

---

**Travail de fin d'études et stage[BR]- Travail de fin d'études : Modeling and experimental evaluation of a reversible air conditioning system of an electric car[BR]- Stage d'insertion professionnelle**

**Auteur :** Peeters, Marie

**Promoteur(s) :** Gendebien, Samuel

**Faculté :** Faculté des Sciences appliquées

**Diplôme :** Master en ingénieur civil électromécanicien, à finalité spécialisée en énergétique

**Année académique :** 2022-2023

**URI/URL :** <http://hdl.handle.net/2268.2/17719>

---

*Avertissement à l'attention des usagers :*

*Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.*

*Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.*

---



**Modeling and experimental evaluation of a reversible  
air conditioning system of an electric car**

*Master's thesis completed in order to obtain the degree of Master of Science in  
Electromechanical Engineering*

Author : **Peeters Marie**  
Promotor : **Gendebien Samuel**

---

**UNIVERSITY OF LIÈGE**  
**Applied Thermodynamic Laboratory**  
Campus du Sart-Tilman  
Allée de la Découverte, 17  
B49 Building, Parking P33  
B 4000 LIEGE (Belgium)  
Tel : +32 4 366 48 00  
Website : <http://www.labohtap.uliege.be/>

Academic year 2022-2023

---

## Abstract

This master thesis deals with the modeling and the experimental evaluation of the reversible air conditioning system of the Renault ZOE. Numerical models have been developed in order to evaluate its performance for different operating conditions in cooling and heating mode. Moreover, experimental campaigns have been carried out on a test bench representing this reversible air conditioning system. Different driving conditions have been tested. For both modes, the influence of outdoor temperature and compressor speed on different parameters has been studied. Finally, the numerical model of the reversible heat pump and the model of the compressor has been validated with data acquired during experimental campaigns. Predictions of consumption and compressor discharge temperature give satisfactory results. However, the compressor model overestimates actual refrigerant flow. Concerning the heat pump model, it does not give satisfactory predictions of heat flow rates for evaporator and condenser. The numerical model developed during this master thesis represents a good basis but still needs to be improved to match the experimental data.

---

## Acknowledgements

First of all, I would like to thank the thermodynamics laboratory Labothap for the warm welcome I received during my internship.

I would particularly like to express my deepest gratitude to my promotor, Sam Gendebien for his availability and kindness.

I would like to thank professor Vincent Lemort for his invaluable advice and help.

I acknowledge the support and guidance provided by all PhD students of the Labothap.

Special thanks to the laboratory technicians for their precious help and patience.

Finally, I would like to thank all my friends and family for the emotional support they have given me throughout this period.

# Contents

<b>Nomenclature</b>	<b>12</b>
<b>I Introduction</b>	<b>13</b>
<b>1 Context</b>	<b>13</b>
<b>2 Objectives</b>	<b>13</b>
<b>3 Literature review</b>	<b>13</b>
<b>II Bench description</b>	<b>16</b>
<b>1 Overview</b>	<b>16</b>
1.1 Bench layout . . . . .	16
1.2 P&ID . . . . .	18
1.3 Refrigerant . . . . .	21
<b>2 Components description</b>	<b>22</b>
2.1 Compressor . . . . .	22
2.1.1 Functioning . . . . .	22
2.2 Control box . . . . .	24
2.3 External exchanger (HTX1) . . . . .	25
2.4 Front fan (FAN2) . . . . .	26
2.5 Orifice tubes . . . . .	26
2.6 Accumulator . . . . .	27
2.7 Electrovalves . . . . .	28
2.8 HVAC module . . . . .	28
2.8.1 Pulser (FAN1) . . . . .	29
2.8.2 Inner condenser (HTX2) . . . . .	29
2.8.3 Inner evaporator (HTX3) . . . . .	30
2.8.4 External heating (PTC) . . . . .	30
2.8.5 Servomotors and flaps . . . . .	31
2.9 Auxiliaries . . . . .	33
2.9.1 Portable Air Conditioning Unit . . . . .	33
2.9.2 Wind tunnel . . . . .	34
<b>3 Measuring device</b>	<b>35</b>
3.1 Sensors . . . . .	35
3.2 Orifice plate . . . . .	36

3.3 Anemometer . . . . .	36
<b>4 Performance</b>	<b>37</b>
4.1 COP . . . . .	37
4.2 Isentropic efficiency . . . . .	37
<b>5 Bench control</b>	<b>38</b>
5.1 Compressor speed control . . . . .	38
5.2 Mobile units control . . . . .	38
5.3 FAN1 control . . . . .	39
5.4 FAN2 control . . . . .	39
<b>6 Bench improvements</b>	<b>41</b>
<b>III Experimental results</b>	<b>42</b>
<b>1 Description of the tests</b>	<b>42</b>
<b>2 Uncertainty propagation</b>	<b>43</b>
2.1 COP <sub>heating</sub> . . . . .	43
2.2 Compressor power . . . . .	44
2.3 Condenser power . . . . .	44
<b>3 Labview's interface</b>	<b>45</b>
<b>4 First experimental campaign</b>	<b>45</b>
<b>5 Second experimental campaign</b>	<b>47</b>
5.1 Cooling mode . . . . .	47
5.1.1 Refrigerant mass flow rate . . . . .	48
5.1.2 Compressor . . . . .	48
5.1.3 Determination of oil mass flow rate . . . . .	50
5.1.4 Evaporator and condenser . . . . .	52
5.1.5 Energy balance . . . . .	52
5.1.6 COP . . . . .	54
5.1.7 Isentropic efficiency . . . . .	54
5.2 Heating mode . . . . .	55
5.2.1 Refrigerant mass flow rate . . . . .	55
5.2.2 Compressor . . . . .	56
5.2.3 Evaporator and condenser . . . . .	56
5.2.4 Energy balance . . . . .	57

---

<b>6 Air flow rate</b>	<b>58</b>
6.1 Anemometer . . . . .	59
6.1.1 Air flow on the external heat exchanger (HTX1) . . . . .	59
6.1.2 Air flow inside the HVAC module . . . . .	59
6.2 Orifice plate . . . . .	60
6.3 Balance on the heat exchanger . . . . .	60
6.3.1 Air flow on HTX1 . . . . .	61
6.3.2 Air flow inside the HVAC module . . . . .	61
6.4 Comparison of the results . . . . .	62
<b>IV Numerical model</b>	<b>64</b>
<b>1 Overview</b>	<b>64</b>
<b>2 Component modeling</b>	<b>66</b>
2.1 Compressor . . . . .	66
2.2 Heat exchangers . . . . .	67
2.2.1 Architecture . . . . .	67
2.2.2 Pressure drops . . . . .	68
2.2.3 Heat transfer coefficient U . . . . .	68
2.2.4 Condenser modelisation . . . . .	68
2.2.5 Evaporator modelisation . . . . .	69
2.3 Orifice tube . . . . .	70
2.4 Accumulator . . . . .	71
<b>3 Reversible heat pump</b>	<b>72</b>
<b>V Results validation</b>	<b>74</b>
<b>1 Methodology</b>	<b>74</b>
<b>2 Compressor</b>	<b>74</b>
<b>3 Validation of the model</b>	<b>77</b>
<b>4 Discussion and perspectives</b>	<b>78</b>
<b>VI Conclusion</b>	<b>79</b>
<b>References</b>	<b>80</b>

---

<b>Annexe A</b>	82
<b>Annexe B</b>	90
<b>Annexe C</b>	96
<b>Annexe D</b>	100



## List of Figures

1	Bench view (1)	16
2	Bench view (2)	17
3	Bench view (3)	17
4	Bench view (4)	18
5	P&ID of the bench	19
6	P&ID: cooling mode	20
7	P&ID: heating mode	20
8	Representation of the different refrigerant states	21
9	DENSO electric scroll compressor ES34C	22
10	Compressor speed according to duty cycle [12]	23
11	Correspondence between duty cycle of $PWM_{out}$ and the compressor power [5]	23
12	Power supply and control of the compressor	24
13	Wiring diagram of the PWM IN input	25
14	Front exchanger (HTX1)	26
15	Front fan (FAN2)	26
16	Calibrated expansion valve [13]	27
17	Accumulator	27
18	Electrovalve [13]	28
19	HVAC module diagram [13]	29
20	Pulser (FAN1) [13]	29
21	Inner condenser (HTX2) [13]	30
22	Inner evaporator (HTX3) [13]	30
23	PTC 12V [13]	31
24	HVAC module	32
25	Roles of the 3 servomotors	33
26	Portable Air Conditioning Unit	34
27	Wind tunnel	34
28	Output air flow measurement	36
29	Anemometer used for the air flow rate measurement (datasheet)	37
30	Control box of the mobile unit	38
31	FAN1 control	39
32	Front fan (FAN2)	40
33	Duty cycle analysis	40
34	Evolution of the compressor outlet and surface temperatures	42
35	Labview's interface	45
36	Evolution of the outlet pressure during the accident	46
37	Evolution of $T_2$ and $p_2$ as a function of the compressor speed	46
38	Refrigerant mass flow rate for different compressor speeds and outdoor temperatures	48

39	Compressor consumption for different compressor speeds and outdoor temperatures	49
40	Evolution of current sensor measurement over time	49
41	Oil mass flow rate for different compressor speeds and outdoor temperatures	51
42	OCR for different compressor speeds and outdoor temperatures	51
43	$\dot{Q}_{ev}$ and $\dot{Q}_{cd}$ for different compressor speeds and outdoor temperatures	52
44	Energy balance over the whole cycle for different compressor speeds and outdoor temperatures	53
45	Evolution of the compressor consumption and the heat transfer rates at both the condenser and evaporator as a function of the compressor speed for three different outdoor temperatures	53
46	COP for different compressor speeds and outdoor temperatures	54
47	Isentropic efficiency for different compressor speeds and outdoor temperatures	54
48	Refrigerant mass flow rate for different compressor speeds and outdoor temperatures	55
49	Compressor consumption for different compressor speeds and outdoor temperatures	56
50	$\dot{Q}_{ev}$ and $\dot{Q}_{cd}$ for different compressor speed and outdoor temperature	57
51	Energy balance over the whole cycle for different compressor speeds and outdoor temperatures	57
52	Evolution of the compressor consumption and the heat transfer rates at both the condenser and evaporator as a function of the compressor speed for three different outdoor temperatures	58
53	Air flow measurement on the front face of the external exchanger (HTX1)	59
54	Air flow measurement at the HVAC output	60
55	Air flow through HTX1	61
56	Air flow through the HVAC	62
57	Flowchart	65
58	Compressor model	66
59	Three division zones of condensation	69
60	Two division zones of evaporation	70
61	Model of the flow streams and pressure drop network inside the accumulator	71
62	Reversible heat pump modelisation	73
63	Comparison between numerical and experimental results for compressor power	75
64	Comparison between numerical and experimental results for refrigerant mass flow rate	76
65	Comparison between numerical and experimental results for compressor exhaust temperature	76
66	Comparison between numerical and experimental results for condenser heat flow rate	77

---

67	Comparison between numerical and experimental results for evaporator heat	
	flow rate . . . . .	78
80	Admission side of the mobile unit . . . . .	96
81	Discharge side of the mobile unit . . . . .	97

## List of Tables

1	Analysis of HVAC inputs and outputs	31
2	Bench sensors characteristics (1)	35
3	Bench sensors characteristics (2)	35
4	Uncertainty propagation on the COP (heating mode)	43
5	Uncertainty propagation on the COP (heating mode)	44
6	Uncertainty propagation on the COP (heating mode)	44
7	Correspondence between duty and compressor speed	47
8	Measured air velocity	59
9	Measured air velocity	60
10	Comparison of the air flow rate results	62
11	Tuned compressor parameters	74
12	Tuned exchangers parameters	77

---

## Nomenclature

### Components:

- EB: Electrical box
  - EB0: Labothap supply
  - EB1: Power
  - EB2: Acquisition
- HVAC: Air conditioning unit of the car
- HTX: Heat exchanger
  - HTX1: Front exchanger
  - HTX2: Inner condenser (HVAC module)
  - HTX3: Inner evaporator (HVAC module)
- FAN1: Pulser in the HVAC module
- FAN2: Front fan

### Subscripts

- su: Supply
- ex: Exhaust
- out: Outdoor
- cp: Compressor
- ev: Evaporator
- cd: Condenser
- refr: Refrigerant
- num: Numerical
- exp: Experimental
- nom: Nominal
- sh: Superheated
- sc: Sub-cooled
- tp: Two-phase

### Acronyms

- COP: coefficient of performance
- OCR: Oil Circulation Rate
- PWM: Pulse With Modulation

## Part I

# Introduction

## 1 Context

Nowadays, air conditioning has become an indispensable element for the safety and comfort of motorists. For electric vehicles, however, this is quite detrimental to the battery's autonomy. This problem is one of the main disadvantages of electric vehicles. It is therefore necessary to optimize these systems so that they consume the minimum amount of electricity. In fact, in electric car both heating and cooling systems consume energy. The motors of electric vehicles hardly heat up at all, so the free heat coming from the motor cannot be recovered as it is the case for thermal engines. The solution developed by Renault and which will be studied in this master thesis is a reversible heat pump allowing the passenger compartment to be cooled and heated. This reversible heat pump offers much better performance than simple heating resistors.

A test bench on the reversible heat pump system of the Renault ZOE electric car has been developed in collaboration with Renault. This bench was entirely designed by students of the first master's degree in electromechanics during the 2021-2022 academic year as part of the "integrated project" course. The main objective of this bench is essentially educational. Indeed, this one will be used to give the laboratories of the course of "Thermodynamique appliqué et introduction aux machines thermiques" of professor V.Lemort [15].

## 2 Objectives

First of all, it is important to define the objectives of this master thesis project. The first objective was to numerically model each components of the Renault ZOE reversible heat pump and then integrate them all into the cycle. This numerical part is described in Part [IV](#). The second objective was to carry out test campaigns on the test bench and characterize its performance in cooling and heating mode. This experimental part is described in [III](#). The final objective was to validate the numerical model using experimental data collected during the different tests. This numerical validation is detailed in Part [V](#).

## 3 Literature review

Starting this master thesis with a literature review can be interesting in order to synthesize existing research on air-conditioning systems. This literature review will be used to identify relevant works, assess their quality, and identify potential problems in order to propose perspectives when it is possible.

### **Air conditioning system: reversible heat pump**

As said before, a whole course was dedicated on the development of this test bench. Several works are therefore available on different aspects of the bench. The project was divided into small group of students who were focused on a very specific work-packages (WP). A more global report produced as part of an internship on the entire test bench is also available. All of the work that has been carried out on this test bench is included in the list below:

- Integrated project course: Hardware part (WP1) [3]
- Integrated project course: Auxiliary part (WP1bis) [10]
- Integrated project course: Measurement Technique part (WP2) [11]
- Integrated project course: Numerical part (WP3) [8]
- Integrated project course: Educational part (WP4) [7]
- STING internship: "*Automotive air conditioning unit for electrical vehicle*" [5]

### **Development and evaluation of an automotive air-conditioning test rig [2]**

This paper describes over the designing process of a test rig that includes an automotive air-conditioning system. Tests have also been realized and are displayed in this document. The test rig is composed of several thermocouples and pressure sensors in order to measure respectively the temperature and pressure at different points in the cycle. Moreover, the COP, the cooling load, and the compressor power consumption have been calculated based upon the evaporator blower speed and the air velocity through the condenser. The conclusions were that increasing the blower speed reduces the COP and the power consumption. This paper was interesting in order to understand the methodology of designing a test bench. In addition, it contains also a lot of interesting data giving an idea of the order of magnitude of most values.

### **Performance evaluation of an integrated automotive air-conditioning and heat pump system [6]**

This article interests in the performance of a reversible automotive air-conditioning system using R134a as refrigerant. Compressor speed and exchanger inlet air temperatures were modified for the tests in heating and cooling mode. COP, capacity, compressor exhaust temperature and rate of exergy destroyed have been calculated for both modes. It has been demonstrated in the results that the heating mode is only efficient in mild weather and its capacity decreases with the outdoor temperature. However, its COP is higher and its rate of exergy destruction per unit capacity is lower than in cooling mode. Solutions to improve heating mode have been presented. For example, the redesign of the indoor exchanger,

or the use of a different refrigerant. This study is very interesting because it deals with similar topics. It is the same reversible cycle than for the Renault ZOE but with a another refrigerant.



## Part II

# Bench description

## 1 Overview

The test bench is situated in the thermodynamics laboratory of the university of Liege. This test bench was entirely designed by students and built with the help of the lab staff. As previously said, this bench will be used to give laboratory sessions.

### 1.1 Bench layout

The bench and its main components are shown in the figures below.

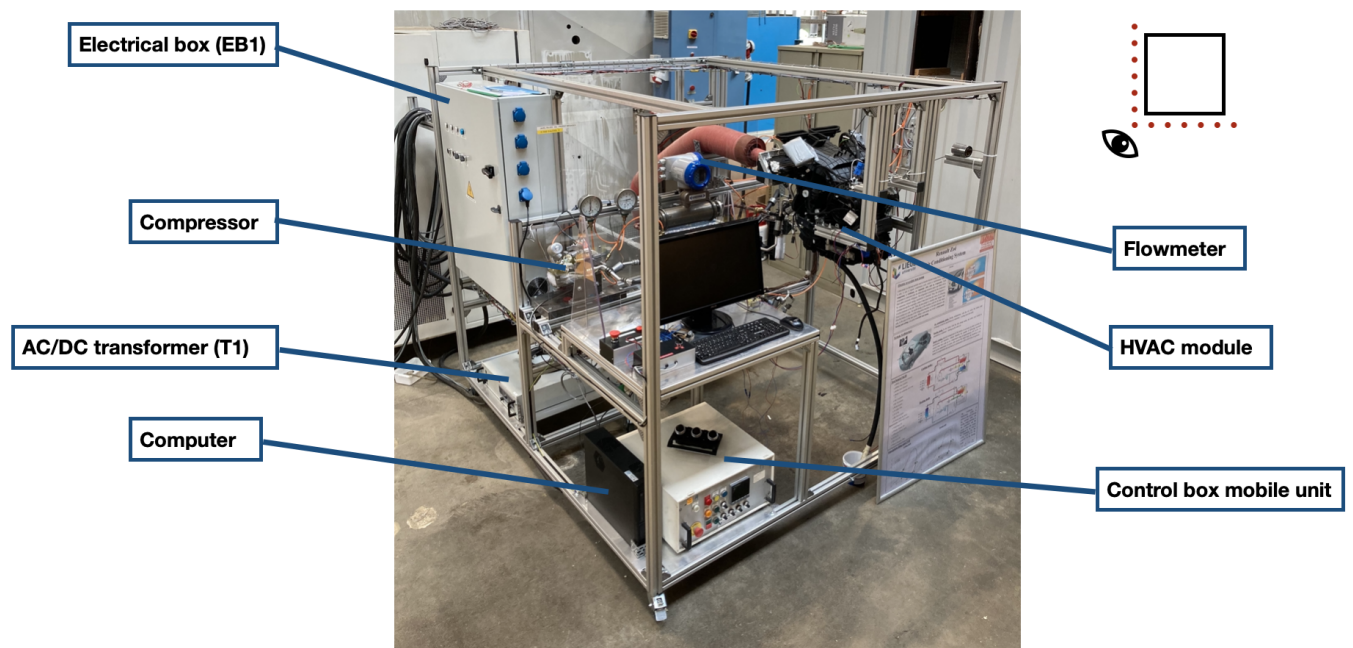


Figure 1: Bench view (1)

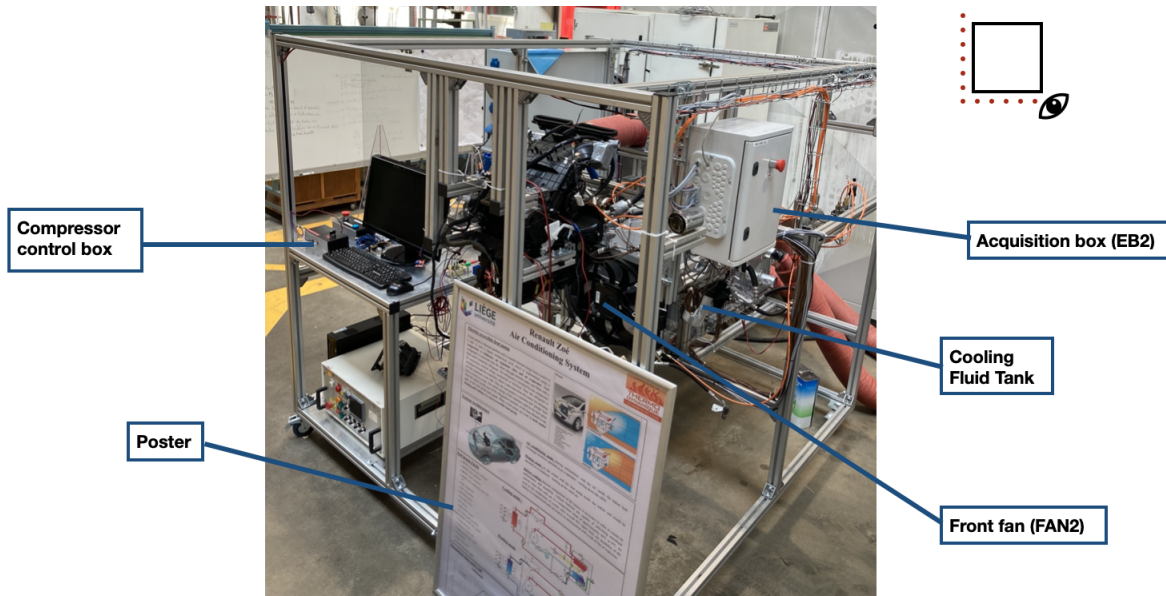


Figure 2: Bench view (2)

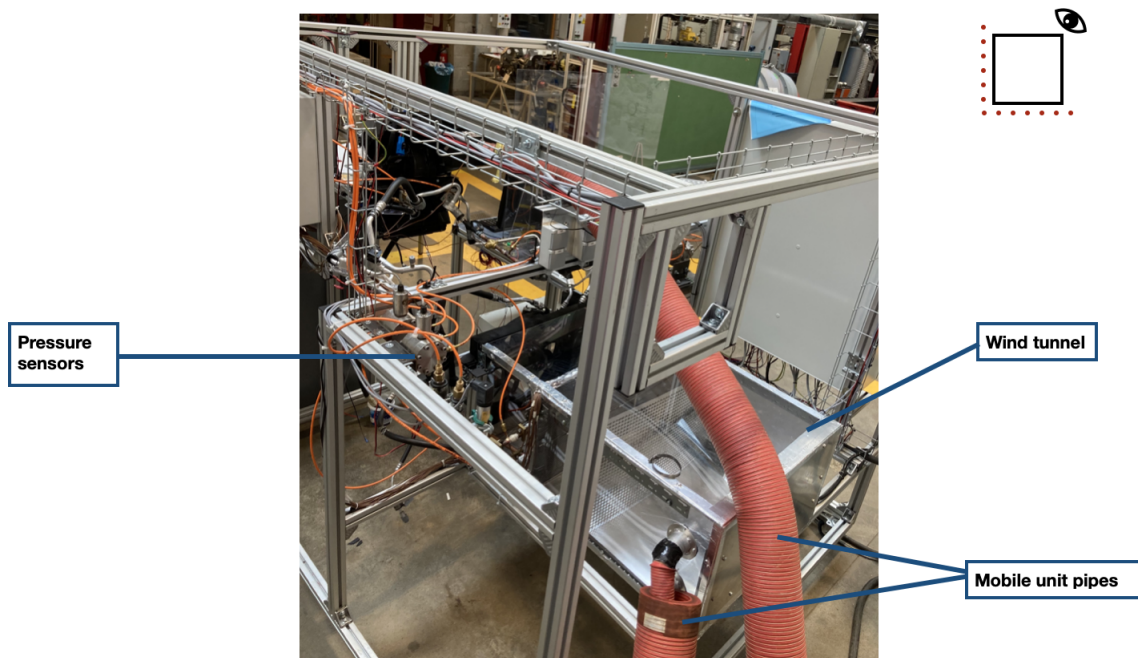


Figure 3: Bench view (3)



Figure 4: Bench view (4)

## 1.2 P&ID

Fig 5 represents the complete P&ID of the bench. All sensors are represented and described in the legend below. The production of cold is ensured by the the evaporation of the refrigerant at low pressure through the inner evaporator HTX3. Heat production is ensured by the refrigerant compression introducing a phase change from gaseous to liquid state through the internal condenser HTX2. A solenoid valve system is implemented in order to switch from one mode to another. In fact, it is physically impossible to simply reverse the direction of flow of the refrigerant for the following reasons:

- Compressor cannot turn in both directions;
- Expansion valve (calibrated orifice) cannot be adapted in both directions;
- Different dimensioning of the internal exchanger.

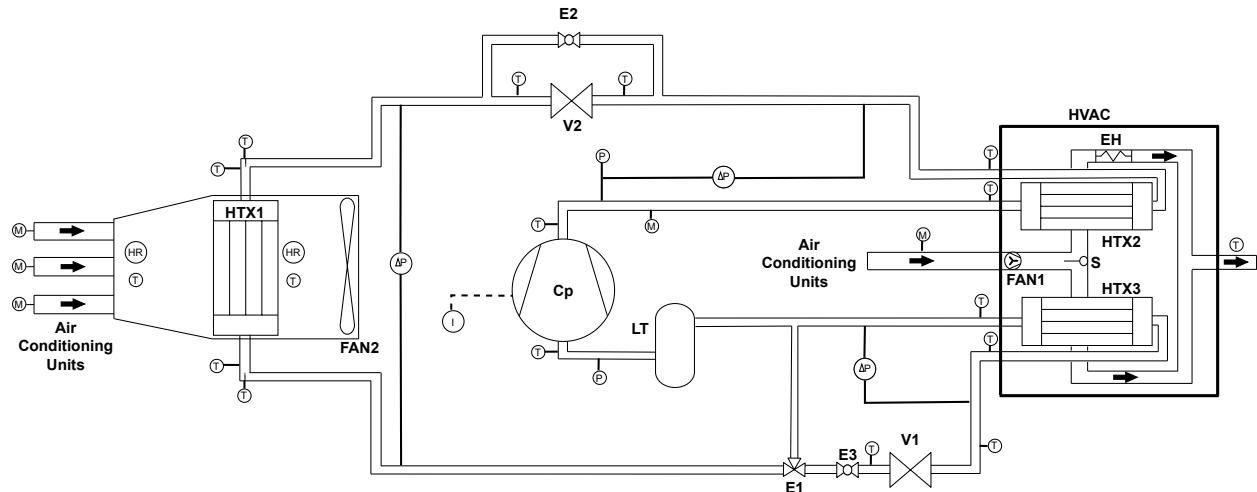


Figure 5: P&amp;ID of the bench

- **HTX1:** External heat exchanger
- **HTX2:** Inner condenser
- **HTX3:** Inner evaporator
- **HVAC:** Air conditioning unit of the car
- **Cp:** Compressor
- **LT:** Accumulator
- **E:** Electrovalves
- **V:** Orifice tubes
- **F:** Air fan
- **EH:** Electric heater
- **S:** Switch inside the HVAC

**Measurements:** (noted with a circle)

- Temperature (T)
- Absolute Pressure (P)
- Differential Pressure ( $\Delta P$ )
- Current (I)
- Flow meter (M)
- Relative Humidity (HR)

**Air flows** are represented by  $\longrightarrow$

Fig.6 shows The refrigerant path in cooling mode. The red color corresponds to the hot refrigerant which is then cooled down through the condenser in order to heat up the air. The color thus becomes orange. After that, the refrigerant is expanded through the orifice tube V1 and its temperature therefore drops drastically (dark blue color). Finally, the refrigerant is heated up through the inner exchanger (HTX3) in order to cool down the air. This warming is illustrated by a light blue color. Air cooling is schematically represented by the passage from the yellow state to green. The air arrives on the condenser and on the evaporator through the mobile unit pipe (see Fig.3).

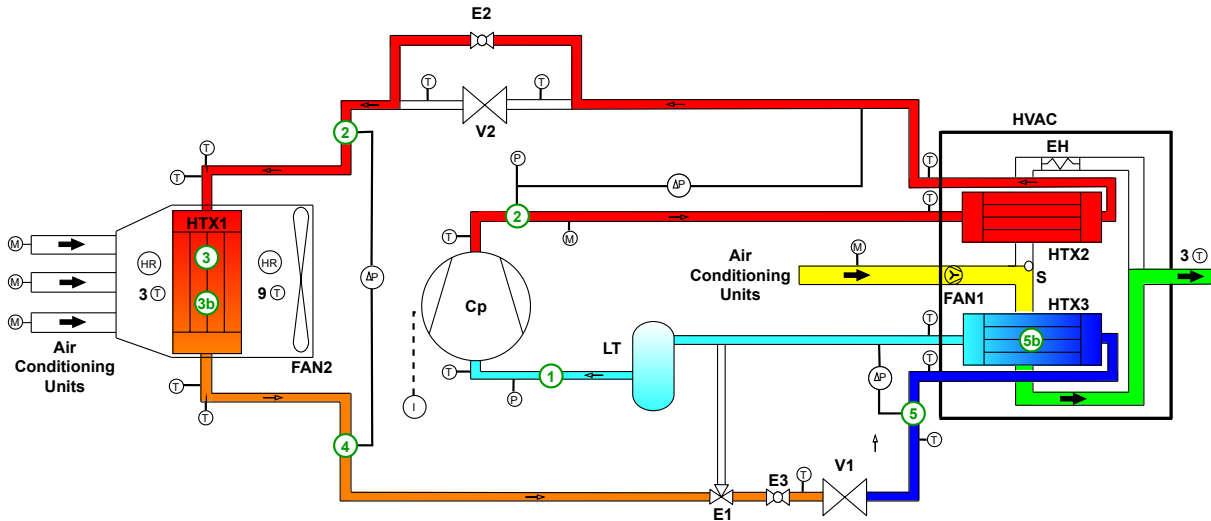


Figure 6: P&ID: cooling mode

In heating mode the refrigerant travels through the cycle as described in Fig.7. In this case the air is heated up through the inner condenser HTX2 (green state to yellow).

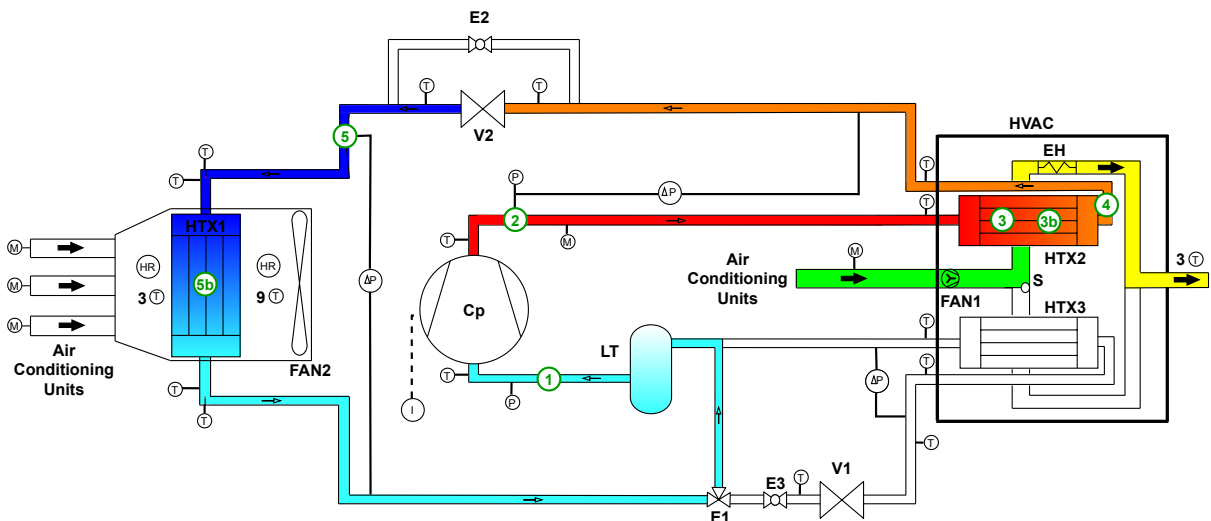


Figure 7: P&ID: heating mode

The states on the cooling and heating P&ID (green circles) can be represented on a Pressure-enthalpy and on a Temperature-entropic diagram, Fig. 8a & 8b respectively. In both modes the refrigerant passes through the same states:

- **State 1:** low pressure, slightly overheated vapour;
- **State 2:** high pressure and temperature, vapour state;
- **State 3:** vapor saturation where quality  $X=1$  in the condenser;
- **State 3b:** liquid saturation where quality  $X=0$  in the condenser;
- **State 4:** high pressure, liquid slightly sub-cooled (liquid state in the expansion valve);
- **State 5:** low pressure and temperature, liquid state;
- **State 5b:** vapor saturation where quality  $X=1$  in the evaporator;

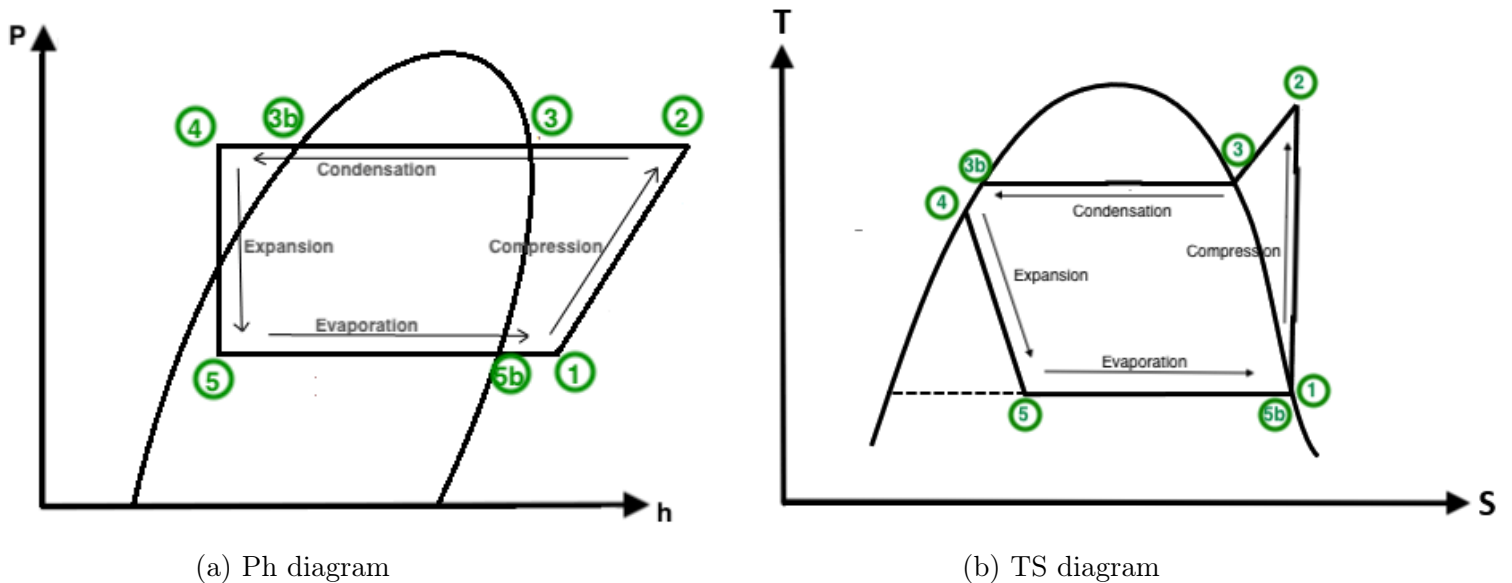


Figure 8: Representation of the different refrigerant states

### 1.3 Refrigerant

R1234yf refrigerant is the one currently used in the Renault ZOE. This one replaced the R134a used in the past because of restriction about the global warming potential. Indeed, R1234yf has a low global warming potential ( $GPW \approx 4$ ) while the one of the refrigerant R134a is higher than 1000 ( $GPW \approx 1300$ ) which is not allowed anymore in the air conditioning system.

## 2 Components description

### 2.1 Compressor

The 288V electric compressor used for this test bench is not the one actually used in the Renault ZOE because datasheet was missing to understand its operation. Another scroll compressor (Fig.9) with a higher displacement was thus used ( $27\text{ cm}^3$  vs  $34\text{ cm}^3$ ).

#### 2.1.1 Functioning

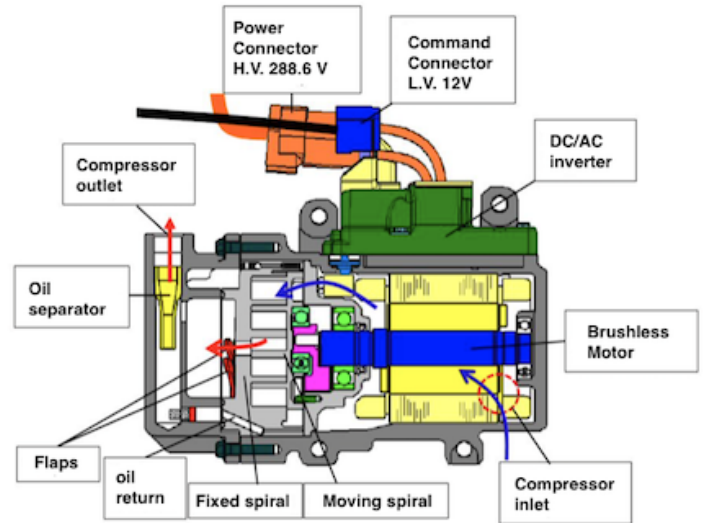
The Denso ES34C requires two power sources: 288 V to supply the motor (instead of 400 V for the compressor supplied by Renault) and 12 V to supply the internal electronics. The power consumed by this compressor can be calculated thanks to the following formula:

$$\dot{W}_{CP} = V \times I \quad (1)$$

With  $V = 288\text{V}$  and  $I$  is measured by the current sensor or is directly read on the screen of the AC/DC transformer (T1).



(a) Test bench compressor



(b) Cross sectional schematic [12]

Figure 9: DENSO electric scroll compressor ES34C

The compressor speed can be controlled with a PWM signal whose duty cycle represents the proportion of high voltage time to the period time. The speed of the compressor can then be modulated by changing this duty cycle. A duty cycle of 100% corresponds to the maximum speed of the compressor, while a duty cycle of 0 does not cause any rotation. The internal

electronics of the compressor has two inputs (PWM IN and STB) and two outputs (PWM OUT and DIAG).

- PWM IN is the signal which determines the rotational speed of the device. The correspondence between duty cycle and compressor speed is illustrated in Fig 10.

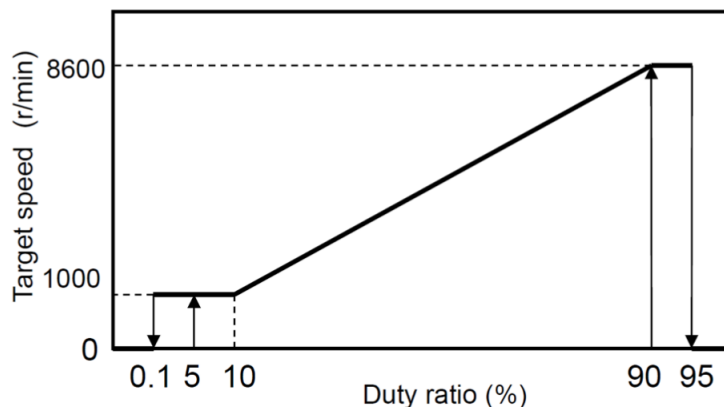


Figure 10: Compressor speed according to duty cycle 12

- STB (Stand by) has to be connected to the ground in order to start the compressor.
- PWM OUT is the signal sent by the internal electronics of the compressor whose duty cycle indicates the power consumed by the compressor. The correspondence between duty cycle and compressor power is illustrated on Fig 11.

