# THERMAL STUDY OF A HEAT EXCHANGER

Authors:
Sara-Medina Šehović
Albert Bartolome Castillo Mestre
Giada Alessi
Heloise Chapelle
Rajnesh Kumar

December 20, 2024

**Abstract**

This document presents a comprehensive study on the thermal performance and evaluation of a counter-flow heat exchanger, addressing fundamental theoretical principles, experimental methodologies employed for data acquisition and semi-analytical result analysis. Detailed descriptions of energy balances, heat transfer calculations, thermodynamic efficiency, and experimental techniques for recording temperatures, flow rates, and pressures are provided. Additionally, the implementation of computational tools (Python, CoolProp) and the generation of graphs and tables for result interpretation and optimization are included.

The study focuses on a tubular counter-flow heat exchanger, evaluating cases under varying conditions: inlet temperatures, flow rates. Finally, conclusions and comments on potential future improvements are presented.

# Contents

# 1 Introduction

Heat exchangers are fundamental components in industrial thermal systems, HVAC, power generation, and various industrial processes. They facilitate the transfer of thermal energy between two fluid streams at different temperatures without the need for mixing. This report documents a thermal study of a tube-in-tube counter-flow heat exchanger, operating under counter-current flow conditions where the hot and cold streams flow in opposite directions. This configuration maximizes the temperature gradient along the entire length of the exchanger, thereby enhancing heat transfer efficiency.

The objectives of this study are:

- To understand the basic principles of energy balance applied to a heat exchanger.
- To determine the heat transfer rate and thermodynamic efficiency.
- To compare experimental results with semi-analytical predictions.
- To document and analyze the impact of operational parameters (flow rate, inlet temperatures, etc.) on the behavior of the heat exchanger.

The document begins with theoretical fundamentals, followed by a description of the experimental setup and procedure. Subsequently, the equations used for analysis are presented, computational tools utilized are detailed, experimental and semi-analytical results are compared.

# 2 Theoretical Fundamentals

## 2.1 Counter-Flow HX Configuration

A heat exchanger (HX) is a device that facilitates the transfer of heat between two fluids at different temperatures. The fluids can be separated by a solid wall to prevent mixing. Common configurations include parallel flow, counter-flow, and cross-flow.

In the counter-flow mode, the hot and cold fluids enter the heat exchanger from opposite ends. This arrangement offers the highest thermal effectiveness since the temperature gradient between the fluids remains relatively constant along the length of the exchanger, enhancing heat transfer.

## 2.2 Energy Balance and Heat Transfer Rate

The steady-state energy balance for the single phase counter-flow heat exchanger is given by:

$$Q = \dot{m}_{\text{hot}} c_{p,\text{hot}} (T_{\text{hot,in}} - T_{\text{hot,out}}) = \dot{m}_{\text{cold}} c_{p,\text{cold}} (T_{\text{cold,out}} - T_{\text{cold,in}})$$

where:

- $Q$ is the heat transfer rate.
- $\dot{m}$ is the mass flow rate.
- $c_p$ is the specific heat capacity at constant pressure.
- $T_{\text{in}}$ and $T_{\text{out}}$ are the inlet and outlet temperatures.

## 2.3 Thermodynamic Efficiency (Effectiveness)

The effectiveness $\varepsilon$ of a heat exchanger indicates the fraction of the maximum possible heat transfer that is achieved under actual operating conditions. It is defined as:

$$\varepsilon = \frac{Q}{Q_{\max}}$$

where:

$$Q_{\max} = C_{\min}(T_{\text{hot,in}} - T_{\text{cold,in}})$$

and

$$C_{\min} = \min(\dot{m}_{\text{hot}} c_{p,\text{hot}}, \dot{m}_{\text{cold}} c_{p,\text{cold}})$$

## 2.4 Overall Heat Transfer Coefficient (U$_o$)

The overall heat transfer coefficient $U$ incorporates all thermal resistances in the heat exchanger:

$$\frac{1}{U_o} = \frac{1}{h_{\text{i}}} \frac{P_o}{P_i} + \frac{P_o \ln(D_{\text{o}}/D_{\text{i}})}{2\pi k_{\text{wall}}} + \frac{1}{h_{\text{o}}}$$

where:

- $h_{\text{i}}$, $h_{\text{o}}$ are the convective heat transfer coefficients for the hot (inner) and cold (outer) fluids.
- $k_{\text{wall}}$ is the thermal conductivity of the exchanger wall.
- $D_{\text{i}}$, $D_{\text{o}}$ are the inner and outer diameters.
- $P_{\text{i}}$, $P_{\text{o}}$ are the inner and outer tube perimeters.

## 2.5 $\varepsilon$-NTU Method

The Number of Transfer Units (NTU) method relates the effectiveness to:

$$NTU = \frac{U_o A_o}{C_{\min}}$$

where $U_o$ is the overall heat transfer coefficient and $A_o$ is the heat transfer area.

For counter-flow heat exchangers, the effectiveness is given by:

$$\varepsilon = \frac{1 - e^{-NTU(1-C_r)}}{1 - C_r e^{-NTU(1-C_r)}}$$

where:

$$C_r = \frac{C_{\min}}{C_{\max}}$$

## 2.6 Pressure Drop

Fluid flow through the heat exchanger results in pressure drops, which can be quantified using the Darcy-Weisbach equation:

$$\Delta P = f_D \frac{L}{D_h} \frac{\rho v^2}{2}$$

where:

- $f_D$ is the Darcy friction factor.
- $D_h$ is the hydraulic diameter.
- $\rho$ is the fluid density.
- $v$ is the fluid velocity.

# 3 Experimental Methodology

## 3.1 Equipment Description

The study utilizes a double-tube counter-flow heat exchanger. One fluid (hot) flows through the inner tube, while the other fluid (cold) flows through the annular space between the inner and outer tubes. Temperature sensors are placed along the length of the exchanger to monitor inlet and outlet temperatures. Flow meters are used to measure flow rates.

## 3.2 Measured Parameters

The following parameters are recorded during the experiments:

- Inlet and outlet temperatures of the hot fluid ($T_{\text{hot,in}}$, $T_{\text{hot,out}}$).
- Inlet and outlet temperatures of the cold fluid ($T_{\text{cold,in}}$, $T_{\text{cold,out}}$).
- Temperatures along the heat exchanger at specific positions.
- Volumetric flow rates of both fluids.

## 3.3 Procedure

The experimental procedure involves:

1. Setting the volumetric flow rate of the cold fluid .
2. Adjusting the volumetric flow rate and temperature of the hot fluid (heated water from a thermal tank).
3. Operating in counter-flow mode and allowing the system to stabilize.
4. Recording data at regular intervals until steady-state conditions are achieved.
5. Repeating the measurements under different operational conditions (flow rates, inlet temperatures).

## 3.4 Instrumentation and Control Units

The system is equipped with data acquisition units connected to type K thermocouples and flow meters. Data is recorded in real-time using dedicated software (LabVIEW, TICC, SCADA).

# 4 Semi-Analytical Calculation Methodology

## 4.1 Calculation of Thermophysical Properties

Thermophysical properties such as density ($\rho$), viscosity ($\mu$), thermal conductivity ($k$), and specific heat capacity ($c_p$) are obtained using the `CoolProp` library in Python. Properties are calculated at the mean temperatures and pressures of each fluid stream.

## 4.2 Calculation of Reynolds, Nusselt Numbers, and Friction Factors

Based on the velocities, areas, hydraulic diameters, and thermophysical properties, the Reynolds number ($Re$) is calculated:

$$Re = \frac{\rho v D_h}{\mu}$$

The flow regime is determined based on $Re$. For $Re > 2000$, turbulent flow is assumed, and correlations from the Formulae are applied to calculate the Nusselt number ($Nu$):

$$Nu = f(Re, Pr)$$

The Darcy-Weisbach friction factor ($f_D$) is also calculated using appropriate correlations.

### 4.3 Convective Heat Transfer Coefficients

With the calculated $Nu$, the convective heat transfer coefficients ($h$) for both hot and cold fluids are determined:

$$h = \frac{Nu \cdot k}{D_h}$$

# 5 Experimental Results

In this section, we present a significantly more detailed experimental analysis, focusing on the seven key figures that illustrate various aspects of the experiment. The objective is to gain a comprehensive understanding of the system's behavior, the instrumentation's influence, and environmental factors that shaped the recorded results. By examining each figure in depth, we aim to identify subtle phenomena and highlight areas where improvements in the experimental setup, data collection, or operating conditions may be necessary.

The figures under consideration are:

1. Temperature evolution over time (Figure 1)
2. Flow rates over time (Figure 2)
3. Temperature profiles along the heat exchanger for each defined mode (Figure 3)
4. Segment-wise heat losses (Figure 4)
5. Per-portion balances of released/absorbed heat (Figure 5)
6. Overall absorbed, released, and lost heat during the entire experiment (Figure 6)

## 5.1 (1) Temperature Evolution Over Time

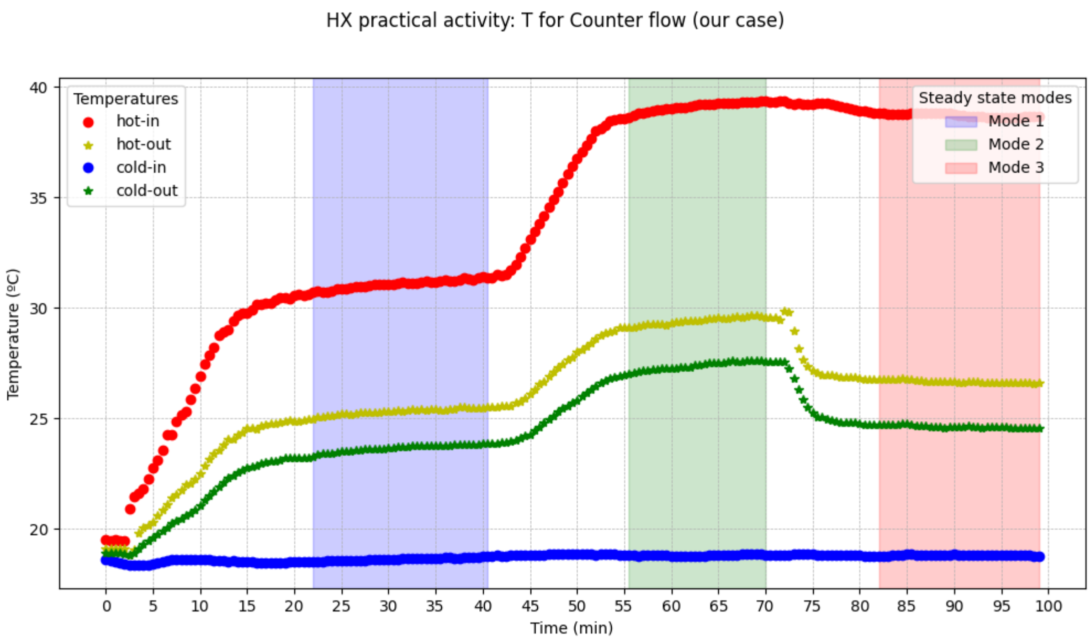


Figure 1: Inlet and outlet temperatures evolution of hot and cold fluids over time with steady-state indicators for different working modes of the experiment.

