

end
```

A.6. Appendix VI: Materials function (materials.m)

```
function [lambda,Cp,rho] = materials(T,Tmat)

% 1st MATERIAL (Stainless Steel)
if Tmat == 1
    lambda = 17;  % Material conduction coeff. [W/mK]
    Cp = 480;     % Specific heat [J/(kg K)]
    rho = 7500;   % Density [m^3/h]

% 2nd MATERIAL (Polyurethane Foam)
elseif Tmat == 2
    lambda = 0.049;
    Cp = 2500;
    rho = 30;

end

% As many materials as zone changes (length(R)-1)

end
```

A.7. Appendix VII: Boundary conditions temperatures function (BC_temperatures.m)

```
function [Tint,Text] = BC_temperatures(time)

% Internal D02 temperature [K]
Tint =  0.000000000000000066861148862.*(time./60+19).^6 - ...
    0.000000000011664075326818 8.*(time./60+19).^5 ...
    + 0.000000007983466317841320 0.*(time./60+19).^4 - ...
        0.000002677823959455010000 0.*(time./60+19).^3 ...
    + 0.000409777874932320000000 0.*(time./60+19).^2 - ...
        0.013118631747985600000000 0.*(time./60+19) ...
    + 21.738203470382900000000000 + 273.15;

% External temperature [K]
Text = 21+273.15;

end
```

A.8. Appendix VIII: Internal heat function (internal_heat.m)

```
function [qvp] = internal_heat(r,time)

% Internal heat volumetric generation [W/m^3]
% Depends on the position (r) and time.

% For the MMS method use:
% qvp = 20*rho*Cp*w*r^2*cos(w*time) - 80*lambda*sin(10^(-3)*time);

% For the usual simulation use:
qvp = 0;

end
```

A.9. Appendix IX: Steady solution function (steady_solver.m)

```
function [T_initial,iteration,Tmat,time] = ...
    steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q)

% Find node materials
for i = 1:size(r_node,1)
    Tmat(i,1) = node_materials(R,r_node(i,1));
end

% Initialization
time = 0;
[Tint, Text] = BC_temperatures(time);
T (:,1) = zeros(length(r_node),1);
T_prev (:,1) = zeros(length(r_node),1);

% 1st assignation
T_prev(:,1) = Tstart + 273.15;
iteration = 0; error = 1;

% Loop
while error > tolerance
    COEFF = steady_coeff(T_prev,R,r_node,r_element,H,Tint,Text,Tmat,Q,time);
    T = TDMA(COEFF);

    error = max(abs(T(:,1)-T_prev(:,1)));
    T_prev(:,1) = T(:,1);
    iteration = iteration + 1;
end

% K to deg
T_initial = T-273.15;

end
```

A.10. Appendix X: Node materials function (node_materials.m)

```
function [Tmat] = node_materials(R,r)

for i = 1:(length(R)-1)
   if r < R(i+1) && r ≥ R(i)
       Tmat = i;
   elseif r == R(length(R))
       Tmat = length(R)-1; % last node
   end
end

end
```

A.11. Appendix XI: Steady coefficients function (steady_coeff.m)

```
function [COEFF] = steady_coeff(T,R,r_node,r_element,H,Tint,Text,Tmat,Q,time)

%{
 COEFFICIENT COLUMNS
    - 1st. ap (Tp coefficient)
    - 2nd. ae (Te coefficient)
    - 3rd. aw (Tw coefficient)
    - 4th. bp (Independent)
%}

COEFF = zeros(length(r_node),4);

%% Convective coefficients calculation
[alpha_int,alpha_ext] = convection_alpha(Q,R,H,T(1,1),T(end,1),time);

%% 1st node
[¬, lambda_e] = harmonic_lambda(T,Tmat,r_node,r_element,1);
dPE = r_node(2,1)-r_node(1,1);

ae = lambda_e/dPE;
ap = ae + alpha_int;
bp = alpha_int*Tint;

COEFF(1,:) = [ap ae 0 bp];

%% Inner nodes
for i = 2:(length(r_node)-1)

    qvp = internal_heat(r_node(i,1),0);
    [lambda_w, lambda_e] = harmonic_lambda(T,Tmat,r_node,r_element,i);
    dPE = r_node(i+1,1)-r_node(i,1);
    dPW = r_node(i,1)-r_node(i-1,1);

    re = r_element(i);
    rw = r_element(i-1);
    Se = 2*pi*re*H;
    Sw = 2*pi*rw*H;
    Vp = pi*(re^2-rw^2)*H;

    ae = lambda_e*Se/dPE;
    aw = lambda_w*Sw/dPW;
    ap = ae + aw;
    bp = qvp*Vp;