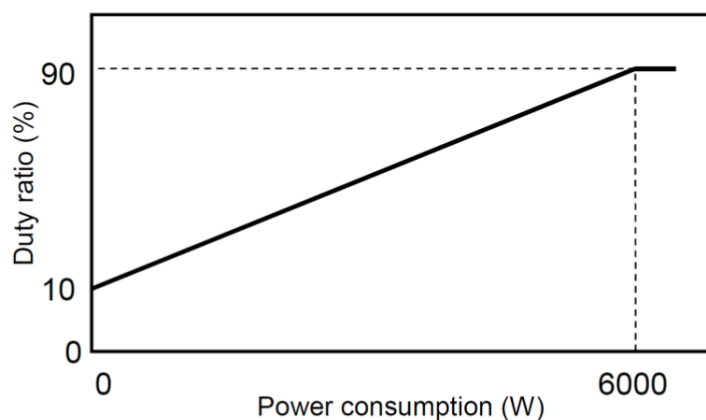


Figure 11: Correspondence between duty cycle of  $PWM_{out}$  and the compressor power 5

- DIAG: PWM signal sent by the compressor in order to identify potential problems.

Fig 12 represents the wiring of the compressor. First of all, the NI9401 module allows to generate a 5V PWM signal with a duty cycle selected by the bench user on the Labview



interface. However, as said before, the supply of the compressor internal electronics is equal to 12 V. An electrical circuit is thus required to transmit the duty cycle of the 5V PWM signal to a 12V signal in order to protect the power electronics integrated into the compressor. This is the role of the MOFSET situated in the control box.

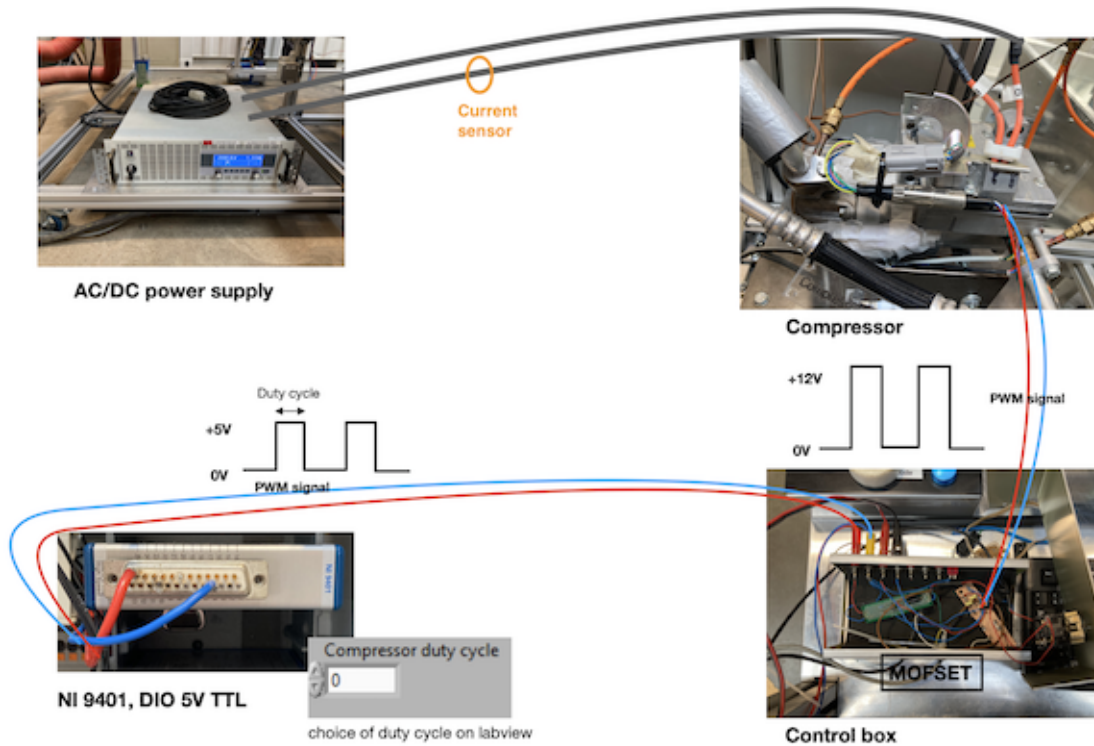


Figure 12: Power supply and control of the compressor

## 2.2 Control box

The control box is shown in Fig. 12, bottom right. The purpose of this box is to transform the signals it receives, via breadboards into usable signals that can be read by both the internal electronics of the compressor and the acquisition card. Fig. 13 represents the electrical circuit of the PWM IN signal which allows to switch from an amplitude of 5V of the acquisition system to one of 12V of the internal electronics of the compressor. This circuit is situated in the control box (green breadboard) and has been realized by Pascal Harmeling, member of the Electronics laboratory.

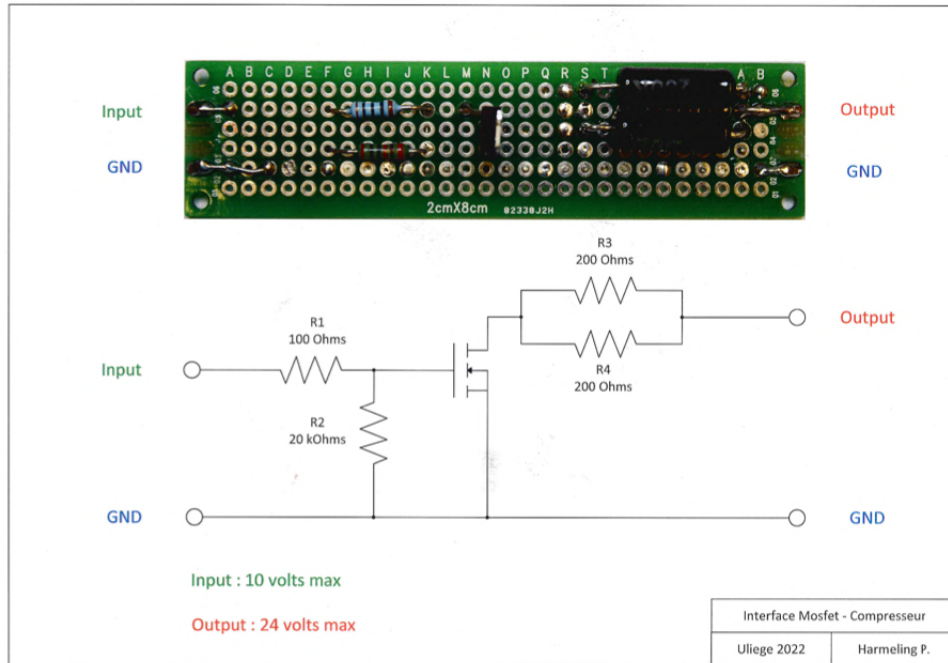


Figure 13: Wiring diagram of the PWM IN input

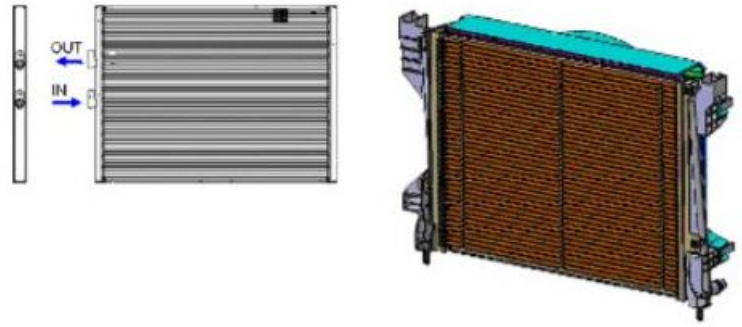
The same breadboard but for the PWM OUT signal have been realized (brown one in the control box) in order to get the compressor power. However, due to the noise of the signal, Labview software was not able to handle it even with filters. Nevertheless, this PWM signal could be displayed on the oscilloscope and the duty cycle of it has been visually determined. The powers calculated with the PWM OUT duty cycle (see Fig. 11) can therefore be compared with the powers calculated with the current sensor.

### 2.3 External exchanger (HTX1)

In cooling mode, HTX1 acts as a condenser and allows the condensation of the refrigerant while in heating mode, HTX1 acts as an evaporator and allows the evaporation of the refrigerant. The exchanger is a fanned and tubular cross-flow exchanger located in the front of the vehicle. Renault did not provide much information about the architecture of this exchanger.



(a) External exchanger of the Renault ZOE (HTX1)



(b) Schematic diagram of the external exchanger [13]

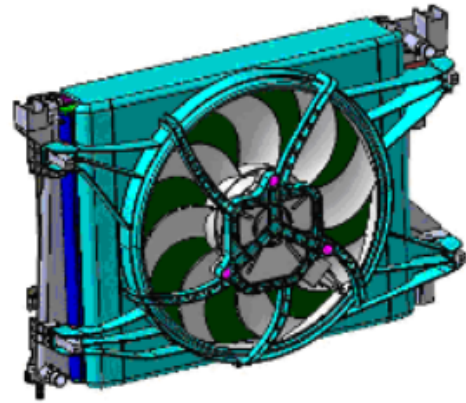
Figure 14: Front exchanger (HTX1)

## 2.4 Front fan (FAN2)

The fan is placed behind HTX1 and is used to force the air to flow through it. It needs a 12V supply and a PWM signal to operate. Once again Renault did not provide any information about this fan.



(a) Front fan of the Renault ZOE (FAN2)



(b) Schematic diagram of the front fan [13]

Figure 15: Front fan (FAN2)

## 2.5 Orifice tubes

As it can be seen on the P&ID in Fig. 5, there are two orifice tubes in the cycle. The first one (V1) expands the refrigerant before entering the internal evaporator (HTX3) in cooling mode. The second (V2) is placed before the external exchanger (HTX1) and acts as an

evaporator in heating mode. In cooling mode (resp. heating mode), V2 (resp. V1) is thus bypassed.

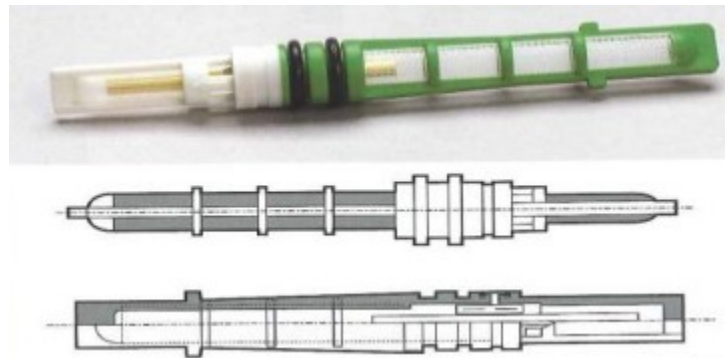


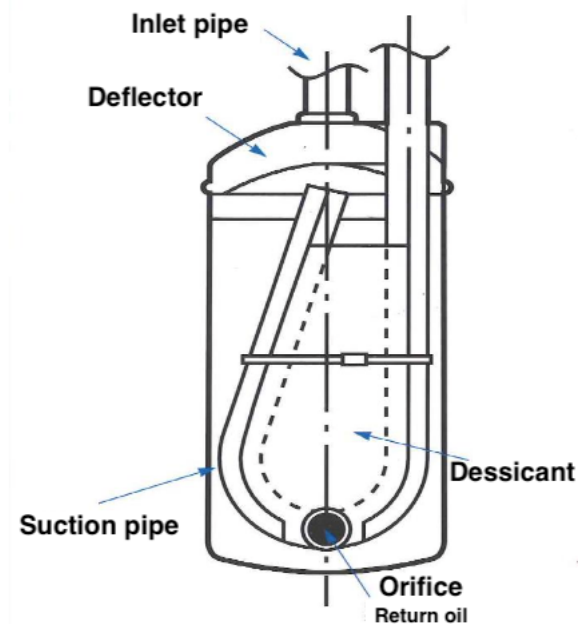
Figure 16: Calibrated expansion valve [13]

## 2.6 Accumulator

The accumulator is placed before the compressor in the cycle and performs various functions. First of all, it allows to store the amount of unused fluid depending on the operating point. The accumulator filters impurities and absorbs moisture in the loop to protect the compressor. Finally, it provides an oil return to the compressor.



(a) Accumulator of the Renault ZOE



(b) Schematic diagram of the accumulator [13]

Figure 17: Accumulator

## 2.7 Electrovalves

An electrovalve is a valve that is electrically controllable by a 12V supply. These last are used to switch from one mode to another thanks to the switch S2 located in the EB1 box. In total there are two two-way solenoid valves and one three-way.

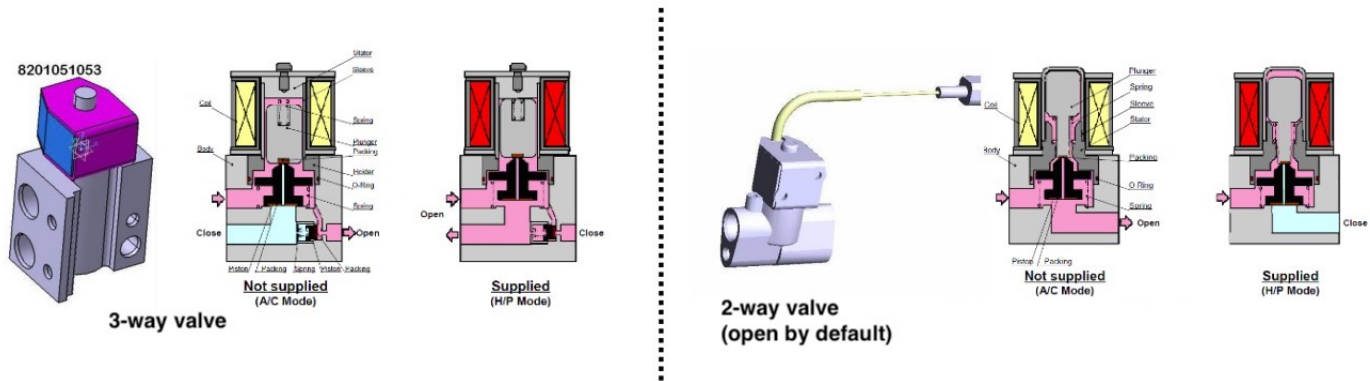


Figure 18: Electrovalve 13

## 2.8 HVAC module

The HVAC ("Heating-Ventilation and Air Conditioning") module contains the following components:

- The pulser (FAN1);
- The exchangers: internal evaporator (HTX3) and internal condenser (HTX2);
- The additional heating (PTC);
- The servomotors and flaps (distribution, recycling & mixing).

Fig.19 shows the organization of the HVAC components.

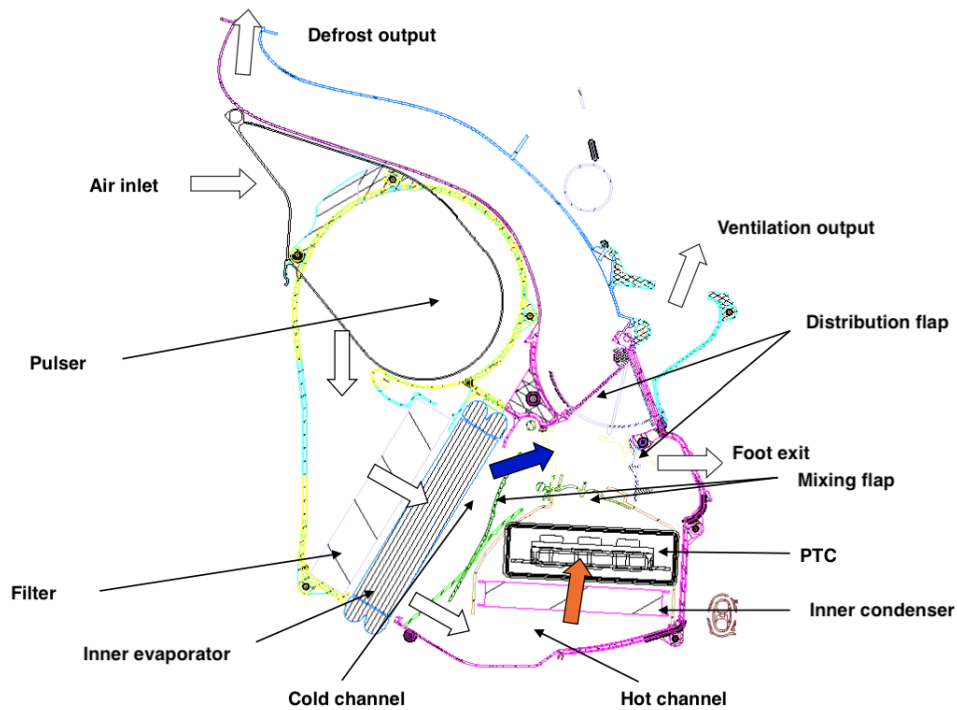


Figure 19: HVAC module diagram [13]

### 2.8.1 Pulser (FAN1)

This fan needs a 12V supply to operate and its speed can be modulated by changing the duty cycle of a PWM signal. It is used for regulating airflow in the passenger compartment.



Figure 20: Pulser (FAN1) [13]

### 2.8.2 Inner condenser (HTX2)

In heating mode, the refrigerant passes through the internal condenser to heat up the air.

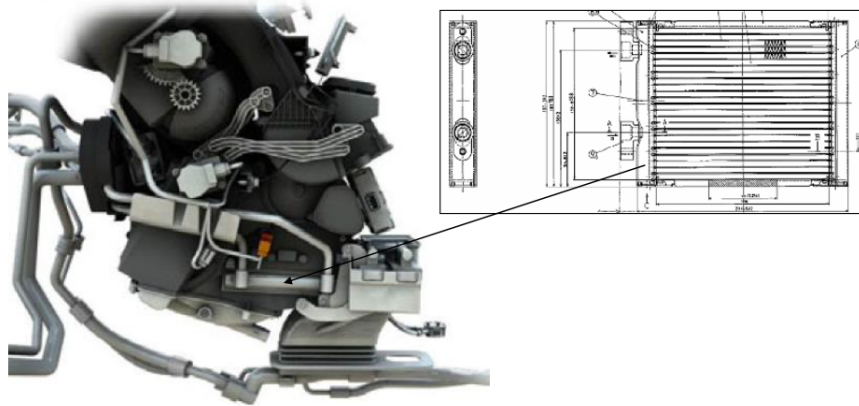


Figure 21: Inner condenser (HTX2) [13]

### 2.8.3 Inner evaporator (HTX3)

In cooling mode, the air passing through this exchanger is cooled down in order to refresh the passenger compartment. In heating mode, this exchanger is bypassed.



Figure 22: Inner evaporator (HTX3) [13]

### 2.8.4 External heating (PTC)

The PTCs ("Positive Temperature Coefficient") are additional heating resistors that can be used if the power developed at the condenser is not sufficient. There are situated downstream the internal condenser and can deliver 1800 W of power.



Figure 23: PTC 12V [13]

### 2.8.5 Servomotors and flaps

The visible inlets and outlets of the HVAC module are shown, numbered and described in Tab.1 and Fig.24.

Table 1: Analysis of HVAC inputs and outputs

Number	Corresponding function
1	Air inlet (no-recycling)
2	Defrost output
3	Defrost output
4	Defrost output
5	Front side exit
6	Front side exit
7	Foot exit
8	Foot exit
9	Front center exit
10	Air inlet (recycling)

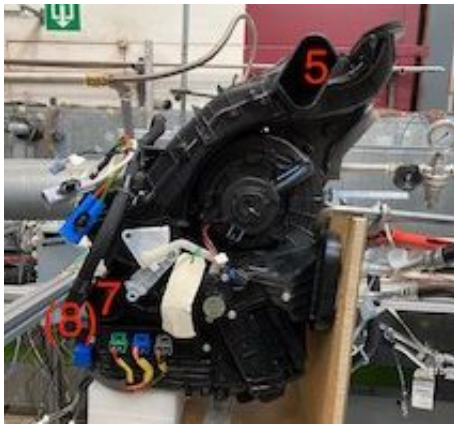




(a) Front view



(b) Top view



(c) Side view - left



(d) Side view - right

Figure 24: HVAC module

By disconnecting the servomotors, the flaps positions could be changed manually in order to better understand the airflow in the HVAC module. There are 4 flaps for the whole HVAC module controlled by 3 servomotors:

- **Servomotor B** controls the recycling flap as represented in Fig. 25a. The air is either taken from outside the vehicle or from inside (recirculation). It is not 0 or 100% recirculation but a gradual opening.
- **Servomotor C** controls the mixing flap and allows to switch from cooling to heating mode as shown in Fig. 25b. It is not 0 or 100% but a gradual opening.
- **Servomotor D** controls 2 flaps that allow to select the different air outlets as illustrated in Fig. 25c.

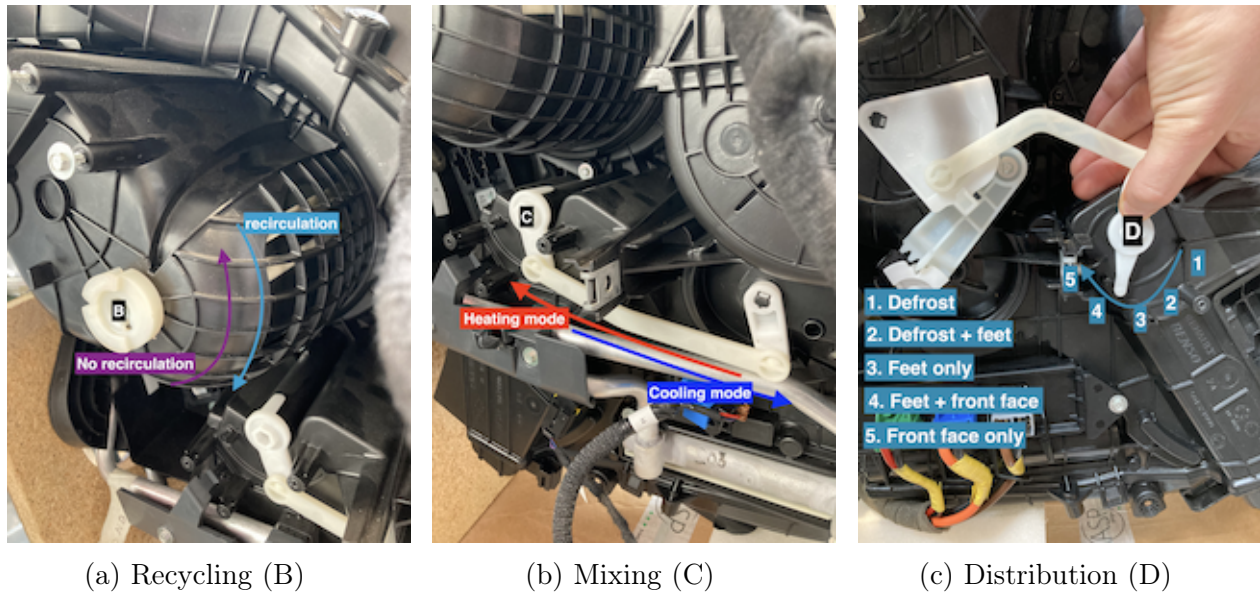


Figure 25: Roles of the 3 servomotors

In normal conditions, the HVAC module of the ZOE draws air through 1 and 10 (Fig. 24) and blows through 2 to 9. In this case, the following decisions have been taken:

- HVAC module of the ZOE always in no-recycling mode + position 5 in Fig. 25c;
- Mixing flap controlled manually;
- Exits 5 and 6 blocked (Fig. 24).

The HVAC module therefore becomes 1 input, air arrives at inlet "1" (see Fig. 24a) and 1 output, air is pulsed through exit "9" (see Fig. 24b). The choice of air conditioning or heating mode is made manually with the mixing flap.

## 2.9 Auxiliaries

### 2.9.1 Portable Air Conditioning Unit

In order to carry out experimental tests and thus simulate real driving conditions, a portable air conditioning unit available in the Labothap has been chosen (see Fig. 26). This unit can deliver a varying air flow rate at desired range temperature:  $[0; 600] \text{ m}^3/\text{h}$  and  $[0; 60] \text{ }^\circ\text{C}$ . Two outputs are available: one is directly connected to the HVAC module and the second to the wind tunnel.



Figure 26: Portable Air Conditioning Unit

### 2.9.2 Wind tunnel

The wind tunnel allows to simulate the air flow in the front exchanger of a car. This one is represented in Fig. 27. The external heat exchanger HTX1 and the front fan FAN2 are placed inside this tunnel. In order to homogenise the flow, a wire mesh is installed upstream the exchanger as it can be seen in Fig. 27. A hole has been made in the wind tunnel to evacuate any condensate from the evaporator (heating mode).

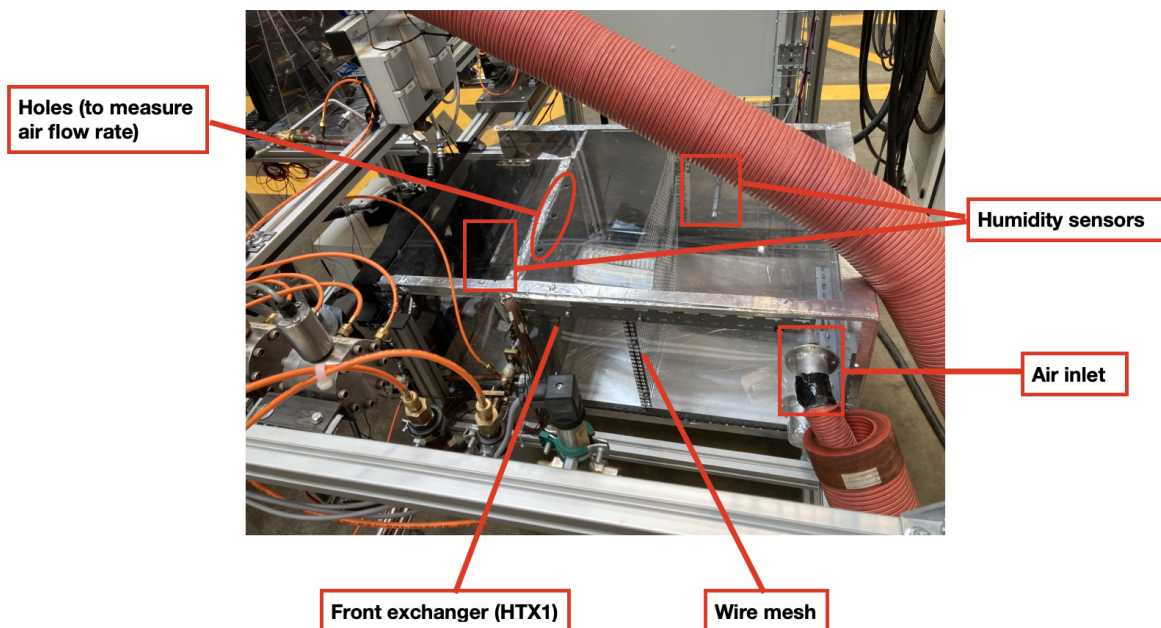


Figure 27: Wind tunnel

## 3 Measuring device

### 3.1 Sensors

All the sensors used on this test bench are listed in Tab. 2 & 3

Table 2: Bench sensors characteristics (1)

Sensor	Name	Input	Output	Range	Error
Type-T thermocouples	Th	/	4-20 mA	[-270;370] °C	1 K
Absolute pressure	PABS30	8-32 Vdc	4-20 mA	[0;30] bar	0.02 bar
	PABS10	8-32 Vdc	4-20 mA	0-10 bar	0.05 bar
Differential pressure	PDIFF1	8-36 Vdc	4-20 mA	[0;1] bar	2.5 mbar
	PDIFF05	15-40 Vdc	4-20 mA	[0;0.5] bar	5 mbar
	PDIFF500	4.75-5.25 Vdc	0.25-4 Vdc	[-20;500] Pa	1 % m.v.
Relative humidity	HUMREL	13.5-35 Vdc	0-10 V	[0;100] %	2% r.h
Mass flow	CORFM	24 Vdc	4-20 mA	[0;1230] kg/h	0.7 %
DC Current	Curr_CP	12-32 Vdc	4-20 mA	[0;25] A	0.175 A

Table 3: Bench sensors characteristics (2)

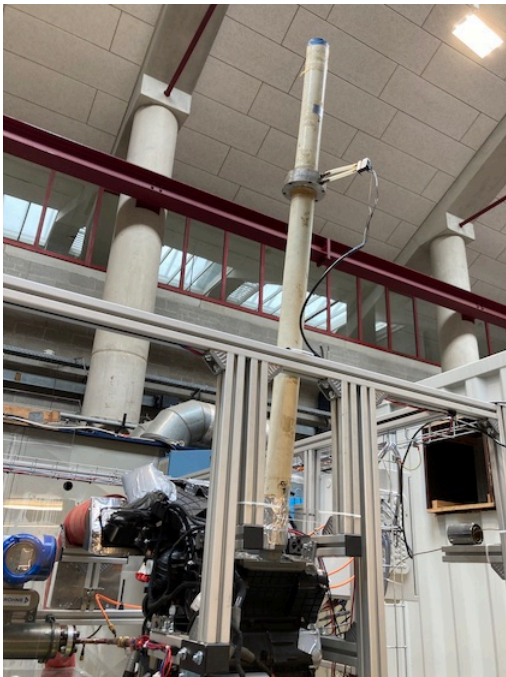
Name	Number	Use	Datasheet
Th	30	Air/ refrigerant / surface temperature	<a href="#">link</a>
PABS30	1	Outlet compressor pressure	<a href="#">link</a>
PABS10	1	Inlet compressor pressure	<a href="#">link</a>
PDIFF1	2	Internal exchangers (HTX2/HTX3) pressure drop	<a href="#">link</a>
PDIFF05	1	External exchangers (HTX1) pressure drop	<a href="#">link</a>
PDIFF500	1	Orifice plate pressure drop	<a href="#">link</a>
HUMREL	2	Relative humidity both sides of external exchanger	<a href="#">link</a>
CORFM	1	Refrigerant mass flow rate	<a href="#">link</a>
Curr_CP	1	Compressor current	<a href="#">link</a>

All the sensors are associated with an absolute or relative error as it is shown in Tab. 2. Indeed, there is always an uncertainty on the measurement even if the sensors are properly installed and calibrated. The uncertainties can come from either the sensor itself, or from the way it is used, or from the environment. According to NIST ("National Institute of Standards and Technology"), a measurement is considered complete only if the measure is attached with an uncertainty on the measurement. It is essential to determine the origin of the main sources of errors and quantify the precision of the results. To do that an uncertainty propagation can be done with the NIST method. This method consists of determining the uncertainty of the quantity  $U_y$  as a function of the measured variables uncertainties via  $U_{x,i}$ .

$$U_y = \sqrt{\sum_{i=1}^n \left(\frac{\partial y}{\partial x_i}\right)^2 \cdot U_{x,i}^2} \quad (2)$$

### 3.2 Orifice plate

Fig.28 represents the orifice plate installation in order to measure the air flow rate leaving the HVAC module. It consists of a long pipe, an orifice plate, and a differential pressure sensor. The orifice consists of a flat, circular plate with a precisely calibrated central hole. When air passes through this restriction, a pressure difference is created between the upstream and downstream sides of the plate. This pressure difference is proportional to the fluid flow rate, according to Bernoulli's law, which relates fluid flow velocity to pressure. The differential pressure sensor used (see Fig.28b) has a range of [-20;500] Pa and an analogic output of 0.25 - 4V. This sensor measures the pressure drop created by the restriction and returns a corresponding voltage to the acquisition system.



(a) Orifice plate



(b) Differential pressure sensor PDIFF500 (see Tab.2)

Figure 28: Output air flow measurement

### 3.3 Anemometer

An anemometer (Fig.29) is a measuring device capable of measuring air speed. This last will be used to measure the speed of the air arriving at the front exchanger HTX1 and at the HVAC module outlet. For temperatures between 0 and 50 °C, this device can measured

air velocities between 0 and 10 m/s.



Figure 29: Anemometer used for the air flow rate measurement ([datasheet](#))

## 4 Performance

### 4.1 COP

The coefficient of performance is the ratio between the amount of energy produced and the total amount of electrical energy absorbed by the system, which is the work consumed by the compressor in this study. In cooling mode, the useful energy is produced by the evaporator. The COP is expressed as follow:

$$COP_{\text{cooling}} = \frac{\dot{Q}_{ev}}{\dot{W}_{cp}} \quad (3)$$

On the other hand, in heating mode, it is the amount of heat absorbed by the condenser which represents the useful energy:

$$COP_{\text{heating}} = \frac{\dot{Q}_{cd}}{\dot{W}_{cp}} \quad (4)$$

### 4.2 Isentropic efficiency

Isentropic efficiency characterizes compressor's ability to compress a gas adiabatically and reversibly, i.e. without energy loss due to friction or heat dissipation. This efficiency is defined as the ratio of isentropic work to real work. Isentropic compression represents the case of a perfect machine that does not exchange heat with the external environment and consumes thus the minimum of energy. However, the system is always faced with losses, it is therefore a theoretical case impossible to reach in reality. The isentropic efficiency can be expressed as follow:

$$\eta_{is} = \frac{\dot{W}_{cp,is}}{\dot{W}_{cp}} = \frac{h_{cp,ex,is} - h_{cp,su}}{h_{cp,ex} - h_{cp,su}} \quad (5)$$

## 5 Bench control

Several parameters can be controlled on the bench in order to simulate different scenarios. Four parameters can be changed:

- Compressor speed via Labview software;
- Outdoor temperature via the mobile unit control box;
- Air flow through the HVAC module via the control boxes of the mobile units and the FAN1;
- Air flow through HTX1 via the control boxes of the mobile units and the FAN2.

### 5.1 Compressor speed control

The compressor rotational speed can be entered manually in the Labview's interface. Fig.10 represents the correspondence between the rotation speed of the compressor and the duty cycle of the signal chosen on Labview. The following relations are obtained:

- For Duty cycle  $\in [0; 10\%]$  ,  $N = 1000$  [RPM];
- For Duty cycle  $\in [10; 90\%]$  ,  $N = 95 x + 50$  [RPM] with x the duty cycle [%];
- For Duty cycle  $\in [90; 95\%]$  ,  $N = 8600$  [RPM].

### 5.2 Mobile units control

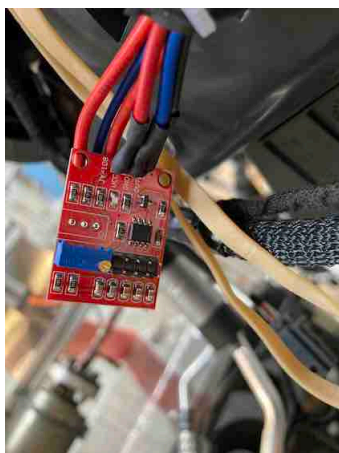


Figure 30: Control box of the mobile unit

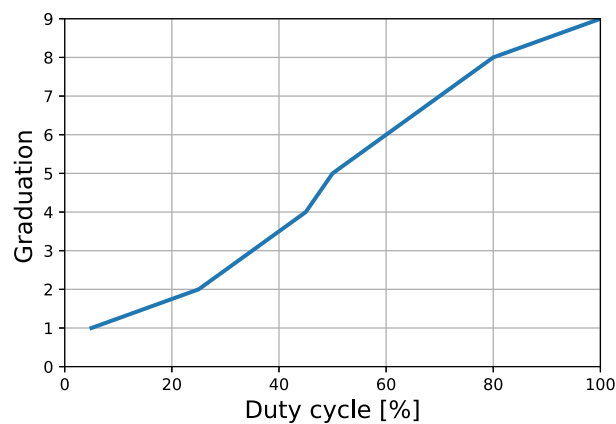
The turbine speed of the portable unit can be controlled thanks to a potentiometer graduated from 1 to 10. It is also possible to choose the proportion of flows in each of the two outputs. The set point temperature who represents the outdoor temperature in driving condition can be manually chosen and is displayed on the screen.

### 5.3 FAN1 control

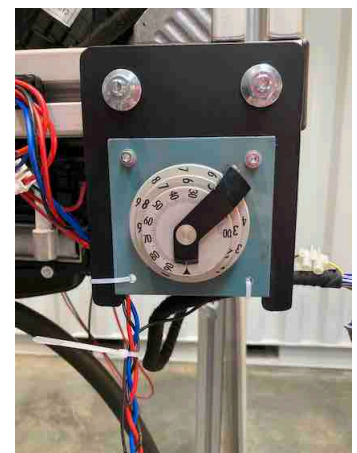
This fan needs a 12V supply and its speed can be modulated by changing the duty cycle of its PWM signal. A square wave signal generator with variable frequency and duty cycle has been used (Subfig. 31a). The only potentiometer visible is the one allowing to change the frequency. This one will normally not be used because the pulser starts with a fixed frequency of 100Hz. The second potentiometer has been removed because its adjustment was too fine for this application. Subfig. 31c represents the new potentiometer allowing to change the duty cycle and thus the rotation speed of the pusler inside the HVAC.



(a) FAN1 control module



(b) Correspondence between graduation and duty cycle



(c) FAN1 potentiometer

Figure 31: FAN1 control

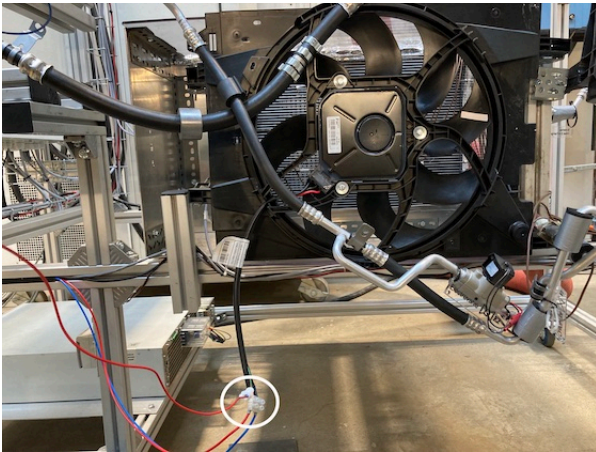
The correspondence between the graduation of the largest knob of the potentiometer and the duty cycle is represented in Fig. 31b. The other two knobs are used to make a finer adjustment of the duty cycle. Since the pulser is stuck inside the HVAC module, its rotational speed cannot be measured and the correspondence between the duty cycle and the rotational speed cannot be established. As duty cycle increases, fan speed decreases.

### 5.4 FAN2 control

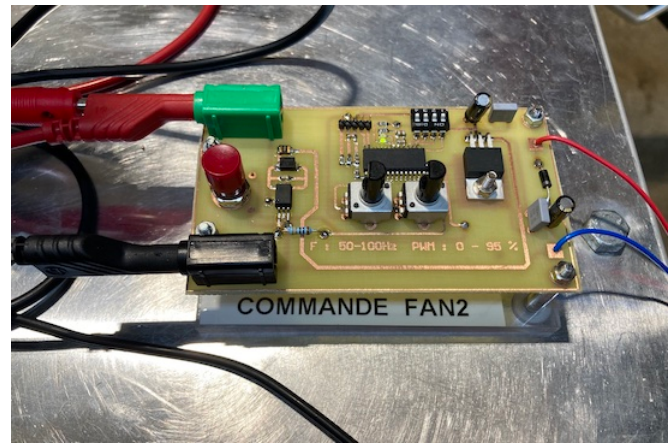
The front fan (Subfig. 32a) needs a 12V supply and a PWM signal to operate. In this figure, the red and blue cables are the power supply. The single red cable provides the PWM signal at 100Hz. The fan speed can be modulated by changing the duty cycle of the PWM signal.



This can be done thanks to an electronic cards represented in Subfig.32b. This set-up has been realized by Pascal Harmeling member of the Electronics laboratory. This card needs a 12V power supply (red and blue cables) and provides a 100Hz PWM signal (red cable connected to the fan). The left-hand potentiometer is used to change the frequency in a narrow range around 100 Hz. The right one is used to change the duty cycle between 0 and 95%. Finally, the red button serves as an emergency stop button and switches off the fan.