Figure 1 represents the inlet and outlet temperature evolution of both hot and cold fluids as functions of time. The shaded color scheme represents the region for different experimental modes, as required by the problem statement. In our theoretical work so far, we have dealt with semi-analytical problems, in which we modeled and analyzed heat exchangers in steady conditions, meaning that their properties do not change over time. In such problems, the inlet and outlet temperatures of the heat exchanger were constant.

However, from Figure 1, we can clearly see that that is not the case in real-life applications. From the obtained figure, we can see how inlet and outlet temperatures evolve over time. Initially, we can see that all four observed temperatures stagnate for a short period of time, which corresponds to Figure 2. During this short period of more or less 2 minutes, the experiment was set up and aligned to the requirements.

After this set-up period, the inlet and outlet temperatures of the hot water and outlet temperatures of the cold water gradually increase, with a steep slope. The inlet temperature of the cold flow was relatively constant throughout the experiment, with slight variations, as expected. After approximately 21 minutes, the slope of the temperatures became almost horizontal. This period can be considered a "steady-state". As can be seen from Figure 1, we have three such states, one for each experimental mode. In these intervals, the temperatures are still increasing, however, with small slopes.

This means that our "steady-state" does not represent the constant conditions, as we have done in the semi-analytical analysis so far, but conditions which change over time. However, those changes can be considered negligible, as it is practically impossible to reach perfect steady-state conditions in real-life applications. Hence, from now on, we will be taking average conditions in our analysis, as that will allow us to conduct a proper analysis of our experiment. Now, we will make a more detailed analysis of our temperature evolution.

The observed steady-state time intervals were chosen according to Figure 1 and Figure 2. The goal was to choose the time intervals properly, in which the temperature and flow rate variations were low. The plot shows a system that does not reach a perfectly constant state but rather exhibits relatively stable "plateaus." This is common in real experiments where it is difficult to control all variables. The selected time-intervals are, in reality, "less variable" phases within a dynamic test. These time intervals, one for each experimental mode are as follows:

- **Mode 1 (Blue):** Between 22 and 40.5 minutes, as temperature and flow fluctuations in this time period are minimal. From Figure 2, we can see that the cold fluid flow rate fluctuated during the first 20 minutes of the experiment, hence, the above-mentioned time interval was used for our steady-state approximation.
- **Mode 2 (Green):** Between 55.5 and 70 minutes. The interval was chosen as it best captures the relatively stable temperature fluctuations during this period.
- **Mode 3 (Red):** Between 81.5 and 99 minutes. The last mode represents the interval with changed volumetric flow rate of hot fluid, which corresponds to the temperature changes. The temperatures in this mode slightly decrease over time.

The observations for temperature evolutions are as follows:

- Inlet temperature of the hot fluid (red): Initially near ambient temperature (19–20°C), as the water was placed in a tank; then it gradually rises to about 30–31°C during Mode 1; during Mode 2, the temperature increases approximately to 39°C. Finally, during Mode 3, the temperature decreases to around 38°C, as a consequence of changed volumetric flow rate. For all three modes, the temperature evolution relatively stabilizes, with relatively small slopes and fluctuations, and these intervals were considered as statistical steady-state.

- Outlet temperature of the hot fluid (yellow): Always below the inlet temperature of hot fluid, as expected, since the hot fluid releases heat along the heat exchanger. The evolution of the outlet temperature of hot fluid follows the one for the inlet temperature of the hot fluid. Once again, the stabilization of temperature change is considered a steady-state in our analysis.

- Inlet temperature of the cold fluid (blue): Remains nearly constant at about 19°C, suggesting a stable cold source, as we are using water from the local network.

- Outlet temperature of the cold fluid (green): This temperature follows a similar evolution to the one of the inlet and outlet hot water temperature. However, this one has the lowest values, which is logical, as we are transferring heat to this flow. The same steady-state assumptions were applied to these temperature observations.

Together, these observations reveal a system that never fully settles into a perfectly constant temperature condition. The interplay between intentional adjustments and environmental factors prevents achieving ideal steady states.

## 5.2 (2) Flow Rates Over Time

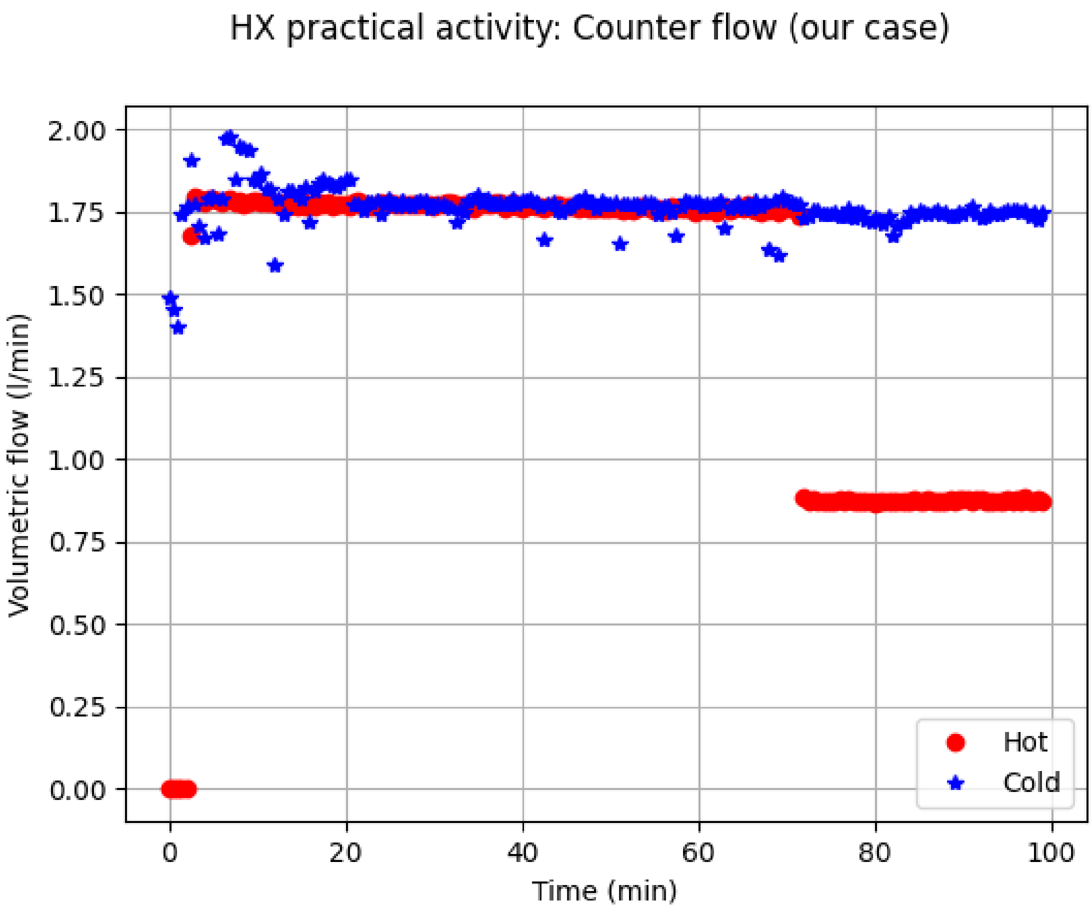


Figure 2: Volumetric flow rates for hot and cold fluids over time.

Figure 2 represents the collected data for the volumetric flow rate of both hot and cold streams. The volumetric flow rates were set in the software at 1.8 L/min for both hot and cold fluids in Mode 1 and Mode 2 and in Mode 3 for the cold fluid. In mode 3, the volumetric flow rate of hot fluid was decreased to 0.9 L/min. The volumetric flow rate in time was measured by SC-1 and SC-2. The observations are as follows:

- **Flow rates for different modes:** From the problem definition, we are required to set volumetric flow rates of both hot and cold fluids. The cold water flow rate is kept at 1.9 L/min throughout the experiment, in all three working modes. On the other hand, the hot water flow rate is kept at 1.9 L/min for Mode 1 and Mode 2 and at 0.9 L/min for Mode 3. Figure 2 represents these modifications well. We can clearly see when we modified our flows, and this allows us for a good analysis of results.

- **Cold Flow Instability:** From Figure 2, we can see that the cold fluid flow fluctuates much more than the hot fluid flow. Initially, we can see that it is still at 0, as that is the time period of the experiment set-up. Once set, the volumetric flow rate of hot fluid captured on the sensors is quite constant and does not experience off values, even after changes made in Mode 3. However, despite this good consistency, the values captured are not perfectly constant, and once again, we need to use the average values of mass flow rates for the previously defined time intervals.

  On the other hand, the volumetric flow rate of the cold fluid experiences large fluctuations. This is especially noticeable during the first 20 minutes of the experiments, where the flow rate of cold water fluctuated significantly. Hence, this time period was excluded from the Mode 1 steady state. After 20 minutes, the flow rate of cold water was more stable. However, we could still observe some noticeable discrepancies, unlike the hot water flow. There are many reasons why this is happening; mostly, it is because we are using water from the local network, and it is prone to flow fluctuations; moreover, the valves used in this are not perfect, and may cause some discrepancies in the flow rate.

- **Correlation with temperature evolution:** As explained before, in order to properly define time intervals for relative steady-states for all three modes, we must also take into account the consistency of volumetric mass flow rate change. In both Figure 1 and Figure 2, we can see instabilities during the first 20 minutes of the experiment. During this period, cold water flow rate fluctuated significantly and it impacted the temperature evolution. Moreover, instability in cold water flow is one of the reasons why temperature fluctuates in the proposed steady state.

In essence, the flow data show that the lack of strict flow control or frequent operational adjustments complicates the attainment of a stable thermal regime.

## 5.3 (3) Temperature Profiles Along the Heat Exchanger for Each Mode

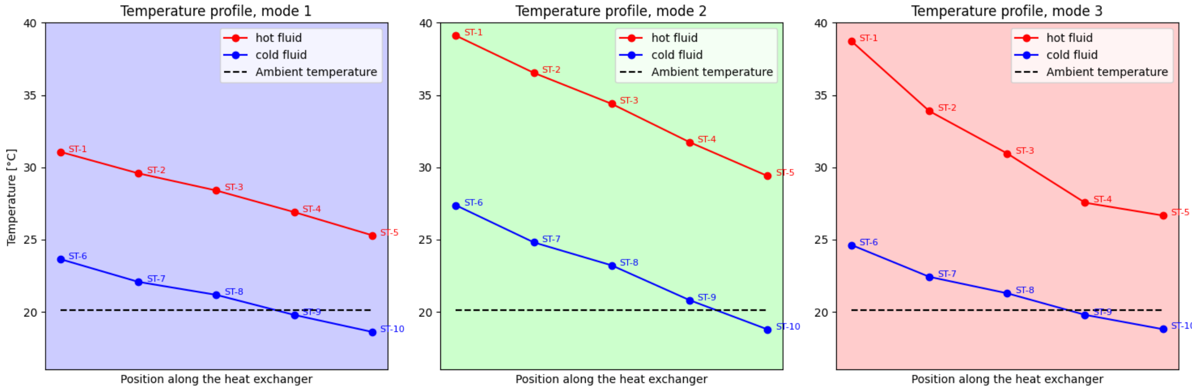


Figure 3: Temperature profiles along the heat exchanger for each operational mode.