    COEFF(i,:) = [ap ae aw bp];
end

%% Last node
[lambda_w,¬] = harmonic_lambda(T,Tmat,r_node,r_element,length(r_node));
dPW = r_node(end,1)-r_node(end-1,1);

aw = lambda_w/dPW;
ap = aw + alpha_ext;
bp = alpha_ext*Text;

COEFF(length(r_node),:) = [ap 0 aw bp];
end
```

A.12. Appendix XII: Unsteady solution function (unsteady_solver.m)

```
function ...
    [T_wall_D02_int,T_wall_D02_ext,T_insulation_ext,T_instant,T_final,Qf,time] = ...
    unsteady_solver(T_initial, ...
    N,H,R,r_node,r_element,Tmat,time,delta_t,time_max,beta,tolerance,save_time,Q)

% Save first values of interest
T_wall_D02_int(1,1) = T_initial(1,1);           % Inner wall of the tank temperature
T_wall_D02_ext(1,1) = T_initial(N(1)+1,1);      % Outter wall of the tank temperature
T_insulation_ext(1,1) = T_initial(sum(N)+2,1);  % Outter wall of the insulator ...
    temperature

% Heat at t = 0
[¬, lambda_i] = harmonic_lambda(T_initial,Tmat,r_node,r_element,1);
Qf(1,1) = -lambda_i*(T_initial(2,1)-T_initial(1,1))/(r_node(2,1)-r_node(1,1))*...
    (2*pi*(r_node(1,1)+r_node(2,1))/2)*H;

% deg to K at time = 0 sec
T_new = T_initial+273.15;

% Initialize instant temperature saving
T_instant = zeros(length(r_node),1);

% Loop
n = 1;
while time < time_max

    % Initialization
    time(n+1,1) = time(n,1) + delta_t;
    T_old = T_new;
    T_prev = T_new;
    iteration = 0; error = 1;

    while abs(error) > tolerance
        iteration = iteration + 1;
        COEFF = unsteady_coeff(T_prev,T_old,r_node,r_element,R,H,Tmat,delta_t, ...
        time(n+1,1),beta,Q);

        if beta ≠ 0
            T_new(:,1) = TDMA(COEFF);
        elseif beta == 0
            T_new(:,1) = explicit(COEFF);
        end

        error = max(abs(T_new(:,1)-T_prev(:,1)));
        T_prev = T_new;
    end

    % Saving values of interest
    T_wall_D02_int(n+1,1) = T_new(1,1)-273.15;
    T_wall_D02_ext(n+1,1) = T_new(N(1)+1,1)-273.15;
    T_insulation_ext(n+1,1) = T_new(sum(N)+2,1)-273.15;

    % Heat calculation
    [¬, lambda_i] = harmonic_lambda(T_new,Tmat,r_node,r_element,1);
    Qf(n+1,1) = -lambda_i*(T_new(2,1)-T_new(1,1))/(r_node(2,1)-r_node(1,1))*...
    (2*pi*(r_node(1,1)+r_node(2,1))/2)*H;

    if time(n+1) == save_time
        T_instant = T_new - 273.15;
    end

    % New loop
```

```
    n = n+1;
end

% Last temperature distribution saving
T_final = T_new - 273.15;

end
```

A.13. Appendix XIII: Unsteady coefficients function (unsteady_coeff.m)

```
function [COEFF] = ...
    unsteady_coeff(T_prev,T_old,r_node,r_element,R,H,Tmat,delta_t,time,beta,Q)

%{
 COEFFICIENT COLUMNS
    - 1st. ap (Tp coefficient)
    - 2nd. ae (Te coefficient)
    - 3rd. aw (Tw coefficient)
    - 4th. bp (Independent)
%}

COEFF = zeros(length(r_node),4);

%% Convective coefficients calculation
[alpha_int_old,alpha_ext_old] = ...
    convection_alpha(Q,R,H,T_old(1,1),T_old(end,1),time-delta_t);
[alpha_int_new,alpha_ext_new] = ...
    convection_alpha(Q,R,H,T_prev(1,1),T_prev(end,1),time);