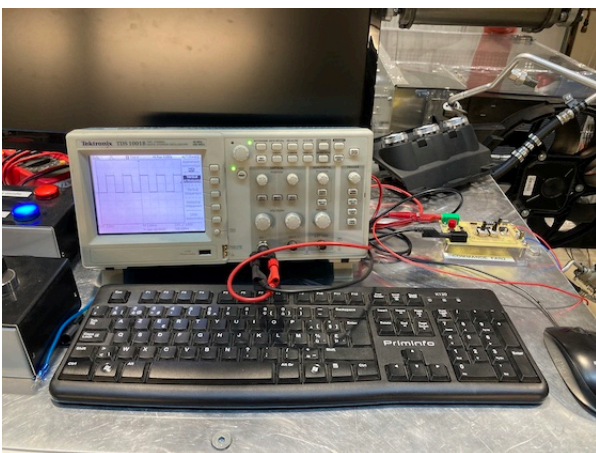
(a) FAN2



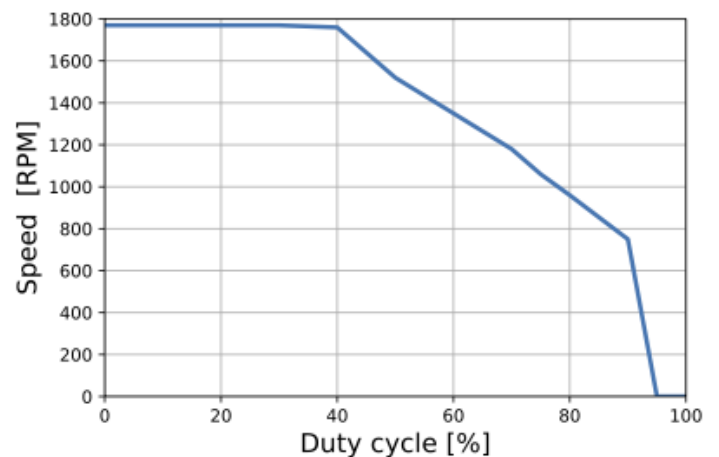
(b) FAN2 control

Figure 32: Front fan (FAN2)

An oscilloscope can be used to get a visual representation of the PWM signal. The connection can be made as shown in Subfig.33a. The correspondence between duty cycle and fan speed is shown in the Subfig.33b.



(a) PWM display



(b) Fan speed according to duty cycle

Figure 33: Duty cycle analysis

## 6 Bench improvements

The following improvements have been realized on the test bench:

- Connection of a 12V power supply (T3) in EB1;
- Installation of protective grids in EB1;
- Installation of a mobile unit power supply socket on the EB1 box;
- Installation of the control modules for FAN1 & 2;
- Grounding of all metal parts (compressor, aluminium plates, cable trays, EB1 door, bosh profile structure);
- Addition of 3 thermocouples (one for ambient temperature measurement and two for compressor surface temperature measurement);
- Addition of an orifice plate and connection of a differential pressure sensor;
- Connection of a 5V power supply (T4) in EB2;
- Update of the electric diagrams (Annex.VI);
- Update of the user guide of the bench [9];
- Labview code;
- Modelisation of heat pump (Python code Annex.VI).

Moreover, after an accident that occurred on Wednesday, April 5 2023 it has been decided to improve the security of the bench.

- When the bench is operational, users have to wear safety glasses;
- A plexiglass plate has been placed between the compressor and the screen and fixed to the bosh structure;
- The flexible pipe at the compressor outlet has been tied with iron wire;
- Limitation of the rotation speed of the compressor so as not to exceed a pressure of 25 bar at its outlet.

## Part III

# Experimental results

## 1 Description of the tests

As explained in Sec. 5, four main variables can be controlled in order to simulate various scenarios. When the bench is started, the compressor is at ambient temperature and the following equation is applicable.

$$\dot{Q} + \dot{W} + \sum_{\text{in}} \dot{m}h - \sum_{\text{out}} \dot{m}h = \left( \frac{d(mu)}{dt} \right)_{\text{system}} \quad (6)$$

The right term represents the thermal mass drift. Once switched on, the compressor parts will heat up over time until they reach a certain constant. During this phase, the system will accumulate energy. The mass drift term will become zero only at the end of this phase. This term depends on time and temperature at any point of the compressor and is therefore difficult to estimate. This is why the data acquisition will start when the temperature at the compressor outlet is stabilized. Fig. 34 represents the evolution of the compressor outlet and surface temperatures for a compressor speed of 1000 RPM and an outdoor temperature of  $25^\circ\text{C}$  (cooling mode). As can be seen in this figure, the temperature stabilizes after more or less 50 minutes of operation.

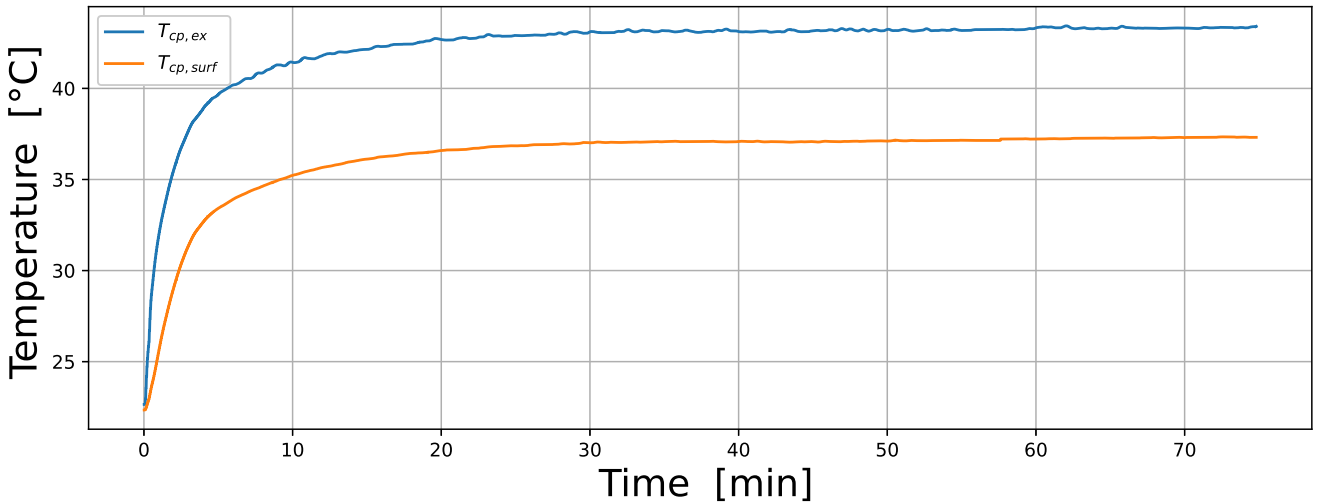


Figure 34: Evolution of the compressor outlet and surface temperatures

The numbering used to designate the different states will be the same as illustrated on the P&ID in Fig.5:

- **State 1:** evaporator outlet/ compressor inlet;
- **State 2:** compressor outlet/ condenser inlet;
- **State 3:** vapor saturation where quality  $X=1$  in the condenser;
- **State 3b:** liquid saturation where quality  $X=0$  in the condenser;
- **State 4:** condenser outlet/ orifice tube inlet;
- **State 5:** orifice tube outlet/ inlet evaporator;
- **State 5b:** vapor saturation where quality  $X=1$  in the evaporator.

## 2 Uncertainty propagation

As previously said in Subsec.3.1, it is essential to determine the main source of errors in order to quantify the precision of the results. It has been decided to do the uncertainty propagation on the computation of the COP, the compressor power and the condenser power in heating mode. An EES code has been developed in order to realize this uncertainty propagation. The numerical values are the experimental data obtained for an outdoor temperature of  $5^{\circ}C$  and a compressor speed of 1000 RPM.

### 2.1 COP<sub>heating</sub>

The coefficient of performance in heating mode can be calculated with Eq.4. The results of the uncertainty propagation are displayed in Tab.4.

Table 4: Uncertainty propagation on the COP (heating mode)

Variables $\pm$ Uncertainty	Partial derivative	% of uncertainty
$COP_{heating} = 3.09 \pm 0.4378$ [/]		
$I = 1.25 \pm 0.175$ [A]	$\partial COP_{heating} / \partial I = -2.484$	98.63 %
$\dot{m}_{refr} = 0.00836 \pm 0.00005852$ [kg/s]	$\partial COP_{heating} / \partial \dot{m}_{refr} = 369.7$	0.24 %
$p_{ex,cp} = 1.254 \times 10^6 \pm 2000$ [Pa]	$\partial COP_{heating} / \partial p_{ex,cp} = -5.747 \times 10^{-7}$	0 %
$T_{ex,cd} = 43.94 \pm 1$ [ $^{\circ}C$ ]	$\partial COP_{heating} / \partial T_{ex,cd} = -0.03473$	0.63 %
$T_{ex,cp} = 50.4 \pm 1$ [ $^{\circ}C$ ]	$\partial COP_{heating} / \partial T_{ex,cp} = 0.02878$	0.43 %
$V = 288 \pm 1$ [ $^{\circ}C$ ]	$\partial COP_{heating} / \partial V = -0.01073$	0.06 %

In this case, the main source of error comes from the current measurement  $I$ . In fact, as can be seen in Tab.2, the current sensor has a large absolute error compared to the other sensors.

## 2.2 Compressor power

The compressor electrical power is computed by multiplying the compressor voltage and the current measured by the current sensor (see Eq. 1). Tab. 5 shows the results of the uncertainty propagation.

Table 5: Uncertainty propagation on the COP (heating mode)

Variables $\pm$ Uncertainty	Partial derivative	% of uncertainty
$\dot{W}_{cp} = 360 \pm 50.42$ [W]		
$I = 1.25 \pm 0.175$ [A]	$\partial\dot{W}_{cp}/\partial I = 288$	99.94 %
$V = 288 \pm 1$ [°C]	$\partial\dot{W}_{cp}/\partial V = 1.25$	0.06 %

As expected, the main source of error is still the current sensor.

## 2.3 Condenser power

The condenser power is expressed as follow:

$$\dot{Q}_{cd} = \dot{m}_{refr} \times (h_{su,cd} - h_{ex,cd}) \quad (7)$$

The condenser supply enthalpy of the refrigerant  $h_{su,cd}$  is calculated with the Coolprop module 4 with inputs the external compressor pressure and the exhaust compressor temperature. On the other hand, the inputs for the calculation of the condenser exhaust enthalpy  $h_{ex,cd}$  are the external condenser pressure and the exhaust condenser temperature.

Table 6: Uncertainty propagation on the COP (heating mode)

Variables $\pm$ Uncertainty	Partial derivative	% of uncertainty
$\dot{Q}_{cd} = 1113 \pm 18.01$ [W]		
$\dot{m}_{refr} = 0.00836 \pm 0.00005852$ [kg/s]	$\partial\text{COP}_{\text{heating}}/\partial\dot{m}_{\text{refr}} = 133078$	18.69 %
$p_{ex,cp} = 1.254 \times 10^6 \pm 2000$ [Pa]	$\partial\text{COP}_{\text{heating}}/\partial p_{ex,cp} = -0.0002069$	0.05 %
$T_{ex,cd} = 43.94 \pm 1$ [°C]	$\partial\text{COP}_{\text{heating}}/\partial T_{ex,cd} = -12.5$	48.17 %
$T_{ex,cp} = 50.4 \pm 1$ [°C]	$\partial\text{COP}_{\text{heating}}/\partial T_{ex,cd} = 10.36$	33.08 %

### 3 Labview's interface

The acquisition system used for this test bench is the Labview software developed by National Instrument. The interface developed for the test bench is shown in Fig.35

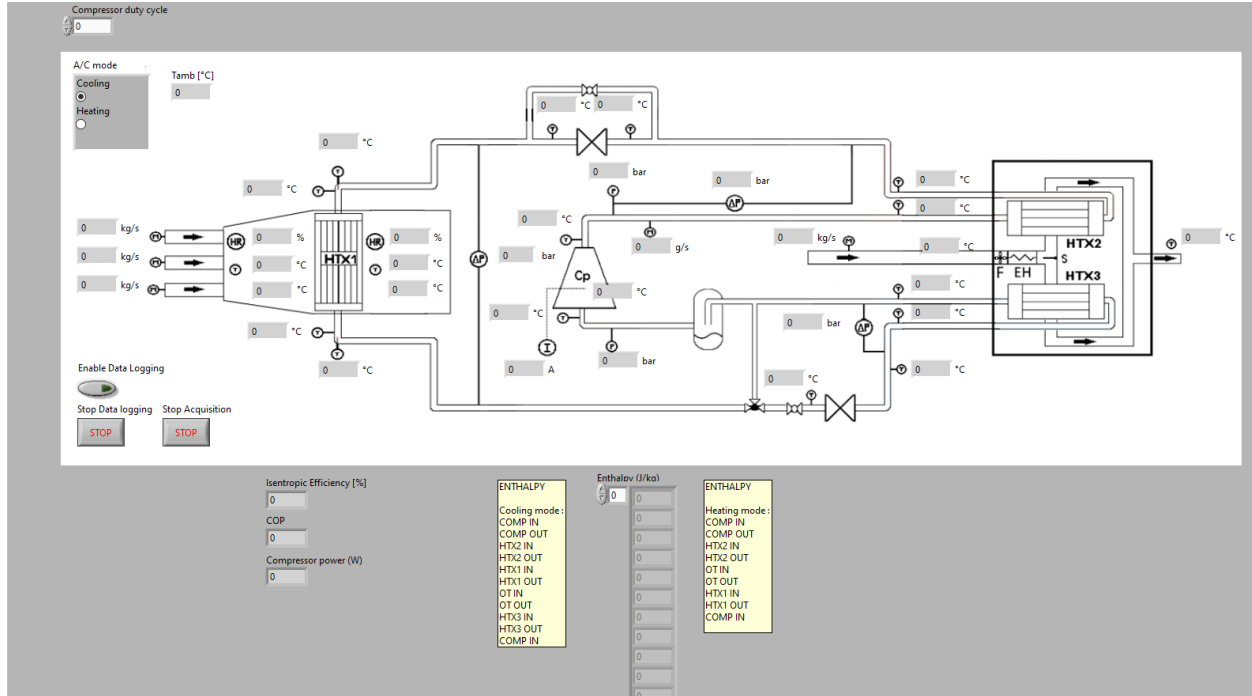


Figure 35: Labview's interface

This graphical interface shows in real time the different temperature and pressures at each point of the cycle directly on the P&ID. The different enthalpies are also listed in a table depending on the mode. Moreover, the isentropic efficiency (Eq.5), the COP (Eq.3 & 4), and the compressor power (Eq.1) are also displayed.

### 4 First experimental campaign

The first idea was to perform several tests in heating and cooling mode and to study the influence of different controllable parameters such as outdoor temperature, compressor rotation speed, and air flow rate. However, during the first test in hot mode, the flexible pipe at the compressor outlet became detached due to a too high pressure. As a result, oil and refrigerant leaked into the entire laboratory. However, these operating conditions had already been tested in previous trials but no problems were observed. Fig.36 represents the evolution of the compressor outlet pressure during the accident. This last reached a maximum pressure of 33.63 bar which caused the rupture of the flexible pipe at the compressor outlet.

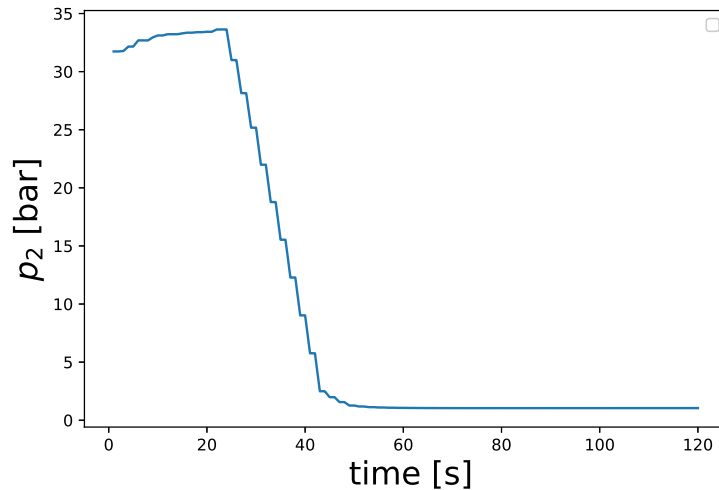
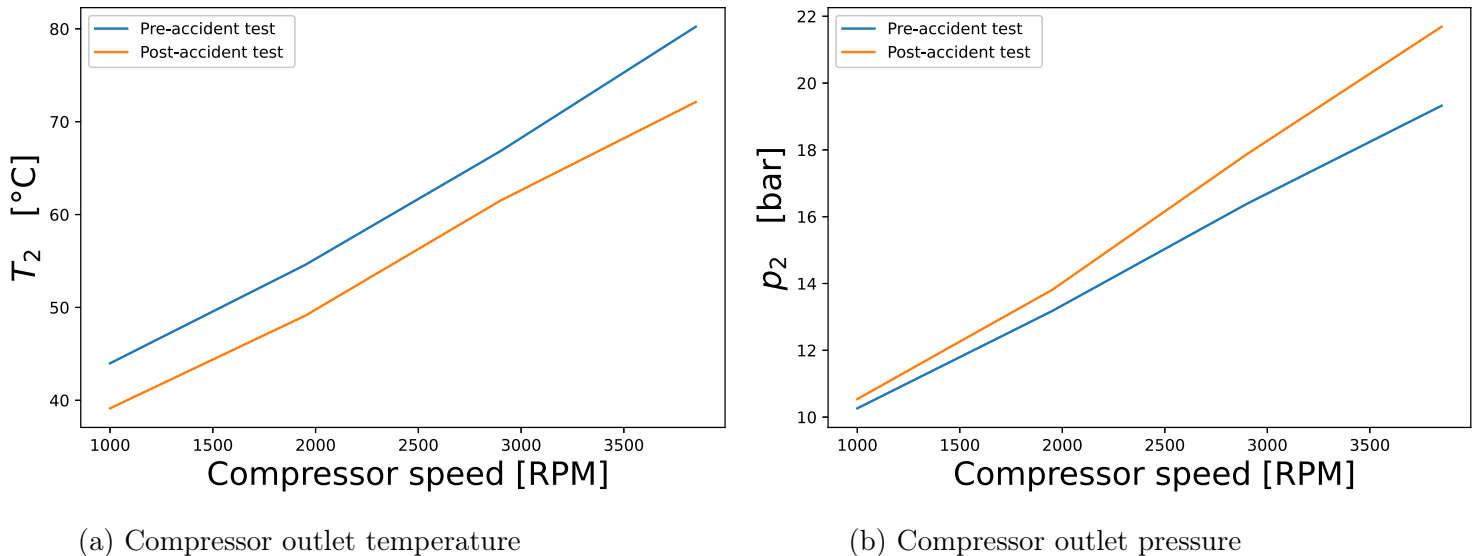


Figure 36: Evolution of the outlet pressure during the accident

Vacuum has been made into the bench in order to evacuate all the residual refrigerant. However, the amount of oil that leaked from the cycle was unknown. It was therefore decided to give a quantity of 100 grams, which is the recommended amount for this loop.

Once the bench was operational again, new tests have been performed. However, even in the same conditions, the results obtained were not the same. As it can be observed in Fig 37a & 37b, the outlet compressor temperature was lower and the outlet pressure higher.

Figure 37: Evolution of  $T_2$  and  $p_2$  as a function of the compressor speed

It was concluded that the difference in results was due to a difference in oil between the

pre- and post-accident tests. In fact, after the accident, not all the oil had escaped, but a certain unknown quantity remained in the cycle. However, the quantity of oil necessary if everything had disappeared was still injected. This may explain the difference in results.

Oil played several functions for compressor operation and is therefore essential. First of all, oil lubricates moving parts inside the compressor. Then it contributes to the compressor sealing and keeps internal components clean by eliminating residues. Finally, it also plays a role in compressor cooling by absorbing the heat generated during compression of the refrigerant gas and helping to dissipate it.

However, too much oil can have a detrimental effect on compressor performance. It is therefore necessary to maintain an adequate level. Excess oil can reduce the compressor's suction volume. In fact, the oil occupies a space normally dedicated to the refrigerant in the compressor, resulting in higher compression and thus a higher outlet pressure as can be observed in Fig. 37b. In addition, too much oil can dilute the refrigerant. The mixture is then less efficient for transferring heat, resulting in a lower compressor outlet temperature as it is observed in Fig. 37a.

## 5 Second experimental campaign

In order to carry out this new test campaign, oil was removed from the cycle in order to try to find an acceptable quantity, more or less 33g has been removed. In this work, the properties of an oil-refrigerant mixture will not be considered. The enthalpies of the refrigerant at different points in the cycle will be calculated from the properties of the refrigerant R1234yf.

After careful analysis of the previous results, there were still inconsistencies in the balances and temperature measurements. After checking the properties of the thermocouples in Labview, it was found that they were of type J when they are actually of type T. The temperatures displayed on the Labview interface were then incorrect. In fact, type J thermocouples have a different conversion law than those of type T. A new test campaign was therefore conducted with the correct thermocouple type specification.

### 5.1 Cooling mode

A total of fifteen tests have been realized in cooling mode with three different outdoor temperatures (25, 35 and 45°C) and five compressor speeds listed in Tab. 7.

Table 7: Correspondence between duty and compressor speed

<b>Duty cycle [%]</b>	10	15	20	25	30
<b>Compressor speed [RPM]</b>	1000	1475	1950	2425	2900



Due to the new safety rules imposed after the accident, the speed of the compressor is limited to not exceed a pressure of 25 bar at the compressor outlet.

### 5.1.1 Refrigerant mass flow rate

The refrigerant mass flow rate is measured by a Coriolis flow meter. Fig.38 shows the evolution of the refrigerant mass flow rate for different compressor speeds and outdoor temperatures. As expected, the flow rate increases with the speed of the compressor.

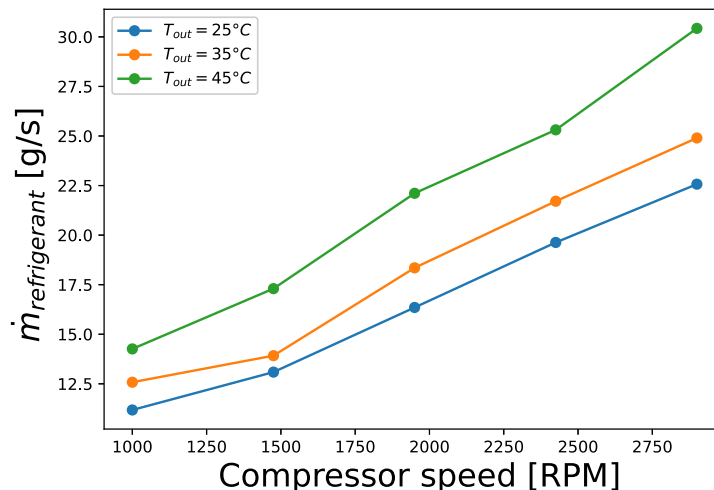


Figure 38: Refrigerant mass flow rate for different compressor speeds and outdoor temperatures

The refrigerant mass flow rate also increases with the outdoor temperature. This can be explained by two reasons. Firstly, in cooling mode, the goal is to cool down the air through the evaporator. However, if the outdoor temperature increases, thermal load increases as well. The refrigerant flow must therefore be increased to allow higher heat transfer. Secondly, when outdoor temperature increases, compressor inlet pressure also increases. As a consequence, refrigerant flow rate increases to maintain an appropriate operating pressure in the system.

### 5.1.2 Compressor

The electrical compressor power displayed in the Labview interface is computed thanks to the product of the compressor voltage and the current measured in real time by the current sensor:

$$\dot{W}_{cp} = V \times I \quad (8)$$

This power is displayed on Fig.39 for different compressor speeds and outdoor temperatures.

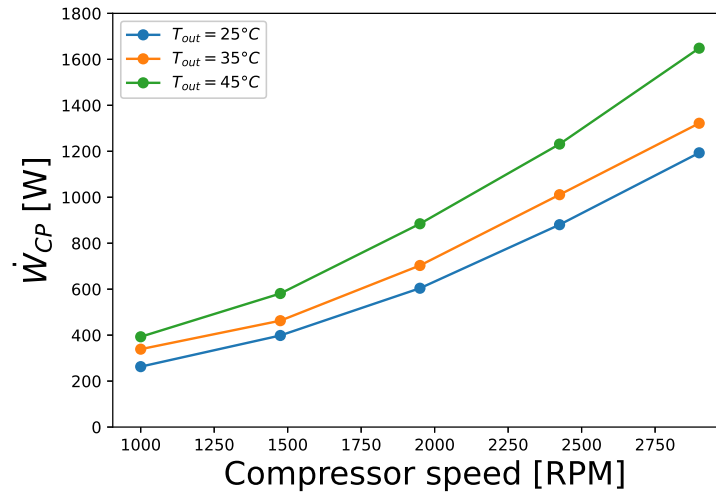


Figure 39: Compressor consumption for different compressor speeds and outdoor temperatures

The faster the compressor turns, the more energy it consumes. Moreover, the compressor power also increases with the outdoor temperature. As for the refrigerant flow, this can be explained by the augmentation of the thermal load.

According to the uncertainty analysis (see Tab. 5), the current sensor represents the main source of error in the compressor power calculation. In fact, as can be observed in Fig. 40, the measured current is very fluctuating. However, for the calculation, an average of its measurements over a period of time can be taken. This current has been checked using a multimeter as well as on the display of the compressor power supply T1.

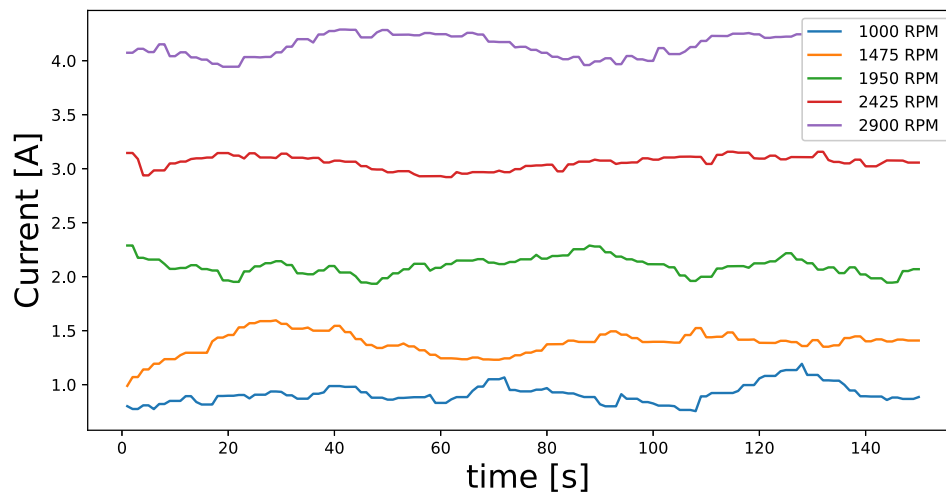


Figure 40: Evolution of current sensor measurement over time

Moreover, as explained in Subsec. 2.1, the internal electronics of the compressor returns a

PWM signal whose duty cycle indicates the power consumed by the compressor. Fig.11 shows the correspondence between duty cycle and power consumption. This PWM signal could be displayed on the oscilloscope and the duty cycle of it have been visually determined. The powers calculated using the graph on Fig.11 were consistent with those calculated with the current sensor and displayed on the Labview interface.

### 5.1.3 Determination of oil mass flow rate

As previously said, excess oil can have a detrimental effect on the compressor work. It is therefore interesting to determine the oil mass flow rate of the compressor. The electrical compressor power can be expressed as the sum of the compressor internal power, the compressor ambient losses and the part of the work associated to the presence of oil.

$$\dot{W}_{cp} = \dot{W}_{in} + \dot{W}_{oil} + \dot{Q}_{amb,cp} \quad (9)$$

The internal power represents the power required to compress the refrigerant.

$$\dot{W}_{in} = \dot{m}_{refr} \times (h_2 - h_1) \quad (10)$$

with the mass flow rate measured by the Coriolis flow meter.

The second term refers to the impact of compressor oil on the compression.

$$\dot{W}_{oil} = \dot{m}_{oil} \times C_{p,oil} \times (T_{oil,ex} - T_{oil,su}) \quad (11)$$

with  $C_{p,oil} \approx 2000 [J/Kkg]$  and the compressor oil inlet (resp. outlet) temperature can be assumed to be equal to the refrigerant inlet (resp. outlet) temperature.

Finally, the third term represents the ambient losses, in fact, compressor surface heats up during operation which leads to a heat exchange with the environment due to the temperature difference. They are expressed as follow:

$$\dot{Q}_{amb,cp} = A_{cp} \times U_{cp} \times (T_{surf,cp} - T_{amb}) \quad (12)$$

The compressor geometry can be considered cylindrical for ease of calculation. The exchange surface  $A_{cp}$  of the compressor is approximately  $0.085 m^2$  and the heat exchange coefficient  $U_{cp}$  can be assumed equal to  $10 W/m^2K$ .

Finally, the oil mass flow rate can be determined and Eq.9 becomes:

$$\dot{m}_{oil} = \frac{\dot{W}_{cp} - \dot{W}_{in} - \dot{Q}_{amb,cp}}{C_{p,oil} \times (T_{ex,oil} - T_{su,oil})} \quad (13)$$

The evolution of the oil mass flow rate in terms of compressor speeds for different outdoor temperatures is illustrated in Fig.41.

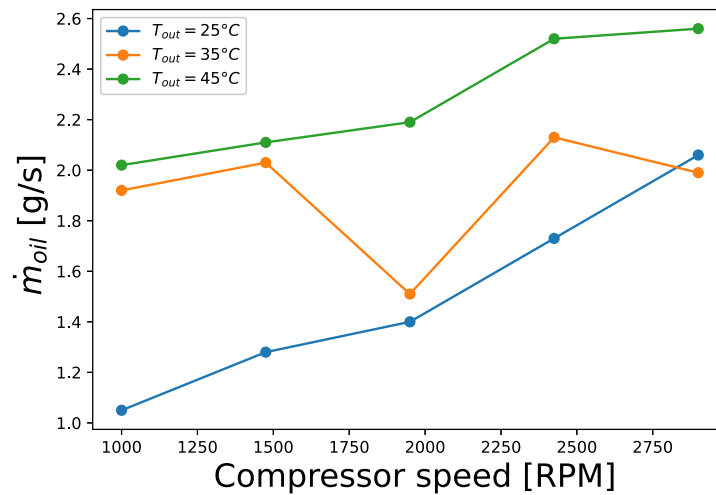


Figure 41: Oil mass flow rate for different compressor speeds and outdoor temperatures

The oil circulation rate OCR represents the proportion of oil in the refrigerant flow rate and can be determined as follows:

$$\text{OCR} = \frac{\dot{m}_{oil}}{\dot{m}_{oil} - \dot{m}_{refr}} \quad (14)$$

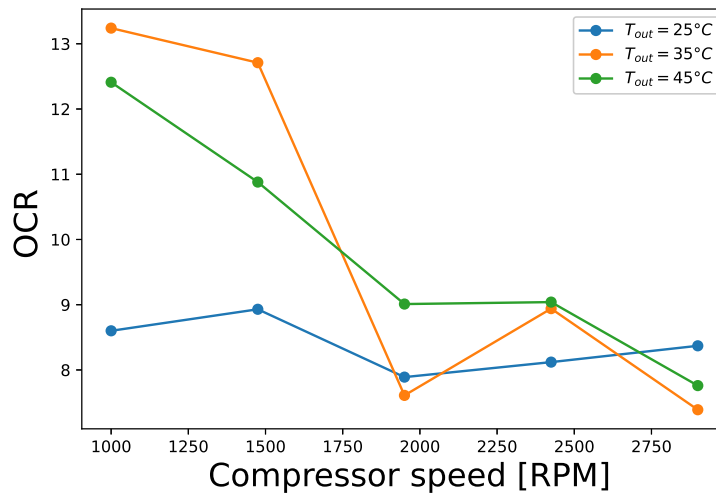


Figure 42: OCR for different compressor speeds and outdoor temperatures

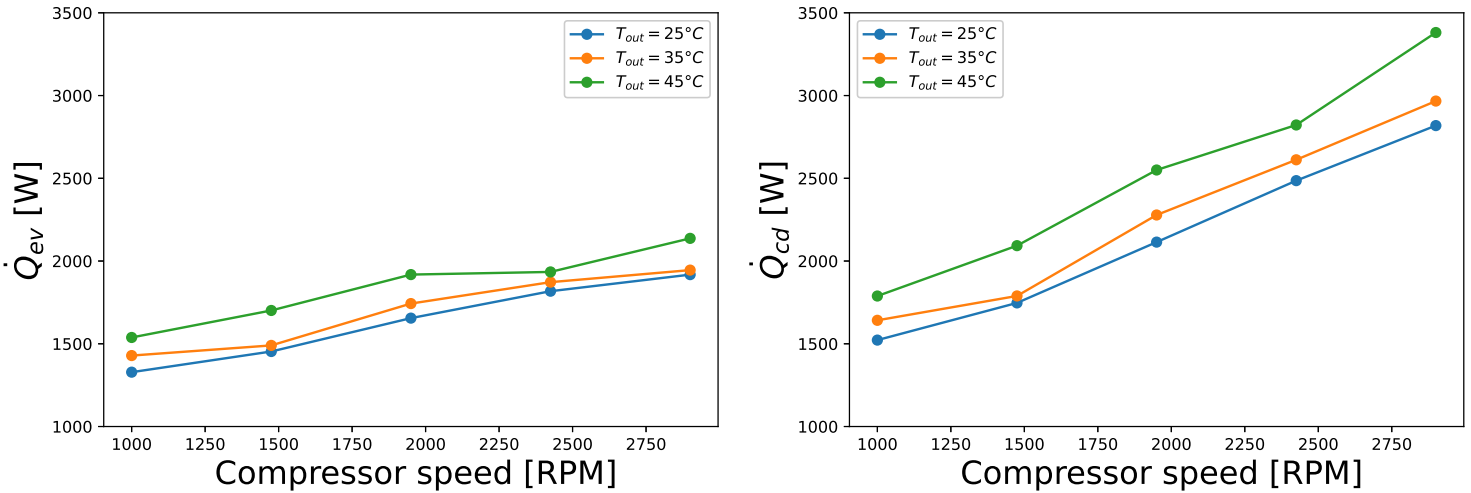
Typical value for this type of compressor is between 0 and 5%. There may still be too much oil in the cycle.

### 5.1.4 Evaporator and condenser

In cooling mode the evaporator is the inner heat exchanger of the HVAC module (HTX3) and the condenser the external one (HTX1). The evaporator and condenser heat transfer rates are shown in Fig.43a & Fig.43b respectively for different compressor speeds and outdoor temperatures. There are calculated as follow:

$$\dot{Q}_{ev} = \dot{m}_{refr} \times (h_1 - h_5) \quad \& \quad \dot{Q}_{cd} = \dot{m}_{refr} \times (h_4 - h_2) \quad (15)$$

With  $\dot{m}_{refr}$  the mass flow rate measured by the flow meter.



(a) Evaporator heat flow rate

(b) Condenser heat flow rate

Figure 43:  $\dot{Q}_{ev}$  and  $\dot{Q}_{cd}$  for different compressor speeds and outdoor temperatures

Logically, as can be observed in Fig.43, condenser and evaporator heat flow rates increase with outside temperature and compressor speed. This confirms the analysis made for Fig.38 & 39.

### 5.1.5 Energy balance

The energy balance over the entire cycle can be written as follows:

$$\dot{Q}_{cd} - \dot{Q}_{ev} - \dot{W}_{cp} + \dot{Q}_{amb,cp} \stackrel{?}{=} 0 \quad (16)$$

With the evaporator and condenser heat flow rates calculated with Eq.15, compressor consumption with Eq.8 and the compressor ambient losses with Eq.12. The energy balance for the different compressor speeds and outdoor temperatures are shown in the Fig.44. At low compressor speeds, the energy balance remains acceptable. However, at higher speeds, the energy balance becomes quite significant. In fact, for higher compressor speeds, some phenomena contribute to a decrease in overall system efficiency and therefore a poorer energy

balance. For example, heat transfer through the exchangers can be incompleted due to the rapid flow of refrigerant through them. Higher compressor speeds also increase mechanical losses and pressure drops across the system components. Finally, it should be noted that only the ambient losses of the compressor are taken into account in the entire cycle.

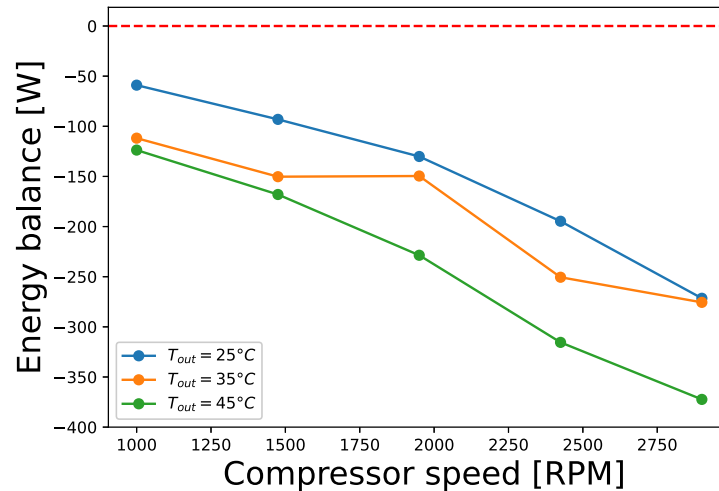


Figure 44: Energy balance over the whole cycle for different compressor speeds and outdoor temperatures

The evolution of these different powers are represented on Fig. 45 for different compressor speeds and outdoor temperatures. The ambient losses (red curve) are negligible compared to the other powers.

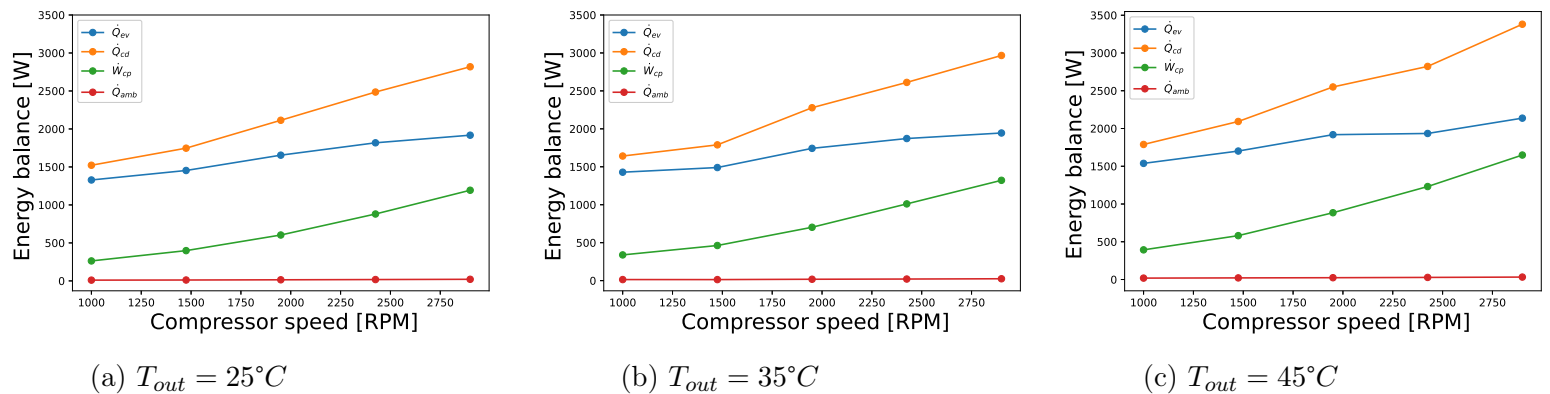


Figure 45: Evolution of the compressor consumption and the heat transfer rates at both the condenser and evaporator as a function of the compressor speed for three different outdoor temperatures

### 5.1.6 COP

The COP in cooling mode can be calculated thanks to Eq.3. Fig.46 represents the COP for different compressor speeds and outdoor temperatures. The faster the compressor rotates, the greater its work (see Fig.39), and the lower the COP. Note that evaporator flow rate also increases with compressor speed (see Fig.43a), but not proportionally with compressor work. It can also be concluded that the system is more efficient for low outside temperatures.

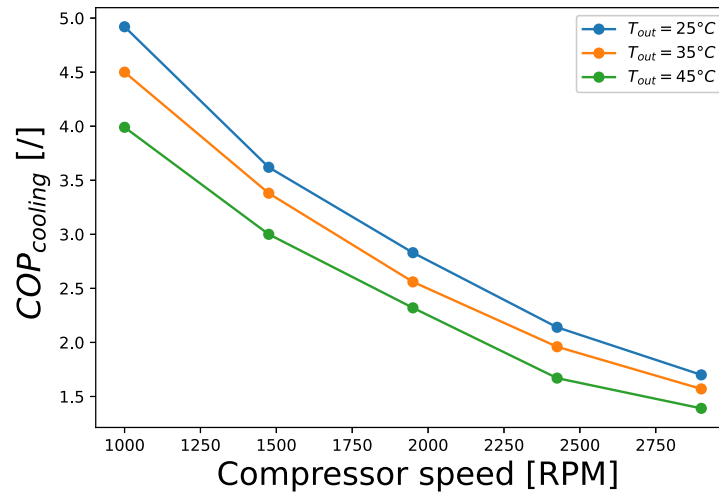


Figure 46: COP for different compressor speeds and outdoor temperatures

### 5.1.7 Isentropic efficiency

The isentropic efficiency can be determined thanks to Eq.5. Fig.47 represents the compressor isentropic efficiency for different compressor speeds and outdoor temperatures.