Figure 3 highlights spatial variations of temperature within the heat under each mode. The goal of this plot is to replicate the temperature evolution presented in the TITCA application module. To obtain these plots, we have used the average temperatures, obtained for the previously defined time intervals for each mode. The time intervals were chosen to comply with the statistical steady-state. The temperatures were taken from the sensor data according to the experiment configuration, at certain positions along the heat exchanger. The observations are the following:

- **Expected Trend (Ideal Case):** In a perfect counterflow configuration, the hot fluid should progressively cool as it travels, while the cold fluid should warm correspondingly. This creates a smooth temperature gradient along the heat exchanger length. Moreover, the exchanged heat depends on the difference between the two stream temperatures.
- **Observed Irregularities:** Some profiles deviate from this predicted behavior. In all three modes, sensor ST-8 has a temperature that is slightly higher than the expected temperature. As we can clearly see on the plot, the slope between ST-7 and ST-8 is smaller compared to other slopes. Furthermore, the temperature profile for Mode 3 significantly deviates from the predicted case. The changes in temperature differences between fluids are not consistent, which might suggest inaccuracy. The next section will help us to understand this phenomenon better. These anomalies suggest:
  - Potential misplacement or inaccuracy of the temperature sensors.
  - Inadequate insulation, allowing localized cooling or heating from the environment.
  - Sensor calibration issues causing systematic temperature offsets.
- **Mode-Dependent Behavior:** While one mode might show a more coherent gradient, another might fail to produce the expected temperature rise in the cold fluid. This inconsistency between modes underlines that experimental conditions were not standardized or controlled closely enough. We can see that the plots for Mode 1 and Mode 2 are quite consistent, while the results for Mode 3 suggest significant deviations. Mode 3 was conducted for the lower volumetric flow rate of cold fluid

Overall, the temperature profiles emphasize the need for meticulous sensor placement, robust insulation, and stable conditions to produce interpretable spatial temperature distributions.

In the next section, we will deal with heat losses along the heat exchanger segments. This will allow us to further observe the collected data, as we will be able to compare heat transfer along the heat exchanger and see whether there are any anomalies.

## 5.4 (4) Segment-wise Heat Losses

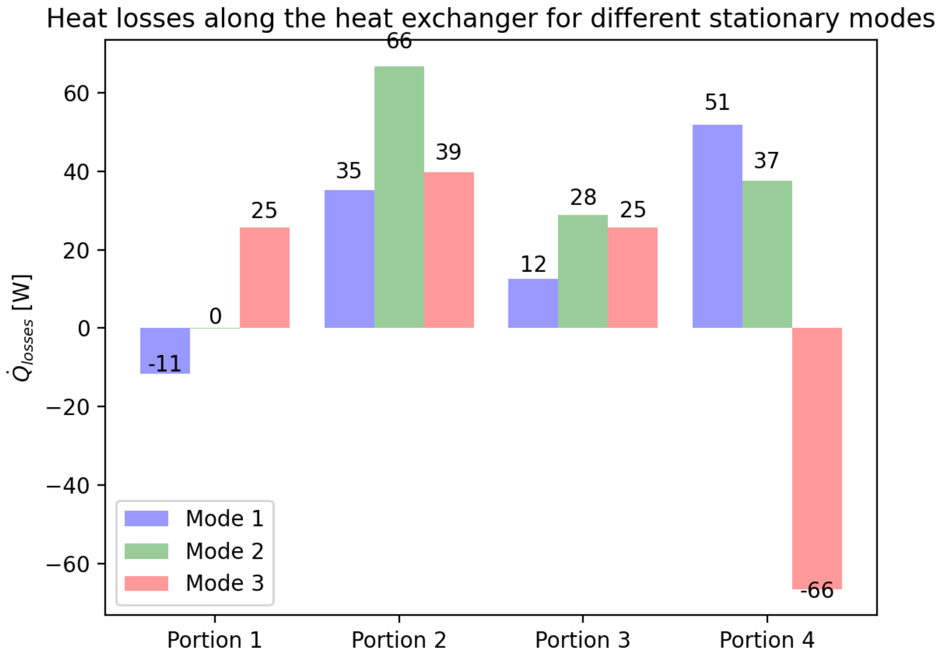


Figure 4: Segment-wise heat losses across different sections of the heat exchanger.

Figure 4 aims to break down heat losses along distinct heat exchanger segments. In a well-insulated heat exchanger, losses among its segments should be small and physically consistent. Since we are dealing with a real-life, non-ideal heat exchanger, we expect to see heat losses along the heat exchanger. From the given figure, our observations are the following:

- **Negative Losses Indicate Measurement Inaccuracies:** Some portions are showing negative losses, meaning that more heat was absorbed by the cold fluid than was released by the hot fluid in a certain segment. This behavior was observed for the first segment of the heat exchanger in Mode 1 and in the last portion of the heat exchanger in Mode 3. This suggests errors in temperature measurement, flow synchronization or simply poor insulation. Such results highlight the challenges of performing local energy balances in a complex experimental setting.
- **High Positive Losses Suggest Environmental Influences:** Positive heat loss means that more heat was released by the hot fluid than was absorbed by the cold fluid. This behavior is expected, as heat losses are usual in practical measurements. However, high values of heat losses might suggest bad insulation, wrong positioning of temperature sensors, as well as measurement inaccuracies.

Ultimately, the segment-wise analysis reveals the experiment's vulnerability to small measurement or environmental errors, making local energy closure difficult to achieve.

### 5.5 (5) Per-portion Balances of Released/Absorbed Heat

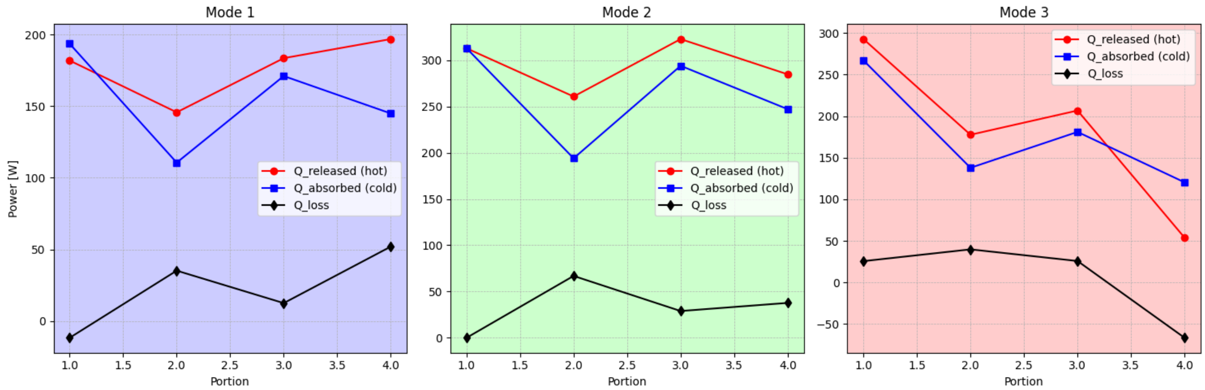


Figure 5: Heat balances ($Q_{\text{loss}}$) for different operational modes.

Figure 5 tracks how $Q_{\text{loss}}$ changes along the heat exchanger, in comparison with released heat by the hot fluid $Q_{\text{released}}$, and absorbed heat $Q_{\text{absorbed}}$ by the cold fluid. The aim of this plot is to further understand the heat transfer evolution. The observations are the following:

- **Expectations vs. Reality:** In a genuine steady state, $Q_{\text{loss}}$ should be relatively constant and small. Instead, we observe large fluctuations, spikes, and dips, sometimes even indicating unphysical trends (negative or extremely high values).
- **Transient Effects and Noise:** Even within selected "steady" intervals, the experimental data never completely stabilize. Minor temperature drifts, flow oscillations, or ambient disturbances lead to temporal variability in computed heat losses.
- **Comparison Between Modes:** Some modes may show smaller fluctuations than others, but none exhibit the stable, low-loss pattern one would expect from a perfectly controlled laboratory scenario. This reaffirms that the declared steady states are only relative reductions in variability, not true equilibrium states. Moreover, Mode 3 once again shows different behavior compared to the other two modes, suggesting that the measurements in Mode 3 were inaccurate. $Q_{\text{loss}}$ curve experiences a similar pattern for Mode 1 and Mode 2. However, in Mode 3, it experiences a significant shift in the last portion of the heat exchanger, as it experiences negative heat losses. This means that in Mode 3, in the last portion of the heat exchanger, more heat was absorbed by the cold fluid than was released by the hot fluid.

In essence, this figure demonstrates the sensitivity of the experiment to any small change in conditions, further complicating data interpretation.

### 5.6 (6) Overall Absorbed, Released, and Lost Heat During the Entire Experiment

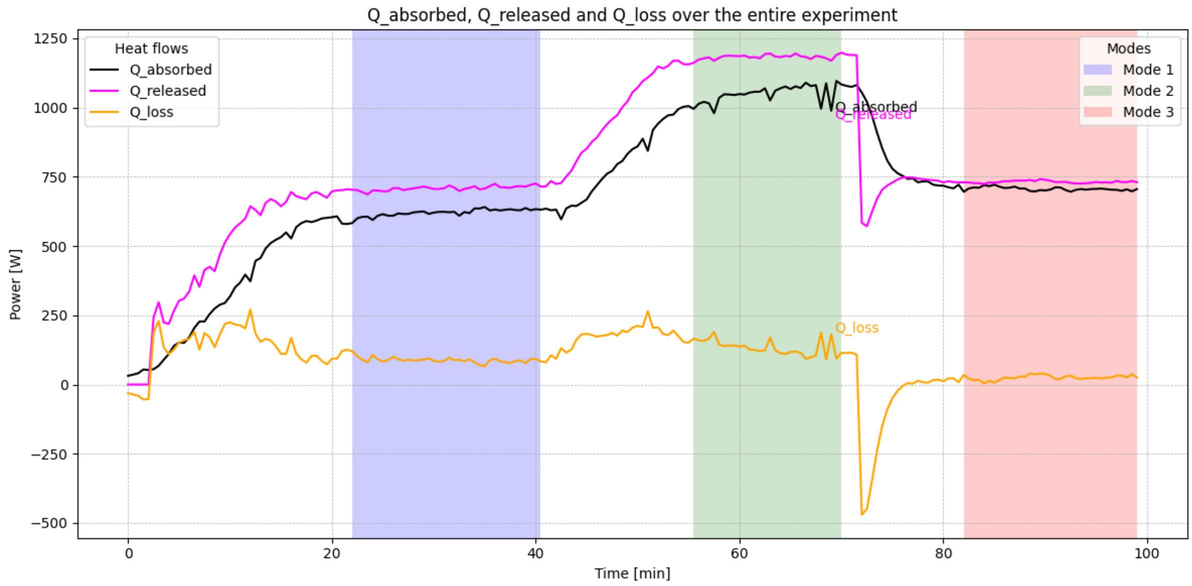


Figure 6: Overall $Q_{\text{absorbed}}$, $Q_{\text{released}}$, and $Q_{\text{lost}}$ over the entire experiment.