%% 1st node

% Initialization
[¬, lambda_e_old] = harmonic_lambda(T_old,Tmat,r_node,r_element,1);
[¬, lambda_e_new] = harmonic_lambda(T_prev,Tmat,r_node,r_element,1);
dPE = r_node(2,1)-r_node(1,1);
re = r_element(1);
Se = 2*pi*re*H;

% sum(Qn) at time_old
[Tint_old, ¬] = BC_temperatures(time-delta_t);
[Tint_new, ¬] = BC_temperatures(time);
sum_Qn = + alpha_int_old*(Tint_old - T_old(1,1))*Se ...
    - Se*lambda_e_old*(T_old(1,1)-T_old(2,1))/dPE;

% New coefficients
ae = beta*lambda_e_new*Se/dPE;
ap = ae + beta*alpha_int_new*Se;
bp = beta*alpha_int_new*Tint_new*Se + (1-beta)*sum_Qn;

COEFF(1,:) = [ap ae 0 bp];

%% Inner nodes
for i = 2:(length(r_node)-1)

    % Initialization
    [lambda_w_old, lambda_e_old] = harmonic_lambda(T_old,Tmat,r_node,r_element,i);
    [lambda_w_new, lambda_e_new] = harmonic_lambda(T_prev,Tmat,r_node,r_element,i);
    [¬,Cp_old,rho_old] = materials(T_old(i,1),Tmat(i,1));
    [¬,Cp_new,rho_new] = materials(T_prev(i,1),Tmat(i,1));
    qvp_old = internal_heat(r_node(i,1),time-delta_t);
    qvp_new = internal_heat(r_node(i,1),time);

    dPE = r_node(i+1,1)-r_node(i,1);
```

```
    dPW = r_node(i,1)-r_node(i-1,1);
    re = r_element(i);
    rw = r_element(i-1);
    Se = 2*pi*re*H;
    Sw = 2*pi*rw*H;
    Vp = pi*(re^2-rw^2)*H;

    % sum(Qn) at time_old
    sum_Qn = + Sw*lambda_w_old*(T_old(i-1,1)-T_old(i,1))/dPW ...
        - Se*lambda_e_old*(T_old(i,1)-T_old(i+1,1))/dPE + qvp_old*Vp;

    % New coefficients
    ae = beta*lambda_e_new*Se/dPE;
    aw = beta*lambda_w_new*Sw/dPW;
    ap = ae + aw + Cp_new*rho_new*Vp/delta_t;
    bp = beta*qvp_new*Vp + (1-beta)*sum_Qn + Cp_old*rho_old*Vp*T_old(i,1)/delta_t;

    COEFF(i,:) = [ap ae aw bp];
end

%% Last node

% Initialization
[lambda_w_old,¬] = harmonic_lambda(T_old,Tmat,r_node,r_element,length(r_node));
[lambda_w_new,¬] = harmonic_lambda(T_prev,Tmat,r_node,r_element,length(r_node));
dPW = r_node(end,1)-r_node(end-1,1);
rw = r_element(length(r_node)-1);
Sw = 2*pi*rw*H;

% sum(Qn) at time_old
[¬,Text_old] = BC_temperatures(time-delta_t);
[¬,Text_new] = BC_temperatures(time);
sum_Qn = - alpha_ext_old*(T_old(length(r_node),1) - Text_old)*Sw ...
     + Sw*lambda_w_old*(T_old(length(r_node)-1,1)-T_old(length(r_node),1))/dPW;

% New coefficients
aw = beta*lambda_w_new*Sw/dPW;
ap = aw + beta*alpha_ext_new*Sw;
bp = beta*alpha_ext_new*Text_new*Sw + (1-beta)*sum_Qn;

COEFF(length(r_node),:) = [ap 0 aw bp];
end
```

A.14. Appendix XIV: Convective coefficients calculation function (convection_alpha.m)

```
function [alpha_int,alpha_ext] = convection_alpha(Q,R,H,Tw_int,Tw_ext,time)

% Fluid temperature
[T_int,T_ext] = BC_temperatures(time);

% Parameters
D = 2*R(1);
[rho_int,Cp_int,mu_int,lambda_int] = fluid_parameters(T_int);
[¬,¬,mu_w_int,¬] = fluid_parameters(Tw_int);
v = Q/rho_int;