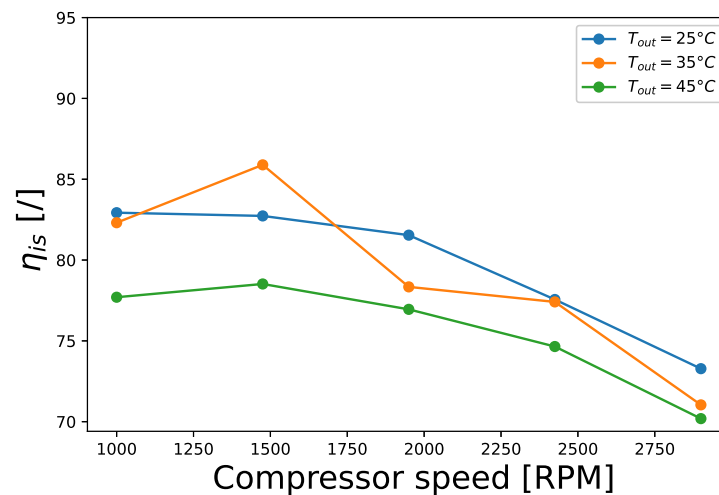


Figure 47: Isentropic efficiency for different compressor speeds and outdoor temperatures

As previously said, isentropic efficiency represents the compressor's ability to increase fluid pressure isentropically i.e. reversibly and adiabatically (without heat exchange with the environment). However, heat losses and reversibilities increase with the compressor speed.

Firstly, the effects of friction and leakage between the compressor's moving parts are greater at higher speeds. Part of the energy supplied to the compressor is then dissipated in the form of heat or mechanical energy losses, leading to a reduction in isentropic efficiency. Secondly, fluid operating conditions in the compressor may diverge from ideal theoretical conditions. For example, superheating, heat losses, irreversibilities due to real compression processes are phenomena that can be amplified by the increase in compressor speed resulting in a reduction in isentropic efficiency.

## 5.2 Heating mode

Due to the new security rule on the compressor outlet pressure, only a few tests have been performed in heating mode. In fact, it was decided not to go beyond a pressure of 25 bar. Only nine tests have been realized for three outdoor temperatures (0, 5 and 10 °C) and three compressor speeds. The graphs presented in this subsection will therefore be less relevant than those in the cooling subsection. However, the conclusion about the different graphs remains the same.

### 5.2.1 Refrigerant mass flow rate

Fig 48 shows the evolution of the refrigerant mass flow rate for different compressor speeds and outdoor temperatures. Curves shape are the same as for the cooling mode. The same conclusions can therefore be drawn.

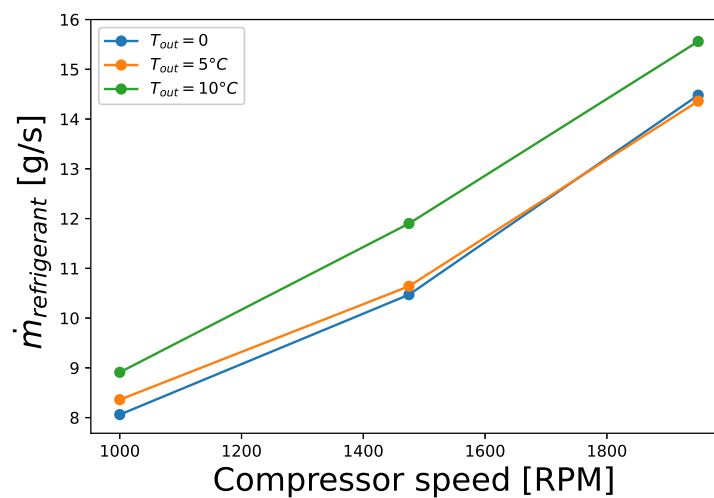


Figure 48: Refrigerant mass flow rate for different compressor speeds and outdoor temperatures



### 5.2.2 Compressor

The compressor power evolution for different compressor speeds and outdoor temperatures is shown in Fig. 49. The compressor consumption increases with its speed and with the outdoor temperature as for the cooling mode.

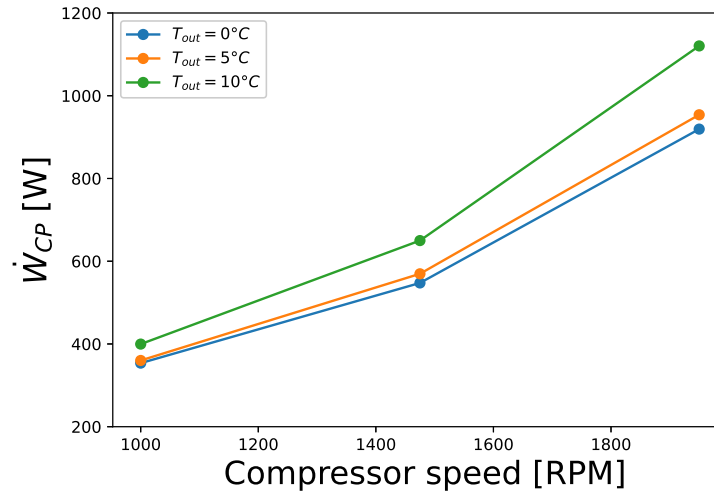


Figure 49: Compressor consumption for different compressor speeds and outdoor temperatures

### 5.2.3 Evaporator and condenser

In heating mode, the evaporator is the external exchanger (HTX1) and the condenser the internal one (HTX2). The evaporator and condenser heat transfer rates are shown in Fig. 43a & Fig. 43b respectively for different compressor speeds and outdoor temperatures. They are calculated with Eq. 15.

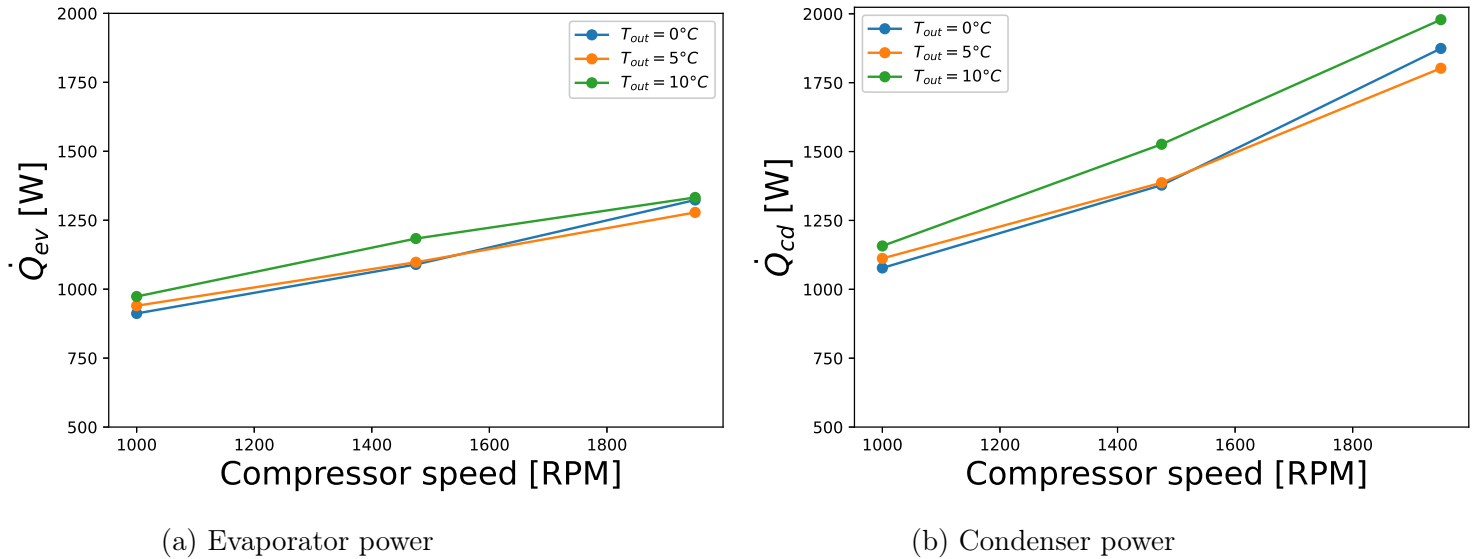


Figure 50:  $\dot{Q}_{ev}$  and  $\dot{Q}_{cd}$  for different compressor speed and outdoor temperature

As for cooling mode, condenser and evaporator heat flow rates increase with outside temperature and compressor speed.

#### 5.2.4 Energy balance

The energy balance over the entire cycle can be calculated with Eq. 16. Fig. 44 represents the energy balance for different compressor speeds and outdoor temperatures. The energy balance is less good in heating mode than in cooling.

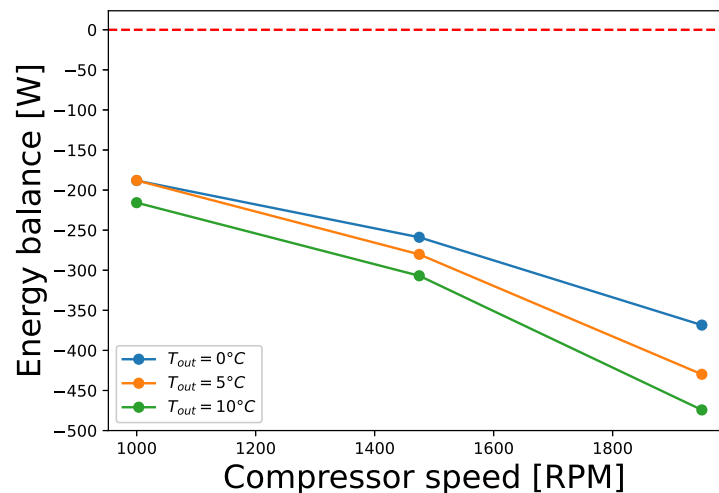


Figure 51: Energy balance over the whole cycle for different compressor speeds and outdoor temperatures

The evolution of these different powers are represent on Fig 52 for different compressor speed and outdoor temperature.

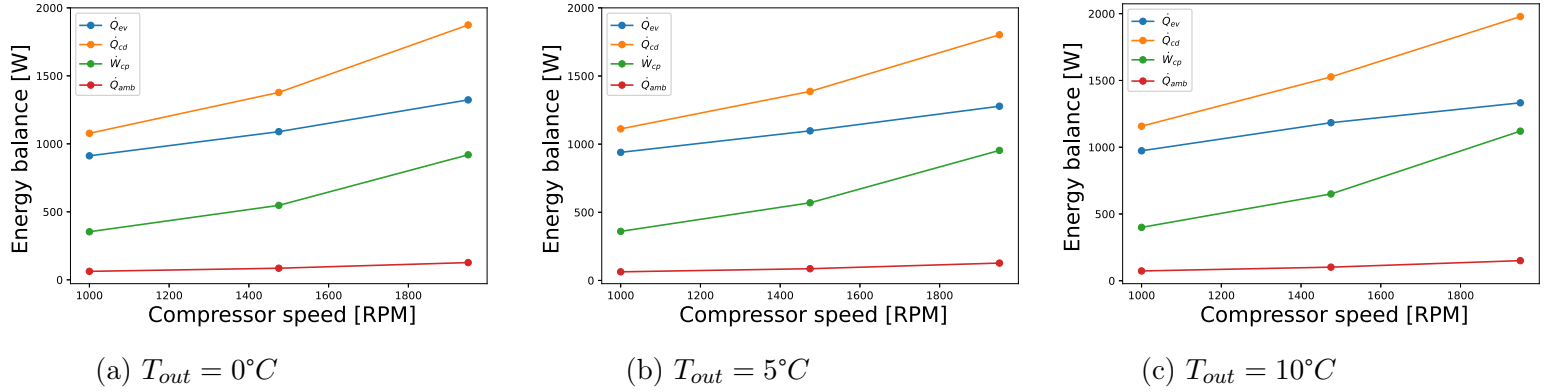


Figure 52: Evolution of the compressor consumption and the heat transfer rates at both the condenser and evaporator as a function of the compressor speed for three different outdoor temperatures

## 6 Air flow rate

As previously said in Subseq 2.9, the portable air conditioning unit can deliver a maximal air flow rate of  $600 \text{ m}^3/\text{h}$ . As can be seen on Fig. 3, two pipes have been plugged in each output of the mobile unit in order to deliver the air flow to the HVAC module and to the wind tunnel. For the whole experimental campaign, the speed of the turbine was set to its maximum (potentiometer on 10) in order to obtain the maximum air flow, i.e  $300 \text{ m}^3/\text{h}$  in each output. However, the air flow rate arriving on the front heat exchanger HTX1 and on the internal one HTX2 or HTX3 will not be equal to this value because of pressure losses created by the system. It is therefore essential to determine the real air flow rate arriving in the exchangers.

A Python code has been developed in order to determine the pressure losses into the portable unit as a function of flow. The whole development is described in the Annex VI. For a maximal turbine speed i.e. potentiometer on "10" the real flow rate arriving to the HVAC module and to the wind tunnel is approximately equal to  $287 \text{ m}^3/\text{h}$ . The air pulse by the front fan (FAN2) and by the pulser (FAN1) must also be taken into account in the air flow rate computation. However, Renault has not provided any information on these fans. Air flow rate must therefore be experimentally determined. Three different approaches have been tested: with an anemometer, with an orifice plate, or by balance on the exchangers.

## 6.1 Anemometer

### 6.1.1 Air flow on the external heat exchanger (HTX1)

Air velocity was measured at nine locations on the front face of the exchanger HTX1 as shown in Fig. 53. The results obtained are represented in Tab. 9. For this test, the front fan (FAN2) and internal fan of the mobile unit were at their maximal speed.

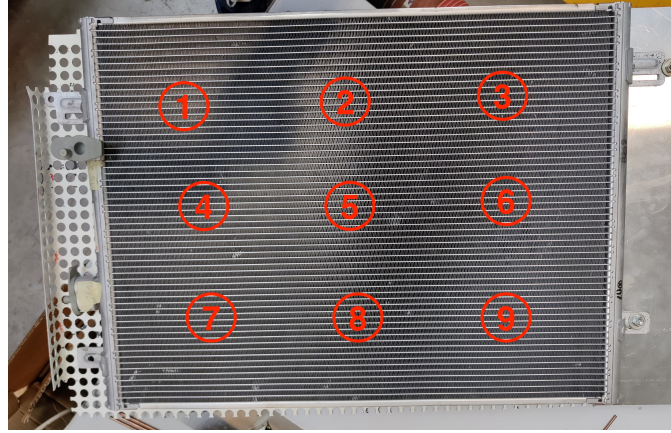


Figure 53: Air flow measurement on the front face of the external exchanger (HTX1)

Table 8: Measured air velocity

	1	2	3	4	5	6	7	8	9
$v_{\max}$ [m/s]	0.5	0.4	0.43	0.25	0.6	0.46	0.34	0.52	0.42
$v_{\min}$ [m/s]	0.44	0.34	0.33	0.2	0.4	0.29	0.27	0.37	0.28
$v_{\text{moy}}$ [m/s]	0.47	0.37	0.38	0.225	0.5	0.375	0.305	0.445	0.35

By taking an average of these nine points, the following airflow is obtained:

$$Q_{air,HTX1,max} = V_{air,HTX1} \times A_{HTX1} = 264 \text{ [m}^3\text{/h]} \quad (17)$$

with  $V_{air,HTX1} = 0.38 \text{ m/s}$

According to the anemometer measurement the maximal air flow passing through the front exchanger HTX1 is equal to  $264 \text{ [m}^3\text{/h]}$ .

### 6.1.2 Air flow inside the HVAC module

Air velocity was measured at three locations at the HVAC outlet, as shown in Fig. 54. The results obtained are represented in Tab. 9. For this test, the pulser (FAN1) and internal fan of the mobile unit were at their maximal speed.

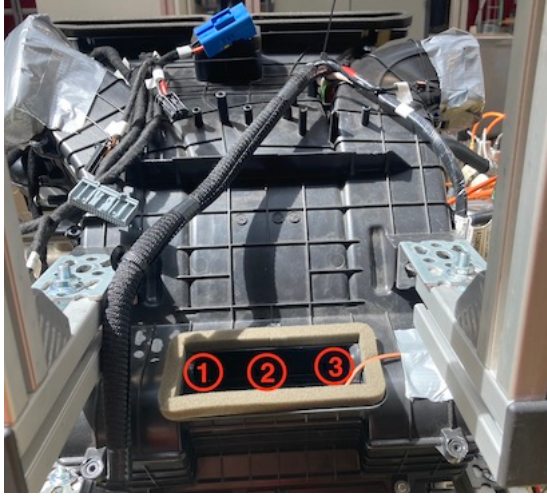


Figure 54: Air flow measurement at the HVAC output

Table 9: Measured air velocity

	<b>1</b>	<b>2</b>	<b>3</b>
$v_{\max}$ [m/s]	6.46	8.14	7.7
$v_{\min}$ [m/s]	6.37	8	7.5
$v_{\text{moy}}$ [m/s]	6.415	8.07	7.6

By taking an average of these three points, the following airflow is obtained:

$$Q_{air,out,HVAC,max} = V_{air,HVAC} \times A_{HVAC,out} = 63.6 \text{ [m}^3\text{/h]} \quad \text{with} \quad A_{HVAC,out} \approx 0.0024 \text{ [m}^2\text{]} \quad (18)$$

According to the anemometer measurements, the maximal air flow leaving the HVAC module is approximately equal to  $63.6 \text{ m}^3\text{/h}$ .

## 6.2 Orifice plate

In order to measure the flow rate at the HVAC outlet in a different way, an orifice plate have been installed. The installation is shown in Fig.28. Under normal test conditions, i.e. with maximum flow to the HVAC module, the pressure difference measured was equal to 120 Pa which corresponds to an air flow rate of  $51 \text{ m}^3\text{/h}$ .

## 6.3 Balance on the heat exchanger

The air mass flow rate can be determined with the following equation:

$$\dot{m}_{air,HTX} = \dot{m}_{refr,th} \times \frac{|h_{refr,out} - h_{refr,in}|}{|h_{HTX,out,air} - h_{HTX,in,air}|} \quad (19)$$

The air flow rate is thus obtained by dividing the mass flow rate by the air density which is dependent on temperature and pressure.

$$Q_{air,HTX} = \frac{\dot{m}_{air,HTX}}{\rho_{air}} \quad (20)$$

The enthalpies on the refrigerant side are calculated with the library Coolprop [4] directly on Labview. For the air enthalpies, if it is a condenser, air can be considered dry, as it is heated through the condenser. However, if the air is cooled through an evaporator, relative humidity has to be taken into account because of condensation. The air temperature at the exchanger inlet and outlet is measured by type-T thermocouples placed respectively upstream and downstream. However, thermocouples are more suitable for measuring fluid or solid surface temperatures than air temperatures. Indeed, environmental conditions have a major influence on their accuracy. Air flow can therefore influence air temperature measurement.

### 6.3.1 Air flow on HTX1

In order to measure the air temperature on both side of HTX1, three thermocouples are placed upstream and nine downstream. As said previously, the external heat exchanger HTX1 can act as a condenser (cooling mode) or as an evaporator (heating mode). Humidity sensors have thus been placed on both sides of the heat exchanger to take into account the condensation when HTX1 acts as an evaporator. The air flow rate is calculated by Eq.20 and is represented in Fig.55 for different compressor speeds and outdoor temperatures.

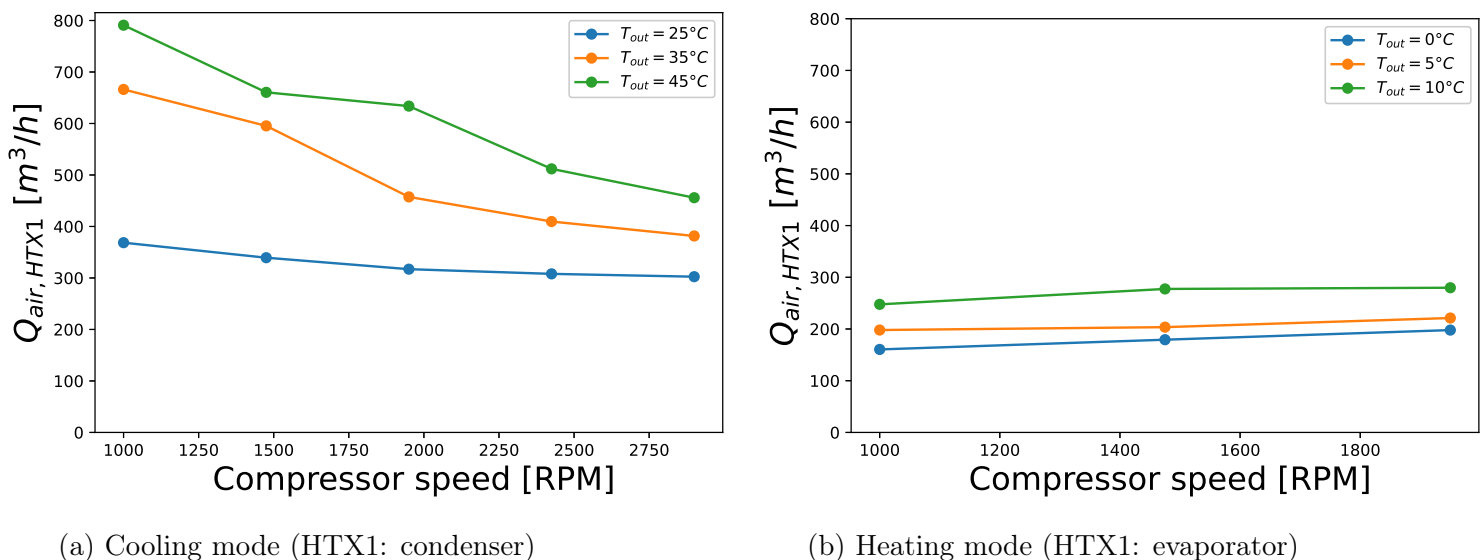


Figure 55: Air flow through HTX1

### 6.3.2 Air flow inside the HVAC module

Two thermocouples are placed at the HVAC module's inlet and outlet to measure air temperature. Since the evaporator is stuck into the HVAC module, humidity sensors could not be placed on either side. Condensation cannot be taken into account. The air flow rate is represented for different compressor speeds and outdoor temperatures in Fig.56.

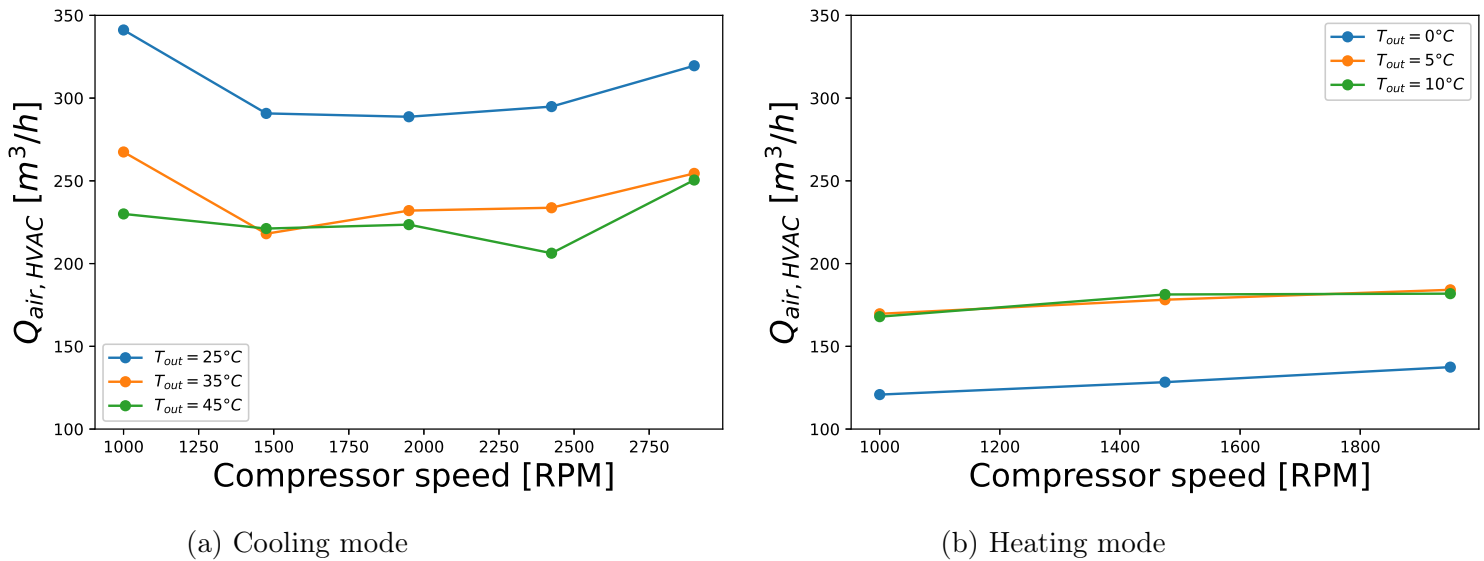


Figure 56: Air flow through the HVAC

## 6.4 Comparison of the results

Tab.10 shown the air flow rate obtained with the different approaches. Widely divergent results are observed.

Table 10: Comparison of the air flow rate results

	$Q_{air, HTX1}$ [ $m^3/h$ ]	$Q_{air, HVAC}$ [ $m^3/h$ ]
Anemometer	264	64
Orifice plate	/	51
Balance (cooling mode)	480	258
Balance (heating mode)	218	161

Determining the HVAC air flow rate by balance on the exchangers gives very different results from anemometer or orifice plate measurements. The reason for this difference is that the air flow arriving at the HVAC inlet is not equal to the air flow coming out. In fact, as previously said, the HVAC can be considered as 1 input (inlet "1" on Fig.24a) and 1 output (output "9" on Fig.24b). The incoming air flow and outgoing should therefore be equal. Nevertheless, although the flaps were positioned to let air out only through outlet "9", this last also came out of the other outlets of the HVAC module (outlets "2", "3", "4", "7", "8" and "10" on Fig.24b). In addition, it was not possible to measure the air flow directly at the HVAC inlet, as the air speed at this point is greater than the anemometer's measuring range. It is therefore difficult to estimate the actual air flow reaching the internal heat exchangers.

The balance performed on the exchangers for both cooling and heating modes also gives

very different results whether for HVAC air flow or HTX1 air flow. This can be explained by the influence of the air flow on sensor accuracy.



## Part IV

# Numerical model

## 1 Overview

The goal of this numerical model is to describe the behaviour of the reversible heat pump into different conditions. The base of the numerical part was modeled by J.Vega and adapted for the case of the Renault ZOE. As on the test bench, 4 variables can be chosen: the rotational speed of the compressor, the air flow from the front exchanger and from the HVAC module and the external temperature. A guess is made on the internal temperature in order to have a first approximation of the evaporation (resp. condensation) temperature in cooling (resp. heating) mode. The external temperature, on the other hand, allows to make a guess on the condensation (resp. evaporation) temperature in cooling mode (resp. heating). The flowchart of the entire code is represented in Fig.57 where an iterative process is described. In fact, some values have to be guessed in order to solve the problem. These last have thus to be updated at each iteration.

All Python codes are given in AnnexVI. The main code is the following:

- `heat_pump_results_single_point.py`

and the different modules associated:

- `ThermoState.py`
- `EffCompressor.py`
- `HX_LMTD_calcUA.py`
- `DeltaP.py`
- `CalcU.py`
- `Accumulator.py`
- `IdealExpValve.py`

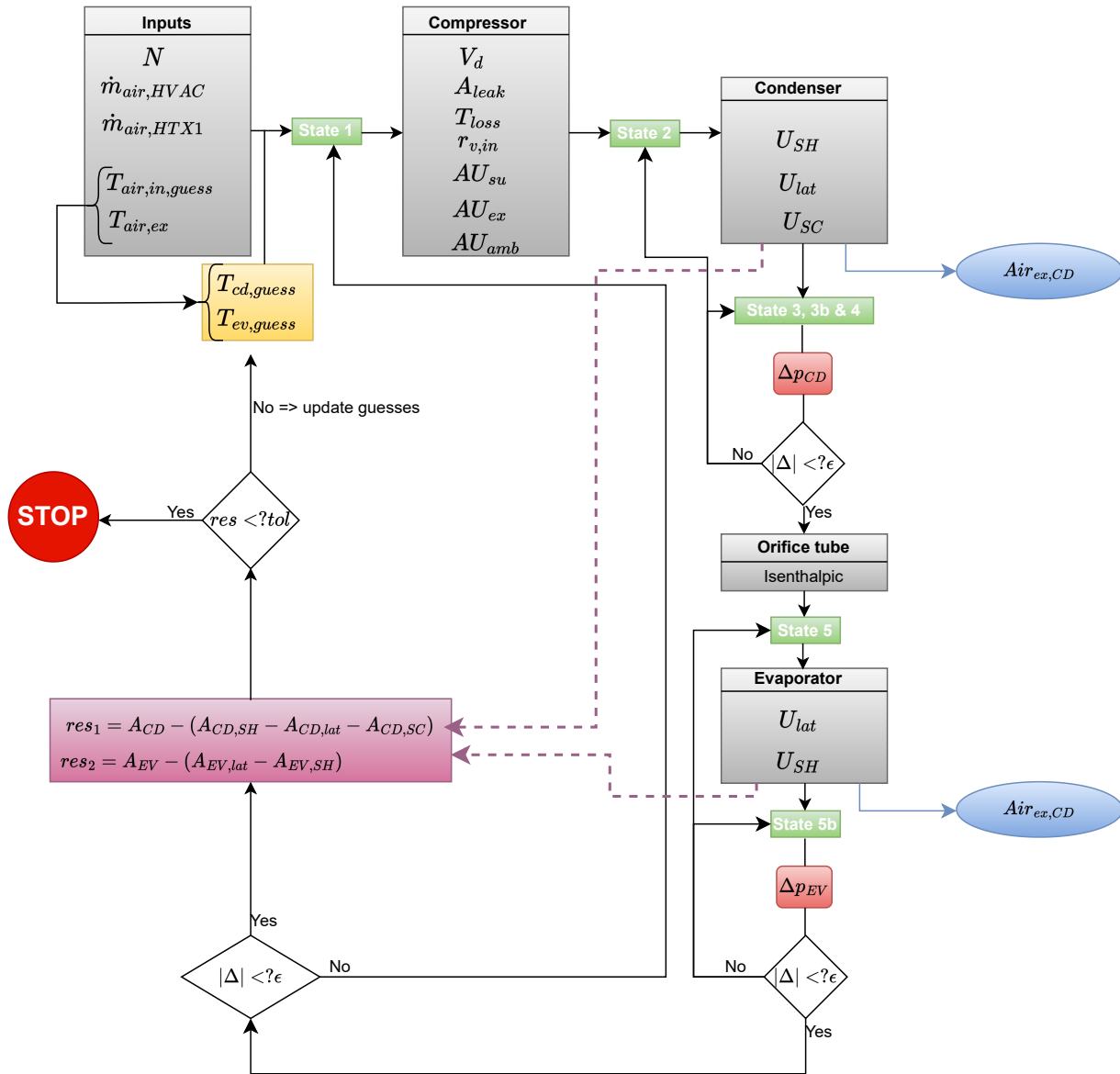


Figure 57: Flowchart

## 2 Component modeling

### 2.1 Compressor

The compressor model is based on the semi-empirical model given by professor V.Lemort in the course of "*Machines et systèmes thermiques*" [14]. In this model supply, exhaust and ambient heat transfers with a fictitious isothermal envelope are considered. Mechanical losses and a leakage mass flow rate through a fixed leak surface are also taken into account. The model can be schematically represented as follow:

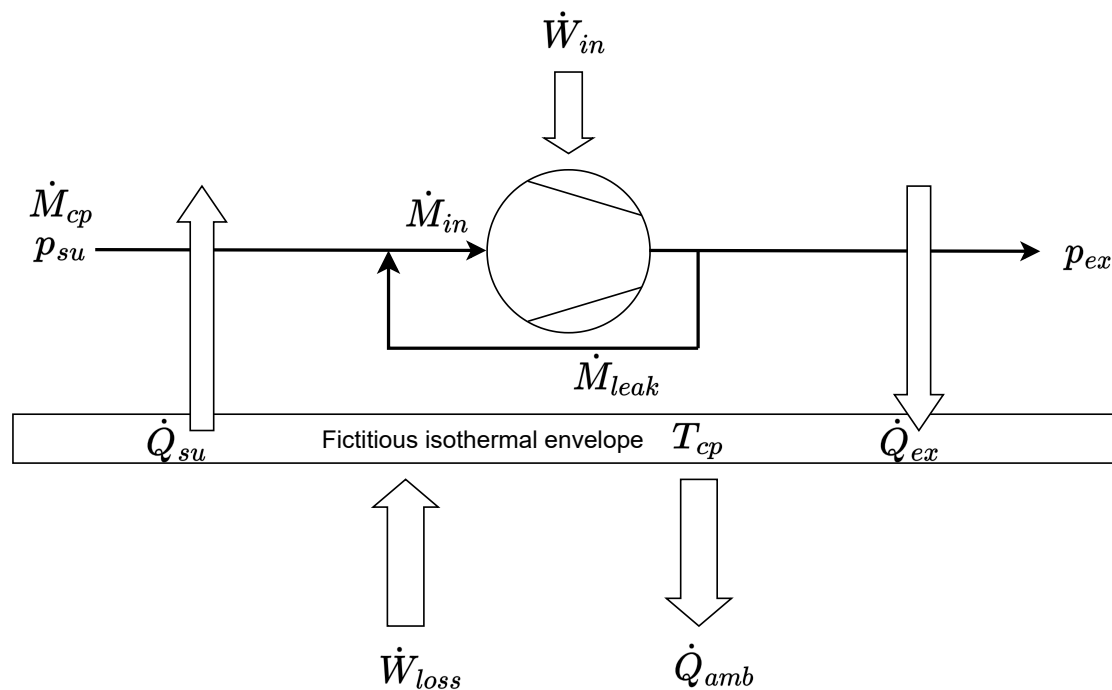


Figure 58: Compressor model

A guess value of the temperature of the fictitious isothermal envelope of the compressor  $T_{cp}$  is needed in order to estimate the exhaust state of the refrigerant. The refrigerant mass flow rate can thus be computed using the input parameters and the leakage mass flow rate. After computing the two compression steps the model can iterate so as to find the real exhaust state. The corresponding python code is the module "`EffCompressor.py`"

#### Assumptions

The following assumptions have been made:

- 2 steps compression: isentropic follow by an isochoric;
- Intake and discharge pressure drops are neglected;

- Ambient temperature:  $T_{amb} = 20^{\circ}C$ .

### Characteristics

Unlike the displacement, compressor characteristics are unknown and have thus to be guessed in a first time and then tuned. The compressor parameters are listed below:

- Displacement:  $V_d = 34 \text{ cm}^3$
- Area of leakage:  $A_{leak}$
- Mechanical loss torque:  $T_{loss}$
- Internal volume ratio:  $r_{v,in}$
- Supply heat exchange coefficient:  $AU_{su}$
- Exhaust heat exchange coefficient:  $AU_{ex}$
- Ambient heat exchange coefficient:  $AU_{amb}$

## 2.2 Heat exchangers

The modelisation of the heat exchangers has been realized by J.Vega and is based on a LMTD-NTU method (module named `HX LMTD CalcUA.py`). Moreover, the interest has been set on the architecture of the heat exchangers used in the bench, the pressure losses occurring through them and the heat transfer coefficient. Two models able to estimate these pressure losses and the heat transfer coefficient have been developed. The modules are respectively named `DeltaP.py` and `CalcU.py`.

### 2.2.1 Architecture

The external heat exchanger (HTX1) can act as a condenser (cooling mode) or as an evaporator (heating mode), its characteristics are listed below:

- Dimensions: 495x390x21 mm
- Number of pipes: 60 <sup>1</sup>
  - Number of pipes in super-heated zone: 15
  - Number of pipes in two-phase zone: 15+15
  - Number of pipes in sub-cooled zone: 15
- Number of passes: 4 <sup>2</sup>

---

<sup>1</sup>Source: [\[8\]](#)

<sup>2</sup>Source: [\[8\]](#)

The internal exchangers HTX2 and HTX3 cannot be analysed because they are stuck into the HVAC module. The inner exchanger model is based on the evaporator model of T. Gillet's thesis ([12]). Their characteristics are the following:

- Dimensions: 200x240x60 mm
- Number of pipes: 23
  - Number of pipes in liquid zone: 9
  - Number of pipes in vapor zone: 14
- Number of passes: 2

### 2.2.2 Pressure drops

Theoretically a heat exchanger can be considered isobar but pressure drop can occur through them. This pressure drop can be evaluated by the following equation [3]:

$$\Delta P_{\text{friction}} = \frac{1}{2} f \times \rho_r v^2 \times \frac{L}{D} \quad (21)$$

$$\text{with } \begin{cases} \lambda_{\text{lam}} = 0.316 \times Re^{-0.25} & \text{if } Re < 2320 \\ \lambda_{\text{turb}} = \left[ -2 \log_{10} \left( \frac{2.51}{Re \sqrt{\lambda_{\text{turb}}}} + \frac{k}{3.7 D_h} \right) \right]^{-\frac{1}{2}} & \text{if } Re > 2320 \end{cases}$$

An iterative process has to be implemented in order to determine this pressure loss. In fact, the density of the fluid has to be calculated but depends itself on the pressure.

### 2.2.3 Heat transfer coefficient U

A similitude correlation [4] between the flow rate and the coefficients based on nominal values has been used in order to determine the heat transfer coefficients of the different exchangers.

$$U = U_{\text{nom}} \left( \frac{\dot{m}}{\dot{m}_{\text{nom}}} \right)^{0.65} \quad (22)$$

The nominal values for the heat transfer coefficients have therefore to be tuned.

### 2.2.4 Condenser modelisation

In cooling mode, the condenser is the external exchanger (HTX1) and in heating mode the internal one (HTX2). The working fluid is the refrigerant R1234yf and the secondary fluid is air. For the modelisation, the condenser is divided into three zones:

---

<sup>3</sup>Source: [8]  
<sup>4</sup>Source: [14]

1. Superheated vapor zone;
2. Two-phase zone;
3. Subcooled liquid zone.

These three division zones can be schematically represented in Fig. 59. The cold outside air arrives on the front face of the condenser and is heated up through this one.

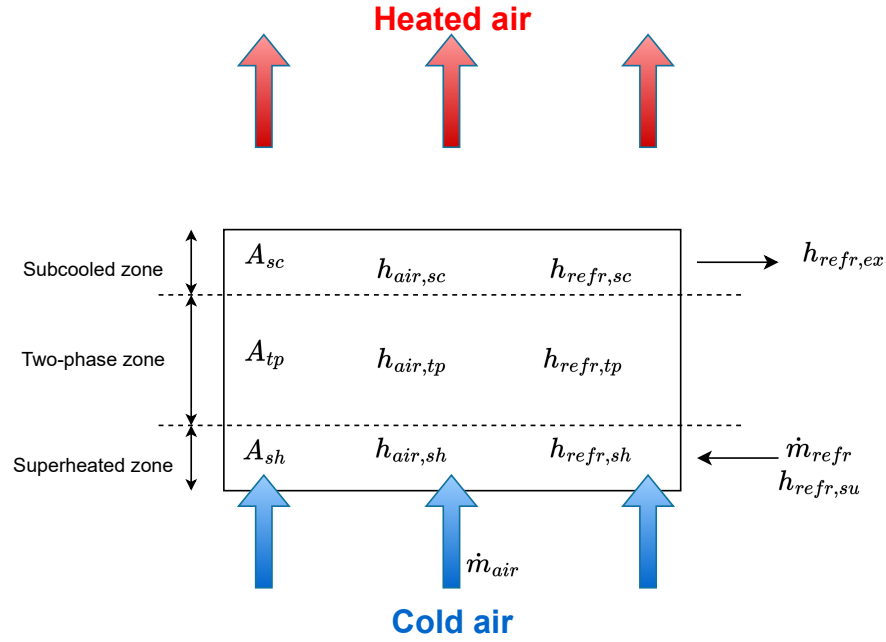


Figure 59: Three division zones of condensation

For each zone "i" , heat balances can be carried out on both refrigerant and air sides.

$$\dot{Q}_{cd,i} = \underbrace{\dot{m}_{refr} \times (h_{refr,ex,i} - h_{refr,su,i})}_{\text{refrigerant side}} = \underbrace{\dot{m}_{air} \times (h_{air,ex,i} - h_{air,su,i})}_{\text{air side}} \quad (23)$$

With  $i = \{sh ; tp ; sc\}$ .