Figure 6 presents an integrated view of the total absorbed, released, and lost heat over the full duration of the experiment. The shaded regions represent the three working modes of the experiment, as explained before. The observations are as follows:

- **Integrated Perspective:** The goal of this observation is to understand the temporal evolution of the heat exchange. This figure allows us to integrate previous observations and understand heat transfer better. We can see that our proposed steady-state intervals observe relatively constant heat exchange in time. The biggest fluctuations in heat transfer are observed in the transient periods; the biggest change is occuring in Mode 3, when released heat significantly drops: this is due to the reduced flow rate of the hot fluid, meaning less released heat. After some time, the exchanged heat stabilizes, and the heat losses experienced in Mode 3 are lower than the losses in Mode 1 and Mode 2. However, we should take into consideration the previously observed behavior for Mode 3, in which we had negative heat losses, meaning that other factors are influencing the heat losses in this mode.
- **Indicative of Systemic Issues:** The inability to align total absorbed and released heat with a small, stable loss fraction suggests that the problem is systemic: sensor miscalibrations, inadequate insulation, flow instability, or data processing errors are not isolated occurrences but ongoing throughout the entire experiment. The next section will compare our experimental results to semi-analytical methods. The goal will be to compare how well semi-analytical models depict real-life implementations. Moreover, it will provide more insight into our experimental analysis, as we will be able to better understand the issues that arose from the experimental analysis.

The global analysis reinforces the message that the current experimental setup and protocols are insufficient to achieve close agreement with idealized expectations.

### 5.7 Summary of the Experimental Analysis Insights

Throughout our experimental analysis, the narrative is consistent:

- True steady states with constant parameters are never achieved; quasi-steady modes, chosen according to the experimental analysis still contain fluctuations and anomalies.
- Flow adjustments and external factors prevent stable thermal conditions, making temperature and energy balances volatile.
- Local segment analysis, intended to provide detailed insight, becomes sensitive to even minor measurement or calculation errors.
- Environmental losses, poor insulation, sensor misalignment, and calibration issues all contribute to discrepancies.

The goal of this experimental analysis was to understand the behavior of heat exchangers in practice. The most important points are the understanding of non-constant parameters and how they can be analyzed. Furthermore, we observed the heat transfer; compared to the idealized semi-analytical methods, in this experiment, we have dealt with the introduction of heat losses.

Furthermore, we understood better the importance of different factors, such as thermal insulation of heat exchanger, equipment used for analysis of heat exchangers etc. Finally, we underline the importance of understanding how incorrect and inconsistent measurements, improper equipment and external factors may impact the experimental results.

## 5.8 Semi-Analytical Results and Comparison with Experimental Data

After carrying out an updated semi-analytical analysis for the three modes under investigation, the results are summarized in Table 1. These calculations provide theoretical predictions for the outlet temperatures of the hot and cold fluids, the heat transfer rates, and the pressure drops across the heat exchanger (HX) under idealized assumptions and standard correlations.

Table 1: Updated Semi-Analytical Calculation Results for the Three Modes

| Mode | $T_{1i}(°C)$ | $T_{1o,\text{calc}}(°C)$ | $T_{2i}(°C)$ | $T_{2o,\text{calc}}(°C)$ | $Q$(W) | $\Delta p_1$(kPa) | $\Delta p_2$(kPa) |
|---|---|---|---|---|---|---|---|
| 1 | 31.06 | 27.25 | 18.61 | 22.41 | 468.58 | 0.1157 | 0.2210 |
| 2 | 39.11 | 32.74 | 18.79 | 25.12 | 773.76 | 0.1102 | 0.2115 |
| 3 | 38.73 | 31.95 | 18.80 | 22.18 | 410.18 | 0.0259 | 0.2173 |

## 5.9 Interpretation of Semi-Analytical Results

From the semi-analytical standpoint:

- **Mode 1:** The model predicts that with a hot inlet temperature of around 31°C and cold inlet of about 18.6°C, the hot fluid exits at about 27.25°C and the cold fluid at approximately 22.41°C. The resulting heat transfer rate ($Q$) is 468.58 W. This case implies a moderate temperature driving force and a rather efficient heat exchange with negligible pressure drops. Predicted frictional pressure drops, $\Delta p_1$ and $\Delta p_2$, are small and this indicates that the flow conditions and the HX geometry assumed in the model result in a minimal frictional penalty.
- **Mode 2:** Under Mode 2 conditions, the hot inlet is significantly higher (near 39.1°C) while the cold inlet remains similar to Mode 1. The model predicts a higher temperature difference along the HX, which leads to a higher predicted heat transfer rate of 773.76 W. This higher Q indicates that either an increase in hot inlet temperature or changes in flow conditions effectively increases the rate of heat transfer. The slight increase of $\Delta p_1$ and $\Delta p_2$ over Mode 1 can be due to either higher flow rates, larger temperature dependent fluid property variations, or the semi-empirical correlations utilized for friction and heat transfer. Nevertheless, the obtained pressure drops are still relatively small.
- **Mode 3:** Mode 3 operates under somewhat different conditions than Mode 2, potentially with lower driving temperature differences or altered flow parameters. The semi-analytical result yields a heat transfer rate of 410.18 W, the lowest among the three modes considered. The outlet temperatures (hot: 31.95°C and cold: 22.18°C) indicate a less effective energy exchange compared to Mode 2, possibly due to reduced thermal gradients or shorter effective exchange time. Pressure drops remain negligible, suggesting no significant hydrodynamic challenges in this mode.

Overall, the semi-analytical model paints a coherent internal picture: as the driving temperature difference increases (as seen from Mode 1 to Mode 2), the predicted heat transfer improves. Conversely, when conditions shift unfavorably (Mode 3), the heat transfer capability diminishes.

### 5.10 Comparison with Experimental Findings

When comparing these semi-analytical predictions with the experimental results (as reported previously), substantial discrepancies arise. Table 2 displays the differences between the experimental and semi-analytical outputs, focusing on the released heat ($Q_{\text{released, exp}}$ versus $Q_{\text{analytical}}$) and absorbed heat ($Q_{\text{absorbed, exp}}$ versus $Q_{\text{analytical}}$). It also includes the percentage differences computed from both a released and absorbed perspective.

Table 2: Detailed Comparison Between Experimental and Semi-Analytical Results

| Mode | $Q_{\text{exp}}$(W) | $Q_{\text{analytical}}$(W) | Diff(%) rel. | Diff(%) abs. |
|---|---|---|---|---|
| 1 | 707.73 | 468.58 | 51.04 | 32.44 |
| 2 | 1179.98 | 773.76 | 52.50 | 35.56 |
| 3 | 729.78 | 410.18 | 77.92 | 72.28 |

- **Mode 1:** The experimental heat released is 707.73 W, more than 50% higher than the semi-analytical estimation of 468.58 W. Even accounting for absorbed heat, the difference is still approximately 32%. This discrepancy suggests that the experimental system may be subjected to supplementary effects not captured by the model, such as unaccounted environmental heat gains or sensor inaccuracies that increase the apparent heat transfer.

- **Mode 2:** Mode 2 is expected to align worse due to a stronger temperature difference between the fluids, in fact, the contrast remain large (over 50% from a released perspective and 35% absorbed). The semi-analytical model may not adequately represent the flow conditions, fluid properties at these elevated temperatures, or the real geometry. Alternatively, experimental conditions may involve calibration errors or analytical procedure may be too rough.

- **Mode 3:** This mode presents the largest deviation. The experimental data indicate a much higher heat transfer rate compared to the model's prediction. With the difference reaching as high as 78% (released perspective) and around 72% (absorbed perspective), such a difference strongly suggests that the experimental setup's complexity (possibly poor insulation, sensor misalignment, or environmental factors) profoundly distorts the measured values relative to the model's simpler assumptions.

## 5.11 Heat Loss Distribution by Portion and Mode

A detailed analysis of heat losses across different portions of the heat exchanger for each operational mode was conducted. The updated distribution of heat losses is summarized in Table 3. This table provides the absorbed heat ($Q_{\text{absorbed}}$), released heat ($Q_{\text{ced}}$), and calculated heat loss ($Q_{\text{lost}}$) for each portion and mode.

Table 3: Updated Heat Loss Distribution by Portion and Mode

| Portion | Mode | $Q_{\text{absorbed}}$(W) | $Q_{\text{ced}}$(W) | $Q_{\text{lost}}$(W) |
|---|---|---|---|---|
| 1 | 1 | 193.65 | 181.91 | -11.74 |
| 1 | 2 | 313.00 | 312.95 | -0.05 |
| 1 | 3 | 267.09 | 292.70 | 25.61 |
| 2 | 1 | 110.61 | 145.77 | 35.15 |
| 2 | 2 | 194.00 | 260.83 | 66.83 |
| 2 | 3 | 137.86 | 177.61 | 39.75 |
| 3 | 1 | 171.11 | 183.59 | 12.48 |
| 3 | 2 | 294.06 | 322.94 | 28.87 |
| 3 | 3 | 181.02 | 206.76 | 25.74 |
| 4 | 1 | 144.96 | 196.92 | 51.96 |
| 4 | 2 | 247.16 | 284.82 | 37.66 |
| 4 | 3 | 120.39 | 53.79 | -66.60 |

### 5.11.1 Analysis of Heat Loss Distribution

The revised data highlight key trends and inconsistencies in heat loss across the exchanger portions and modes:

- **Negative Heat Losses**: Portions 1 and 4 exhibit negative heat losses in Mode 1 and 2, respectively. This simply means that within that particular portion, the system is gaining heat. For example, $Q_{\text{lost}} = -11.74$ W is experienced in Portion 1 of Mode 1 and $Q_{\text{lost}} = -66.60$ W in Mode 3 of Portion 4. These results might be due to the very specific geometry of the bends set-up, in which, in fact, during the bends the two fluids take different path without any heat exchange with each other but rather with ambient
- **Positive Heat Losses**: Positive heat losses are observed in most portions for all modes, with particularly high values in Mode 2. For example, Portion 2 in Mode 2 shows $Q_{\text{lost}} = 66.83$ W, the highest among all cases. This is consistent with Mode 2 having the largest overall heat transfer rate, meaning that more energy is dissipated to the ambient space or there is higher experimental inefficiency

### 5.11.2 Cumulative Heat Losses per Mode

The cumulative heat losses for the entire heat exchanger across the three modes are as follows:

- **Mode 1:** Total heat losses = 87.86 W
- **Mode 2:** Total heat losses = 133.30 W
- **Mode 3:** Total heat losses = 24.51 W

Mode 2 exhibits the highest cumulative heat losses, which correlates with its higher heat transfer rate but also highlights its susceptibility to inefficiencies. On the other hand, Mode 3 presents the smallest heat losses, which may point to superior performance or less interference from outside factors but has also presented significant deviations in some areas.

By addressing these points, the experimental results can better align with theoretical expectations, enabling more reliable analysis and optimization of the heat exchanger's performance.

### 5.11.3 Sources of Discrepancies

The large differences between experimental and semi-analytical values underscore potential issues:

- **Instrument Calibration and Placement:** Thermocouples might read incorrectly if not calibrated or if placed near metallic supports or areas of local cooling. Flow meters could introduce errors if pulsations or transient states are not accounted for.
- **Environmental Heat Losses:** The semi-analytical model typically assumes negligible external losses. The environment might draw off much heat if the HX is poorly insulated and will therefore show higher or lower effective transfers.
- **Data Synchronization and Sampling:** If temperatures and flows are not recorded simultaneously or averaged over consistent periods, transient fluctuations can artificially inflate or deflate measured Q values.
- **Model Assumptions and Correlations:** Selection of Nusselt number and friction factor correlations may not accurately represent the real internal HX geometry, surface roughness, or combined convection regimes. Any modification or re-fitting of these correlations to the conditions of the experiment might improve alignment.