% Dimensionless groups
Re_int = rho_int*v*D/mu_int; % mean Reynolds
Pr_int = mu_int*Cp_int/lambda_int; % mean Prandtl

%% INTERNAL ALPHA
```

```
% Turbulent Nusselt, always ( Re > 2000 ), viscous liquid
C = 0.027;
m = 0.8;
n = 0.33;
K = (mu_int/mu_w_int)^0.14;
Nu_int = C*Re_int^m*Pr_int^n*K;

% Convective coefficient calculation
alpha_int = lambda_int*Nu_int/D;

%% EXTERNAL ALPHA
% Parameters of the external fluid (air) and others
g0 = 9.81; % Gravity
T_m = (T_ext+Tw_ext)/2; % Film temperature
P_ext = 101208; % Pressure in Cerdanyola (ISA)
beta_ext = 1/T_m; % Expansion coefficient
rho_ext = P_ext/(287*T_m); % Air density
mu_ext = (1.458*10^(-6)*T_m^(1.5))/(T_m+110.4); % Air viscosity
Cp_ext = 1034.09-2.849*10^(-1)*T_m+ 7.817*10^(-4)*T_m^2 - ...
    4.971*10^(-7)*T_m^3 + 1.077*10^(-10)*T_m^4; % Air Cp
lambda_ext = 2.648*10^(-3)*sqrt(T_m) / (1 + (245.4/T_m)*10^(-12/T_m)); % Air ...
    conductive coeff

% Dimensionless groups
Gr_ext = g0*beta_ext*rho_ext^2*abs(Tw_ext-T_ext)*H^3/mu_ext^2; % Grashof number
Pr_ext = mu_ext*Cp_ext/lambda_ext; % Prandtl number
Ra_ext = Pr_ext*Gr_ext; % Rayleigh number

% Nusselt number
C = 0.686;
n = 1/4;
K = (Pr_ext/(1+1.05*Pr_ext))^1/4;
Nu_ext = C*Ra_ext^n*K;

% Convective coefficient calculation
alpha_ext = lambda_ext*Nu_ext/H;


end
```

A.15. Appendix XV: Harmonic mean function (harmonic_lambda.m)

```
function [lambda_w, lambda_e] = harmonic_lambda(T,Tmat,r_node,r_element,node)

if node == 1
    [lambda_E,¬,¬] = materials(T(node+1,1),Tmat(node+1,1));
    lambda_e = lambda_E;
    lambda_w = 0;

elseif node == length(r_node)
    [lambda_W,¬,¬] = materials(T(node-1,1),Tmat(node-1,1));
    lambda_w = lambda_W;
    lambda_e = 0;

elseif node == length(r_node)-1
    [lambda_P,¬,¬] = materials(T(node,1),Tmat(node,1));
    [lambda_W,¬,¬] = materials(T(node-1,1),Tmat(node-1,1));

    dPW = r_node(node,1)-r_node(node-1,1);
    dPw = r_node(node,1)-r_element(node-1,1);
    dWw = r_element(node-1,1)-r_node(node-1,1);

    lambda_e = lambda_P;
```

```
    lambda_w = dPW / ( (dPw/lambda_P) + (dWw/lambda_W) );

else
    [lambda_P,¬,¬] = materials(T(node,1),Tmat(node,1));
    [lambda_E,¬,¬] = materials(T(node+1,1),Tmat(node+1,1));
    [lambda_W,¬,¬] = materials(T(node-1,1),Tmat(node-1,1));

    dPE = r_node(node+1,1)-r_node(node,1);
    dPe = r_element(node,1)-r_node(node,1);
    dEe = r_node(node+1,1)-r_element(node,1);
    dPW = r_node(node,1)-r_node(node-1,1);
    dPw = r_node(node,1)-r_element(node-1,1);
    dWw = r_element(node-1,1)-r_node(node-1,1);

    lambda_e = dPE / ( (dPe/lambda_P) + (dEe/lambda_E) );
    lambda_w = dPW / ( (dPw/lambda_P) + (dWw/lambda_W) );
end

end
```

A.16. Appendix XVI: Plot of any saved instant function (instant_plot.m)

```
function instant_plot(T_tdma,r,time)

% Initialization
Ri = r(1,1);
Re = r(end,1);
[Tint,Text] = BC_temperatures(time);
T_array = [Tint-273.15, Text-273.15];