The total condenser heat flow rate is the sum of the rate in each zone:

$$\dot{Q}_{cd} = \dot{Q}_{cd,sh} + \dot{Q}_{cd,tp} + \dot{Q}_{cd,sc} \quad (24)$$

### 2.2.5 Evaporator modelisation

In cooling mode, the evaporator is the internal exchanger HTX3 and in heating mode the external one HTX1. For the modelisation the evaporator is divided into two zones:

1. Two-phase zone
2. Superheated vapor zone

Fig.60 illustrates this two division zones. The hot outside air arrives in the front face of the evaporator and is cooled down through this one.

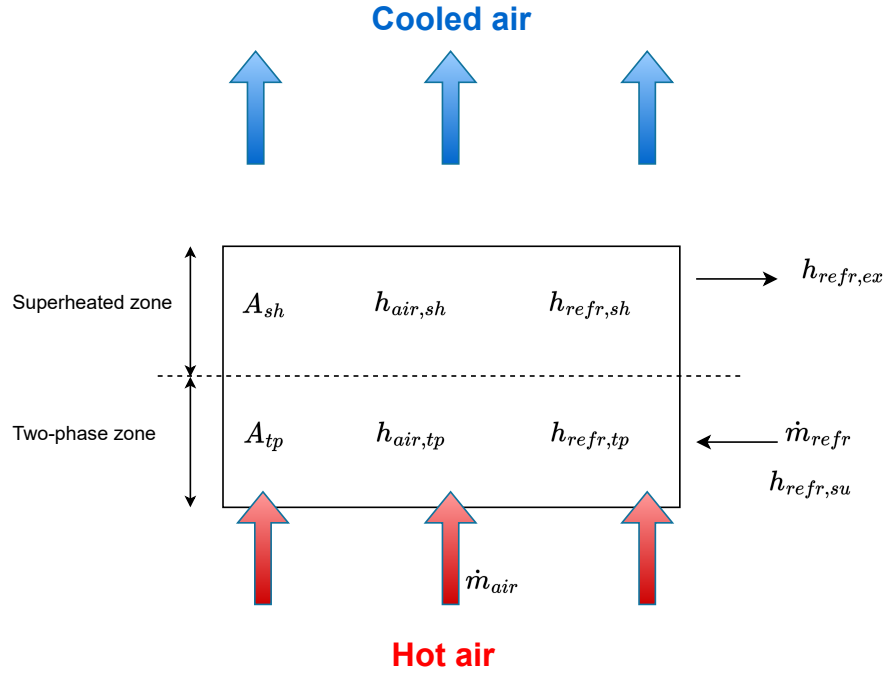


Figure 60: Two division zones of evaporation

For each zones "i" , heat balances can be carried out on both refrigerant and air sides.

$$\dot{Q}_{ev,i} = \underbrace{\dot{m}_{refr} \times (h_{refr,ex,i} - h_{refr,su,i})}_{\text{refrigerant side}} = \underbrace{\dot{m}_{air} \times (h_{air,ex,i} - h_{air,su,i})}_{\text{air side}} \quad (25)$$

With  $i = \{tp ; sh\}$ .

The total evaporator heat flow rate is the sum of the rate in each zone:

$$\dot{Q}_{ev} = \dot{Q}_{ev,tp} + \dot{Q}_{ev,sh} \quad (26)$$

## 2.3 Orifice tube

The orifice tube has been modeled as an ideal isenthalpic expansion valve. It takes as inputs the supply state of the valve, the exhaust pressure, and the supply refrigerant mass flow rate. The model returns as output the outlet refrigerant state.

## 2.4 Accumulator

The accumulator model used for this modelisation is the one developed by Wanga et al [1]. This model comes from a thesis on the modelisation and the experimental investigation of accumulators for automotive air conditioning systems. In this model, the mass flow is separated into several flows as can be seen in Fig.61. These streams are:

- **Stream 1:** two-phase flow stream from inlet pipe to vapor bulk in the container;
- **Stream 2:** main vapor flow stream through the j-tube;
- **Stream 3:** liquid stream flowing through the pinhole at the bottom of j-tube;
- **Stream 4:** vapor flow stream through the anti-siphon hole;
- **Stream 5:** liquid and vapor mixed flow stream;
- **Stream 6:** liquid and vapor mixed flow stream.

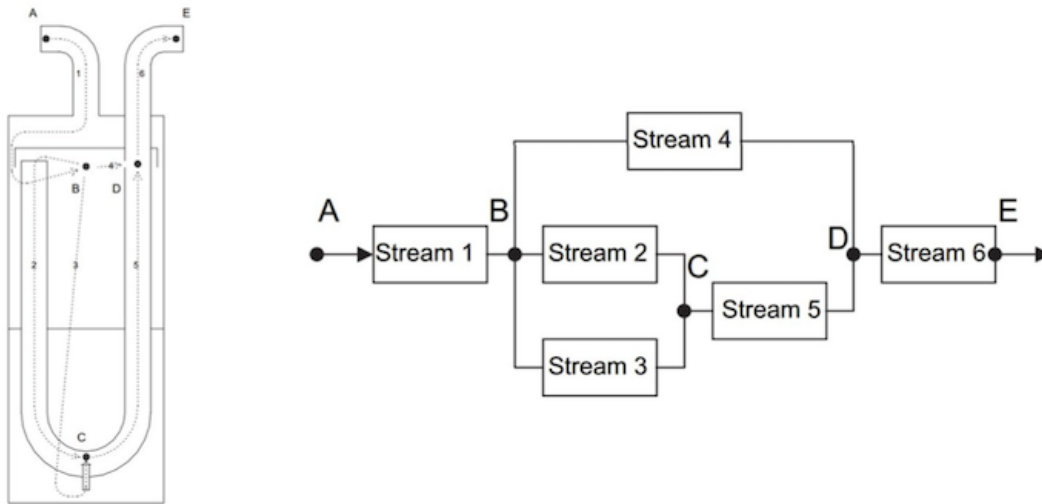


Figure 61: Model of the flow streams and pressure drop network inside the accumulator

### Assumptions

Some assumptions were made for the test bench model compared to the paper model:

- Streams are only composed of R1234yf refrigerant (oil is not taken into account);
- Stream 3 is not considered (pressure at point C and vapor-liquid equilibrium cannot be computed because the height of refrigerant in the accumulator is not known);



- Pressure drops due to sudden expansion and sudden contraction are not considered (Lack of information about the internal architecture of the accumulator who has a huge impact on the pressure drops);
- Stream 4 is not considered (expansion followed by a contraction which are not computed);
- Impact of the inlet vapor quality cannot be measured (sub-heating imposed by the evaporator induces that the flow entering the accumulator is fully vaporized).

### Characteristics

The Characteristics of the model are the following:

- $D_{pipe} = 1.6 \text{ cm}$
- $\rho = 1100 \text{ kg/m}^3$
- $L_{AB} = 5 \text{ cm}$
- $r_{bends} = 4.5 \text{ cm}$
- $L_{BC} = 20 \text{ cm}$
- $L_{CD} = L_{BC} \ \& \ L_{DE} = L_{AB}$

## 3 Reversible heat pump

Fig. 62 represents the schematic representation of the reversible heat pump. The red variables correspond to the inputs, these are chosen by the bench user. Those gray represent the parameters that have to be tuned. Finally, the blue variables are the outputs of the numerical model.

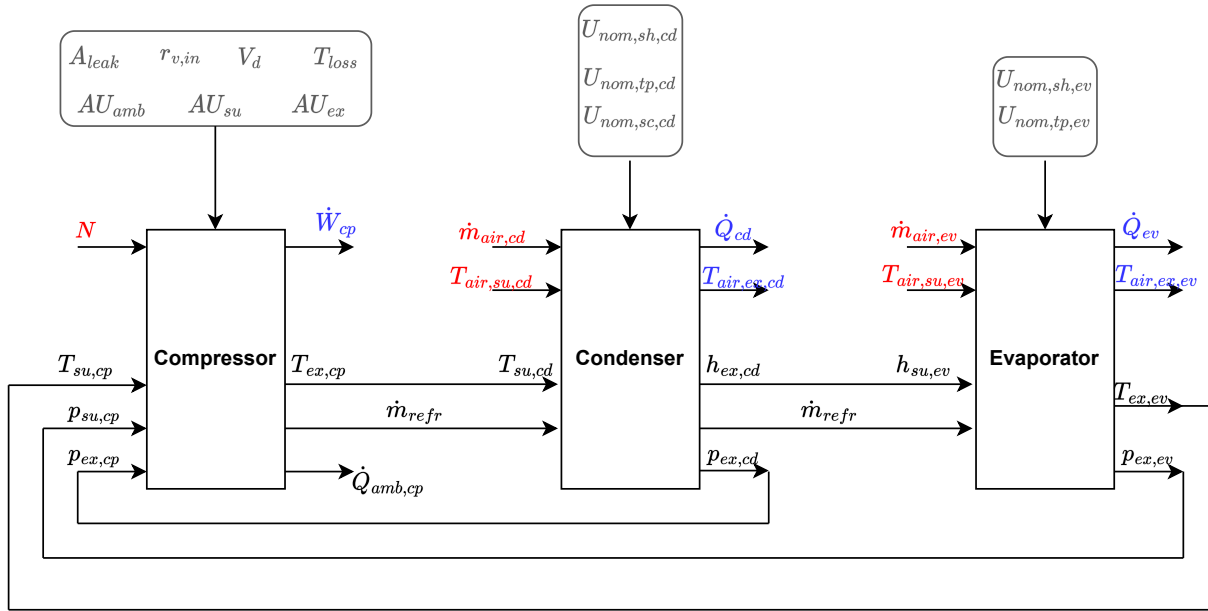


Figure 62: Reversible heat pump modelisation

## Part V

# Results validation

The aim of this section is to validate the numerical model with the data acquired during the experimental campaign. In order to do that, the parameters of the compressor, the condenser and the evaporator have to be determined.

## 1 Methodology

As can be seen in the schematic representation of the heat pump in Fig.62, the compressor model has 7 parameters ( $V_d, r_{v,in}, T_{loss}, A_{leak}, AU_{amb}, AU_{su}$  &  $AU_{ex}$ ). Only the displacement  $V_d$  is known and is set as a fixed variable. The goal is thus to determine the most suitable parameters in order to match the experimental and numerical values. To do that, the objective is to minimize the error between the measurements (experimental data) and the predictions of the model (numerical data). The numerical solution to this problem is to iterate different sets of parameters to minimize the following objective function:

$$g = \sqrt{\frac{\sum_{i=1}^n \left( \left( \frac{\dot{W}_{num} - \dot{W}_{exp}}{\dot{W}_{num}} \right)^2 + \left( \frac{\dot{m}_{num} - \dot{m}_{exp}}{\dot{m}_{num}} \right)^2 + \left( \frac{T_{ex,num} - T_{ex,exp}}{T_{ex,num}} \right)^2 \right)}{n}} \quad (27)$$

However, due to a lack of time, the model calibration has been manually performed. The determination of parameters by minimization of the objective function (Eq.27) can be carried out for future work on this test bench. The tuning of the parameters was performed manually on a specific operating point in cooling mode ( $T_{out} = 25^\circ C$  &  $N = 1000$  RPM).

## 2 Compressor

The tuned compressor parameters obtained are listed in Tab.11.

Table 11: Tuned compressor parameters

$r_{v,in}$	$T_{loss}$	$A_{leak}$	$AU_{amb}$	$AU_{su}$	$AU_{ex}$
2.5	0.5 Nm	$4 \times 10^{-7} m^2$	0.85 W/K	0.5 W/K	0.5 W/K

Fig.63 shown the comparison between numerical and experimental results for compressor consumption. The error on the compressor power measurement is calculated thanks to the uncertainty analysis (see Tab.5) and is represented on the graph. The maximum error between experimental work and prediction work is situated within a range of -10 to 10 %. The numerical model prediction of compressor consumption can be considered acceptable.

In fact, as shown in Fig. 63, the experimental values considered with their uncertainty still intersect the straight line of the graph.

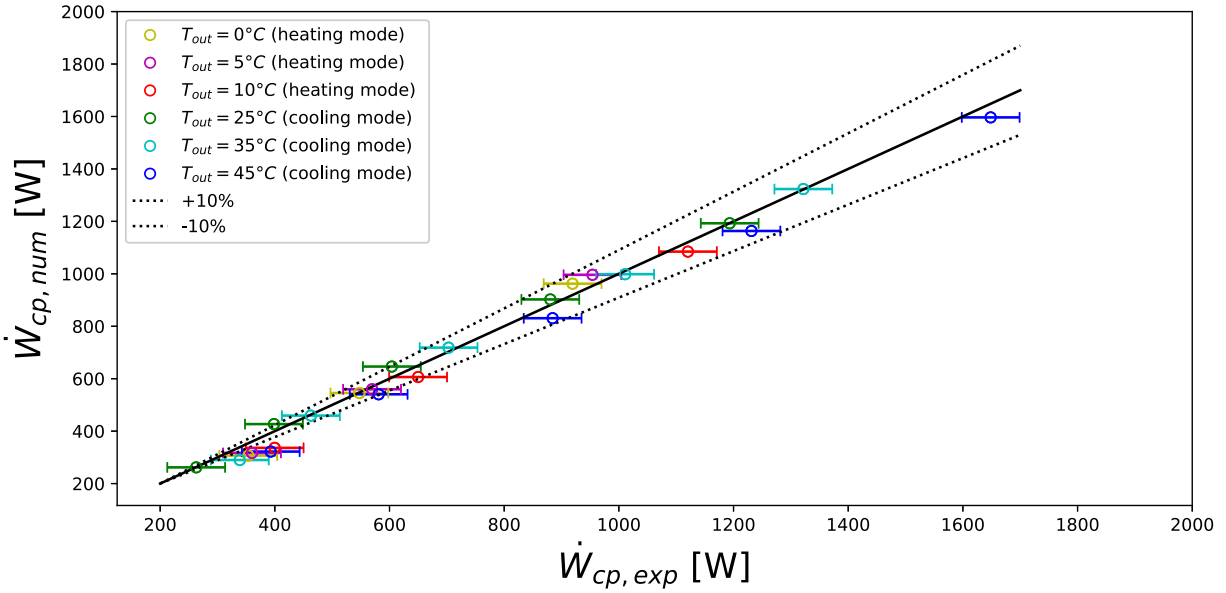


Figure 63: Comparison between numerical and experimental results for compressor power

The comparison between the refrigerant mass flow rate measured by the Coriolis flow meter and the one predicted by the numerical model is displayed in Fig. 64. The error between numerical and experimental results is situated in the area of 25 %. The compressor numerical model mainly overestimates the refrigerant mass flow rate in both modes. The compressor model's prediction of refrigerant flow is not accurate.

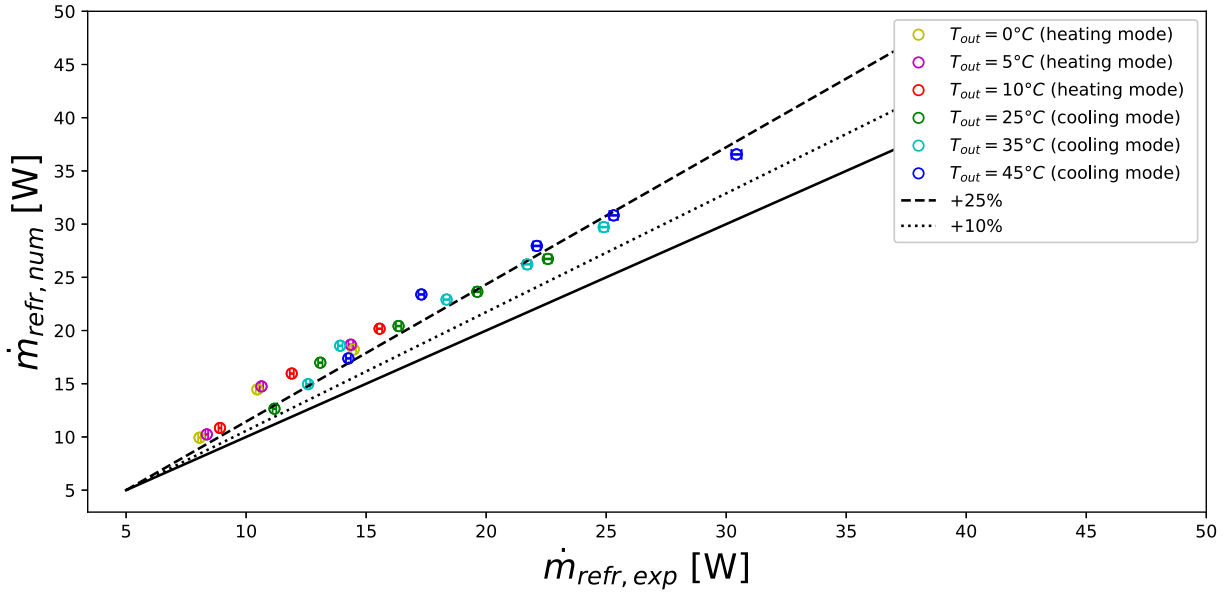


Figure 64: Comparison between numerical and experimental results for refrigerant mass flow rate

The compressor exhaust temperature predicted by the model as a function of that measured experimentally is shown in Fig. 65. Predictions on the exhaust temperature are really good. The maximal error stays in a range from -10% to 10%.

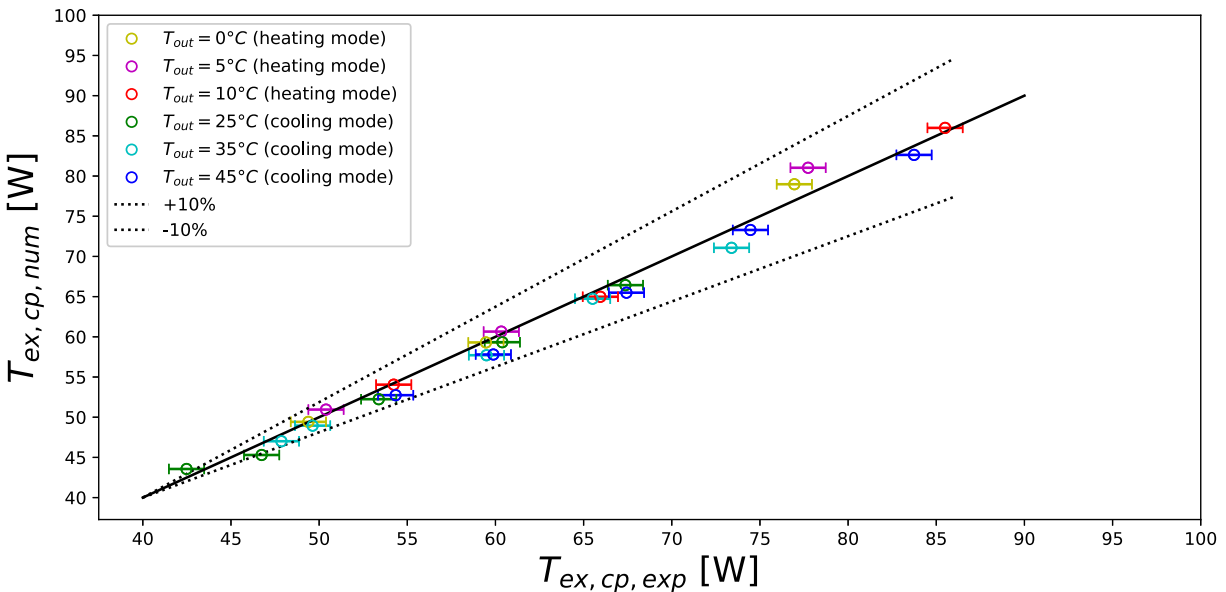


Figure 65: Comparison between numerical and experimental results for compressor exhaust temperature

### 3 Validation of the model

The nominal heat exchange coefficients will be manually tuned to calibrate the heat pump model. Tab.12 displayed the tuned heat transfer coefficient for a specific operating point in cooling mode ( $T_{out} = 25^\circ C$  &  $N = 1000$  RPM).

Table 12: Tuned exchangers parameters

Condenser			Evaporator	
$U_{nom,sh,cd}$	$U_{nom,tp,cd}$	$U_{nom,sc,cd}$	$U_{nom,tp,ev}$	$U_{nom,sh,ev}$
40 $W/m^2K$	400 $W/m^2K$	80 $W/m^2K$	400 $W/m^2K$	40 $W/m^2K$

The comparison between numerical and experimental results for the condenser heat flow rate is displayed in Fig.66. As a reminder, in cooling mode, the condenser is the external exchanger (HTX1) and in heating mode the internal one (HTX2). In this case, the predictions for the heating mode are better than for the cooling mode. In fact, as can be seen in Subfig.67b the maximal error between experimental and numerical data is situated within a range of -10 to 10 %. On the other hand, in cooling mode, this maximal error lies more within a range of -25 to 25 %.

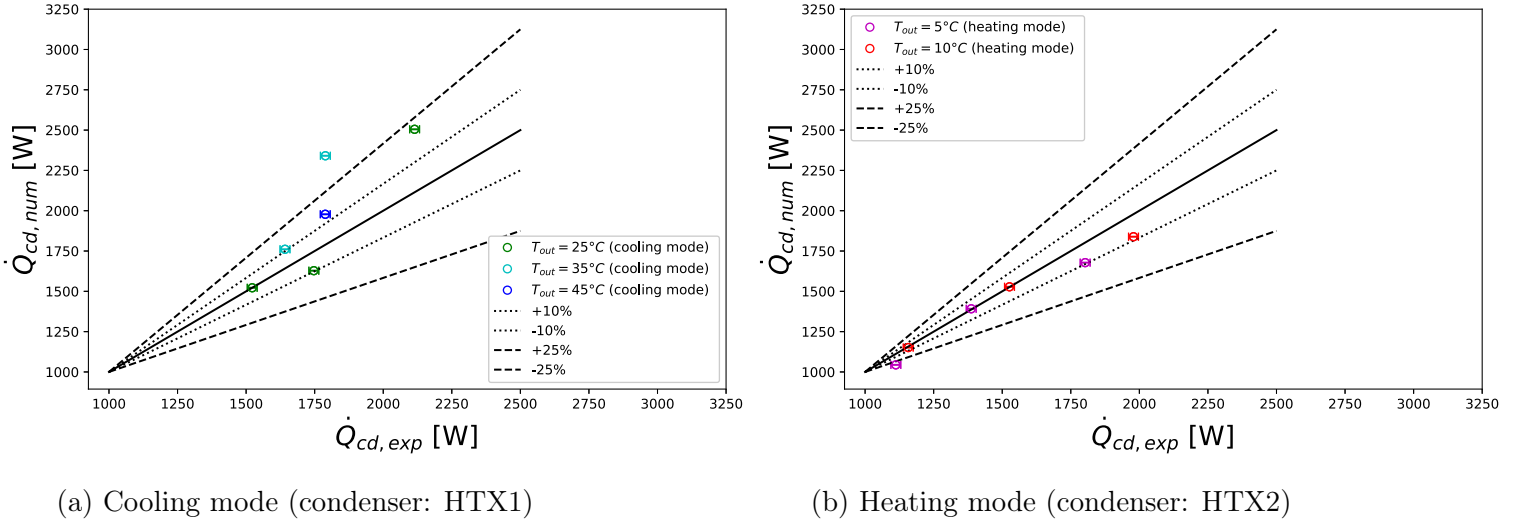
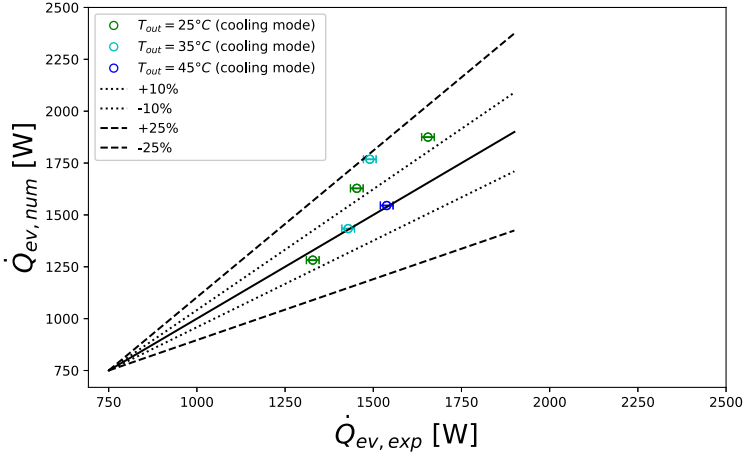
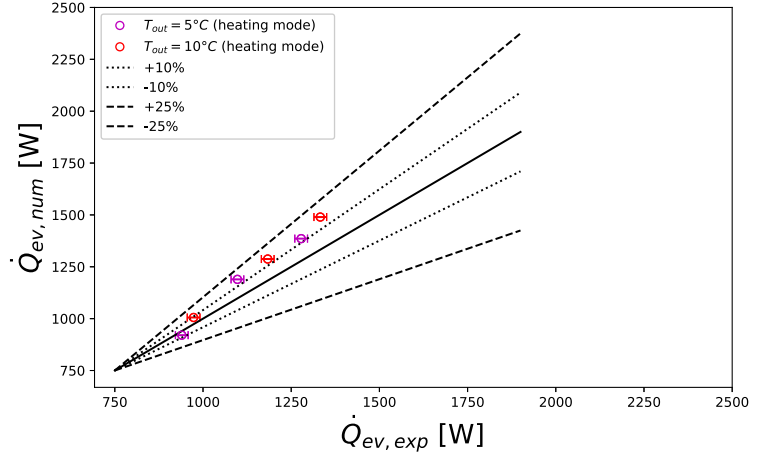


Figure 66: Comparison between numerical and experimental results for condenser heat flow rate

Fig.67 displayed the comparison between numerical and experimental results for the evaporator heat flow rate. As a reminder, in cooling mode, the evaporator is the internal exchanger (HTX3) and in heating mode the external one (HTX1). For both modes, predictions are overestimated. This can be partially explained by the fact that the compressor model overestimates the refrigerant mass flow rate.



(a) Cooling mode (evaporator: HTX3)



(b) Heating mode (evaporator: HTX1)

Figure 67: Comparison between numerical and experimental results for evaporator heat flow rate

## 4 Discussion and perspectives

The compressor model provides a good prediction of compressor consumption and exhaust temperature. However, it overestimates refrigerant mass flow rate. It might be interesting to improve this model by trying to make less simplifying assumptions. The reversible heat pump model gives poor results for the condenser and evaporator heat flow rates. For unknown reasons, the model does not run for certain input conditions. First of all, there seems to be a problem with the guess on the evaporation temperature. It might be interesting to explore the subject further. Once this compilation error has been corrected, the heat pump model can be validated. In order to do that, the predicted COP can be validated with the one determined experimentally.

---

## Part VI

# Conclusion

Despite some minor technical problems encountered during this master thesis realization, the main objectives have been achieved. As a reminder, the main objectives were:

1. Development of a numerical model for the reversible heat pump;
2. Realization of experimental campaigns in heating and cooling mode in order to assess the heat pump performance;
3. Validation of the numerical model with the experimental data obtained during the test campaigns.

Moreover, as described in Sec. 6, some modifications have been added on the test bench. The bench can now be considered fully operational and can be operated safely. However, it should be noted that the amount of oil present in the cycle is still unknown. Although the oil flow in the compressor has been estimated in the experimental part, the total quantity present in the cycle remains unknown. However, improvements can still be realized on the bench. Firstly, from an experimental point of view, it could be interesting to implement new solutions to measure air flows on exchangers accurately. Secondly, as discussed above, improvements can be added to the numerical models in order to improve the modelisation of the reversible heat pump. Finally, a more complete calibration model procedure can be carried out in order to correctly validate the numerical model.



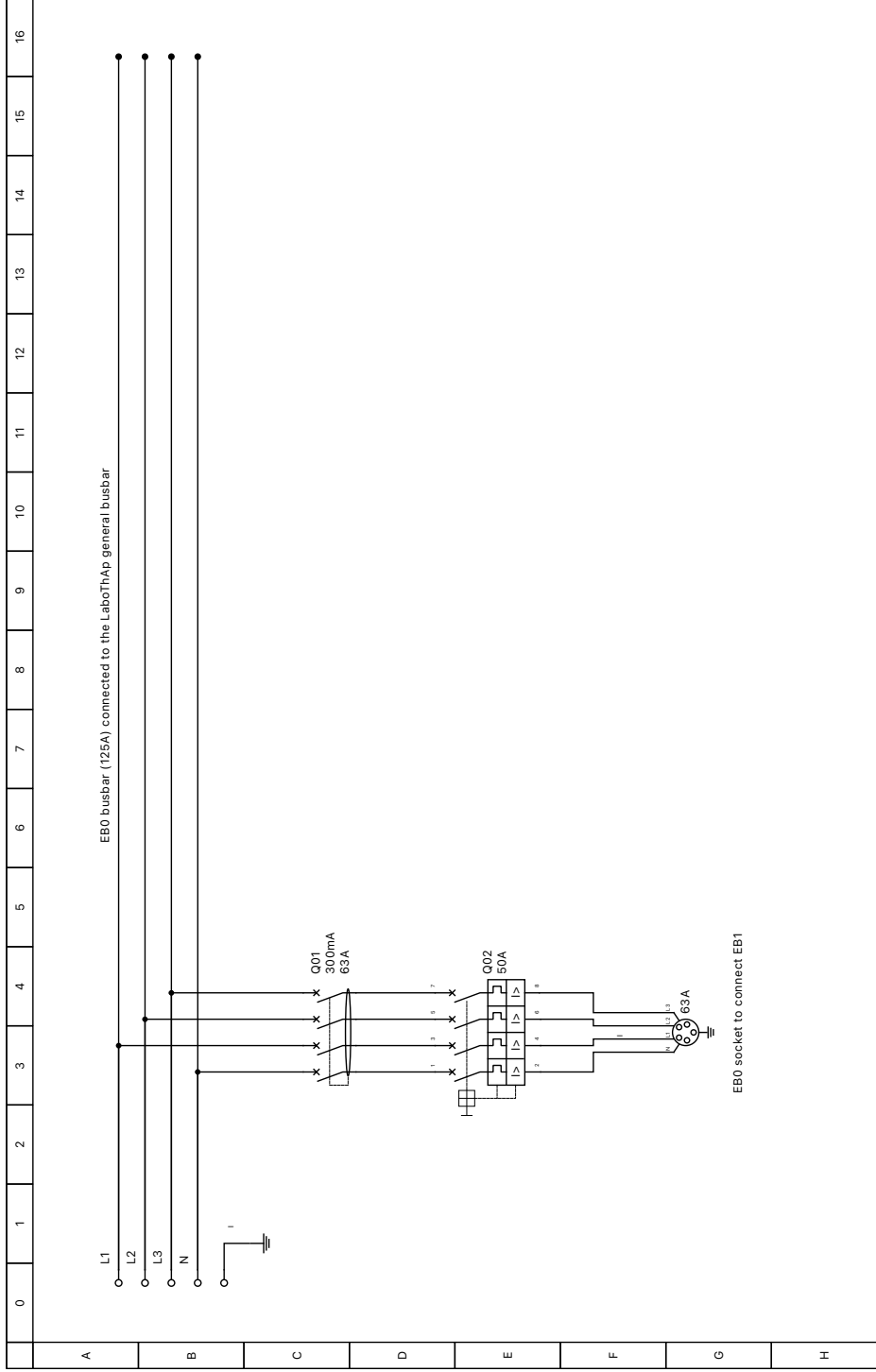
## References

- [1] Wanyong Li et al. “Modeling and experimental investigation of accumulators for automotive air conditioning systems”. English. 2021. URL: [https://www.sciencedirect.com/science/article/pii/S0140700706000636?fr=RR-2&ref=pdf\\_download&rr=70b3d652cb4083c6](https://www.sciencedirect.com/science/article/pii/S0140700706000636?fr=RR-2&ref=pdf_download&rr=70b3d652cb4083c6).
- [2] A. Z. Amran et al. “Development and evaluation of an automotive air-conditioning test rig”. English (US). In: Jurnal Teknologi Volume 78, No 10-2 (2016).
- [3] B.Chaudoir, B.Fantoli, and B.Himbert. “Energetics Integrated Project: Hardware part”. English. 2022.
- [4] Ian H. Bell et al. “Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp”. In: Industrial & Engineering Chemistry Research 53.6 (2014), pp. 2498–2508. DOI: [10.1021/ie4033999](https://doi.org/10.1021/ie4033999), eprint: <http://pubs.acs.org/doi/pdf/10.1021/ie4033999>. URL: <http://pubs.acs.org/doi/abs/10.1021/ie4033999>.
- [5] C.Aubourg. “Automotive air conditioning unit for electrical vehicle”. English. 2022.
- [6] M. Hosoz and M. Direk. “Performance evaluation of an integrated automotive air-conditioning and heat pump system”. English (US). In: Energy Conversion and Management Volume 47, Issue 5- (2005), pp. 545–559.
- [7] M.Lemarchand, B.Van Geel, and A.Waltregny-Dengis. “Energetics Integrated Project: Educational part”. English. 2022.
- [8] M.Messens, T.Rulot, and N.Wauthier. “Energetics Integrated Project: Numerical part”. English. 2022.
- [9] M.Peeters et al. “Renault ZOE test bench: User guide”. English. 2022.
- [10] N.Hustinx et al. “Energetics Integrated Project: Auxiliary part”. English. 2022.
- [11] P.Depaepe et al. “Energetics Integrated Project: Measurement Technique part”. English. 2022.
- [12] T.Gillet. “Étude expérimentale et modélisation numérique d’un système de climatisation multi-évaporateurs pour véhicule électrifié”. French. 2018.
- [13] T.Gillet. “RENAULT ZOE: POMPE A CHALEUR”. Présentation à l’université de Liège. 2022.
- [14] V.Lemort. Machines et systèmes thermiques (MECA0006-1). <https://www.programmes.uliege.be/cocoon/20222023/cours/MECA0006-1.html>.
- [15] V.Lemort. Thermodynamique appliqué et introduction aux machines thermiques (MECA0002-1). <https://www.programmes.uliege.be/cocoon/20222023/cours/MECA0002-1.html>.