# 6 Conclusions

The comprehensive analysis of the counter-flow heat exchanger, encompassing both semi-analytical and experimental methodologies, has yielded valuable insights into its thermal and fluid dynamic performance. Despite the rigorous approach, significant discrepancies between theoretical predictions and experimental measurements were observed, underscoring areas that require further refinement and investigation. The key conclusions of this study are as follows:

- **Model Performance:** The semi-analytical calculations effectively predicted essential performance metrics, including heat transfer rates and pressure drops, providing a foundational baseline for comparison. However, discrepancies between experimental and theoretical heat transfer rates reached up to approximately 78%, particularly in Mode 3.This large deviation suggests possible weaknesses in the assumptions or simplifications made in the model and hence indicates that the present semi-analytical framework is perhaps too simplistic to model the actual system.
- **Experimental Heat Transfer Efficiency:** Experimental measurements indicated consistent heat losses across all operational modes, with Mode 2 exhibiting the highest losses at 133.30 W. The energy ratios ($Q_{\text{absorbed}}/Q_{\text{released}}$) ranged from 0.88 to 0.97, which shows a good efficiency in heat transfer. However, there is still significant loss in some modes indicating inherent inefficiencies within the experimental setup.
- **Need for Model Refinement:** The large deviations of the semi-analytical predictions from the experimental findings bring out the need to revisit and refine the NTU-effectiveness approach. More precisely, the adopted correlations to determine the Nusselt number, ($Nu$), and Darcy friction factor, ($f_D$), may need further tuning and adjustment to accurately represent the real operating condition and geometrical configuration of the heat exchanger. The use of more accurate or different correlations may lead to an improvement in the predictive capability of the model.
- **Heat Loss Distribution Insights:** The distribution of heat loss showed that the heat exchanger is not behaving uniformly. Whereas some sections gave quite adequate results, others revealed larger inefficiencies or discrepancies, probably because of measurement inaccuracies or the influence of environmental factors. This gives reason for an uneven distribution and points to the need for insulation of areas and accurate positioning of sensors to reduce heat loss.

Intrinsic complexity in the experimental arrangement, along with the limitations of instrumentation and environmental influences, precludes any accurate matching of semi-analytical models to the results obtained experimentally. Overcoming these challenges is essential to bridge the gap between theoretical predictions and practical applications, thereby enhancing the reliability and efficiency of future heat exchanger performance analyses. Addressing these issues will not only improve the accuracy of experimental measurements but also refine theoretical models, leading to more effective and optimized heat exchanger designs in subsequent studies.

# 7 Annex

## 7.1 Python and Utilized Libraries

The analysis is performed using Python, using the following libraries:

- `CoolProp`: For thermophysical property calculations.
- `NumPy`: For numerical computations.
- `Pandas`: For data manipulation and exporting to Excel.
- `Matplotlib`: For generating graphs (temperature vs. time, performance diagrams, etc.).
- `SciPy`: For additional scientific functions.

## 7.2 Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
from CoolProp.CoolProp import *
import matplotlib.patches as mpatches

#____________________USEFUL FUNCTIONS____________________________________

def toCelsius(T):
    return T-273.15


def toKelvin(T) :
    return T+273.15

def calcul_properties(p,T,Tw,fluid,v,Dh) :
    mu = PropsSI("V","P", p, "T",T,fluid)
    muw = PropsSI("V","P", p, "T",Tw,fluid)
    rho = PropsSI("D", "P", p, "T", T, fluid)
    k = PropsSI("L", "P", p, "T", T, fluid)
    Cp = PropsSI("C", "P", p, "T", T, fluid)

    Re = v * Dh * rho / mu
    Pr = mu * Cp / k
    Gz =  Dh * Re * Pr /  L
    return mu,muw,rho,k,Cp,Re,Pr,Gz

def calcul_alpha_Nu(mu,muw,rho,k,Cp,Re,Pr,Dh,Gz,gas) :
    if Re < 2000:
        if Gz >10 :
            C = 1.86
            m = 1/3
            n = 1/3
            K = ((Dh/L)**(1/3))*(mu/muw)**(0.14)
        else :
            C = 3.66
            m = 0
            n = 0
            K = 1
```

```python
    elif Re > 2000 and Pr>0.6 and Pr<100 and gas == False:
        C = 0.027
        m = 0.8
        n = 0.33
        K = (mu / muw) ** 0.14

    else :
        C = 0.023
        m = 0.8
        n = 0.4
        K = 1
    Nu = C * (Re ** m) * (Pr ** n) * K
    alpha = Nu * k / Dh
    return alpha,Nu

def calcul_ff(Re,rr) :
    if Re<2000 : f = 16/Re
    elif rr <=0.0001 :
        if Re <3*10e4 : f = 0.079*Re**(-0.25)
        else : f = 0.046*Re**(-0.2)
    elif rr == 0.004 :
        if Re <3*10e4 : f = 0.096*Re**(-0.25)
        else : f = 0.078*Re**(-0.2)
    return f

#_________CREATE A DATAFRAME WITH DATA FROM ATENEA_______________________
tabexp = pd.read_csv('datos_CF_titca_formatted.dat', sep='\t+', skipinitialspace=True, comment='#', engine='python')
tabexp.to_csv('output_analysis.csv', index=False)

#___________CODE PROFESSOR________________________________
print ("----------------------------------COUNTERFLOW-----------------------------------------------
#Plot flows
fig, ax = plt.subplots(1,1, squeeze=False)
fig.suptitle('HX practical activity: Counter flow (our case)')

ax[0,0].plot(tabexp["Time(s)"]/60.0, tabexp["SC-1"],'ro')
ax[0,0].plot(tabexp["Time(s)"]/60.0, tabexp["SC-2"],'b*')
ax[0,0].set_xlabel('Time (min)')
ax[0,0].set_ylabel('Volumetric flow (l/min)')
ax[0,0].grid(True)
ax[0,0].legend(['Hot','Cold'])


fig, ax = plt.subplots(1, 1, squeeze=False, figsize=(12, 6))
fig.suptitle('HX practical activity: T for Counter flow (our case)')
curve1, = ax[0, 0].plot(tabexp["Time(s)"] / 60.0, tabexp["ST-1"], 'ro', label='hot-in')
curve2, = ax[0, 0].plot(tabexp["Time(s)"] / 60.0, tabexp["ST-5"], 'y*', label='hot-out')
curve3, = ax[0, 0].plot(tabexp["Time(s)"] / 60.0, tabexp["ST-10"], 'bo', label='cold-in')
curve4, = ax[0, 0].plot(tabexp["Time(s)"] / 60.0, tabexp["ST-6"], 'g*', label='cold-out')
ax[0,0].set_xlabel('Time (min)')
ax[0,0].set_ylabel('Temperature (ºC)')
# Ajouter des ticks pour l'axe x toutes les 5 minutes
```

```python
ax[0, 0].set_xticks(range(0, int(max(tabexp["Time(s)"] / 60.0)) + 5, 5))
# Activer la grille (lignes verticales et horizontales)
ax[0, 0].grid(True, which='both', linestyle='--', linewidth=0.5)
#ax[0, 0].legend(['hot-in', 'hot-out', 'cold-in', 'cold-out'], loc='upper left', fontsize=10)
zone1 = ax[0, 0].axvspan(tabexp["Time(s)"][44] / 60.0, tabexp["Time(s)"][81] / 60.0, color='blue', alpha=0.2, label='Mode 1')
# Mode 2 : indices 105 à 146
zone2 = ax[0, 0].axvspan(tabexp["Time(s)"][111] / 60.0, tabexp["Time(s)"][140] / 60.0, color='green', alpha=0.2, label='Mode 2')
# Mode 3 : indices 4 à 198
zone3 = ax[0, 0].axvspan(tabexp["Time(s)"][164] / 60.0, tabexp["Time(s)"][198] / 60.0, color='red', alpha=0.2, label='Mode 3')

# Ajouter la première légende pour les courbes
legend_curves = ax[0, 0].legend(handles=[curve1, curve2, curve3, curve4], loc='upper left', fontsize=10, title='Temperatures')
# Ajouter la deuxième légende pour les zones
legend_modes = ax[0, 0].legend(handles=[mpatches.Patch(color='blue', alpha=0.2, label='Mode 1'),
                                        mpatches.Patch(color='green', alpha=0.2, label='Mode 2'),
                                        mpatches.Patch(color='red', alpha=0.2, label='Mode 3')],
                               loc='upper right', fontsize=10, title='Steady state modes')
# Ajouter les deux légendes à l'axe
ax[0, 0].add_artist(legend_curves)
plt.show()


#__________________INPUT DATA__________________________________
fluid_1 = "Water"
fluid_2 = "Water"
Di = 0.016
Do = 0.018
De1 = 0.026
De2 = 0.028
L = 4
rr = 1e-4
lambda_intermediate = 400 #for conduction between
p1i = 101325
p2i = 101325

#_______________MODES DEFINITION_______________________________________
T1i_1 = np.array(tabexp["ST-1"][44:81])
T1i_2 = np.array(tabexp["ST-1"][111:140])
T1i_3 = np.array(tabexp["ST-1"][164:198])
T1o_1 = np.array(tabexp["ST-5"][44:81])
T1o_2 = np.array(tabexp["ST-5"][111:140])
T1o_3 = np.array(tabexp["ST-5"][164:198])

T2i_1 = np.array(tabexp["ST-10"][44:81])
T2i_2 = np.array(tabexp["ST-10"][111:140])
T2i_3 = np.array(tabexp["ST-10"][164:198])
T2o_1 = np.array(tabexp["ST-6"][44:81])
T2o_2 = np.array(tabexp["ST-6"][111:140])
T2o_3 = np.array(tabexp["ST-6"][164:198])
```

```python

#flow rate m^3/s
Q1_1 = np.array(tabexp["SC-1"][44:81])/60000
Q1_2 = np.array(tabexp["SC-1"][111:140])/60000
Q1_3 = np.array(tabexp["SC-1"][164:198])/60000

Q2_1 = np.array(tabexp["SC-2"][44:81])/60000
Q2_2 = np.array(tabexp["SC-2"][111:140])/60000
Q2_3 = np.array(tabexp["SC-2"][164:198])/60000

T1i_mean = np.array([np.mean(T1i_1),np.mean(T1i_2),np.mean(T1i_3)])
T2i_mean = np.array([np.mean(T2i_1),np.mean(T2i_2),np.mean(T2i_3)])
T1o_mean = np.array([np.mean(T1o_1),np.mean(T1o_2),np.mean(T1o_3)])
T2o_mean = np.array([np.mean(T2o_1),np.mean(T2o_2),np.mean(T2o_3)])

Q1_mean = np.array([np.mean(Q1_1),np.mean(Q1_2),np.mean(Q1_3)])
Q2_mean = np.array([np.mean(Q2_1),np.mean(Q2_2),np.mean(Q2_3)])

#_______________________TEMPERATURE DIAGRAM STEADY STATE FOR 3 MODES___________________________
label_1i = ['ST-1','ST-2','ST-3','ST-4']
label_1o = ['ST-2','ST-3','ST-4','ST-5']
label_2i = ['ST-7','ST-8','ST-9','ST-10']
label_2o = ['ST-6','ST-7','ST-8','ST-9']
stations_hot = ["ST-1", "ST-2", "ST-3", "ST-4", "ST-5"]
stations_cold = ["ST-6", "ST-7", "ST-8", "ST-9", "ST-10"]