Tint(1:length(r),1) = Tint-273.15;
Text(1:length(r),1) = Text-273.15;
r = r*1000; % [m] to [mm] conversion
Ri = Ri*1000; Re = Re*1000;

% Figure
figure ('Name', 'Temperature distribution')
yyaxis left
plot (r(:)-Ri,T_tdma(:))
ylabel('Wall Temperature [deg]')

yyaxis right
plot(r(:)-Ri,Tint(:),r(:)-Ri,Text(:))
ylim([min(T_array)-0.5 max(T_array)+0.5])
legend('Wall Temperature', 'Average Internal Temperature', 'External Temperature')
grid on

title(['Tank temperature distribution at t = ',num2str(time),' sec.'])
xlabel('Wall thickness [mm]')
ylabel('BC Temperatures [deg]')
xlim([0, Re-Ri])

% Save plot
%{
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[pos(3), pos(4)])
print(h,'filename','-dpdf','-r0')
%}

end
```

A.17. Appendix XVII: Heat and temperatures evolution plot function (time_plot.m)

```
function time_plot(T_wall_D02_int,T_wall_D02_ext,T_insulation_ext,Qf,time)

% Inner and outer temperature calculation
Tint = zeros(length(time),1);
Text = zeros(length(time),1);
[Tint(:,1),Text(:,1)] = BC_temperatures(time);

% TOP PLOT
figure('Name', 'Temperature of walls vs time')

subplot(2,1,1)
plot(time,T_wall_D02_int, time,T_wall_D02_ext, time,T_insulation_ext)
ylabel('Temperature [deg]')

grid on

title('Temperatures & Heat Evolution in the D02 Tank')
xlabel('Time [sec]')
xlim([0 time(end)])
legend('Inner D02 wall', 'Outer D02 wall', 'Outer Insulation wall', ...
    'location','northwest')

% BOTTOM PLOT
subplot(2,1,2)
yyaxis left
plot(time,Tint-273.15, time,Text-273.15)
ylim([Text(1)-273.15-0.2 23.85])
ylabel('Temperature [deg]')

yyaxis right
plot(time,Qf)
ylabel('Heat Power [W]')

grid on
xlabel('Time [sec]')
xlim([0 time(end)])
legend('Inner Temperature', 'Air Temperature','Heat [W]', 'location','northeast')

end
```

A.18. Appendix XVIII: Steady heat balance verification function (steady_verification.m)

```
function [Q_in_st,Q_out_st] = steady_verification(T_initial,Tmat,r_node,r_element,H)

% Heat at the inner wall
[¬, lambda_i] = harmonic_lambda(T_initial,Tmat,r_node,r_element,1);
Q_in_st = -lambda_i*(T_initial(2,1)-T_initial(1,1))/(r_node(2,1)-r_node(1,1))*...
    (2*pi*(r_node(1,1)+r_node(2,1))/2)*H;

% Heat at the outer wall
[lambda_o,¬] = harmonic_lambda(T_initial,Tmat,r_node,r_element,length(r_node));
Q_out_st = ...
    -lambda_o*(T_initial(end,1)-T_initial(end-1,1))/(r_node(end,1)-r_node(end-1,1))*...
    (2*pi*(r_node(end-1,1)+r_node(end,1))/2)*H;

end
```

A.19. Appendix XIX: Insulator radius main code variation (insulator.m)

```
%-------------------------------------------------------------------------
%                          CODE VARIATION
%                   MINIMUM INSULATOR RADIUS STUDY
%-------------------------------------------------------------------------
%% MAIN
clc, clear all

%% -----------------------------------------------------------------------
%                                1. INPUT
%           User definition of the different solver's inputs
%-------------------------------------------------------------------------

% 1.1. Volumetric Flow
Q = 650;                    % [m^3/h]

% 1.2. Geometry
% Internal tank wall, external tank wall, external insulation wall
R = [2.944, 2.950];         % Radius [m]
H = 5.38;                   % TANK height [m]

Rins = linspace(R(2)+1/10000,R(2)+.25,100); % Insulator external radius

% 1.3. Numerical data
N = [20, 100];              % Number of elements of tank and insulation walls
tolerance = 1e-6;           % Convergence tolerance
Tstart = 20;                % Start temperature for first/steady solution [deg]
delta_t = 60;               % Time increment for unsteady solutions [sec]
time_max = 525*60;          % Maximum time for the unsteady simulation [sec]
save_time = 300*60;         % Save temperature distribution at this instant [sec]
beta = 0.5;                 % Heat integration method [0 (EXP), 0.5 (CN), 1 (IMP)]