# Annexes A

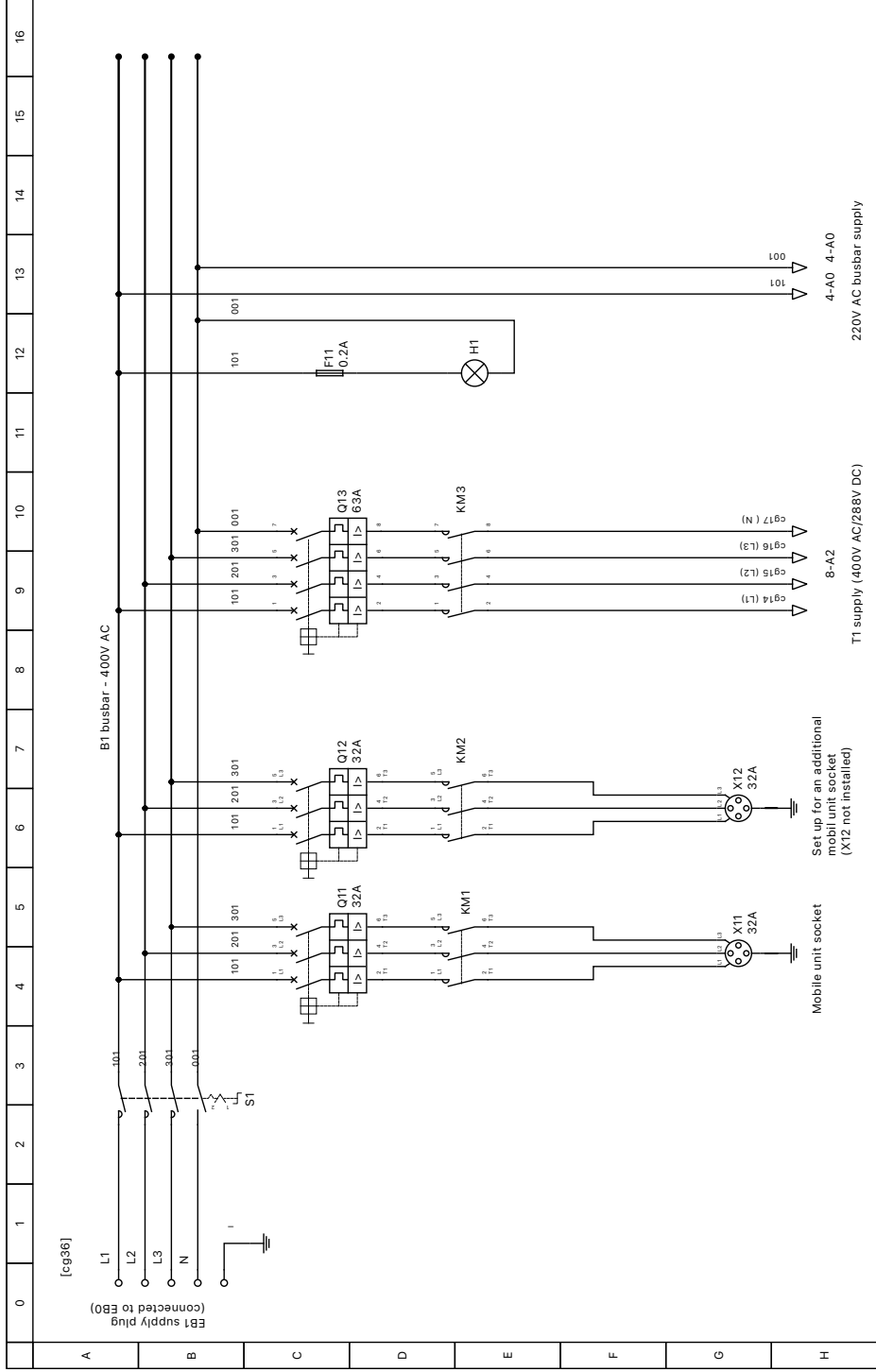
## Electrical diagram

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16																																																																																																																					
A	<p><b>Energetics Integrated Project 2022</b></p> <p><b>Reversible Heat pump Bench - Renault Zoe</b></p>																																																																																																																																					
B																																																																																																																																						
C																																																																																																																																						
D																																																																																																																																						
E																																																																																																																																						
F																																																																																																																																						
G																																																																																																																																						
H																																																																																																																																						
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%; vertical-align: top;"> <p><b>Components no-standard Nomenclature</b></p> <p>b = Phoenix contact terminal block</p> <p>B = busbar</p> <p>cg = cable gland ("presse-étoupe")</p> <p>ESB = Emergency Stop Button</p> <p>"Fx" using in the diagram, but only "x" on the components (Wago fusibles) in the box</p> </td> <td style="width: 33%; vertical-align: top;"> <p><b>Wires: nomenclature [XYZ]</b></p> <p>X = voltage</p> <p>0 - N</p> <p>1 - L1</p> <p>2 - L2</p> <p>3 - L3</p> <p>4 - 24V DC</p> <p>5 - 0V (24V DC)</p> <p>YZ = n°</p> </td> <td style="width: 33%; vertical-align: top;"> <p><b>Wires: color code</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>L1 (400V AC)</td><td>Black</td></tr> <tr><td>L1 (220V AC)</td><td>Black</td></tr> <tr><td>L2 (400V)</td><td>Black + brown</td></tr> <tr><td>L3 (400V)</td><td>Brown</td></tr> <tr><td>Neutral</td><td>Blue</td></tr> <tr><td>Ground</td><td>Green + yellow</td></tr> <tr><td>400V DC</td><td>Grey + orange</td></tr> <tr><td>0V (440V DC)</td><td>Grey</td></tr> <tr><td>24V DC</td><td>Purple</td></tr> <tr><td>0V (24V DC)</td><td>White</td></tr> <tr><td>12V DC + 0V</td><td>Red/white twisted</td></tr> <tr><td>Security (24V)</td><td>Orange</td></tr> </table> </td> </tr> <tr> <td colspan="18" style="text-align: right;"> <p>REV : 2.1      Sheet : 1/8</p> <p>Date : 20/06/22</p> </td> </tr> <tr> <td colspan="18" style="text-align: right;"> <p>Draw by : Thomé Olivier</p> </td> </tr> <tr> <td colspan="18" style="text-align: center;"> <p><b>LIÈGE</b> université</p> </td> </tr> <tr> <td colspan="18" style="text-align: center;"> <p><b>TITLE : Project Informations</b></p> </td> </tr> <tr> <td colspan="18" style="text-align: center;"> <p>Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe</p> </td> </tr> </table>																		<p><b>Components no-standard Nomenclature</b></p> <p>b = Phoenix contact terminal block</p> <p>B = busbar</p> <p>cg = cable gland ("presse-étoupe")</p> <p>ESB = Emergency Stop Button</p> <p>"Fx" using in the diagram, but only "x" on the components (Wago fusibles) in the box</p>	<p><b>Wires: nomenclature [XYZ]</b></p> <p>X = voltage</p> <p>0 - N</p> <p>1 - L1</p> <p>2 - L2</p> <p>3 - L3</p> <p>4 - 24V DC</p> <p>5 - 0V (24V DC)</p> <p>YZ = n°</p>	<p><b>Wires: color code</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>L1 (400V AC)</td><td>Black</td></tr> <tr><td>L1 (220V AC)</td><td>Black</td></tr> <tr><td>L2 (400V)</td><td>Black + brown</td></tr> <tr><td>L3 (400V)</td><td>Brown</td></tr> <tr><td>Neutral</td><td>Blue</td></tr> <tr><td>Ground</td><td>Green + yellow</td></tr> <tr><td>400V DC</td><td>Grey + orange</td></tr> <tr><td>0V (440V DC)</td><td>Grey</td></tr> <tr><td>24V DC</td><td>Purple</td></tr> <tr><td>0V (24V DC)</td><td>White</td></tr> <tr><td>12V DC + 0V</td><td>Red/white twisted</td></tr> <tr><td>Security (24V)</td><td>Orange</td></tr> </table>	L1 (400V AC)	Black	L1 (220V AC)	Black	L2 (400V)	Black + brown	L3 (400V)	Brown	Neutral	Blue	Ground	Green + yellow	400V DC	Grey + orange	0V (440V DC)	Grey	24V DC	Purple	0V (24V DC)	White	12V DC + 0V	Red/white twisted	Security (24V)	Orange	<p>REV : 2.1      Sheet : 1/8</p> <p>Date : 20/06/22</p>																		<p>Draw by : Thomé Olivier</p>																		<p><b>LIÈGE</b> université</p>																		<p><b>TITLE : Project Informations</b></p>																		<p>Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe</p>																	
<p><b>Components no-standard Nomenclature</b></p> <p>b = Phoenix contact terminal block</p> <p>B = busbar</p> <p>cg = cable gland ("presse-étoupe")</p> <p>ESB = Emergency Stop Button</p> <p>"Fx" using in the diagram, but only "x" on the components (Wago fusibles) in the box</p>	<p><b>Wires: nomenclature [XYZ]</b></p> <p>X = voltage</p> <p>0 - N</p> <p>1 - L1</p> <p>2 - L2</p> <p>3 - L3</p> <p>4 - 24V DC</p> <p>5 - 0V (24V DC)</p> <p>YZ = n°</p>	<p><b>Wires: color code</b></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>L1 (400V AC)</td><td>Black</td></tr> <tr><td>L1 (220V AC)</td><td>Black</td></tr> <tr><td>L2 (400V)</td><td>Black + brown</td></tr> <tr><td>L3 (400V)</td><td>Brown</td></tr> <tr><td>Neutral</td><td>Blue</td></tr> <tr><td>Ground</td><td>Green + yellow</td></tr> <tr><td>400V DC</td><td>Grey + orange</td></tr> <tr><td>0V (440V DC)</td><td>Grey</td></tr> <tr><td>24V DC</td><td>Purple</td></tr> <tr><td>0V (24V DC)</td><td>White</td></tr> <tr><td>12V DC + 0V</td><td>Red/white twisted</td></tr> <tr><td>Security (24V)</td><td>Orange</td></tr> </table>	L1 (400V AC)	Black	L1 (220V AC)	Black	L2 (400V)	Black + brown	L3 (400V)	Brown	Neutral	Blue	Ground	Green + yellow	400V DC	Grey + orange	0V (440V DC)	Grey	24V DC	Purple	0V (24V DC)	White	12V DC + 0V	Red/white twisted	Security (24V)	Orange																																																																																																												
L1 (400V AC)	Black																																																																																																																																					
L1 (220V AC)	Black																																																																																																																																					
L2 (400V)	Black + brown																																																																																																																																					
L3 (400V)	Brown																																																																																																																																					
Neutral	Blue																																																																																																																																					
Ground	Green + yellow																																																																																																																																					
400V DC	Grey + orange																																																																																																																																					
0V (440V DC)	Grey																																																																																																																																					
24V DC	Purple																																																																																																																																					
0V (24V DC)	White																																																																																																																																					
12V DC + 0V	Red/white twisted																																																																																																																																					
Security (24V)	Orange																																																																																																																																					
<p>REV : 2.1      Sheet : 1/8</p> <p>Date : 20/06/22</p>																																																																																																																																						
<p>Draw by : Thomé Olivier</p>																																																																																																																																						
<p><b>LIÈGE</b> université</p>																																																																																																																																						
<p><b>TITLE : Project Informations</b></p>																																																																																																																																						
<p>Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe</p>																																																																																																																																						



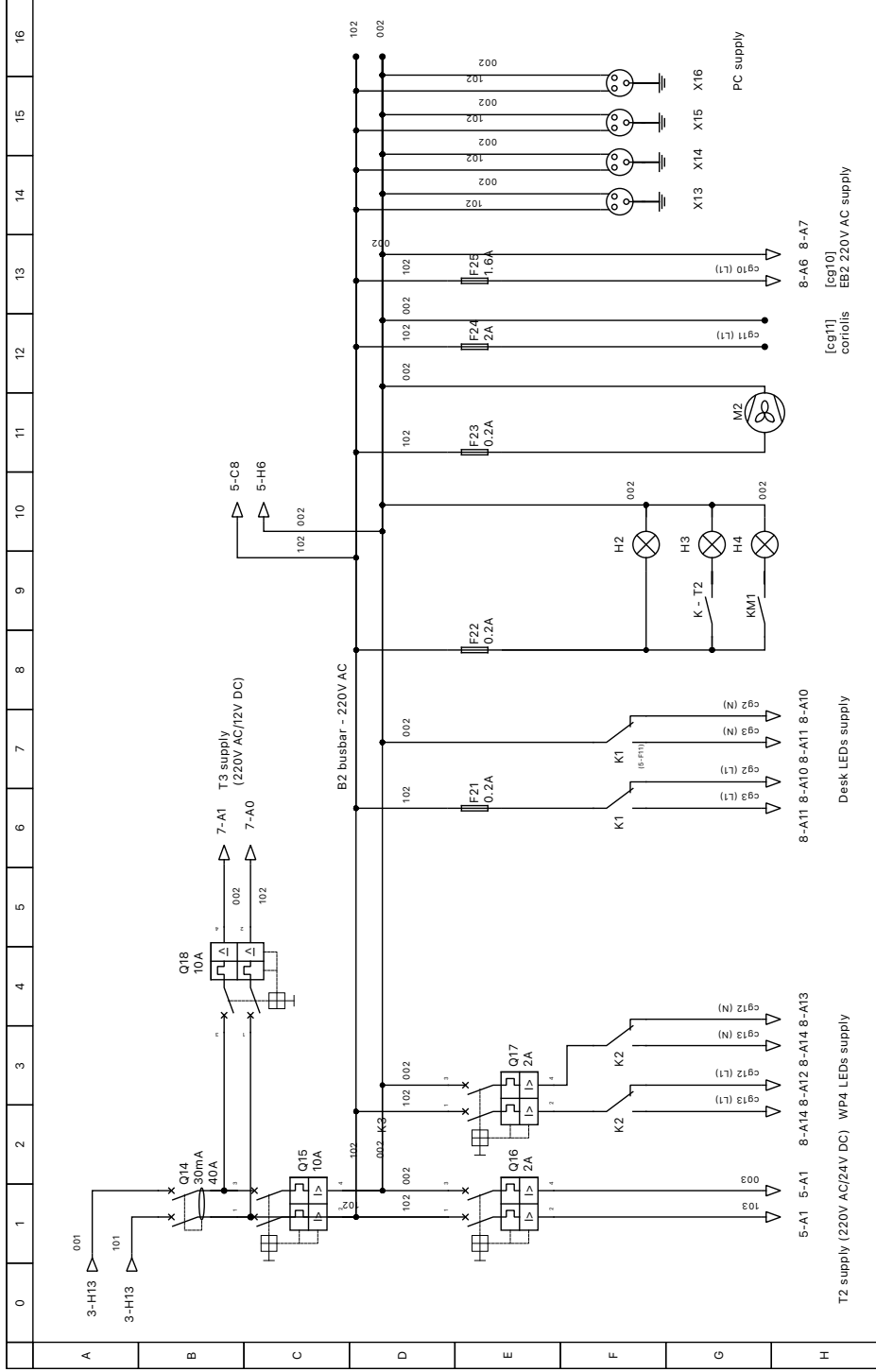
REV : 2.1		Sheet : 2/8
Date : 20/06/22		
TITLE : EBO		
Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe		Draw by : Thomé Olivier




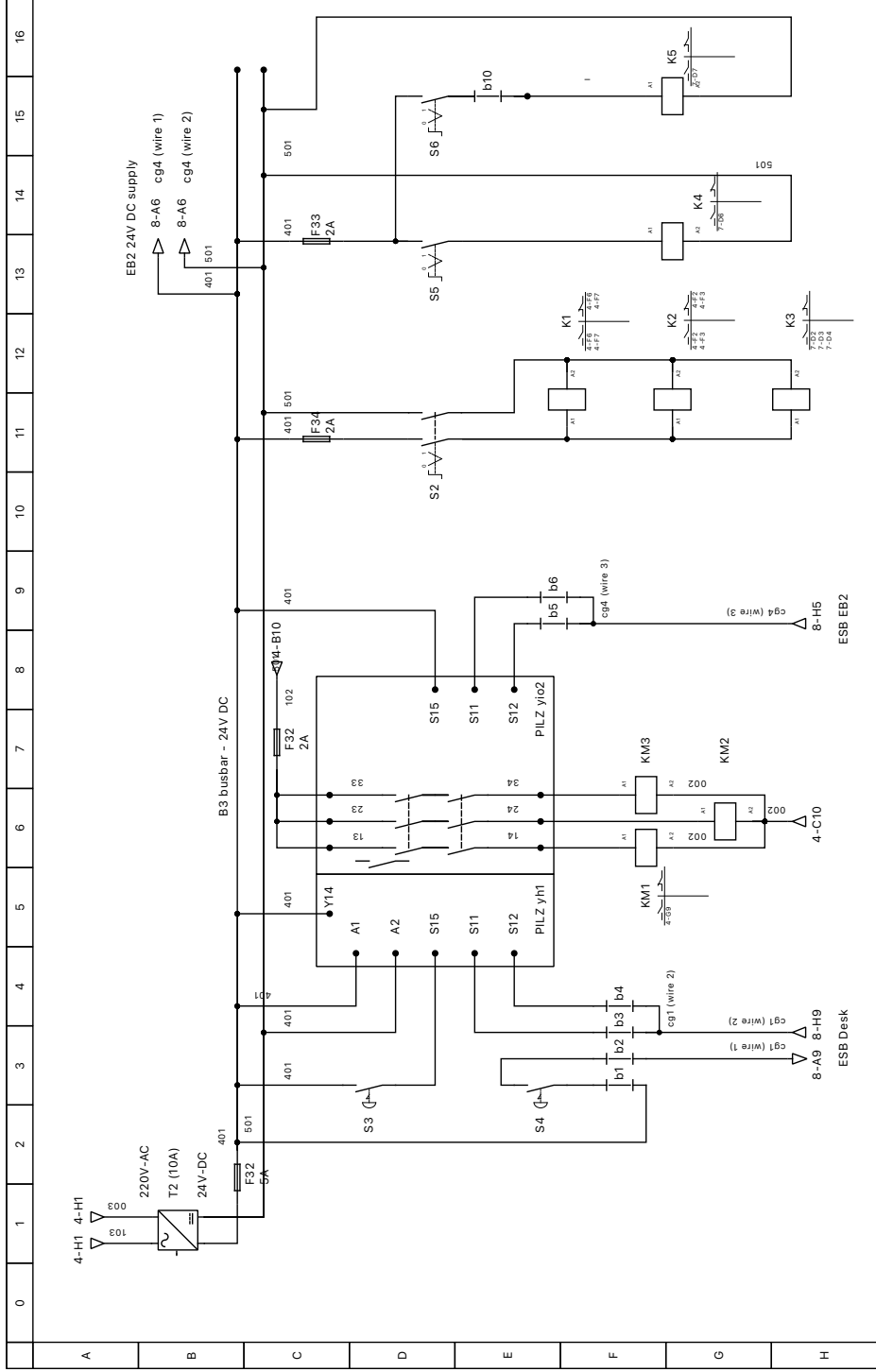


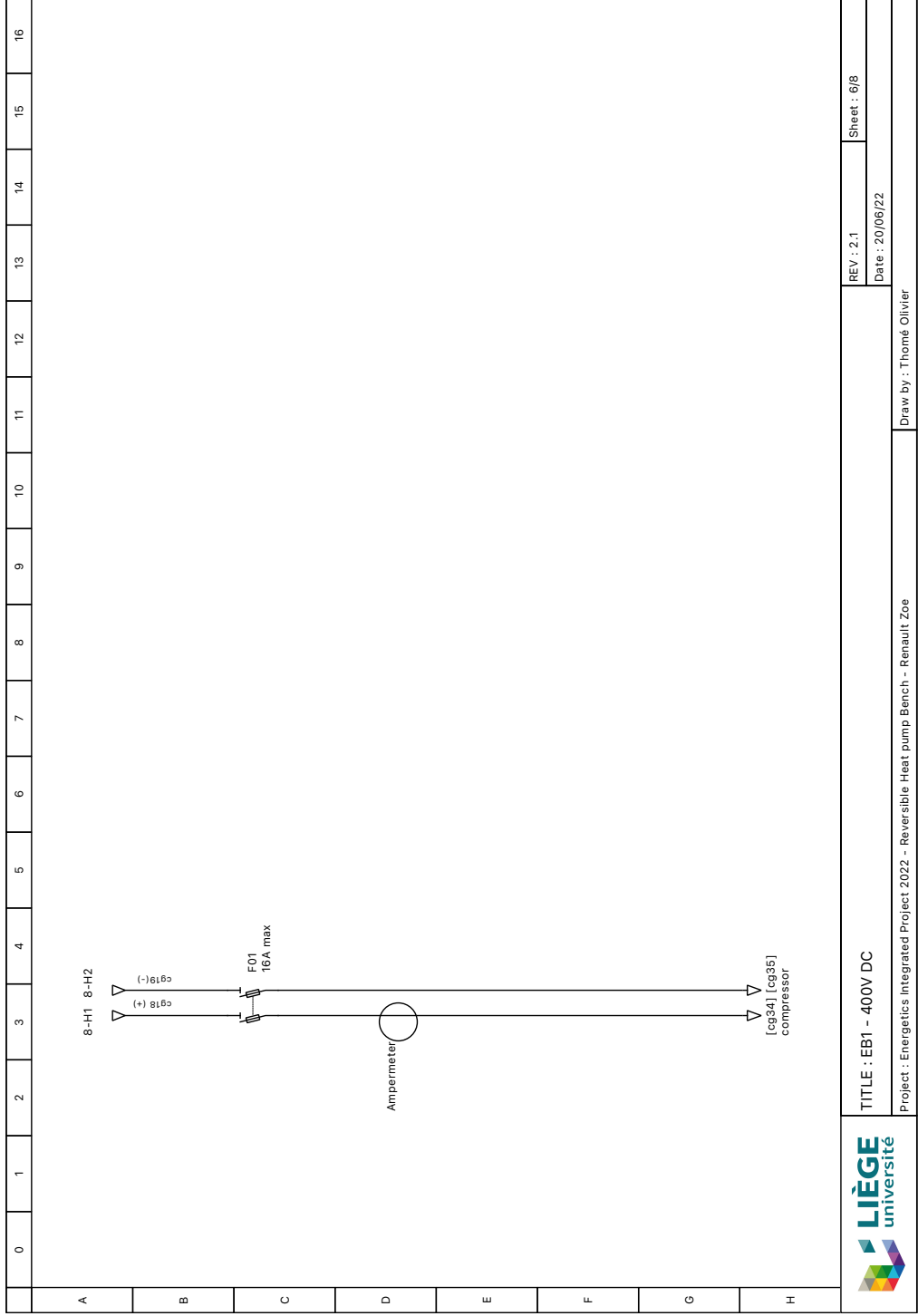
[c936]		B1 busbar - 400V AC		220V AC busbar supply		REV : 2.1		Sheet : 3/8	
E81 supply plug (connected to E80)		T1 supply (400V AC/288V DC)		8-A2		Date : 20/06/22		Draw by : Thomé Olivier	
Mobile unit socket		Set up for an additional mobile unit socket (X12 not installed)		4-AO 4-AO					
TITLE : EB1 - 400V AC		Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe							

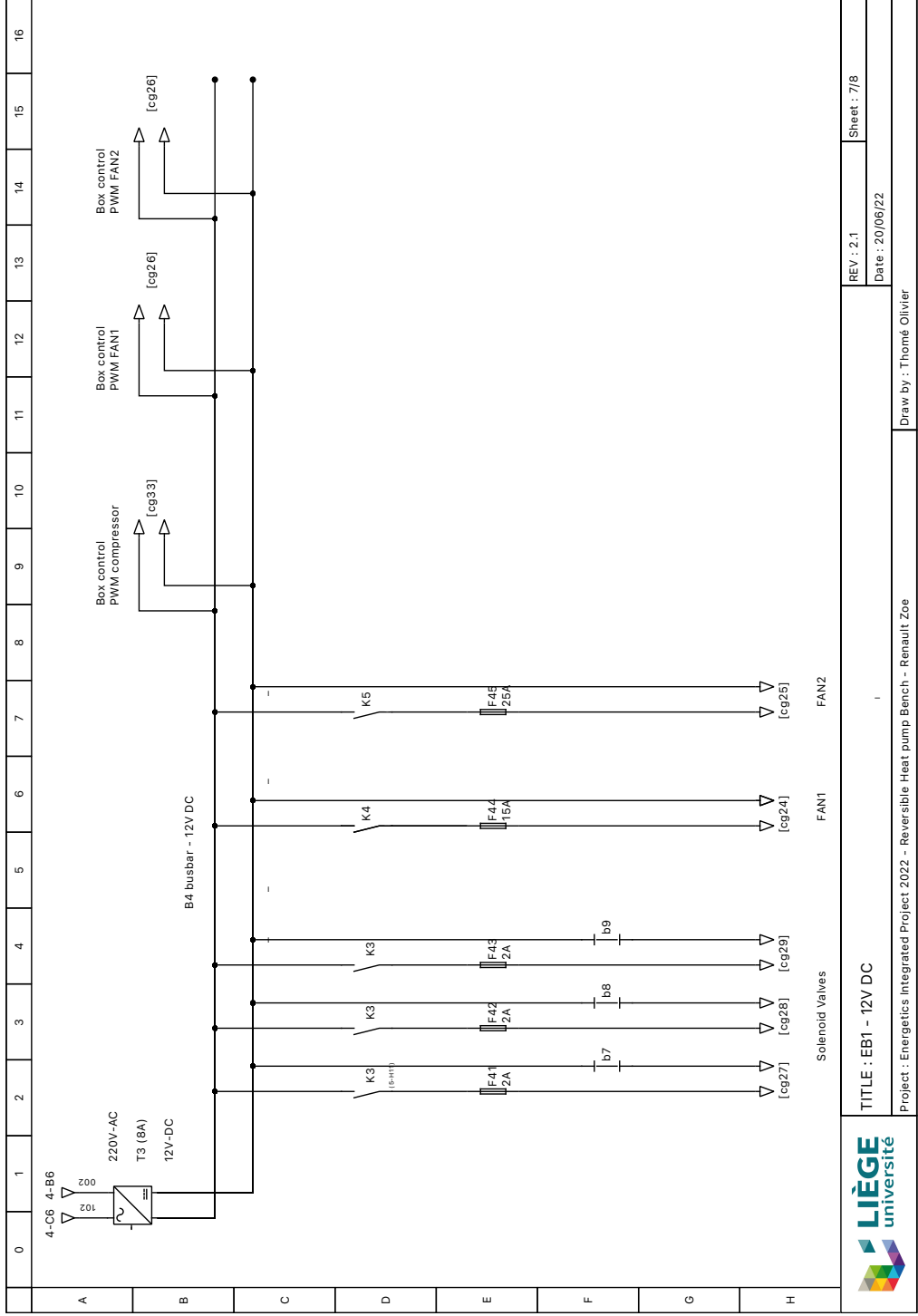




		<b>TITLE : EB1 - 220V AC</b>		Sheet : 4/8	
Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe				Date : 20/06/22	
Draw by : Thomé Olivier				REV : 2.1	







TITLE : EB1 - 12V DC

Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe

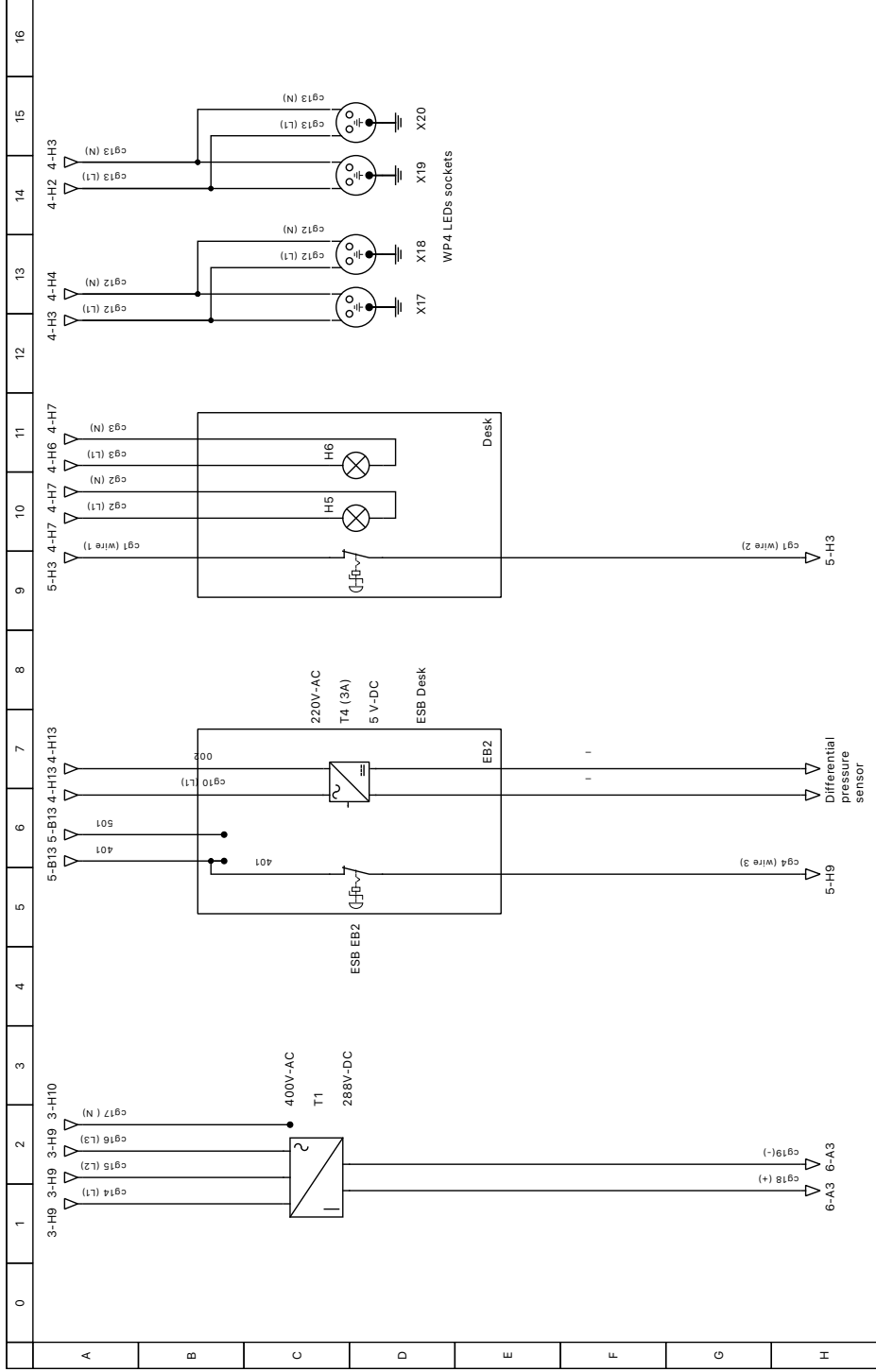
Draw by : Thomé Olivier


REV : 2.1

Date : 20/06/22

Sheet : 7/8



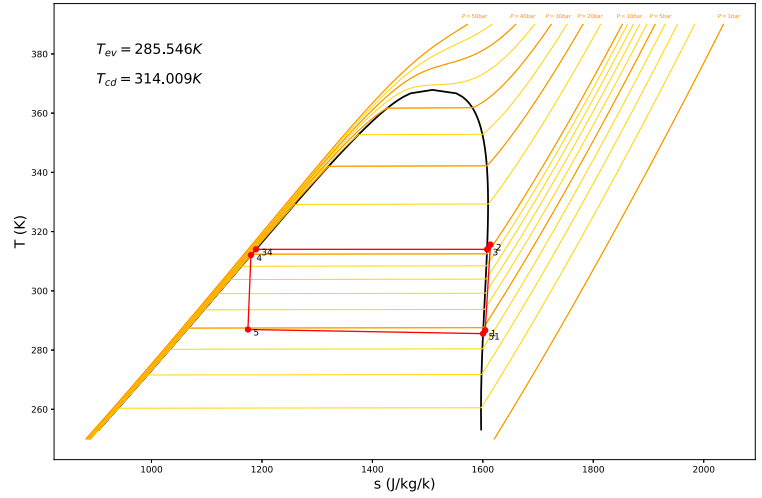
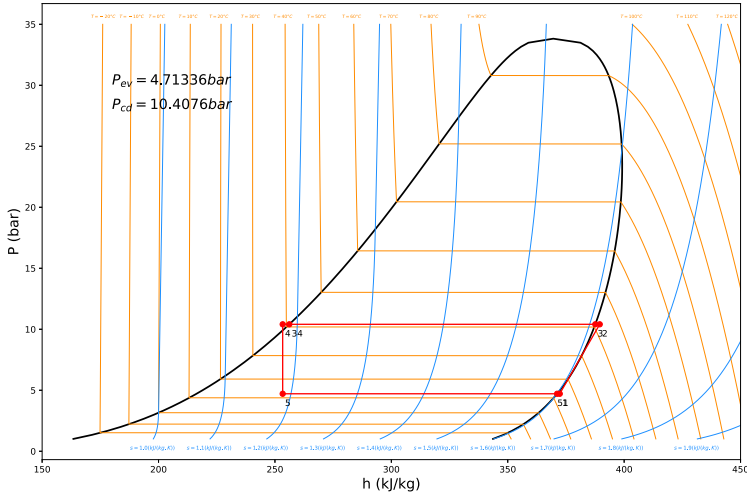


	<b>TITLE : External Components</b>	
	Project : Energetics Integrated Project 2022 - Reversible Heat pump Bench - Renault Zoe	
REV : 2.1	Sheet : 8/8	Date : 20/06/22
Draw by : Thomé Olivier		

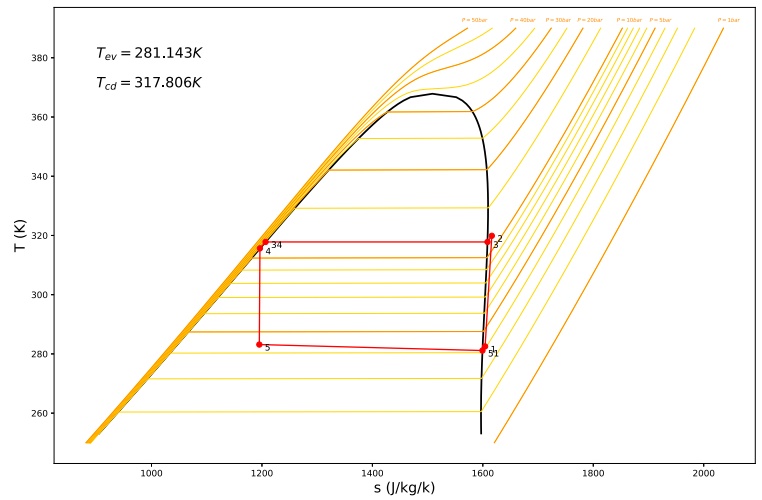
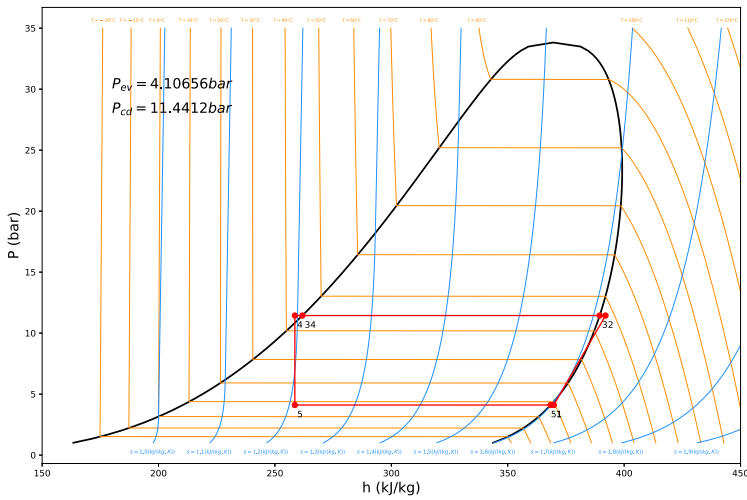
# Annexes B

## Ph and TS diagram: cooling mode

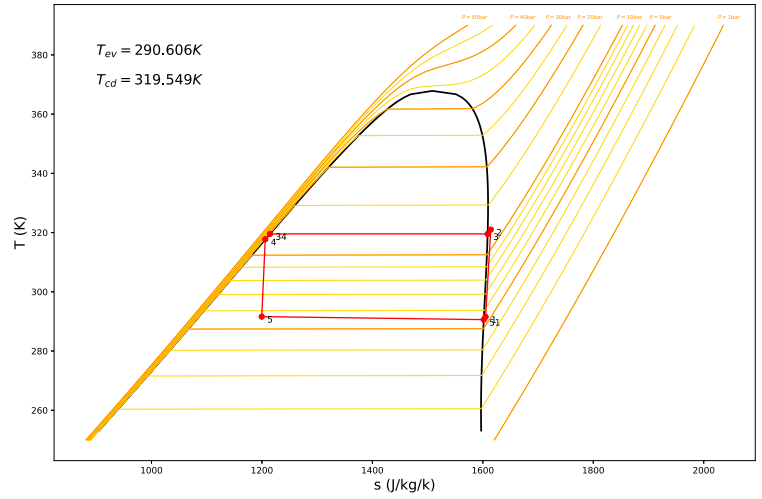
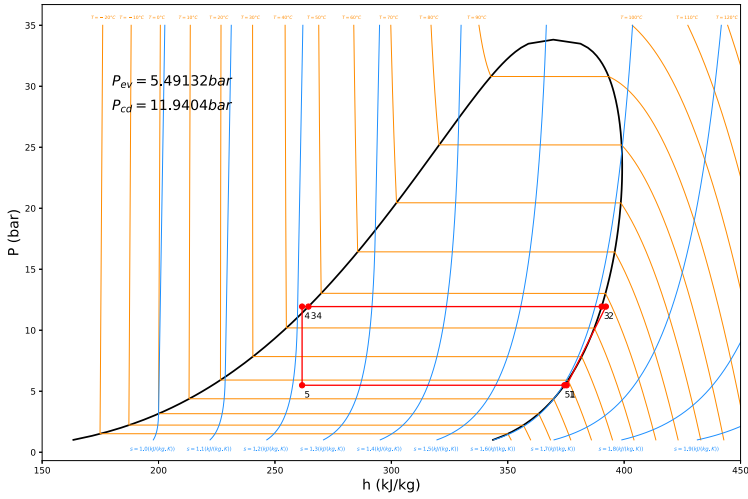
$T_{out} = 25^{\circ}\text{C}$  &  $N = 1000$  RPM



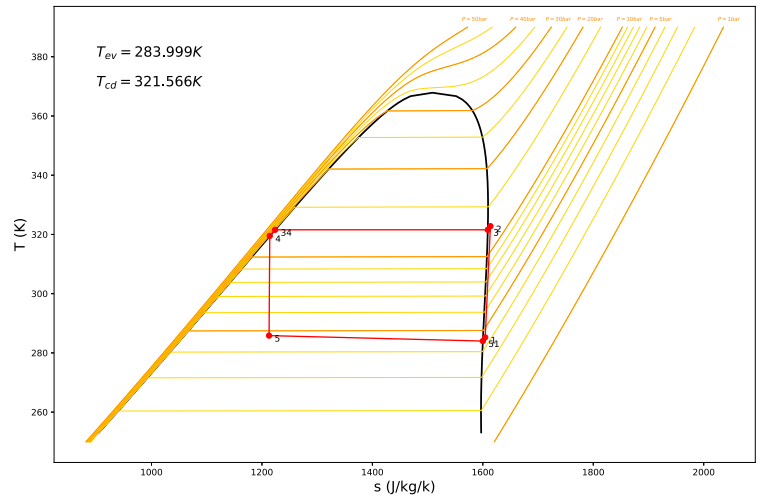
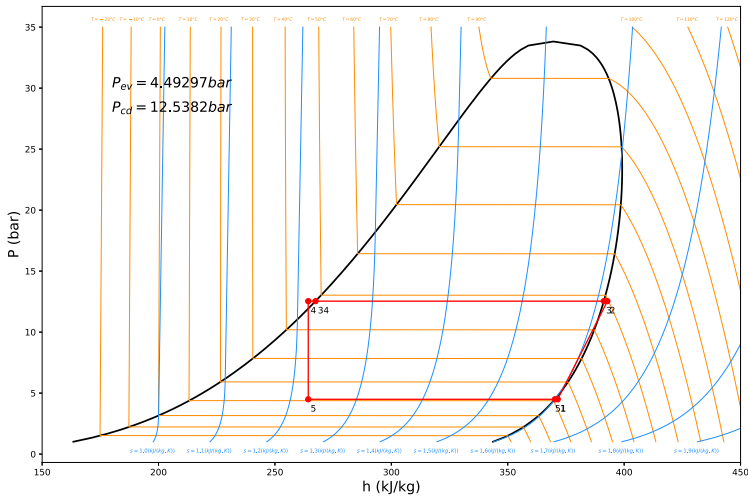
$T_{out} = 25^{\circ}\text{C}$  &  $N = 1475$  RPM



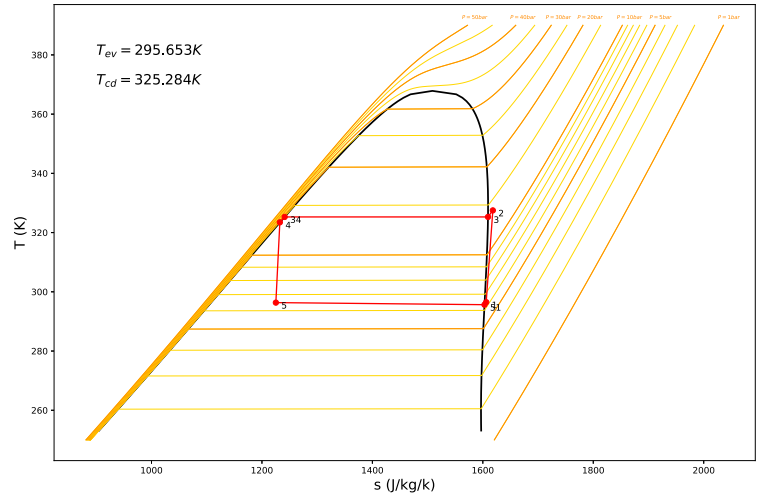
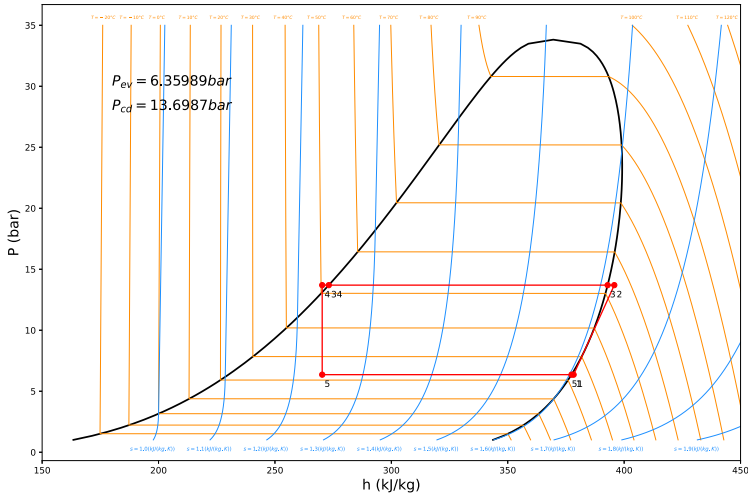
**$T_{out} = 35^{\circ}\text{C}$  &  $N = 1000$  RPM**



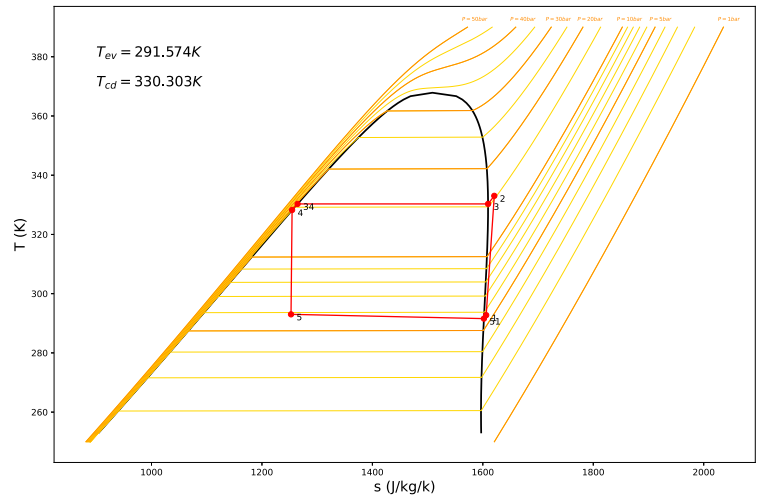
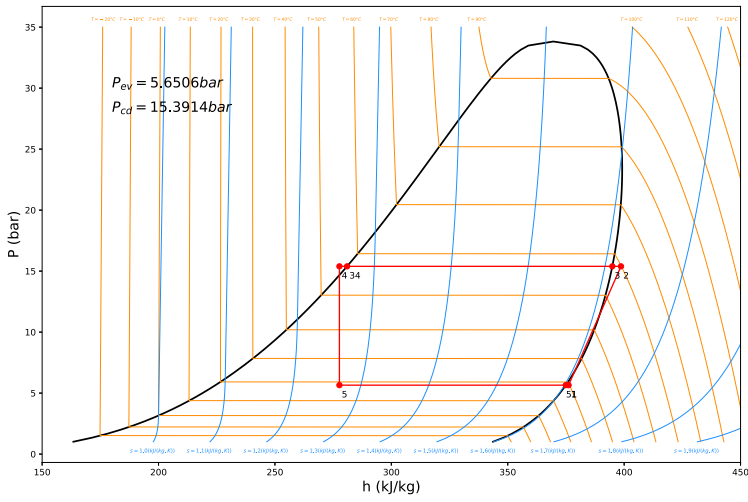
**$T_{out} = 35^{\circ}\text{C}$  &  $N = 1475$  RPM**