T1_1 = np.zeros(3)
T1_2 = np.zeros(3)
T1_3 = np.zeros(3)
T1_4 = np.zeros(3)
T1_5 = np.zeros(3)

T2_1 = np.zeros(3)
T2_2 = np.zeros(3)
T2_3 = np.zeros(3)
T2_4 = np.zeros(3)
T2_5 = np.zeros(3)

T1_1[0] = np.mean(np.array(tabexp['ST-1'][44:81]))
T1_2[0] = np.mean(np.array(tabexp['ST-2'][44:81]))
T1_3[0] = np.mean(np.array(tabexp['ST-3'][44:81]))
T1_4[0] = np.mean(np.array(tabexp['ST-4'][44:81]))
T1_5[0] = np.mean(np.array(tabexp['ST-5'][44:81]))

T2_1[0] = np.mean(np.array(tabexp['ST-6'][44:81]))
T2_2[0] = np.mean(np.array(tabexp['ST-7'][44:81]))
T2_3[0] = np.mean(np.array(tabexp['ST-8'][44:81]))
T2_4[0] = np.mean(np.array(tabexp['ST-9'][44:81]))
T2_5[0] = np.mean(np.array(tabexp['ST-10'][44:81]))

T1_1[1] = np.mean(np.array(tabexp['ST-1'][111:140]))
T1_2[1] = np.mean(np.array(tabexp['ST-2'][111:140]))
T1_3[1] = np.mean(np.array(tabexp['ST-3'][111:140]))
T1_4[1] = np.mean(np.array(tabexp['ST-4'][111:140]))
T1_5[1] = np.mean(np.array(tabexp['ST-5'][111:140]))

T2_1[1] = np.mean(np.array(tabexp['ST-6'][111:140]))
```

```python
T2_2[1] = np.mean(np.array(tabexp['ST-7'][111:140]))
T2_3[1] = np.mean(np.array(tabexp['ST-8'][111:140]))
T2_4[1] = np.mean(np.array(tabexp['ST-9'][111:140]))
T2_5[1] = np.mean(np.array(tabexp['ST-10'][111:140]))

T1_1[2] = np.mean(np.array(tabexp['ST-1'][164:198]))
T1_2[2] = np.mean(np.array(tabexp['ST-2'][164:198]))
T1_3[2] = np.mean(np.array(tabexp['ST-3'][164:198]))
T1_4[2] = np.mean(np.array(tabexp['ST-4'][164:198]))
T1_5[2] = np.mean(np.array(tabexp['ST-5'][164:198]))

T2_1[2] = np.mean(np.array(tabexp['ST-6'][164:198]))
T2_2[2] = np.mean(np.array(tabexp['ST-7'][164:198]))
T2_3[2] = np.mean(np.array(tabexp['ST-8'][164:198]))
T2_4[2] = np.mean(np.array(tabexp['ST-9'][164:198]))
T2_5[2] = np.mean(np.array(tabexp['ST-10'][164:198]))

# Création d'une figure avec 3 sous-graphiques côte à côte pour les 3 modes
fig, axs = plt.subplots(1, 3, figsize=(15,5))

# Couleurs de fond pour chaque mode (inspirées du code du professeur)
bg_colors = [(0,0,1,0.2), (0,1,0,0.2), (1,0,0,0.2)]  # bleu, vert, rouge avec alpha

positions = [1,2,3,4,5]
T_amb = [20.1,20.1,20.1,20.1,20.1]

for i in range(3):
    # Définir la couleur de fond du subplot
    axs[i].set_facecolor(bg_colors[i])
    hot_temps = [T1_1[i], T1_2[i], T1_3[i], T1_4[i], T1_5[i]]
    cold_temps = [T2_1[i], T2_2[i], T2_3[i], T2_4[i], T2_5[i]]

    # Tracé des températures du fluide chaud pour le mode i
    axs[i].plot(positions, [T1_1[i], T1_2[i], T1_3[i], T1_4[i], T1_5[i]],
                "-", color="red", marker="o", label="hot fluid")
    for idx, (x, y) in enumerate(zip(positions, hot_temps)):
        axs[i].text(x+0.1, y, stations_hot[idx], fontsize=8, color="red")

    # Tracé des températures du fluide froid pour le mode i
    axs[i].plot(positions, [T2_1[i], T2_2[i], T2_3[i], T2_4[i], T2_5[i]],
                "-", color="blue", marker="o", label="cold fluid")
    for idx, (x, y) in enumerate(zip(positions, cold_temps)):
        axs[i].text(x+0.1, y, stations_cold[idx], fontsize=8, color="blue")

    # Tracé de la température ambiante
    axs[i].plot(positions, T_amb, "--", color="black", label="Ambient temperature")

    axs[i].legend()
    axs[i].set_title("Temperature profile, mode {}".format(i+1))
    axs[i].set_xlabel("Position along the heat exchanger")
    axs[i].set_xticks([])  # On enlève les ticks sur x
    axs[i].set_ylim([16,40])
    if i == 0:
        axs[i].set_ylabel("Temperature [°C]")

plt.tight_layout()
plt.savefig('T_modes_comparison.png')
plt.show()
```

```python


#___________________________SEMI_ANALYTICAL CALCULATION_________________________________
T1o_analytical = np.zeros(3)
T2o_analytical = np.zeros(3)
Q_analytical   = np.zeros(3)
delta_p1_analytical = np.zeros(3)
delta_p2_analytical = np.zeros(3)
p1o = p1i
p2o = p2i

for i in range(3):  # Pour les 3 modes
    # Conditions d'entrée
    T1i = toKelvin(T1i_mean[i])
    T2i = toKelvin(T2i_mean[i])

    # Hypothèses initiales pour démarrer l'itération
    T1o = T1i - 10
    T2o = T2i + 10

    # Surfaces et diamètres hydrauliques
    A1 = np.pi * Di * L
    S1 = np.pi * Di**2 / 4
    Dh1 = Di  # diam. hydraulique pour un tube circulaire simple = Di
    v1 = Q1_mean[i] / S1

    # Annulaire
    # On considère que le second fluide s'écoule dans l'espace annulaire entre Do et De2
    S2 = np.pi * (De1**2 - Do**2) / 4
    Dh2 = 4 * S2 / (np.pi*(De1+Do))
    v2 = Q2_mean[i] / S2

    #Mass flow rate
    m1 = Q1_mean[i]*PropsSI('D', 'T', T1i, 'P', p1i, fluid_1)
    m2 = Q2_mean[i]*PropsSI('D', 'T', T2i, 'P', p2i, fluid_2)

    # Boucle d'itérations pour convergence
    tolerance = 1e-12
    for j in range(5000000):
        T1 = np.mean([T1i, T1o])
        T2 = np.mean([T2i, T2o])
        p1 = np.mean([p1i, p1o])
        p2 = np.mean([p2i, p2o])

        Tw = np.mean([T1,T2])

        # Calcul des propriétés pour chaque fluide
        mu1, muw1, rho1, k1, Cp1, Re1, Pr1, Gz1 = calcul_properties(p1, T1, Tw, fluid_1, v1, Dh1)
        mu2, muw2, rho2, k2, Cp2, Re2, Pr2, Gz2 = calcul_properties(p2, T2, Tw, fluid_2, v2, Dh2)

        # Coefficients de transfert interne et externe
        hi, Nui = calcul_alpha_Nu(mu1, muw1, rho1, k1, Cp1, Re1, Pr1, Dh1, Gz1, False)
        ho, Nuo = calcul_alpha_Nu(mu2, muw2, rho2, k2, Cp2, Re2, Pr2, Dh2, Gz2, False)

```

```
        # Pas de rugosité interne ajoutée pour le moment, pas de Rfi et Rfo
        Rconv1 = Do/(hi *Di)
        Rcond  = Do*np.pi*np.log(Do/Di)/(2*np.pi*lambda_intermediate)
        Rconv2 = 1/(ho)
        UA = np.pi*Do*L/(Rconv1 + Rcond + Rconv2)

        # Calcul NTU et epsilon pour contre-courant
        Cmin = min(Cp1*m1, Cp2*m2)
        Cmax = max(Cp1*m1, Cp2*m2)
        Qmax = Cmin * (T1i - T2i)

        NTU = UA / Cmin
        Z   = Cmin / Cmax
        epsilon = (1 - np.exp(-NTU*(1-Z)))/(1-Z*np.exp(-NTU*(1-Z)))

        Q = epsilon * Qmax
        T2o_new = T2i + Q/(m2*Cp2)
        T1o_new = T1i - Q/(m1*Cp1)

        if abs(T1o_new - T1o) < tolerance and abs(T2o_new - T2o) < tolerance:
            T1o = T1o_new
            T2o = T2o_new

            # Calcul des pertes de charge
            f1 = calcul_ff(Re1, rr)
            tau1 = f1 * rho1 * v1**2 / 2
            delta_p1 = tau1 * A1 / S1
            p1o = p1i - delta_p1

            f2_i = calcul_ff(Re2, rr)
            tau2_i = f2_i * rho2 * v2**2 / 2
            A2_i = np.pi * Do * L

            f2_e = calcul_ff(Re2, rr)
            tau2_e = f2_e * rho2 * v2**2 / 2
            A2_e = np.pi * De2 * L

            delta_p2 = (tau2_i * A2_i + tau2_e * A2_e)/S2
            p2o = p2i - delta_p2
            break

        # Mise à jour
        T1o = T1o_new
        T2o = T2o_new

        # Calcul des pertes de charge
        f1 = calcul_ff(Re1, rr)
        tau1 = f1 * rho1 * v1**2 / 2
        delta_p1 = tau1 * A1 / S1
        p1o = p1i - delta_p1

        f2_i = calcul_ff(Re2, rr)
        tau2_i = f2_i * rho2 * v2**2 / 2
        A2_i = np.pi * Do * L

        f2_e = calcul_ff(Re2, rr)
        tau2_e = f2_e * rho2 * v2**2 / 2
        A2_e = np.pi * De2 * L
```

```python

        delta_p2 = (tau2_i * A2_i + tau2_e * A2_e)/S2
        p2o = p2i - delta_p2


    # Stockage des résultats
    Q_analytical[i] = Q
    T1o_analytical[i] = T1o
    T2o_analytical[i] = T2o
    delta_p1_analytical[i] = delta_p1
    delta_p2_analytical[i] = delta_p2


# Création d'un tableau récapitulatif pour les 3 modes
data = {
    'Mode': [1,2,3],
    'T1i(°C)': T1i_mean,
    'T1o_calc(°C)': T1o_analytical - 273.15,
    'T2i(°C)': T2i_mean,
    'T2o_calc(°C)': T2o_analytical - 273.15,
    'Q(W)': Q_analytical,
    'dp1(kPa)': delta_p1_analytical/1000,
    'dp2(kPa)': delta_p2_analytical/1000
}

df_res = pd.DataFrame(data)
print("_____________Semi-analytical calculation______________")
print(df_res)
#___________________EXPERIMENTAL CALCULATION__________________________
Q_released_exp = np.zeros(3)
Q_absorbed_exp = np.zeros(3)
p1_out = p1i
p2_out = p2i

for j in range(3):
    # Convert average inlet/outlet temperatures to Kelvin
    T_hot_in = toKelvin(T1i_mean[j])
    T_cold_in = toKelvin(T2i_mean[j])
    T_hot_out = toKelvin(T1o_mean[j])
    T_cold_out = toKelvin(T2o_mean[j])