% 1.4. MODIFY THE MATERIALS FUNCTION (materials.m)
% 1.5. MODIFY INTERNAL//EXTERNAL TEMPERATURE FUNCTION (BC_temperatures.m)
% 1.6. MODIFY INTERNAL VOLUMETRIC HEAT GENERATION FUNCTION (internal_heat.m)
% 1.7. MODIFY FLUID PARAMETERS FUNCTION IF NOT WATER (fluid_parameters.m)

%% -----------------------------------------------------------------------
%                             2. CALCULUS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

for i = 1:1:length(Rins)

    % Set external radius
    R(3) = Rins(i);

    % 2.1. Geometry discretization
    [r_node,r_element] = node_distribution(R,N);

    % 2.2. TDMA based SOLVER for steady solution (time=0)
    [T_initial,iteration,Tmat,time] = ...
        steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q);

    % 2.3. TDMA based SOLVER for unsteady solutions
    [¬,¬,¬,¬,¬,Qf,time] = unsteady_solver( ...
        T_initial,N,H,R,r_node,r_element,Tmat, ...
        time,delta_t,time_max,beta,tolerance,save_time,Q);

    % Global heat balance ( + if outgoing, - if ingoing )
    Qw(i) = trapz(time,Qf); % [J]

end
```

```
62
63 %% ------------------------------------------------------------------------
64 %                            3. RESULTS
65 %           Coefficient determination and algorithm functions
66 %------------------------------------------------------------------------
67
68 figure('Name','Radius of the insulator')
69 plot((Rins-R(2))*1000,(Qw-Qw(1))/Qw(1)*100), grid on
70 title('Delivered heat variation against insulator thickness')
71 xlabel('Insulator thickness [mm]')
72 ylabel('\Delta Q [%]')
```

A.20. Appendix XX: Convergence study main code variation (convergence.m)

```
1  %------------------------------------------------------------------------
2  %                          CODE VARIATION
3  %                        CONVERGENCE STUDY
4  %------------------------------------------------------------------------
5  %% MAIN
6  clc, clear all, close all
7
8  %% ------------------------------------------------------------------------
9  %                              1. INPUT
10 %           User definition of the different solver's inputs
11 %------------------------------------------------------------------------
12
13 % 1.1. Volumetric Flow
14 Q = 650;                    % [m^3/h]
15
16 % 1.2. Geometry
17 % Internal tank wall, external tank wall, external insulation wall
18 R = [2.944, 2.950, 3.2];    % Radius [m]
19 H = 5.38;                   % TANK height [m]
20
21 % 1.3. Numerical data
22 N = [300, 300];             % Number of elements of tank and insulation walls
23 tolerance = 1e-6;           % Convergence tolerance
24 Tstart = 20;                % Start temperature for first/steady solution [deg]
25 delta_t = 60;               % Time increment for unsteady solutions [sec]
26 time_max = 525*60;          % Maximum time for the unsteady simulation [sec]
27 save_time = 300*60;         % Save temperature distribution at this instant [sec]
28 beta = 0.5;                 % Heat integration method [0 (EXP), 0.5 (CN), 1 (IMP)]
29
30 % 1.4. Convergence parameters
31 NN = [10,20,40,80,100,150,300,1000]; % Number of elements for both walls
32 TT = divisors(time_max); % delta_t used
33 TT = TT(5:length(TT)-12); % Change limits (if needed)
34
35 % 1.5. MODIFY THE MATERIALS FUNCTION (materials.m)
36 % 1.6. MODIFY INTERNAL//EXTERNAL TEMPERATURE FUNCTION (BC_temperatures.m)
37 % 1.7. MODIFY INTERNAL VOLUMETRIC HEAT GENERATION FUNCTION (internal_heat.m)
38 % 1.8. MODIFY FLUID PARAMETERS FUNCTION IF NOT WATER (fluid_parameters.m)
39
40 %% ------------------------------------------------------------------------
41 %                            2. CALCULUS
42 %           Coefficient determination and algorithm functions
43 %------------------------------------------------------------------------
44
45 % 2.1. Iterative convergence study of number of elements
46 for i = 1:length(NN)
47
```