$T_{out} = 45^\circ\text{C}$  &  $N = 1000$  RPM

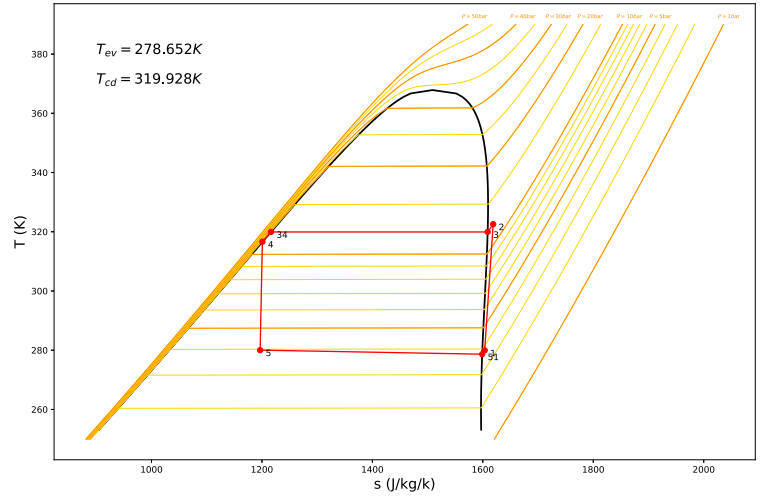
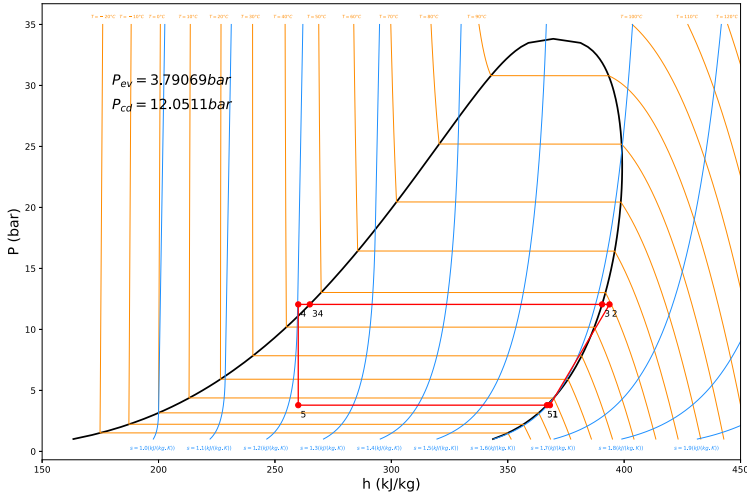


$T_{out} = 45^\circ\text{C}$  &  $N = 1475$  RPM

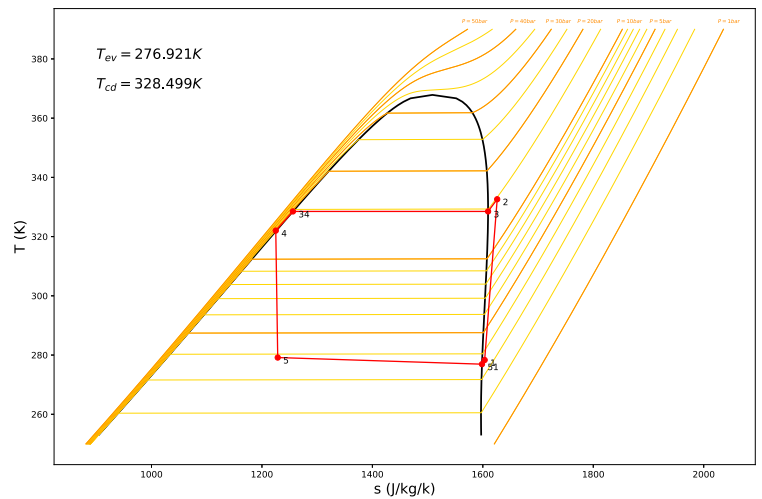
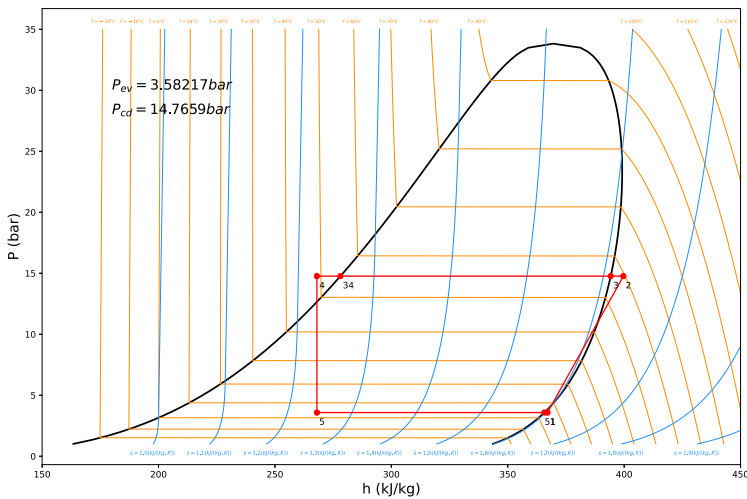


# P-h and T-S diagram: heating mode

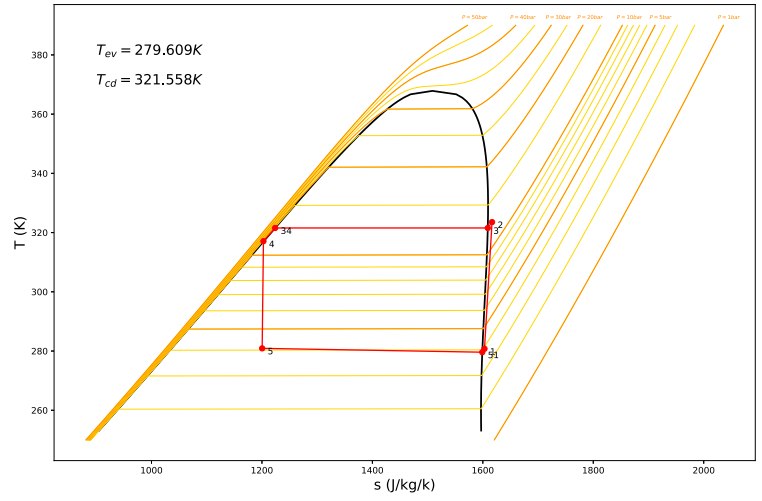
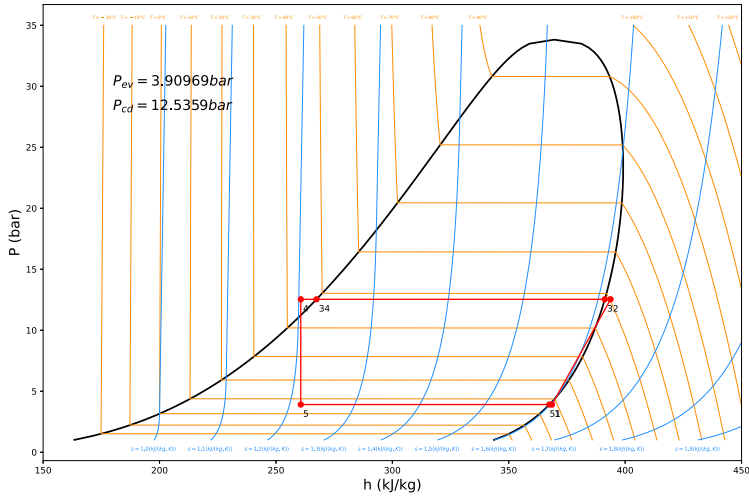
$T_{out} = 0^{\circ}\text{C}$  &  $N = 1000$  RPM



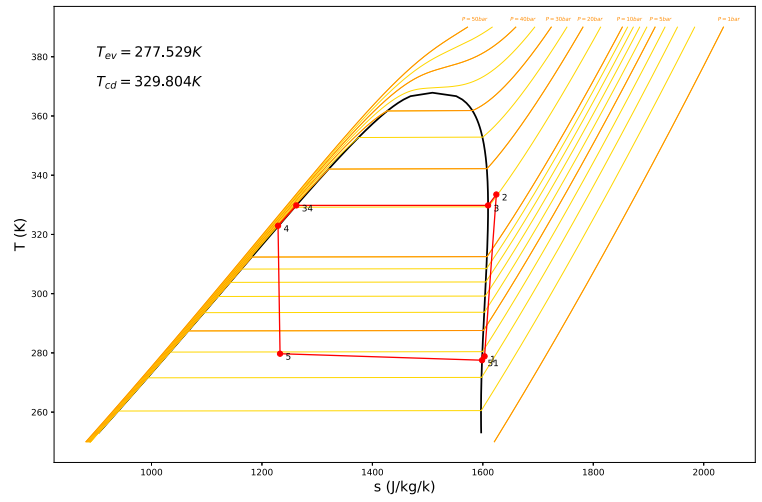
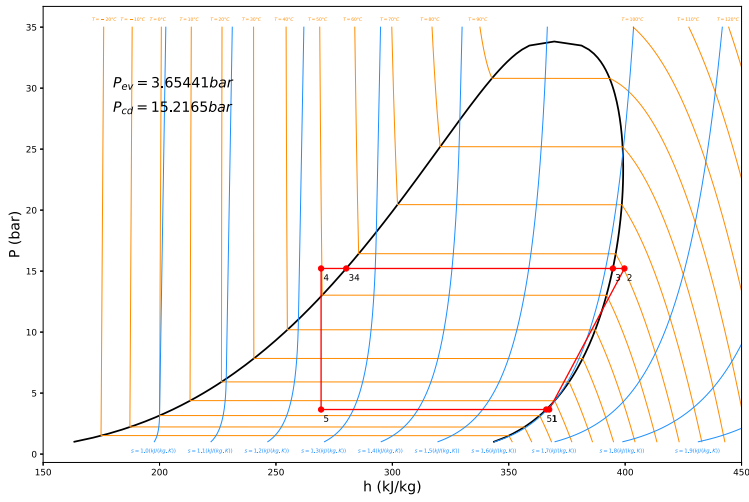
$T_{out} = 0^{\circ}\text{C}$  &  $N = 1475$  RPM



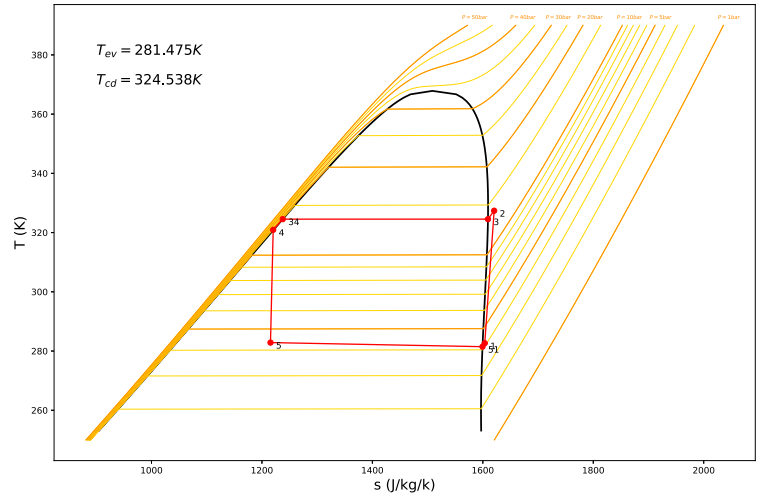
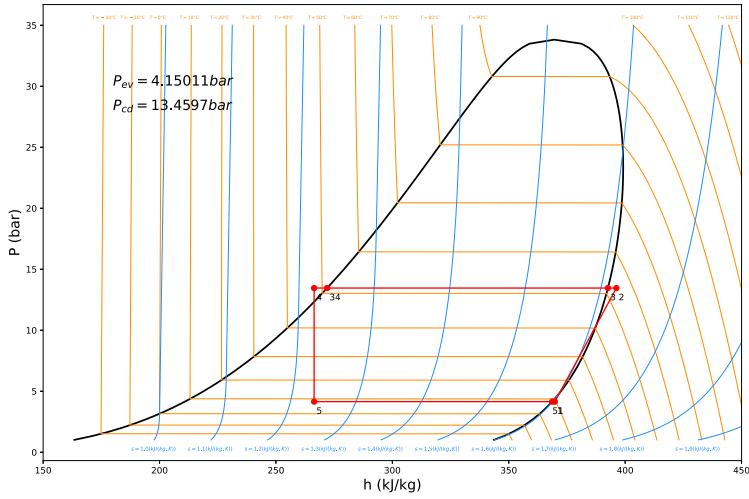
$T_{out} = 5^{\circ}\text{C}$  &  $N = 1000$  RPM



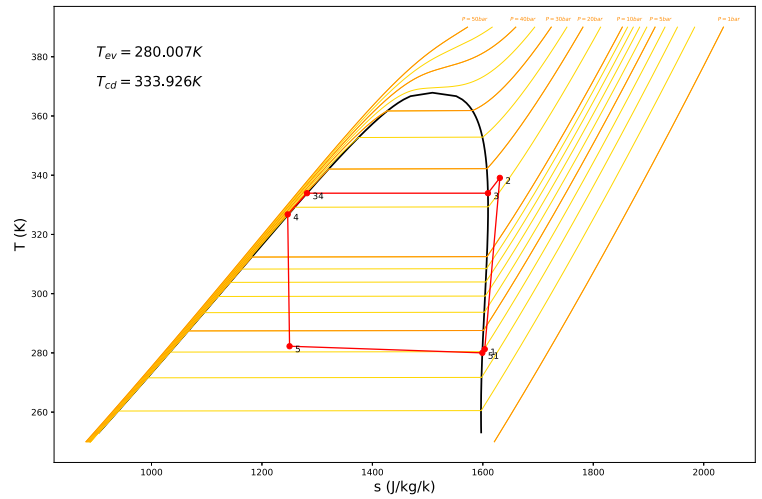
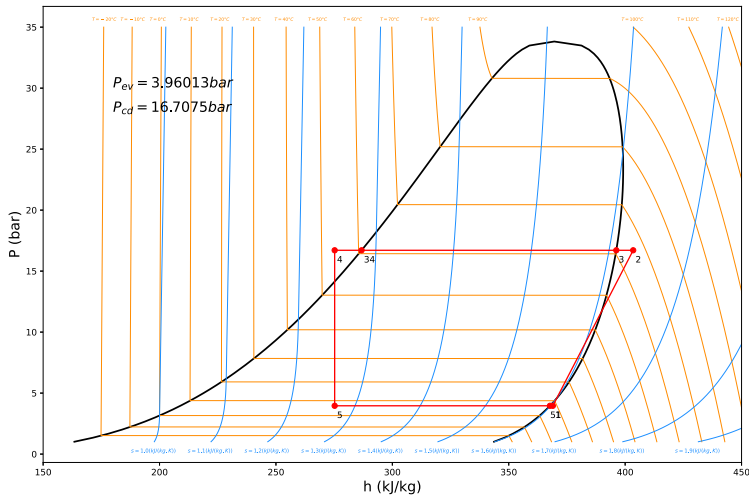
$T_{out} = 5^{\circ}\text{C}$  &  $N = 1475$  RPM



**$T_{out} = 10^{\circ}\text{C}$  &  $N = 1000$  RPM**



**$T_{out} = 10^{\circ}\text{C}$  &  $N = 1475$  RPM**



## Annexes C

### Evaluation of the portable unit pressure losses

#### Air path in the portable unit

The air path in the mobile unit was determined in order to determine where pressure drops were occurring.

1. Firstly, the fan draws ambient air from above the unit through a stainless steel pipe ( $\text{Ø}12$  cm) as represented in Fig 80. Linear losses occur in this pipe but also singular one in the elbow.

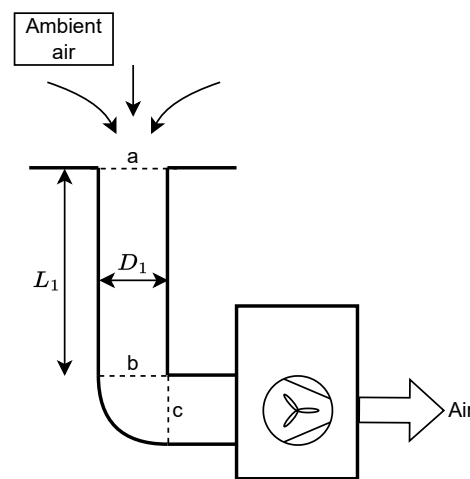


Figure 80: Admission side of the mobile unit

2. The fan blows the air to a mixing chamber containing the heat exchangers (30x50x166 cm). Pressure losses are negligible in this case. The air passes then by a second chamber (12x50x166 cm) containing the discharge pipes through a rectangular hole in the wall (10x8 cm). The volumes before and after the hole are important so pressure losses can still be neglected
3. As schematically represented in Fig 81 air is pulled through two identical pipes ( $\text{Ø}6$  cm) by conical inlets. Linear and singular losses are thus induced.



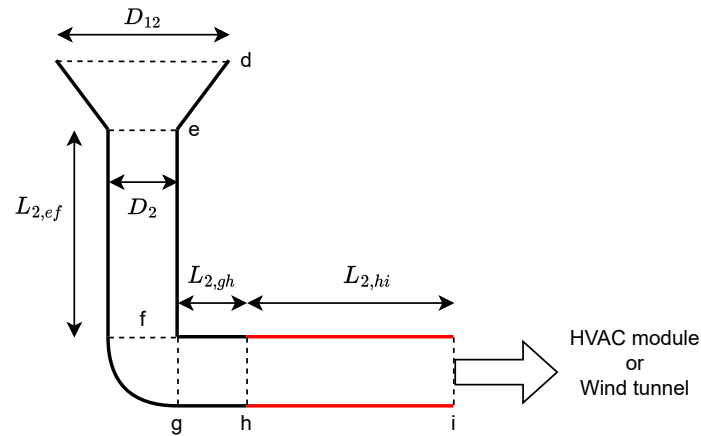


Figure 81: Discharge side of the mobile unit

4. Finally, air flow reach the wind tunnel and the HVAC module through two red flexible and insulated plastic pipes ( $\varnothing 6$  cm). Once again linear pressure losses are induced.

### Pyhton code

```

1 import numpy as np
2 from math import pi, sqrt
3 import matplotlib.pyplot as plt
4
5
6 nbr_it = 6
7 g = 9.81
8
9 #function of temperature and pressure => ambient temperature 20 C
   and atmospheric pressure 101325 Pa
10 rho = 1.2 #[kg/m3], density
11 mu = 1.5*1E-5 #[m2/s], dynamic viscosity
12 nu = mu/rho #[m2/s], cinematic viscosity
13
14
15
16 Q_start_m3h = 50
17 Q_start_m3s = Q_start_m3h/3600
18
19
20 Q_m3h = np.zeros(nbr_it, dtype=float)
21 Q_m3s = np.zeros(nbr_it, dtype=float)
22 pdc = np.zeros(nbr_it, dtype=float)
23 V_out = np.zeros(nbr_it, dtype=float)

```

```

24 Q_out = np.zeros(nbr_it, dtype=float)
25
26
27 "1"
28 L1_ab = 0.8 #m
29 D1 = 0.12 #m
30 r1 = D1/2
31 A1 = pi*r1**2
32
33
34 "2"
35 L2_ef = 1.22
36 L2_gh = 0.41
37 L2_hi = 2.5
38 L2 = L2_ef + L2_gh + L2_hi
39
40 D2 = 0.06 #m
41 r2 = D2/2
42 A2 = pi*r2**2
43
44
45
46 for i in range (0,nbr_it):
47
48     Q_m3h[i] = Q_start_m3h + i*50
49     Q_m3s[i] = Q_m3h[i]/3600
50     Q = Q_m3s[i]
51
52
53     "Inlet "
54     V1 = Q/A1 #m/s
55     Re1 = (V1*D1)/nu
56     #f1 = 64/Re1 # valable pour Re<3000, pas le cas ici
57
58     #Linear pressure losses (Darcy-Weisbach)
59     f1 = 0.3164*Re1**(-0.25)
60
61     deltaP1_ab = f1 * (L1_ab/D1) * (V1**2/(2*g))
62
63     #Singular pressure losses (large elbow)
64     k_bc = 0.13
65     deltaP1_bc = (k_bc*V1**2)/(2*g)
66

```

```

67 #Total pressure losses a the inlet
68 deltaP1 = deltaP1_ab + deltaP1_bc
69
70 V1_out = sqrt(V1**2 - deltaP1)
71 Q1_out = V1_out * A1
72
73
74
75 "Outlet"
76 V2 = Q1_out/A2 #m/s
77 Re2 = (V2*D2)/nu
78 #Singular pressure losses (cone)
79 k_de = 0.1
80 deltaP2_de = (k_de*V2**2)/(2*g)
81
82 #Linear pressure losses (Darcy-Weisbach) e->f , g->h and h->i
83 f2 = 0.02 #Moody
84 deltaP2_ef = f2 * (L2_ef/D2) * (V2**2/(2*g))
85 deltaP2_gh = f2 * (L2_gh/D2) * (V2**2/(2*g))
86 deltaP2_hi = f2 * (L2_hi/D2) * (V2**2/(2*g))
87
88 #Singular pressure losses (elbow)
89 k_fg = 0.15
90 deltaP2_fg = (k_fg*V2**2)/(2*g)
91
92 #Total pressure losses a the outlet
93 deltaP2 = deltaP2_de + deltaP2_ef + deltaP2_gh + deltaP2_hi +
94         deltaP2_fg
95 V2_out = sqrt(V2**2 - deltaP2)
96 Q2_out = V2_out * A2
97
98
99
100 "Total ressure losses of the system"
101 pdc[i] = deltaP1 + deltaP2
102
103 print(Q_m3h)
104 print(pdc)
105 print(Q_out)

```

# Annexes D

## Python codes

heat\_pump\_results\_single\_point

```
1 from Improved_heat_pump_with_all_deltaP import GenericInputs
2 from Improved_heat_pump_with_all_deltaP import SimpleHeatPump
3 from Improved_heat_pump_with_all_deltaP import Saturation_Curve
4
5
6 import matplotlib.pyplot as plt
7 import numpy as np
8 from statistics import mean
9
10
11 import CoolProp.CoolProp as CP
12
13
14 #-----
15 params = GenericInputs()
16 params.Comp.displ = 34e-06
17 params.Comp.rv_in= 2.5
18 params.Comp.A_leak = 2*10**-7
19 params.Comp.T_loss = 0.1
20 params.Comp.AU = [0.85, 0.5, 0.5]
21
22 #-----
23 params.Cond.U_sh = 20
24 params.Cond.U_lat = 200
25 params.Cond.U_sb = 40
26 #params.Cond.A = 0.7
27 params.Cond.typeLAT = "SemiIso"
28 params.Cond.typeHX = "CrossFlow"
29
30 params.Evap.U_sat = 200
31 params.Evap.U_sh = 20
32
33 params.Evap.typeHX = "CrossFlow"
34 params.Evap.typeLAT = "SemiIso"
35
36 params.HTX1.A = 1.824
37 params.HTX1.N_pipe = [15, 15, 15, 15]
```

```

38 params.HTX1.L = 0.495
39 params.HTX1.l = 0.001
40 params.HTX1.h = 0.021
41
42 params.HTX2.A = 0.7
43 params.HTX2.N_pipe = [14, 9]
44 params.HTX2.L = 0.2
45 params.HTX2.l = 0.001
46 params.HTX2.h = 0.06
47
48 params.HTX3.A = 0.7
49 params.HTX3.N_pipe = [9, 14]
50 params.HTX3.L = 0.2
51 params.HTX3.l = 0.001
52 params.HTX3.h = 0.06
53 #-----
54 params.Cycle.SH = 2
55 params.Cycle.SC = 3
56
57 params.Mdot_n = 0.019
58 #-----
59 fluid = "R1234yf"
60 #=====
61 AA_HeatPump = SimpleHeatPump(fluid, params)
62 #=====
63 P_air_ev = P_air_cd = P_air = 101325
64 T_PP_cd = 20
65 T_PP_ev = 15
66
67 N = 1950/60 #Hz
68 T_air_ext = 45+273.15
69 T_air_int = 0+273.15
70 rho_air_ext = CP.PropsSI('D', 'T', T_air_ext, 'P', P_air, 'Air')
71 rho_air = 1.225 #kg/m3
72
73
74 "HTX1 "
75 v_air_HTX1 = 0.38 #0.38 #m/s
76 A_HTX1 = 0.495*0.39 #m^2
77 V_dot_air_m3min_HTX1 = v_air_HTX1*A_HTX1*60 #m^3/min
78 V_dot_air_HTX1 = V_dot_air_m3min_HTX1/60 #m^3/s
79 Air_nom_flow_HTX1 = V_dot_air_HTX1*rho_air_ext

```

```

80 m_dot_air_HTX1 = 1*Air_nom_flow_HTX1 # air flow at the external
    echanger kg/s
81
82
83 "HVAC"
84 v_air_HVAC = 2 #m/s
85 A_HVAC = 0.048 #m^2
86 V_dot_air_m3min_HVAC = v_air_HVAC*A_HVAC*60 #m^3/min
87 V_dot_air_HVAC = V_dot_air_m3min_HVAC/60 #m^3/s
88 Air_nom_flow_HVAC = V_dot_air_HVAC*rho_air_ext
89 m_dot_air_HVAC = 1*Air_nom_flow_HVAC # air flow into the HVAC
    supposed constant for both exchangers kg/s
90
91 #=====
92
93 if T_air_ext > T_air_int:
94     mode = 'Cooling'
95     T_air_ev = T_air_int
96     T_air_cd = T_air_ext
97 if T_air_int > T_air_ext:
98     mode = 'Heating'
99     T_air_ev = T_air_ext
100    T_air_cd = T_air_int
101 print("mode = ", mode)
102 AA_HeatPump_results_bis = AA_HeatPump.solveSys(N, m_dot_air_HTX1,
    m_dot_air_HVAC, T_air_ev, T_air_cd, P_air_ev, P_air_cd, T_PP_cd,
    T_PP_ev, mode)
103
104
105 Sat_Curve = Saturation_Curve(fluid)
106
107 Sat_Curve_results = Sat_Curve.Temperature_entropy_curve()
108
109 print("N = %.2f RPM " %(AA_HeatPump.N*60))
110 print("COP_c = ", AA_HeatPump.COP_c)
111 print("COP_h = ", AA_HeatPump.COP_h)
112 print("Isentropic efficiency = ", AA_HeatPump.Comp.e_is)
113 print("m_dot = %.2f g/s " %(AA_HeatPump.Comp.Mdot*1000))
114 print("m_dot_air_HTX1 = %.2f m^3/h " %(V_dot_air_HTX1*3600))
115 print("m_dot_air_HVAC = %.2f m^3/h " %(V_dot_air_HVAC*3600))
116
117 print("-----")
118 print("Bilan:")

```

```

119 print("W_dot_CP = %.2f W " %(AA_HeatPump.Comp.Wdot))
120 print("W_dot_loss = %.2f W " %(AA_HeatPump.Comp.w_loss))
121
122 print("Q_dot_ev = %.2f W " %(AA_HeatPump.Q_dot_ev))
123 print("Q_dot_cd = %.2f W " %(AA_HeatPump.Q_dot_cd))
124 balance = AA_HeatPump.Q_dot_cd - AA_HeatPump.Q_dot_ev -
        AA_HeatPump.Comp.Wdot
125 print("balance = %.2f W " %(balance))
126
127
128
129 if mode=='Cooling':
130     print("T_air_in = ", AA_HeatPump.air_ex_state_ev_sat.T - 273.15)
131
132 if mode=='Heating':
133     print("T_air_in = ", AA_HeatPump.air_ex_state_cd_sb.T - 273.15)
134
135 Tsat = Sat_Curve.Tsat
136
137 T_array = np.linspace(250, 390, num=1000)
138
139 P_cst = [1e5, 2e5, 3e5, 4e5, 5e5, 6e5, 7e5, 8e5, 9e5, 10e5, 15e5,
        20e5, 25e5, 30e5, 35e5, 40e5, 45e5, 50e5]
140 s_for_P_cst = np.zeros((len(P_cst), len(T_array)))
141 for i in range(len(P_cst)):
142     s_for_P = Sat_Curve.Entropy_from_T_S_curve_for_P_cst(T_array,
        P_cst[i])
143     s_for_P_cst_tmp = Sat_Curve.s_cst
144     for j in range(len(T_array)):
145         s_for_P_cst[i][j] = s_for_P_cst_tmp[j]
146
147
148
149 fig1, ax = plt.subplots(figsize=(12, 8))
150 ax.plot(Sat_Curve.Ssat, Sat_Curve.Tsat, 'k-', linewidth=2)
151 ax.set_xlabel(u's (J/kg/k)', fontsize=16)
152 ax.set_ylabel(u'T (K)', fontsize=16)
153
154 for i in range(len(P_cst)):
155     ax.plot(s_for_P_cst[i][:], T_array, color='gold', label='T =
        %d' %P_cst[i], linewidth=1.0)
156     if i == 0 or i ==4 or i == 9 or i == 11 or i == 13 or i == 15 or
        i == 17:

```

```

157     ax.plot(s_for_P_cst[i][:], T_array, color='darkorange',
158           label='T = %d' % P_cst[i], linewidth=1.0)
159     ax.text((s_for_P_cst[i][len(T_array) - 1] - 10), 392., r'$P
160           = %i bar$' % (P_cst[i] / 1.e5), color='darkorange',
161           fontsize=6)
162
163 ax.plot(AA_HeatPump.Point_1.s, AA_HeatPump.Point_1.T, 'ro')
164 ax.plot(AA_HeatPump.Point_2.s, AA_HeatPump.Point_2.T, 'ro')
165 ax.plot(AA_HeatPump.Point_3.s, AA_HeatPump.Point_3.T, 'ro')
166 ax.plot(AA_HeatPump.Point_3b.s, AA_HeatPump.Point_3b.T, 'ro')
167 ax.plot(AA_HeatPump.Point_4.s, AA_HeatPump.Point_4.T, 'ro')
168 ax.plot(AA_HeatPump.Point_5.s, AA_HeatPump.Point_5.T, 'ro')
169 ax.plot(AA_HeatPump.Point_5b.s, AA_HeatPump.Point_5b.T, 'ro')
170 ax.text(900, 380, r'$T_{ev} = %g K$' %((AA_HeatPump.Point_5b.T)),
171         fontsize=16)
172 ax.text(900, 370, r'$T_{cd} = %g K$' %((AA_HeatPump.Point_3b.T)),
173         fontsize=16)
174 ax.text(AA_HeatPump.Point_1.s+10, AA_HeatPump.Point_1.T-2, '%d' %1)
175 ax.text(AA_HeatPump.Point_2.s+10, AA_HeatPump.Point_2.T-2, '%d' %2)
176 ax.text(AA_HeatPump.Point_3.s+10, AA_HeatPump.Point_3.T-2, '%d' %3)
177 ax.text(AA_HeatPump.Point_3b.s+10, AA_HeatPump.Point_3b.T-2, '%d'
178         %34)
179 ax.text(AA_HeatPump.Point_4.s+10, AA_HeatPump.Point_4.T-2, '%d' %4)
180 ax.text(AA_HeatPump.Point_5.s+10, AA_HeatPump.Point_5.T-2, '%d' %5)
181 ax.text(AA_HeatPump.Point_5b.s+10, AA_HeatPump.Point_5b.T-2, '%d'
182         %51)
183 ax.plot(np.linspace(AA_HeatPump.Point_1.s, AA_HeatPump.Point_2.s,
184                   num=100), np.linspace(AA_HeatPump.Point_1.T,
185                   AA_HeatPump.Point_2.T, num=100), 'r-')
186 ax.plot(np.linspace(AA_HeatPump.Point_2.s, AA_HeatPump.Point_3.s,
187                   num=100), np.linspace(AA_HeatPump.Point_2.T,
188                   AA_HeatPump.Point_3.T, num=100), 'r-')
189 ax.plot(np.linspace(AA_HeatPump.Point_3.s, AA_HeatPump.Point_3b.s,
190                   num=100), np.linspace(AA_HeatPump.Point_3.T,
191                   AA_HeatPump.Point_3b.T, num=100), 'r-')
192 ax.plot(np.linspace(AA_HeatPump.Point_3b.s, AA_HeatPump.Point_4.s,
193                   num=100), np.linspace(AA_HeatPump.Point_3b.T,
194                   AA_HeatPump.Point_4.T, num=100), 'r-')
195 ax.plot(np.linspace(AA_HeatPump.Point_4.s, AA_HeatPump.Point_5.s,
196                   num=100), np.linspace(AA_HeatPump.Point_4.T,
197                   AA_HeatPump.Point_5.T, num=100), 'r-')

```



```

182 ax.plot(np.linspace(AA_HeatPump.Point_5.s, AA_HeatPump.Point_5b.s,
    num=100), np.linspace(AA_HeatPump.Point_5.T,
    AA_HeatPump.Point_5b.T, num=100), 'r-')
183 ax.plot(np.linspace(AA_HeatPump.Point_5b.s, AA_HeatPump.Point_1.s,
    num=100), np.linspace(AA_HeatPump.Point_5b.T,
    AA_HeatPump.Point_1.T, num=100), 'r-')
184 fig1.savefig('./Temperature_entropy_curve.png')
185
186 #Pour un r gime de fonctionnement: P-h
187
188
189
190 Sat_Curve = Saturation_Curve(fluid)
191 Sat_Curve_results_bis = Sat_Curve.Pressure_enthalpy_curve()
192
193 P_array = np.linspace(101325, 35*1e5, num=3500)
194 T_cst = 20 + 273.15
195 h_for_T_cst_20 =
    Sat_Curve.Enthalpy_from_P_H_curve_for_T_cst(P_array, T_cst)
196
197 T_cst = [-20, -10, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110,
    120]
198 h_for_T_cst = np.zeros((len(T_cst), len(P_array)))
199 for i in range(len(T_cst)):
200     T_cst[i] = T_cst[i] + 273.15
201     h_for_T = Sat_Curve.Enthalpy_from_P_H_curve_for_T_cst(P_array,
    T_cst[i])
202     h_for_T_cst_tmp = Sat_Curve.h_cst
203     for j in range(len(P_array)):
204         h_for_T_cst[i][j] = h_for_T_cst_tmp[j]
205
206 s_cst = [1e3, 1.1e3, 1.2e3, 1.3e3, 1.4e3, 1.5e3, 1.6e3, 1.7e3,
    1.8e3, 1.9e3, 2e3]
207 h_for_s_cst = np.zeros((len(s_cst), len(P_array)))
208 for i in range(len(s_cst)):
209     h_for_s = Sat_Curve.Enthalpy_from_P_H_curve_for_s_cst(P_array,
    s_cst[i])
210     h_for_s_cst_tmp = Sat_Curve.h_cst
211     for j in range(len(P_array)):
212         h_for_s_cst[i][j] = h_for_s_cst_tmp[j]
213
214 fig2, ax = plt.subplots(figsize=(12,8))

```

```

215 ax.plot(Sat_Curve.Hsat/1.e3, (Sat_Curve.Psat/1.e5), 'k-',
          linewidth=2)
216 ax.set_xlabel(u'h (kJ/kg)', fontsize=16)
217 ax.set_ylabel(u'P (bar)', fontsize=16)
218
219 ax.plot(Sat_Curve.h_cst/1.e3, P_array/1.e5, color='orange', label='T
      = 20', linewidth=1.0)
220
221 for i in range(len(T_cst)):
222     ax.plot(h_for_T_cst[i][:]/1.e3, P_array / 1.e5,
              color='darkorange', label='T = %f' %T_cst[i], linewidth=1.0)
223     ax.text(((h_for_T_cst[i][len(P_array)-1] - 5000)/1.e3), 35.5,
              r'$T = %i C$ ' % (T_cst[i]-273.15), color='darkorange',
              fontsize=5)
224 for i in range(len(s_cst)):
225     ax.plot(h_for_s_cst[i][:]/1.e3, P_array / 1.e5,
              color='dodgerblue', label='T = %d' %T_cst[i], linewidth=1.0)
226     if i < 10:
227         ax.text(((h_for_s_cst[i][0] - 10000)/1.e3), 10000. / 1.e5,
                  r'$s = %s (kJ/(kg.K))$' % (s_cst[i] / 1.e3),
                  color='dodgerblue', fontsize=6)
228
229
230 plt.xlim(150000./1.e3, 450000./1.e3)
231 ax.plot(AA_HeatPump.Point_1.h/1.e3, AA_HeatPump.Point_1.P/1.e5, 'ro')
232 ax.plot(AA_HeatPump.Point_2.h/1.e3, AA_HeatPump.Point_2.P/1.e5, 'ro')
233 ax.plot(AA_HeatPump.Point_3.h/1.e3, AA_HeatPump.Point_3.P/1.e5, 'ro')
234 ax.plot(AA_HeatPump.Point_3b.h/1.e3, AA_HeatPump.Point_3b.P/1.e5,
          'ro')
235 ax.plot(AA_HeatPump.Point_4.h/1.e3, AA_HeatPump.Point_4.P/1.e5, 'ro')
236 ax.plot(AA_HeatPump.Point_5.h/1.e3, AA_HeatPump.Point_5.P/1.e5, 'ro')
237 ax.plot(AA_HeatPump.Point_5b.h/1.e3, AA_HeatPump.Point_5b.P/1.e5,
          'ro')
238 ax.text(180000/1.e3, 3000000./1e5, r'$P_{ev} = %g bar$'
          %((AA_HeatPump.Point_1.P)/1.e5), fontsize=16)
239 ax.text(180000/1.e3, 2800000./1e5, r'$P_{cd} = %g bar$'
          %((AA_HeatPump.Point_2.P)/1.e5), fontsize=16)
240 ax.text((AA_HeatPump.Point_1.h+1000)/1.e3,
          (AA_HeatPump.Point_1.P-1.e5)/1.e5, '%d' %1)
241 ax.text((AA_HeatPump.Point_2.h+1000)/1.e3,
          (AA_HeatPump.Point_2.P-1.e5)/1.e5, '%d' %2)
242 ax.text((AA_HeatPump.Point_3.h+1000)/1.e3,
          (AA_HeatPump.Point_3.P-1.e5)/1.e5, '%d' %3)

```

```

243 ax.text((AA_HeatPump.Point_3b.h+1000)/1.e3,
        (AA_HeatPump.Point_3b.P-1.e5)/1.e5, '%d' %34)
244 ax.text((AA_HeatPump.Point_4.h+1000)/1.e3,
        (AA_HeatPump.Point_4.P-1.e5)/1.e5, '%d' %4)
245 ax.text((AA_HeatPump.Point_5.h+1000)/1.e3,
        (AA_HeatPump.Point_5.P-1.e5)/1.e5, '%d' %5)
246 ax.text((AA_HeatPump.Point_5b.h+100)/1.e3,
        (AA_HeatPump.Point_5b.P-1.e5)/1.e5, '%d' %51)
247 ax.plot(np.linspace(AA_HeatPump.Point_1.h/1.e3,AA_HeatPump.Point_2.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_1.P)/1.e5,(AA_HeatPump.Point_2.P)/1.e5,
        num=100), 'r-')
248 ax.plot(np.linspace(AA_HeatPump.Point_2.h/1.e3,AA_HeatPump.Point_3.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_2.P)/1.e5,(AA_HeatPump.Point_3.P)/1.e5,
        num=100), 'r-')
249 ax.plot(np.linspace(AA_HeatPump.Point_3.h/1.e3,AA_HeatPump.Point_3b.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_3.P)/1.e5,(AA_HeatPump.Point_3b.P)/1.e5,
        num=100), 'r-')
250 ax.plot(np.linspace(AA_HeatPump.Point_3b.h/1.e3,AA_HeatPump.Point_4.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_3b.P)/1.e5,(AA_HeatPump.Point_4.P)/1.e5,
        num=100), 'r-')
251 ax.plot(np.linspace(AA_HeatPump.Point_4.h/1.e3,AA_HeatPump.Point_5.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_4.P)/1.e5,(AA_HeatPump.Point_5.P)/1.e5,
        num=100), 'r-')
252 ax.plot(np.linspace(AA_HeatPump.Point_5.h/1.e3,AA_HeatPump.Point_5b.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_5.P)/1.e5,(AA_HeatPump.Point_5b.P)/1.e5,
        num=100), 'r-')
253 ax.plot(np.linspace(AA_HeatPump.Point_5b.h/1.e3,AA_HeatPump.Point_1.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_5b.P)/1.e5,(AA_HeatPump.Point_1.P)/1.e5,
        num=100), 'r-')
254 fig2.savefig('./Pressure_enthalpy_curve.png')
255
256
257
258 T_air_su_cd = AA_HeatPump.air_su_state_cd.T
259 T_air_su_cd_tp = AA_HeatPump.air_ex_state_cd_sb.T
260 T_air_su_cd_sh = AA_HeatPump.air_ex_state_cd_lat.T