    # Compute average properties at mean temperatures and pressures
    Cp_hot = PropsSI("C","T", np.mean([T_hot_in, T_hot_out]), "P",
    ↪  np.mean([p1i,p1_out]), fluid_1)
    Cp_cold = PropsSI("C","T", np.mean([T_cold_in,T_cold_out]), "P", np.mean([p2i,
    ↪  p2_out]), fluid_2)

    rho_hot = PropsSI("D","T", T_hot_in, "P", p1i, fluid_1)
    rho_cold = PropsSI("D","T", T_cold_in, "P",p2i, fluid_2)

    # Mass flow rates based on density and volumetric flow
    # (Previously named deb1_mean, deb2_mean, now Q1_mean, Q2_mean to ensure
    ↪  consistency)
    m_hot = Q1_mean[j]*rho_hot
    m_cold = Q2_mean[j]*rho_cold

    # Calculate experimental heat exchange
    # Q_released: hot fluid releases heat, Q_absorbed: cold fluid absorbs heat
```

```
    Q_released = m_hot * Cp_hot * (T_hot_in - T_hot_out)
    Q_absorbed = m_cold * Cp_cold * (T_cold_out - T_cold_in)

    Q_loss =   Q_released - Q_absorbed
    efficiency = np.abs(Q_absorbed / Q_released)

    Q_released_exp[j] = Q_released
    Q_absorbed_exp[j]=Q_absorbed

    # Print results for each mode
    print("_______Experimental Case {}____________".format(j+1))
    print("Q_absorbed_exp = {:.2f} W".format(Q_absorbed))
    print("Q_released_exp = {:.2f} W".format(Q_released))
    print("Q_loss_exp = {:.2f} W".format(Q_loss))
    print("Energy ratio (Q_abs/Q_rel) = {:.2f}".format(efficiency))
    print("_________________________________________")

# Now compare experimental and analytical results in a single table
# We already have Q_analytical from the analytical calculations.
# Let's calculate percentage difference.

difference_percent = np.abs(Q_analytical - Q_released_exp)/Q_analytical * 100
difference_percent_abs = np.abs(Q_analytical - Q_absorbed_exp)/Q_analytical * 100

comparison_data = {
    'Mode': [1, 2, 3],
    'Q_exp(W)': Q_released_exp,
    'Q_analytical(W)': Q_analytical,
    'Difference(%) released': difference_percent,
    'Difference(%) absorbed': difference_percent_abs
    }

df_comparison = pd.DataFrame(comparison_data)
print("______Comparison between Experimental and Analytical Results_________")
print(df_comparison)

#___________________LOSSES DISTRIBTION__________________________
def losses():
    # On se base sur les variables globales déjà définies :
    # T1i_mean, T1o_mean, T2i_mean, T2o_mean, Q1_mean, Q2_mean, p1i, p2i, fluid_1,
    ↪  fluid_2
    # et le tableau tabexp

    label_1i = ['ST-1','ST-2','ST-3','ST-4']
    label_1o = ['ST-2','ST-3','ST-4','ST-5']
    label_2i = ['ST-7','ST-8','ST-9','ST-10']
    label_2o = ['ST-6','ST-7','ST-8','ST-9']

    Qlost_all = np.zeros((4,3))
    Qced_all = np.zeros((4,3))
    Qabs_all = np.zeros((4,3))
    n_BC_all = np.zeros((4,3))

    # On dispose déjà de T1i_mean, T1o_mean, T2i_mean, T2o_mean, Q1_mean, Q2_mean
    ↪  calculés globalement.
    # Ici, on va supposer que la répartition par portion (j) ne sert qu'à illustrer
    # un calcul local, mais on ne va pas recalculer les moyennes de débit. On
    ↪  utilisera Q1_mean et Q2_mean globaux.
```

```
    # Les ranges T1i_range_x, etc. sont lus mais on n'en a pas réellement besoin pour le calcul final,
    # car on a déjà les moyennes globales. Si on veut réellement faire un calcul portion par portion,
    # il faudrait redéfinir la logique. Pour éviter les NaN, on utilisera les moyennes globales déjà existantes.

    for j in range(4):
        # On lit les données portion par portion (on pourrait l'enlever si inutile)
        T1i_range_1 = np.array(tabexp[label_1i[j]][44:81])
        T1i_range_2 = np.array(tabexp[label_1i[j]][111:140])
        T1i_range_3 = np.array(tabexp[label_1i[j]][164:198])
        T1o_range_1 = np.array(tabexp[label_1o[j]][44:81])
        T1o_range_2 = np.array(tabexp[label_1o[j]][111:140])
        T1o_range_3 = np.array(tabexp[label_1o[j]][164:198])

        T2i_range_1 = np.array(tabexp[label_2i[j]][44:81])
        T2i_range_2 = np.array(tabexp[label_2i[j]][111:140])
        T2i_range_3 = np.array(tabexp[label_2i[j]][164:198])
        T2o_range_1 = np.array(tabexp[label_2o[j]][44:81])
        T2o_range_2 = np.array(tabexp[label_2o[j]][111:140])
        T2o_range_3 = np.array(tabexp[label_2o[j]][164:198])

        # On calcule les moyennes par mode pour cette portion
        T1i_portion = np.array([np.mean(T1i_range_1), np.mean(T1i_range_2), np.mean(T1i_range_3)])
        T1o_portion = np.array([np.mean(T1o_range_1), np.mean(T1o_range_2), np.mean(T1o_range_3)])
        T2i_portion = np.array([np.mean(T2i_range_1), np.mean(T2i_range_2), np.mean(T2i_range_3)])
        T2o_portion = np.array([np.mean(T2o_range_1), np.mean(T2o_range_2), np.mean(T2o_range_3)])

        p1o = p1i
        p2o = p2i
        for i in range(3):
            # Convertir en Kelvin
            T1i_K = toKelvin(T1i_portion[i])
            T1o_K = toKelvin(T1o_portion[i])
            T2i_K = toKelvin(T2i_portion[i])
            T2o_K = toKelvin(T2o_portion[i])

            # Utiliser Q1_mean[i] et Q2_mean[i] au lieu de deb1_mean[i], deb2_mean[i]
            m1 = Q1_mean[i] * PropsSI("D","T",T1i_K,"P",p1i,fluid_1)
            m2 = Q2_mean[i] * PropsSI("D","T",T2i_K,"P",p2i,fluid_2)

            Cp1 = PropsSI("C","T",np.mean([T1i_K,T1o_K]),"P",np.mean([p1i,p1o]),fluid_1)
            Cp2 = PropsSI("C","T", np.mean([T2i_K,T2o_K]), "P", np.mean([p2i, p2o]), fluid_2)

            Qced = m1*Cp1*(T1i_K-T1o_K)
            Qabs = m2*Cp2*(T2o_K-T2i_K)

            n_BC_all[j][i] = Qabs/Qced
            Qabs_all[j][i] = Qabs
            Qced_all[j][i] = Qced
            Qlost_all[j][i] = Qced - Qabs
```

```
            print("=============Portion {}, Mode {}=============".format(j+1,i+1))
            print("T1i = {:.2f} °C, T2o = {:.2f} °C, T1o = {:.2f} °C, T2i = {:.2f} °C".format(
                T1i_portion[i], T2o_portion[i], T1o_portion[i], T2i_portion[i]))

    for j in range(4):
        for i in range(3):
            print("=============Portion {}, Mode {}=============".format(j+1,i+1))
            print("Qabs = {:.2f} W\nQced = {:.2f} W\nQlost = {:.2f} W\nn_BC = {:.2f}".format(
                Qabs_all[j][i],Qced_all[j][i],Qlost_all[j][i],n_BC_all[j][i]))

    N = 4
    ind = np.arange(N)  # the x locations for the groups
    width = 0.27       # the width of the bars

    fig = plt.figure()
    ax = fig.add_subplot(111)

    val_1 = Qlost_all.T[0]
    val_2 = Qlost_all.T[1]
    val_3 = Qlost_all.T[2]

    # Pour éviter les erreurs, on peut s'assurer qu'il n'y a pas de NaN
    val_1 = np.nan_to_num(val_1, nan=0.0)
    val_2 = np.nan_to_num(val_2, nan=0.0)
    val_3 = np.nan_to_num(val_3, nan=0.0)

    rects1 = ax.bar(ind, val_1, width, color='blue', alpha = 0.4)
    rects2 = ax.bar(ind + width, val_2, width, color='green', alpha = 0.4)
    rects3 = ax.bar(ind + width * 2, val_3, width, color='red', alpha = 0.4)

    for i in range(3):
        print("-------Mode{}-------".format(i+1))
        print("losses = {:.2f}".format(sum(Qlost_all.T[i])))

    ax.set_ylabel('$\dot{Q}_{losses}$ [W]')
    ax.set_xticks(ind+width)
    ax.set_xticklabels(('Portion 1', 'Portion 2', 'Portion 3', 'Portion 4'))
    ax.legend((rects1[0], rects2[0], rects3[0]), ('Mode 1', 'Mode 2', 'Mode 3'), loc='lower left')

    def autolabel(rects):
        for rect in rects:
            h = rect.get_height()
            # Conversion en int seulement si h n'est pas NaN
            if not np.isnan(h):
                ax.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%d'%int(h),
                        ha='center', va='bottom')

    autolabel(rects1)
    autolabel(rects2)
    autolabel(rects3)
    plt.title('Heat losses along the heat exchanger for different stationary modes')
    plt.savefig("losses")
    plt.show()
```

```python
# Appel
L = losses()

#________________________________TEST______________________________
def losses():
    # On se base sur les variables globales déjà définies :
    # T1i_mean, T1o_mean, T2i_mean, T2o_mean, Q1_mean, Q2_mean, p1i, p2i, fluid_1, fluid_2
    # et le tableau tabexp

    label_1i = ['ST-1','ST-2','ST-3','ST-4']
    label_1o = ['ST-2','ST-3','ST-4','ST-5']
    label_2i = ['ST-7','ST-8','ST-9','ST-10']
    label_2o = ['ST-6','ST-7','ST-8','ST-9']

    Qlost_all = np.zeros((4,3))
    Qced_all = np.zeros((4,3))
    Qabs_all = np.zeros((4,3))
    n_BC_all = np.zeros((4,3))

    for j in range(4):
        T1i_range_1 = np.array(tabexp[label_1i[j]][44:81])
        T1i_range_2 = np.array(tabexp[label_1i[j]][111:140])
        T1i_range_3 = np.array(tabexp[label_1i[j]][164:198])
        T1o_range_1 = np.array(tabexp[label_1o[j]][44:81])
        T1o_range_2 = np.array(tabexp[label_1o[j]][111:140])
        T1o_range_3 = np.array(tabexp[label_1o[j]][164:198])

        T2i_range_1 = np.array(tabexp[label_2i[j]][44:81])
        T2i_range_2 = np.array(tabexp[label_2i[j]][111:140])
        T2i_range_3 = np.array(tabexp[label_2i[j]][164:198])
        T2o_range_1 = np.array(tabexp[label_2o[j]][44:81])
        T2o_range_2 = np.array(tabexp[label_2o[j]][111:140])
        T2o_range_3 = np.array(tabexp[label_2o[j]][164:198])