```
    % Number of elements
    N = [NN(i),NN(i)];

    % Geometry discretization
    [r_node,r_element] = node_distribution(R,N);

    % TDMA based SOLVER for steady solution (time=0)
    [T_initial,¬,Tmat,time] = steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q);

    % TDMA based SOLVER for unsteady solutions
    [¬,¬,¬,¬,¬,Qf,time] = unsteady_solver( ...
        T_initial,N,H,R,r_node,r_element,Tmat, ...
        time,delta_t,time_max,beta,tolerance,save_time,Q);

    % Global heat balance ( + if outgoing, - if ingoing )
    Qw_n(i,1) = trapz(time,Qf); % [J]

end

% 2.2. Iterative convergence study of time increment
N = [300,300]; % input

% Geometry discretization
[r_node,r_element] = node_distribution(R,N);

% TDMA based SOLVER for steady solution (time=0)
[T_initial,iteration,Tmat,¬] = steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q);

for i = 1:length(TT)

    % Initialization of time vector
    time = 0;

    % delta_t update
    delta_t = TT(i);

    % TDMA based SOLVER for unsteady solutions
    [¬,¬,¬,¬,¬,Qf,time] = unsteady_solver( ...
        T_initial,N,H,R,r_node,r_element,Tmat, ...
        time,delta_t,time_max,beta,tolerance,save_time,Q);

    % Global heat balance ( + if outgoing, - if ingoing )
    Qw_t(i,1) = trapz(time,Qf); % [J]

    clear time
end


%% ------------------------------------------------------------------------
%                              3. RESULTS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 3.1. Number of elements convergence plot
figure('Name','Convergence Study (numer of elements)')
loglog(NN,abs((Qw_n-Qw_n(end)))/Qw_n(end)*100), grid on
title('Element Convergence Study')
xlabel('Number of elements')
ylabel('\Delta Q_w [%]')

% 3.2. Number of elements convergence plot
figure('Name','Convergence Study (time increment)')
plot(TT,abs((Qw_t-Qw_t(1)))/Qw_t(1)*100), grid on
title('Time Increment Convergence Study')
xlabel('\Delta t [sec]')
ylabel('\Delta Q_w [%]')
```

A.21. Appendix XXI: MMS main code variation (verification.m)

```
%-------------------------------------------------------------------------
%                            CODE VARIATION
%                       MMS UNSTEADY VERIFICATION
%-------------------------------------------------------------------------
%% MAIN
clc, clear all, close all

%% -----------------------------------------------------------------------
%                                1. INPUT
%           User definition of the different solver's inputs
%-------------------------------------------------------------------------

% 1.1. Geometry
% Internal tank wall, external tank wall
R = [0, 2];                  % Radius [m]
H = 10;                      % TANK height [m]

% 1.2. Numerical data
N = [5000];                  % Number of elements of tank and insulation walls
tolerance = 1e-8;            % Convergence tolerance
Tstart = 20;                 % Start temperature for first/steady solution [deg]
delta_t = 1;                 % Time increment for unsteady solutions [sec]
time_max = 600;              % Maximum time for the unsteady simulation [sec]
save_time = 500;             % Save temperature distribution at this instant [sec]
beta = 0.5;                  % Heat integration method [0 (EXP), 0.5 (CN), 1 (IMP)]

% 1.3. MODIFY THE MATERIALS FUNCTION (materials.m)
% 1.4. SET CONVECTIVE COEFF TO 0 W/m^2K
% 1.5. MODIFY INTERNAL//EXTERNAL TEMPERATURE FUNCTION (BC_temperatures.m)
% 1.6. MODIFY INTERNAL VOLUMETRIC HEAT GENERATION FUNCTION (internal_heat.m)

%% -----------------------------------------------------------------------
%                              2. CALCULUS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 2.1. Geometry discretization
[r_node,r_element] = node_distribution(R,N);

% 2.2. Find node materials
for i = 1:size(r_node,1)
    Tmat(i,1) = node_materials(R,r_node(i,1));
end

% 2.3. Initial temperature distribution
time(1,1) = 0;
T_initial(1:length(r_node),1) = 30;

% 2.4. TDMA based SOLVER for unsteady solutions
[T_instant,T_final,time] = unsteady_solver(T_initial, ...
    H,r_node,r_element,Tmat,time,delta_t,time_max,beta,tolerance,save_time);