        T1i_portion = np.array([np.mean(T1i_range_1), np.mean(T1i_range_2), np.mean(T1i_range_3)])
        T1o_portion = np.array([np.mean(T1o_range_1), np.mean(T1o_range_2), np.mean(T1o_range_3)])
        T2i_portion = np.array([np.mean(T2i_range_1), np.mean(T2i_range_2), np.mean(T2i_range_3)])
        T2o_portion = np.array([np.mean(T2o_range_1), np.mean(T2o_range_2), np.mean(T2o_range_3)])

        p1o = p1i
        p2o = p2i
        for i in range(3):
            T1i_K = toKelvin(T1i_portion[i])
            T1o_K = toKelvin(T1o_portion[i])
            T2i_K = toKelvin(T2i_portion[i])
            T2o_K = toKelvin(T2o_portion[i])

            m1 = Q1_mean[i] * PropsSI("D","T",T1i_K,"P",p1i,fluid_1)
            m2 = Q2_mean[i] * PropsSI("D","T",T2i_K,"P",p2i,fluid_2)

            Cp1 = PropsSI("C","T",np.mean([T1i_K,T1o_K]),"P",np.mean([p1i,p1o]),fluid_1)
```

```
            Cp2 = PropsSI("C","T", np.mean([T2i_K,T2o_K]), "P", np.mean([p2i, p2o]), fluid_2)

            Qced = m1*Cp1*(T1i_K-T1o_K)
            Qabs = m2*Cp2*(T2o_K-T2i_K)

            n_BC_all[j][i] = Qabs/Qced
            Qabs_all[j][i] = Qabs
            Qced_all[j][i] = Qced
            Qlost_all[j][i] = Qced - Qabs

            print("============Portion {}, Mode {}=============".format(j+1,i+1))
            print("T1i = {:.2f} °C, T2o = {:.2f} °C, T1o = {:.2f} °C, T2i = {:.2f} °C".format(
                T1i_portion[i], T2o_portion[i], T1o_portion[i], T2i_portion[i]))

    for j in range(4):
        for i in range(3):
            print("============Portion {}, Mode {}=============".format(j+1,i+1))
            print("Qabs = {:.2f} W\nQced = {:.2f} W\nQlost = {:.2f} W\nn_BC = {:.2f}".format(
                Qabs_all[j][i],Qced_all[j][i],Qlost_all[j][i],n_BC_all[j][i]))

    return Qabs_all, Qced_all, Qlost_all, n_BC_all

# Appel de la fonction losses()
Qabs_all, Qced_all, Qlost_all, n_BC_all = losses()

# Maintenant que Qced_all, Qabs_all, Qlost_all, n_BC_all sont définis, on peut tracer les graphiques.

# 1) Graphiques Q_loss, Q_absorbed et Q_released pour chaque mode en fonction de la portion
fig, axs = plt.subplots(1, 3, figsize=(15,5))

bg_colors = [(0,0,1,0.2), (0,1,0,0.2), (1,0,0,0.2)]
positions = np.array([1,2,3,4])

for i in range(3):
    axs[i].set_facecolor(bg_colors[i])

    axs[i].plot(positions, Qced_all[:,i], 'o-', color='red', label='Q_released (hot)')
    axs[i].plot(positions, Qabs_all[:,i], 's-', color='blue', label='Q_absorbed (cold)')
    axs[i].plot(positions, Qlost_all[:,i], 'd-', color='black', label='Q_loss')

    axs[i].set_xlabel("Portion")
    axs[i].set_title("Mode {}".format(i+1))
    axs[i].grid(True, linestyle='--', linewidth=0.5)
    if i == 0:
        axs[i].set_ylabel("Power [W]")
    axs[i].legend()

plt.tight_layout()
plt.savefig('Q_comparison_per_portion.png')
plt.show()

# 2) Graphique Q_loss moyen en fonction du temps
```

```python
# Les indices des modes sont déjà définis : mode1_indices, mode2_indices,
↪  mode3_indices
# On utilise la fonction calc_Q_loss_time déjà définie.

def calc_Q_loss_time(interval_indices, fluid_1, fluid_2):
    Q_loss_time = []
    times = tabexp["Time(s)"][interval_indices]/60.0  # en minutes
    for idx in interval_indices:
        # Températures instantanées
        T1i_inst = toKelvin(tabexp["ST-1"][idx])
        T1o_inst = toKelvin(tabexp["ST-5"][idx])
        T2i_inst = toKelvin(tabexp["ST-10"][idx])
        T2o_inst = toKelvin(tabexp["ST-6"][idx])

        # Débits volumiques instantanés
        Q1_inst = tabexp["SC-1"][idx]/60000.0
        Q2_inst = tabexp["SC-2"][idx]/60000.0

        # Propriétés
        p1_mean = p1i  # On suppose pression constante
        p2_mean = p2i

        T_hot_mean = np.mean([T1i_inst,T1o_inst])
        T_cold_mean = np.mean([T2i_inst,T2o_inst])

        Cp_hot = PropsSI("C","T",T_hot_mean,"P",p1_mean,fluid_1)
        Cp_cold = PropsSI("C","T",T_cold_mean,"P",p2_mean,fluid_2)

        rho_hot = PropsSI("D","T",T_hot_mean,"P",p1_mean,fluid_1)
        rho_cold = PropsSI("D","T",T_cold_mean,"P",p2_mean,fluid_2)

        m_hot = Q1_inst * rho_hot
        m_cold = Q2_inst * rho_cold

        Qced_inst = m_hot*Cp_hot*(T1i_inst - T1o_inst)
        Qabs_inst = m_cold*Cp_cold*(T2o_inst - T2i_inst)

        Q_loss_inst = Qced_inst - Qabs_inst
        Q_loss_time.append(Q_loss_inst)

    return times, np.array(Q_loss_time)


times1, Q_loss_1 = calc_Q_loss_time(range(44,81), fluid_1, fluid_2)
times2, Q_loss_2 = calc_Q_loss_time(range(111,140), fluid_1, fluid_2)
times3, Q_loss_3 = calc_Q_loss_time(range(164,198), fluid_1, fluid_2)

fig, ax = plt.subplots()
ax.plot(times1, Q_loss_1, '-', color='orange', label='Q_loss Mode 1')
ax.plot(times2, Q_loss_2, '-', color='blue', label='Q_loss Mode 2')
ax.plot(times3, Q_loss_3, '-', color='purple', label='Q_loss Mode 3')
ax.set_xlabel('Time [min]')
ax.set_ylabel('Q_loss [W]')
ax.set_title('Q_loss over time for different modes')
ax.grid(True, linestyle='--', linewidth=0.5)
ax.legend()
plt.savefig('Q_loss_over_time.png')
plt.show()
```

```python


# Recalcul des Q_abs, Q_rel, Q_loss sur toute la durée
def calc_Q_values_time(tabexp, fluid_1, fluid_2, p1i, p2i):
    times = tabexp["Time(s)"].values/60.0  # en minutes
    Q_abs_time = []
    Q_rel_time = []
    Q_loss_time = []
    for idx in range(len(tabexp)):
        T1i_inst = toKelvin(tabexp["ST-1"][idx])
        T1o_inst = toKelvin(tabexp["ST-5"][idx])
        T2i_inst = toKelvin(tabexp["ST-10"][idx])
        T2o_inst = toKelvin(tabexp["ST-6"][idx])

        Q1_inst = tabexp["SC-1"][idx]/60000.0
        Q2_inst = tabexp["SC-2"][idx]/60000.0

        p1_mean = p1i
        p2_mean = p2i

        T_hot_mean = np.mean([T1i_inst,T1o_inst])
        T_cold_mean = np.mean([T2i_inst,T2o_inst])

        # Propriétés moyennes
        Cp_hot = PropsSI("C","T",T_hot_mean,"P",p1_mean,fluid_1)
        Cp_cold = PropsSI("C","T",T_cold_mean,"P",p2_mean,fluid_2)

        rho_hot = PropsSI("D","T",T_hot_mean,"P",p1_mean,fluid_1)
        rho_cold = PropsSI("D","T",T_cold_mean,"P",p2_mean,fluid_2)

        m_hot = Q1_inst * rho_hot
        m_cold = Q2_inst * rho_cold

        Q_released_inst = m_hot * Cp_hot * (T1i_inst - T1o_inst)
        Q_absorbed_inst = m_cold * Cp_cold * (T2o_inst - T2i_inst)
        Q_loss_inst = Q_released_inst - Q_absorbed_inst

        Q_abs_time.append(Q_absorbed_inst)
        Q_rel_time.append(Q_released_inst)
        Q_loss_time.append(Q_loss_inst)

    return times, np.array(Q_abs_time), np.array(Q_rel_time), np.array(Q_loss_time)

times, Q_abs_full, Q_rel_full, Q_loss_full = calc_Q_values_time(tabexp, fluid_1, fluid_2, p1i, p2i)

fig, ax = plt.subplots(figsize=(12,6))
# Tracé des courbes avec nouvelles couleurs
line_abs, = ax.plot(times, Q_abs_full, '-', color='black', label='Q_absorbed')
line_rel, = ax.plot(times, Q_rel_full, '-', color='magenta', label='Q_released')
line_loss,= ax.plot(times, Q_loss_full, '-', color='orange', label='Q_loss')

ax.set_xlabel('Time [min]')
ax.set_ylabel('Power [W]')
ax.set_title('Q_absorbed, Q_released and Q_loss over the entire experiment')
ax.grid(True, linestyle='--', linewidth=0.5)

```

```python
# Ajout des zones pour les modes, mêmes couleurs (blue, green, red) en alpha=0.2 sans hachures
# Mode 1 : indices 32 à 85
ax.axvspan(times[44], times[81], facecolor='blue', alpha=0.2, label='Mode 1')
# Mode 2 : indices 111 à 143
ax.axvspan(times[111], times[140], facecolor='green', alpha=0.2, label='Mode 2')
# Mode 3 : indices 162 à 198
ax.axvspan(times[164], times[198], facecolor='red', alpha=0.2, label='Mode 3')

# Gestion des légendes
# La première légende pour les courbes Q
legend_curves = ax.legend(handles=[line_abs, line_rel, line_loss], loc='upper left', title='Heat flows')
ax.add_artist(legend_curves)

# Deuxième légende pour les modes
from matplotlib.patches import Patch
legend_modes = [Patch(facecolor='blue', alpha=0.2, label='Mode 1'),
                Patch(facecolor='green', alpha=0.2, label='Mode 2'),
                Patch(facecolor='red', alpha=0.2, label='Mode 3')]
ax.legend(handles=legend_modes, loc='upper right', title='Modes')

# Ajout de petites annotations sur le graphique pour identifier les courbes
ax.text(times[-1]*0.7, np.max(Q_abs_full)*0.9, "Q_absorbed", color='black', fontsize=10)
ax.text(times[-1]*0.7, np.max(Q_rel_full)*0.8, "Q_released", color='magenta', fontsize=10)
ax.text(times[-1]*0.7, np.max(Q_loss_full)*0.7, "Q_loss", color='orange', fontsize=10)

plt.tight_layout()
plt.savefig('Q_all_over_time_with_modes_colored.png')
plt.show()
```