%% -----------------------------------------------------------------------
%                              3. RESULTS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 3.1. Saved instant Temperature plot
instant_plot (T_instant,r_node,save_time);

% 3.2. Analytical plot at saved instant
analytical_plot(r_node,save_time);
```

A.22. Appendix XXII: Analytical MMS plot function (analytical_plot.m)

```
function analytical_plot(r,time)

% Analytical Expression
w = 1e-3;
T(:,1) = 30 + 20*r.^2*sin(w*time);

% Initialization
r = r*1000; % [m] to [mm] conversion

% Figure
figure ('Name', 'Temperature distribution')
plot (r(:),T(:)), grid on
xlim([1000, 1100])
xlabel('Wall thickness [mm]')
ylabel('Wall Temperature [deg]')
title(['ANALYTICAL: temperature distribution at t = ',num2str(time),' sec.'])

end
```

A.23. Appendix XXIII: Volumetric flow dependency main code variation (volumetric_flow.m)

```
%-------------------------------------------------------------------------
%                          CODE VARIATION
%                    VOLUMETRIC FLOW DEPENDENCY
%-------------------------------------------------------------------------
%% MAIN
clc, clear all, close all

%% -----------------------------------------------------------------------
%                                1. INPUT
%          User definition of the different solver's inputs
%-------------------------------------------------------------------------

% 1.1. Volumetric flow vector
Qv = 400:5:650;              % [m^3/h]

% 1.2. Geometry
% Internal tank wall, external tank wall, external insulation wall
R = [2.944, 2.950];          % Radius [m]
H = 5.38;                    % TANK height [m]

% 1.3. Numerical data
N = [20];                    % Number of elements of tank and insulation walls
tolerance = 1e-6;            % Convergence tolerance
Tstart = 20;                 % Start temperature for first/steady solution [deg]
delta_t = 60;                % Time increment for unsteady solutions [sec]
time_max = 525*60;           % Maximum time for the unsteady simulation [sec]
save_time = 300*60;          % Save temperature distribution at this instant [sec]
beta = 0.5;                  % Heat integration method [0 (EXP), 0.5 (CN), 1 (IMP)]

% 1.4. MODIFY THE MATERIALS FUNCTION (materials.m)
% 1.5. MODIFY INTERNAL//EXTERNAL TEMPERATURE FUNCTION (BC_temperatures.m)
% 1.6. MODIFY INTERNAL VOLUMETRIC HEAT GENERATION FUNCTION (internal_heat.m)
% 1.7. MODIFY FLUID PARAMETERS FUNCTION IF NOT WATER (fluid_parameters.m)


```

```
%% ------------------------------------------------------------------------
%                              2. CALCULUS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% 2.1. Geometry discretization
[r_node,r_element] = node_distribution(R,N);

for i = 1:length(Qv)

    % New calculation
    Q = Qv(i);

    % 2.2. TDMA based SOLVER for steady solution (time=0)
    [T_initial,iteration,Tmat,time] = ...
        steady_solver(Tstart,R,H,tolerance,r_node,r_element,Q);

    % 2.3. TDMA based SOLVER for unsteady solutions
    [¬,¬,¬,¬,¬,Qf,time] = unsteady_solver( ...
        T_initial,N,H,R,r_node,r_element,Tmat, ...
        time,delta_t,time_max,beta,tolerance,save_time,Q);

    % Global heat balance ( + if outgoing, - if ingoing )
    Qw(i,1) = trapz(time,Qf); % [J]

end

%% ------------------------------------------------------------------------
%                              3. RESULTS
%           Coefficient determination and algorithm functions
%-------------------------------------------------------------------------

% Heat vs volumetric flow plot
figure('Name','Heat against volumetric flow')
plot(Qv,Qw), grid on
title('Heat against volumetric flow without insulator')
xlabel('Volumetric flow [m^3/h]')
ylabel('Delivered heat [J]')
```

A.24. Appendix XXIV: Explicit function (explicit.m)

```
function [T_new] = explicit(COEFF)

% In case of beta = 0, use this function instead of TDMA.m

T_new = zeros(size(COEFF,1),1);

for i = 2:size(COEFF,1)-1
    T_new(i,1) = COEFF(i,4)/COEFF(i,1);
end

T_new(1,1) = T_new(2,1);
T_new(size(COEFF,1),1) = T_new(size(COEFF,1)-1,1);

end
```