```

```

261 T_air_ex_cd = AA_HeatPump.air_ex_state_cd_sh.T
262
263 T_r_su_cd = AA_HeatPump.Point_2.T
264 T_r_su_cd_tp = AA_HeatPump.Point_3.T
265 T_r_su_cd_sc = AA_HeatPump.Point_3b.T
266 T_r_ex_cd = AA_HeatPump.Point_4.T
267
268 Q_dot_cd_sh = AA_HeatPump.Q_dot_cd_sh
269 Q_dot_cd_tp = AA_HeatPump.Q_dot_cd_lat
270 Q_dot_cd_sc = AA_HeatPump.Q_dot_cd_sb
271
272 Q_dot_cd = [0, Q_dot_cd_sh, Q_dot_cd_sh+Q_dot_cd_tp,
             Q_dot_cd_sc+Q_dot_cd_tp+Q_dot_cd_sh]
273 T_r_cd = [T_r_su_cd, T_r_su_cd_tp, T_r_su_cd_sc, T_r_ex_cd]
274 T_air_cd = [T_air_ex_cd, T_air_su_cd_sh, T_air_su_cd_tp, T_air_su_cd]
275
276 fig, ax1 = plt.subplots()
277 ax1.plot(Q_dot_cd, T_r_cd)
278 ax1.plot(Q_dot_cd, T_air_cd)
279 ax1.legend(("T_r_cd", "T_air_cd"))
280 ax1.set_ylabel("T [K]")
281 ax1.set_xlabel("Q_dot_cd [W]")
282 ax1.text((Q_dot_cd[0] + 50), (T_r_cd[0] - 0.3), '%d' % 2)
283 ax1.text((Q_dot_cd[1] + 50), (T_r_cd[1] + 1), '%d' % 3)
284 ax1.text((Q_dot_cd[2] + 50), (T_r_cd[2] - 0.3), '%d' % 34)
285 ax1.text((Q_dot_cd[3] + 50), (T_r_cd[3] - 0.3), '%d' % 4)
286
287 T_air_su_ev = AA_HeatPump.air_su_state_ev.T
288 T_air_su_ev_tp = AA_HeatPump.air_ex_state_ev.T
289 T_air_ex_ev = AA_HeatPump.air_ex_state_ev_sat.T
290
291 T_r_su_ev = AA_HeatPump.Point_5.T
292 T_r_su_ev_sh = AA_HeatPump.Point_5b.T
293 T_r_ex_ev = AA_HeatPump.Point_1.T
294
295 Q_dot_ev_tp = AA_HeatPump.Q_dot_ev_lat
296 Q_dot_ev_sh = AA_HeatPump.Q_dot_ev_sh
297
298 Q_dot_ev = [0, Q_dot_ev_tp, Q_dot_ev_tp + Q_dot_ev_sh]
299 T_r_ev = [T_r_su_ev, T_r_su_ev_sh, T_r_ex_ev]
300 T_air_ev = [T_air_ex_ev, T_air_su_ev_tp, T_air_su_ev]
301
302 fig, ax1 = plt.subplots()

```

```

303 ax1.plot(Q_dot_ev, T_r_ev)
304 ax1.plot(Q_dot_ev, T_air_ev)
305 ax1.legend(("T_r_ev", "T_air_ev"))
306 ax1.set_ylabel("T [K]")
307 ax1.set_xlabel("Q_dot_ev [W]")
308 ax1.text((Q_dot_ev[0]+50), (T_r_ev[0]-0.3), '%d' % 5)
309 ax1.text((Q_dot_ev[1]+50), (T_r_ev[1]-0.3), '%d' % 51)
310 ax1.text((Q_dot_ev[2]+50), (T_r_ev[2]-0.3), '%d' % 1)

```

### heat\_pump\_results\_single\_point

```

1  from Improved_heat_pump_with_all_deltaP import GenericInputs
2  from Improved_heat_pump_with_all_deltaP import SimpleHeatPump
3  from Improved_heat_pump_with_all_deltaP import Saturation_Curve
4
5
6  import matplotlib.pyplot as plt
7  import numpy as np
8  from statistics import mean
9
10
11 import CoolProp.CoolProp as CP
12
13
14 #=====
15 params = GenericInputs()
16 params.Comp.displ = 34e-06
17 params.Comp.rv_in= 2.5
18 params.Comp.A_leak = 2*10**-7
19 params.Comp.T_loss = 0.1
20 params.Comp.AU = [0.85, 0.5, 0.5]
21
22 #-----
23 params.Cond.U_sh = 20
24 params.Cond.U_lat = 200
25 params.Cond.U_sb = 40
26 #params.Cond.A = 0.7
27 params.Cond.typeLAT = "SemiIso"
28 params.Cond.typeHX = "CrossFlow"
29
30 params.Evap.U_sat = 200
31 params.Evap.U_sh = 20
32

```

```

33 params.Evap.typeHX = "CrossFlow"
34 params.Evap.typeLAT = "SemiIso"
35
36 params.HTX1.A = 1.824
37 params.HTX1.N_pipe = [15, 15, 15, 15]
38 params.HTX1.L = 0.495
39 params.HTX1.l = 0.001
40 params.HTX1.h = 0.021
41
42 params.HTX2.A = 0.7
43 params.HTX2.N_pipe = [14, 9]
44 params.HTX2.L = 0.2
45 params.HTX2.l = 0.001
46 params.HTX2.h = 0.06
47
48 params.HTX3.A = 0.7
49 params.HTX3.N_pipe = [9, 14]
50 params.HTX3.L = 0.2
51 params.HTX3.l = 0.001
52 params.HTX3.h = 0.06
53 #-----
54 params.Cycle.SH = 2
55 params.Cycle.SC = 3
56
57 params.Mdot_n = 0.019
58 #-----
59 fluid = "R1234yf"
60 #=====
61 AA_HeatPump = SimpleHeatPump(fluid, params)
62 #=====
63 P_air_ev = P_air_cd = P_air = 101325
64 T_PP_cd = 20
65 T_PP_ev = 15
66
67 N = 1950/60 #Hz
68 T_air_ext = 45+273.15
69 T_air_int = 0+273.15
70 rho_air_ext = CP.PropsSI('D', 'T', T_air_ext, 'P', P_air, 'Air')
71 rho_air = 1.225 #kg/m3
72
73
74 "HTX1 "
75 v_air_HTX1 = 0.38 #0.38 #m/s

```

```

76 A_HTX1 = 0.495*0.39 #m^2
77 V_dot_air_m3min_HTX1 = v_air_HTX1*A_HTX1*60 #m^3/min
78 V_dot_air_HTX1 = V_dot_air_m3min_HTX1/60 #m^3/s
79 Air_nom_flow_HTX1 = V_dot_air_HTX1*rho_air_ext
80 m_dot_air_HTX1 = 1*Air_nom_flow_HTX1 # air flow at the external
    echanger kg/s
81
82
83 "HVAC"
84 v_air_HVAC = 2 #m/s
85 A_HVAC = 0.048 #m^2
86 V_dot_air_m3min_HVAC = v_air_HVAC*A_HVAC*60 #m^3/min
87 V_dot_air_HVAC = V_dot_air_m3min_HVAC/60 #m^3/s
88 Air_nom_flow_HVAC = V_dot_air_HVAC*rho_air_ext
89 m_dot_air_HVAC = 1*Air_nom_flow_HVAC # air flow into the HVAC
    supposed constant for both exchangers kg/s
90
91 #=====
92
93 if T_air_ext > T_air_int:
94     mode = 'Cooling'
95     T_air_ev = T_air_int
96     T_air_cd = T_air_ext
97 if T_air_int > T_air_ext:
98     mode = 'Heating'
99     T_air_ev = T_air_ext
100    T_air_cd = T_air_int
101 print("mode = ", mode)
102 AA_HeatPump_results_bis = AA_HeatPump.solveSys(N, m_dot_air_HTX1,
    m_dot_air_HVAC, T_air_ev, T_air_cd, P_air_ev, P_air_cd, T_PP_cd,
    T_PP_ev, mode)
103
104
105 Sat_Curve = Saturation_Curve(fluid)
106
107 Sat_Curve_results = Sat_Curve.Temperature_entropy_curve()
108
109 print("N = %.2f RPM " %(AA_HeatPump.N*60))
110 print("COP_c = ", AA_HeatPump.COP_c)
111 print("COP_h = ", AA_HeatPump.COP_h)
112 print("Isentropic efficiency = ", AA_HeatPump.Comp.e_is)
113 print("m_dot = %.2f g/s " %(AA_HeatPump.Comp.Mdot*1000))
114 print("m_dot_air_HTX1 = %.2f m^3/h " %(V_dot_air_HTX1*3600))

```

```

115 print("m_dot_air_HVAC = %.2f m^3/h " %(V_dot_air_HVAC*3600))
116
117 print("-----")
118 print("Bilan:")
119 print("W_dot_CP = %.2f W " %(AA_HeatPump.Comp.Wdot))
120 print("W_dot_loss = %.2f W " %(AA_HeatPump.Comp.w_loss))
121
122 print("Q_dot_ev = %.2f W " %(AA_HeatPump.Q_dot_ev))
123 print("Q_dot_cd = %.2f W " %(AA_HeatPump.Q_dot_cd))
124 balance = AA_HeatPump.Q_dot_cd - AA_HeatPump.Q_dot_ev -
        AA_HeatPump.Comp.Wdot
125 print("balance = %.2f W " %(balance))
126
127
128
129 if mode=='Cooling':
130     print("T_air_in = ", AA_HeatPump.air_ex_state_ev_sat.T - 273.15)
131
132 if mode=='Heating':
133     print("T_air_in = ", AA_HeatPump.air_ex_state_cd_sb.T - 273.15)
134
135 Tsat = Sat_Curve.Tsat
136
137 T_array = np.linspace(250, 390, num=1000)
138
139 P_cst = [1e5, 2e5, 3e5, 4e5, 5e5, 6e5, 7e5, 8e5, 9e5, 10e5, 15e5,
        20e5, 25e5, 30e5, 35e5, 40e5, 45e5, 50e5]
140 s_for_P_cst = np.zeros((len(P_cst), len(T_array)))
141 for i in range(len(P_cst)):
142     s_for_P = Sat_Curve.Entropy_from_T_S_curve_for_P_cst(T_array,
        P_cst[i])
143     s_for_P_cst_tmp = Sat_Curve.s_cst
144     for j in range(len(T_array)):
145         s_for_P_cst[i][j] = s_for_P_cst_tmp[j]
146
147
148
149 fig1, ax = plt.subplots(figsize=(12, 8))
150 ax.plot(Sat_Curve.Ssat, Sat_Curve.Tsat, 'k-', linewidth=2)
151 ax.set_xlabel(u's (J/kg/k)', fontsize=16)
152 ax.set_ylabel(u'T (K)', fontsize=16)
153
154 for i in range(len(P_cst)):

```



```

155 ax.plot(s_for_P_cst[i][:], T_array, color='gold', label='T =
      %d' %P_cst[i], linewidth=1.0)
156 if i == 0 or i ==4 or i == 9 or i == 11 or i == 13 or i == 15 or
      i == 17:
157     ax.plot(s_for_P_cst[i][:], T_array, color='darkorange',
              label='T = %d' % P_cst[i], linewidth=1.0)
158     ax.text((s_for_P_cst[i][len(T_array) - 1] - 10), 392., r'$P
              = %i bar$' % (P_cst[i] / 1.e5), color='darkorange',
              fontsize=6)
159
160
161 ax.plot(AA_HeatPump.Point_1.s, AA_HeatPump.Point_1.T, 'ro')
162 ax.plot(AA_HeatPump.Point_2.s, AA_HeatPump.Point_2.T, 'ro')
163 ax.plot(AA_HeatPump.Point_3.s, AA_HeatPump.Point_3.T, 'ro')
164 ax.plot(AA_HeatPump.Point_3b.s, AA_HeatPump.Point_3b.T, 'ro')
165 ax.plot(AA_HeatPump.Point_4.s, AA_HeatPump.Point_4.T, 'ro')
166 ax.plot(AA_HeatPump.Point_5.s, AA_HeatPump.Point_5.T, 'ro')
167 ax.plot(AA_HeatPump.Point_5b.s, AA_HeatPump.Point_5b.T, 'ro')
168 ax.text(900, 380, r'$T_{ev} = %g K$' %((AA_HeatPump.Point_5b.T)),
          fontsize=16)
169 ax.text(900, 370, r'$T_{cd} = %g K$' %((AA_HeatPump.Point_3b.T)),
          fontsize=16)
170 ax.text(AA_HeatPump.Point_1.s+10, AA_HeatPump.Point_1.T-2, '%d' %1)
171 ax.text(AA_HeatPump.Point_2.s+10, AA_HeatPump.Point_2.T-2, '%d' %2)
172 ax.text(AA_HeatPump.Point_3.s+10, AA_HeatPump.Point_3.T-2, '%d' %3)
173 ax.text(AA_HeatPump.Point_3b.s+10, AA_HeatPump.Point_3b.T-2, '%d'
          %34)
174 ax.text(AA_HeatPump.Point_4.s+10, AA_HeatPump.Point_4.T-2, '%d' %4)
175 ax.text(AA_HeatPump.Point_5.s+10, AA_HeatPump.Point_5.T-2, '%d' %5)
176 ax.text(AA_HeatPump.Point_5b.s+10, AA_HeatPump.Point_5b.T-2, '%d'
          %51)
177 ax.plot(np.linspace(AA_HeatPump.Point_1.s, AA_HeatPump.Point_2.s,
                      num=100), np.linspace(AA_HeatPump.Point_1.T,
                      AA_HeatPump.Point_2.T, num=100), 'r-')
178 ax.plot(np.linspace(AA_HeatPump.Point_2.s, AA_HeatPump.Point_3.s,
                      num=100), np.linspace(AA_HeatPump.Point_2.T,
                      AA_HeatPump.Point_3.T, num=100), 'r-')
179 ax.plot(np.linspace(AA_HeatPump.Point_3.s, AA_HeatPump.Point_3b.s,
                      num=100), np.linspace(AA_HeatPump.Point_3.T,
                      AA_HeatPump.Point_3b.T, num=100), 'r-')
180 ax.plot(np.linspace(AA_HeatPump.Point_3b.s, AA_HeatPump.Point_4.s,
                      num=100), np.linspace(AA_HeatPump.Point_3b.T,
                      AA_HeatPump.Point_4.T, num=100), 'r-')

```

```

181 ax.plot(np.linspace(AA_HeatPump.Point_4.s, AA_HeatPump.Point_5.s,
    num=100), np.linspace(AA_HeatPump.Point_4.T,
    AA_HeatPump.Point_5.T, num=100), 'r-')
182 ax.plot(np.linspace(AA_HeatPump.Point_5.s, AA_HeatPump.Point_5b.s,
    num=100), np.linspace(AA_HeatPump.Point_5.T,
    AA_HeatPump.Point_5b.T, num=100), 'r-')
183 ax.plot(np.linspace(AA_HeatPump.Point_5b.s, AA_HeatPump.Point_1.s,
    num=100), np.linspace(AA_HeatPump.Point_5b.T,
    AA_HeatPump.Point_1.T, num=100), 'r-')
184 fig1.savefig('./Temperature_entropy_curve.png')
185
186 #Pour un r gime de fonctionnement: P-h
187
188
189
190 Sat_Curve = Saturation_Curve(fluid)
191 Sat_Curve_results_bis = Sat_Curve.Pressure_enthalpy_curve()
192
193 P_array = np.linspace(101325, 35*1e5, num=3500)
194 T_cst = 20 + 273.15
195 h_for_T_cst_20 =
    Sat_Curve.Enthalpy_from_P_H_curve_for_T_cst(P_array, T_cst)
196
197 T_cst = [-20, -10, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110,
    120]
198 h_for_T_cst = np.zeros((len(T_cst), len(P_array)))
199 for i in range(len(T_cst)):
200     T_cst[i] = T_cst[i] + 273.15
201     h_for_T = Sat_Curve.Enthalpy_from_P_H_curve_for_T_cst(P_array,
    T_cst[i])
202     h_for_T_cst_tmp = Sat_Curve.h_cst
203     for j in range(len(P_array)):
204         h_for_T_cst[i][j] = h_for_T_cst_tmp[j]
205
206 s_cst = [1e3, 1.1e3, 1.2e3, 1.3e3, 1.4e3, 1.5e3, 1.6e3, 1.7e3,
    1.8e3, 1.9e3, 2e3]
207 h_for_s_cst = np.zeros((len(s_cst), len(P_array)))
208 for i in range(len(s_cst)):
209     h_for_s = Sat_Curve.Enthalpy_from_P_H_curve_for_s_cst(P_array,
    s_cst[i])
210     h_for_s_cst_tmp = Sat_Curve.h_cst
211     for j in range(len(P_array)):
212         h_for_s_cst[i][j] = h_for_s_cst_tmp[j]

```

```

213
214 fig2, ax = plt.subplots(figsize=(12,8))
215 ax.plot(Sat_Curve.Hsat/1.e3, (Sat_Curve.Psat/1.e5), 'k-',
          linewidth=2)
216 ax.set_xlabel(u'h (kJ/kg)', fontsize=16)
217 ax.set_ylabel(u'P (bar)', fontsize=16)
218
219 ax.plot(Sat_Curve.h_cst/1.e3, P_array/1.e5, color='orange', label='T
          = 20', linewidth=1.0)
220
221 for i in range(len(T_cst)):
222     ax.plot(h_for_T_cst[i][:]/1.e3, P_array / 1.e5,
              color='darkorange', label='T = %f' %T_cst[i], linewidth=1.0)
223     ax.text(((h_for_T_cst[i][len(P_array)-1] - 5000)/1.e3), 35.5,
              r'$T = %i C$ ' % (T_cst[i]-273.15), color='darkorange',
              fontsize=5)
224 for i in range(len(s_cst)):
225     ax.plot(h_for_s_cst[i][:]/1.e3, P_array / 1.e5,
              color='dodgerblue', label='T = %d' %T_cst[i], linewidth=1.0)
226     if i < 10:
227         ax.text(((h_for_s_cst[i][0] - 10000)/1.e3), 10000. / 1.e5,
                  r'$s = %s (kJ/(kg.K))$' % (s_cst[i] / 1.e3),
                  color='dodgerblue', fontsize=6)
228
229
230 plt.xlim(150000./1.e3, 450000./1.e3)
231 ax.plot(AA_HeatPump.Point_1.h/1.e3, AA_HeatPump.Point_1.P/1.e5, 'ro')
232 ax.plot(AA_HeatPump.Point_2.h/1.e3, AA_HeatPump.Point_2.P/1.e5, 'ro')
233 ax.plot(AA_HeatPump.Point_3.h/1.e3, AA_HeatPump.Point_3.P/1.e5, 'ro')
234 ax.plot(AA_HeatPump.Point_3b.h/1.e3, AA_HeatPump.Point_3b.P/1.e5,
          'ro')
235 ax.plot(AA_HeatPump.Point_4.h/1.e3, AA_HeatPump.Point_4.P/1.e5, 'ro')
236 ax.plot(AA_HeatPump.Point_5.h/1.e3, AA_HeatPump.Point_5.P/1.e5, 'ro')
237 ax.plot(AA_HeatPump.Point_5b.h/1.e3, AA_HeatPump.Point_5b.P/1.e5,
          'ro')
238 ax.text(180000/1.e3, 3000000./1e5, r'$P_{ev} = %g bar$',
          %((AA_HeatPump.Point_1.P)/1.e5), fontsize=16)
239 ax.text(180000/1.e3, 2800000./1e5, r'$P_{cd} = %g bar$',
          %((AA_HeatPump.Point_2.P)/1.e5), fontsize=16)
240 ax.text((AA_HeatPump.Point_1.h+1000)/1.e3,
          (AA_HeatPump.Point_1.P-1.e5)/1.e5, '%d' %1)
241 ax.text((AA_HeatPump.Point_2.h+1000)/1.e3,
          (AA_HeatPump.Point_2.P-1.e5)/1.e5, '%d' %2)

```

```

242 ax.text((AA_HeatPump.Point_3.h+1000)/1.e3,
        (AA_HeatPump.Point_3.P-1.e5)/1.e5, '%d' %3)
243 ax.text((AA_HeatPump.Point_3b.h+1000)/1.e3,
        (AA_HeatPump.Point_3b.P-1.e5)/1.e5, '%d' %34)
244 ax.text((AA_HeatPump.Point_4.h+1000)/1.e3,
        (AA_HeatPump.Point_4.P-1.e5)/1.e5, '%d' %4)
245 ax.text((AA_HeatPump.Point_5.h+1000)/1.e3,
        (AA_HeatPump.Point_5.P-1.e5)/1.e5, '%d' %5)
246 ax.text((AA_HeatPump.Point_5b.h+100)/1.e3,
        (AA_HeatPump.Point_5b.P-1.e5)/1.e5, '%d' %51)
247 ax.plot(np.linspace(AA_HeatPump.Point_1.h/1.e3,AA_HeatPump.Point_2.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_1.P)/1.e5,(AA_HeatPump.Point_2.P)/1.e5,
        num=100), 'r-')
248 ax.plot(np.linspace(AA_HeatPump.Point_2.h/1.e3,AA_HeatPump.Point_3.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_2.P)/1.e5,(AA_HeatPump.Point_3.P)/1.e5,
        num=100), 'r-')
249 ax.plot(np.linspace(AA_HeatPump.Point_3.h/1.e3,AA_HeatPump.Point_3b.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_3.P)/1.e5,(AA_HeatPump.Point_3b.P)/1.e5,
        num=100), 'r-')
250 ax.plot(np.linspace(AA_HeatPump.Point_3b.h/1.e3,AA_HeatPump.Point_4.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_3b.P)/1.e5,(AA_HeatPump.Point_4.P)/1.e5,
        num=100), 'r-')
251 ax.plot(np.linspace(AA_HeatPump.Point_4.h/1.e3,AA_HeatPump.Point_5.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_4.P)/1.e5,(AA_HeatPump.Point_5.P)/1.e5,
        num=100), 'r-')
252 ax.plot(np.linspace(AA_HeatPump.Point_5.h/1.e3,AA_HeatPump.Point_5b.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_5.P)/1.e5,(AA_HeatPump.Point_5b.P)/1.e5,
        num=100), 'r-')
253 ax.plot(np.linspace(AA_HeatPump.Point_5b.h/1.e3,AA_HeatPump.Point_1.h/1.e3,
        num=100),
        np.linspace((AA_HeatPump.Point_5b.P)/1.e5,(AA_HeatPump.Point_1.P)/1.e5,
        num=100), 'r-')
254 fig2.savefig('./Pressure_enthalpy_curve.png')
255
256
257
258 T_air_su_cd = AA_HeatPump.air_su_state_cd.T

```

```

259 T_air_su_cd_tp = AA_HeatPump.air_ex_state_cd_sb.T
260 T_air_su_cd_sh = AA_HeatPump.air_ex_state_cd_lat.T
261 T_air_ex_cd = AA_HeatPump.air_ex_state_cd_sh.T
262
263 T_r_su_cd = AA_HeatPump.Point_2.T
264 T_r_su_cd_tp = AA_HeatPump.Point_3.T
265 T_r_su_cd_sc = AA_HeatPump.Point_3b.T
266 T_r_ex_cd = AA_HeatPump.Point_4.T
267
268 Q_dot_cd_sh = AA_HeatPump.Q_dot_cd_sh
269 Q_dot_cd_tp = AA_HeatPump.Q_dot_cd_lat
270 Q_dot_cd_sc = AA_HeatPump.Q_dot_cd_sb
271
272 Q_dot_cd = [0, Q_dot_cd_sh, Q_dot_cd_sh+Q_dot_cd_tp,
             Q_dot_cd_sc+Q_dot_cd_tp+Q_dot_cd_sh]
273 T_r_cd = [T_r_su_cd, T_r_su_cd_tp, T_r_su_cd_sc, T_r_ex_cd]
274 T_air_cd = [T_air_ex_cd, T_air_su_cd_sh, T_air_su_cd_tp, T_air_su_cd]
275
276 fig, ax1 = plt.subplots()
277 ax1.plot(Q_dot_cd, T_r_cd)
278 ax1.plot(Q_dot_cd, T_air_cd)
279 ax1.legend(("T_r_cd", "T_air_cd"))
280 ax1.set_ylabel("T [K]")
281 ax1.set_xlabel("Q_dot_cd [W]")
282 ax1.text((Q_dot_cd[0] + 50), (T_r_cd[0] - 0.3), '%d' % 2)
283 ax1.text((Q_dot_cd[1] + 50), (T_r_cd[1] + 1), '%d' % 3)
284 ax1.text((Q_dot_cd[2] + 50), (T_r_cd[2] - 0.3), '%d' % 34)
285 ax1.text((Q_dot_cd[3] + 50), (T_r_cd[3] - 0.3), '%d' % 4)
286
287 T_air_su_ev = AA_HeatPump.air_su_state_ev.T
288 T_air_su_ev_tp = AA_HeatPump.air_ex_state_ev.T
289 T_air_ex_ev = AA_HeatPump.air_ex_state_ev_sat.T
290
291 T_r_su_ev = AA_HeatPump.Point_5.T
292 T_r_su_ev_sh = AA_HeatPump.Point_5b.T
293 T_r_ex_ev = AA_HeatPump.Point_1.T
294
295 Q_dot_ev_tp = AA_HeatPump.Q_dot_ev_lat
296 Q_dot_ev_sh = AA_HeatPump.Q_dot_ev_sh
297
298 Q_dot_ev = [0, Q_dot_ev_tp, Q_dot_ev_tp + Q_dot_ev_sh]
299 T_r_ev = [T_r_su_ev, T_r_su_ev_sh, T_r_ex_ev]
300 T_air_ev = [T_air_ex_ev, T_air_su_ev_tp, T_air_su_ev]

```

```

301
302 fig, ax1 = plt.subplots()
303 ax1.plot(Q_dot_ev, T_r_ev)
304 ax1.plot(Q_dot_ev, T_air_ev)
305 ax1.legend(("T_r_ev", "T_air_ev"))
306 ax1.set_ylabel("T [K]")
307 ax1.set_xlabel("Q_dot_ev [W]")
308 ax1.text((Q_dot_ev[0]+50), (T_r_ev[0]-0.3), '%d' % 5)
309 ax1.text((Q_dot_ev[1]+50), (T_r_ev[1]-0.3), '%d' % 51)
310 ax1.text((Q_dot_ev[2]+50), (T_r_ev[2]-0.3), '%d' % 1)

```

## ThermoState

```

1 """
2 Title: ThermoState
3 """
4
5 import CoolProp.CoolProp as CP
6
7 class ThermoState():
8
9     def __init__(self, fluid):
10         self.fluid = fluid
11
12     def pT_ThermoState(self, P, T):
13         self.T = T
14         self.TdegC = self.T - 273.15
15         self.P = P
16         self.rho = CP.PropsSI("D", "T", T, "P", P, self.fluid)
17         self.vmass = 1/self.rho
18         self.h = CP.PropsSI("H", "T", T, "P", P, self.fluid)
19         self.s = CP.PropsSI("S", "T", T, "P", P, self.fluid)
20         self.Cp = CP.PropsSI("CPMASS", "T", T, "P", P, self.fluid)
21         self.Cv = CP.PropsSI("CVMASS", "T", T, "P", P, self.fluid)
22         self.gamma = self.Cp/self.Cv
23         self.P_critFlow = ((2/(self.gamma+1))**\
24                             (self.gamma/(self.gamma-1+1e-09)))*self.P
25
26     def ps_ThermoState(self, P, s):
27         self.P = P
28         self.s = s
29         self.T = CP.PropsSI("T", "S", s, "P", P, self.fluid)
30         self.TdegC = self.T - 273.15

```

```

31 self.rho = CP.PropsSI("D", "S", s, "P", P, self.fluid)
32 self.vmass = 1/self.rho
33 self.h = CP.PropsSI("H", "S", s, "P", P, self.fluid)
34 self.Cp = CP.PropsSI("CPMASS", "S", s, "P", P, self.fluid)
35 self.Cv = CP.PropsSI("CVMASS", "S", s, "P", P, self.fluid)
36 self.gamma = self.Cp/self.Cv
37
38 ##Evaluate saturation enthalpies to determine if a 2-phase
    quality can be determined.
39 #If statement compares the enthalpies and calculate if needed.
40 Pcrit = CP.PropsSI("Pcrit", self.fluid)
41
42 if P<Pcrit:
43     h_sl = CP.PropsSI("H", "Q", 0, "P", P, self.fluid)
44     h_sg = CP.PropsSI("H", "Q", 1, "P", P, self.fluid)
45     if self.h>h_sl and self.h<h_sg:
46         self.x = CP.PropsSI("Q", "H", self.h, "P", P,
            self.fluid)
47
48 self.P_critFlow = ((2/(self.gamma+1))**\
49     (self.gamma/(self.gamma-1+1e-09)))*self.P
50
51 def ph_ThermoState(self, P, h):
52     self.P = P
53     self.h = h
54     self.T = CP.PropsSI("T", "H", h, "P", P, self.fluid)
55     self.TdegC = self.T - 273.15
56     self.rho = CP.PropsSI("D", "H", h, "P", P, self.fluid)
57     self.vmass = 1/self.rho
58     self.s = CP.PropsSI("S", "H", h, "P", P, self.fluid)
59     self.Cp = CP.PropsSI("CPMASS", "H", h, "P", P, self.fluid)
60     self.Cv = CP.PropsSI("CVMASS", "H", h, "P", P, self.fluid)
61     self.gamma = self.Cp/self.Cv
62
63 ##Evaluate saturation enthalpies to determine if a 2-phase
    quality can be determined.
64 #If statement compares the enthalpies and calculate if needed.
65 Pcrit = CP.PropsSI("Pcrit", self.fluid)
66
67 if P<Pcrit:
68     h_sl = CP.PropsSI("H", "Q", 0, "P", P, self.fluid)
69     h_sg = CP.PropsSI("H", "Q", 1, "P", P, self.fluid)
70     if h>h_sl and h<h_sg:

```

```

71         self.x = CP.PropsSI("Q", "H", h, "P", P, self.fluid)
72
73     self.P_critFlow = ((2/(self.gamma+1))**\
74         (self.gamma/(self.gamma-1+1e-09)))*self.P
75
76     def px_ThermoState(self, P, x):
77         self.P = P
78         self.x = x
79         self.h = CP.PropsSI("H", "P", P, "Q", x, self.fluid)
80         self.T = CP.PropsSI("T", "Q", x, "P", P, self.fluid)
81         self.TdegC = self.T - 273.15
82         self.rho = CP.PropsSI("D", "Q", x, "P", P, self.fluid)
83         self.vmass = 1/self.rho
84         self.s = CP.PropsSI("S", "Q", x, "P", P, self.fluid)
85         self.Cp = CP.PropsSI("CPMASS", "H", self.h, "P", P, self.fluid)
86         self.Cv = CP.PropsSI("CVMASS", "H", self.h, "P", P, self.fluid)
87         self.gamma = self.Cp / self.Cv
88
89     def vs_ThermoState(self, v, s):
90         self.v = v
91         self.s = s
92         self.rho = 1/self.v
93         self.h = CP.PropsSI("H", "S", s, "D", self.rho, self.fluid)
94         self.P = CP.PropsSI("P", "S", s, "D", self.rho, self.fluid)
95         self.T = CP.PropsSI("T", "S", s, "D", self.rho, self.fluid)
96         self.TdegC = self.T - 273.15
97         self.x = CP.PropsSI("Q", "S", s, "D", self.rho, self.fluid)
98         self.Cp = CP.PropsSI("CPMASS", "H", self.h, "P", self.P,
99             self.fluid)
100        self.Cv = CP.PropsSI("CVMASS", "H", self.h, "P", self.P,
            self.fluid)
        self.gamma = self.Cp / self.Cv

```

## EffCompressor

```

1  """
2  Title: EffCompressor
3  """
4
5
6  from Modules.ThermoState import ThermoState
7  from scipy.optimize import fsolve
8  import math

```



```

9
10 class EffCompressor:
11
12     def __init__(self, displ, fluid, rv_in, A_leak, T_loss, AU, T_amb):
13         self.displ = displ
14         self.fluid = fluid
15         self.rv_in = rv_in
16         self.A_leak = A_leak
17         self.T_loss = T_loss
18         self.AU_amb = AU[0]
19         self.AU_su = AU[1]
20         self.AU_ex = AU[2]
21         self.T_amb = T_amb
22
23     def Compressor(self, x, su_state, ex_is_state, N):
24         self.su_state = su_state
25         "Guess for fsolve function, exhaust temperature"
26         T_ex_guess = x[0]
27         self.T_cp_guess = x[1]
28
29         "We make sure we stay within a reasonable range of values"
30         if T_ex_guess > 367.85 or T_ex_guess < 219.9:
31             self.res1 = 0
32             self.res2 = 0
33             self.Statut_cp = '!!!!!!!!!!!!!!!!Problem!!!!!!!!!!!!!!!!'
34             return self.res1, self.res2
35
36         self.ex_state_guess = ThermoState(self.fluid)
37         self.ex_state_guess.pT_ThermoState(ex_is_state.P, T_ex_guess)
38
39         "Frequency of rotation"
40         self.N = N
41
42         "Losses of mechanical power"
43         self.w_loss = 2*math.pi*self.N*self.T_loss
44
45         "Heating by metallic parts"
46         self.Qsu = self.AU_su*(su_state.T - self.T_cp_guess)
47         self.su1_state = ThermoState(self.fluid)
48         self.su1_state.pT_ThermoState(su_state.P, su_state.h-self.Qsu)
49
50         "Mixing with leakage"
51         self.thr_state = ThermoState(self.fluid)

```

52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90

```
self.thr_state.ps_ThermoState(self.su1_state.P,
    self.ex_state_guess.s)
C_thr = math.sqrt(2*(self.ex_state_guess.h-self.thr_state.h))
V_dot_leak = C_thr*self.A_leak
self.m_dot_leak = V_dot_leak*self.thr_state.rho

def mixing(g):
    self.m_dot_in = self.N*self.displ*g[0]
    self.Mdot = self.m_dot_in - self.m_dot_leak

    "Energy equilibrium"
    h_su2 =
        (self.Mdot*self.su1_state.h+self.m_dot_leak*self.ex_state_guess.h)/self.Mdot

    self.su2_state = ThermoState(self.fluid)
    self.su2_state.ph_ThermoState(self.su1_state.P, h_su2)
    return g[0] - self.su2_state.rho

rho_init = self.su1_state.rho
g = [rho_init]
fsolve(mixing, g)

"Volumetric efficiency"
m_dot_cp_th = self.N*self.displ*su_state.rho
self.e_v = self.Mdot/m_dot_cp_th

"Iisentropic compression"
v_ex_is = self.su2_state.vmass/self.rv_in
state_ex_is = ThermoState(self.fluid)
state_ex_is.vs_ThermoState(v_ex_is, self.su2_state.s)
self.w_dot_is = self.m_dot_in*(state_ex_is.h-self.su2_state.h)

"Isochoric compression"
self.w_dot_isoc = self.m_dot_in*v_ex_is*(ex_is_state.P -
    state_ex_is.P)

"Exhaust"
h_out = self.su2_state.h + (self.w_dot_is +
    self.w_dot_isoc)/self.m_dot_in
self.ex_state_bc = ThermoState(self.fluid)
self.ex_state_bc.ph_ThermoState(self.ex_state_guess.P, h_out)

"Cooling by metallic parts"
```

```

91 self.Qex = self.AU_ex*(self.ex_state_guess.T - self.T_cp_guess)
92 self.ex_state = ThermoState(self.fluid)
93 self.ex_state.ph_ThermoState(self.ex_state_guess.P,
    self.ex_state_bc.h-self.Qex)
94
95 "Ambiant loss"
96 self.Qamb =
    self.AU_amb*((self.T_amb+self.T_cp_guess)/2-self.T_cp_guess)
97
98 "Power"
99 self.Wdot = self.w_dot_is + self.w_dot_isoc + self.w_loss
100 self.w_dot_is_tot = self.Mdot*(ex_is_state.h-su_state.h)
101 self.e_is = self.w_dot_is_tot /self.Wdot
102
103 "Residuals"
104 res1 = self.ex_state_guess.T-self.ex_state_bc.T
105 res2 = self.Qsu + self.Qex + self.Qamb + self.w_loss
106 self.Statut_cp = 'No problem'
107 return res1, res2
108
109 def solveComp(self, su_state, P_ex, N):
110     ex_is_state = ThermoState(self.fluid)
111     ex_is_state.ps_ThermoState(P_ex, su_state.s)
112
113     h_out_guess = su_state.h + (ex_is_state.h-su_state.h)/0.8
114     ex_state_guess = ThermoState(self.fluid)
115     ex_state_guess.ph_ThermoState(ex_is_state.P, h_out_guess)
116     T_ex_guess = ex_state_guess.T
117
118     if T_ex_guess > 367.85 or T_ex_guess < 219.9:
119         self.Statut_cp = '!!!!!!!!!!!!!!!!Problem!!!!!!!!!!!!!!!!'
120         return
121     else:
122         args = (su_state, ex_is_state, N)
123         x = [T_ex_guess, 273.15 + 30]
124         fsolve(self.Compressor, x, args=args)

```

#### HX\_LMTD\_calcUA

```

1  """
2  Title: EffHXCrossFlow
3  """
4

```

```

5 import math
6 import numpy as np
7 from scipy.optimize import fsolve
8 from math import log
9 #%%
10 class HX_LMTD_calcUA:
11     def __init__(self, typeHX):
12         self.typeHX = typeHX
13         #=====
14     def F_lmtd2(self, R,P):
15         #-----
16         def res_NTU_crossflow(Ntu, epsilon, C_r):
17             Ntu = max(1e-4, Ntu)
18             epsilon_g =
19                 1-np.exp((1/C_r)*Ntu**0.22*(np.exp(-C_r*Ntu**0.78)-1))
20             res = (epsilon-epsilon_g)
21             return res
22
23         #R = (T_h_i - T_h_ex)/(T_c_ex - T_c_i) --> must be positive
24         [0; infinte [
25         #P = (T_c_ex - T_c_i)/(T_h_i - T_c_i) --> must be within 0
26         and 1
27         #-----
28         #R and P must be positive
29         R = abs(R)
30         P = abs(P)
31         #-----
32         R_min = 1e-4
33         R_max = 100
34         R = max(R_min, min(R_max, R))
35         # print("R =", R)
36         if R < 0.41:
37             P_max_for_given_R = 0.99999
38         else:
39             P_max_for_given_R = 0.995*((0.966475661367996)*R**3 +
40                 (-1.431274296912407)*R**2 + (0.247230065033875)*R +
41                 (0.309118607270897)) / (R**4 +
42                 (-1.766309686745371)*R**3 + (1.287179055148762)*R**2
43                 + (-0.902512766020191)*R + (0.484880205333508))
44         P = max(0, min(P_max_for_given_R, P))
45
46         if R <= 1: # Cdot_min = Cdot_h
47             epsilon = P

```

```

41         Cr = R
42         Pbis = P
43         Rbis = R
44     else: # Cdot_max = Cdot_c
45         epsilon = P*R
46         Cr = 1/R
47         Pbis = P*R
48         Rbis = 1/R
49
50     f = lambda x: res_NTU_crossflow(x, epsilon, Cr)
51
52     out_fsolve = fsolve(f, 1, full_output= 1)#, args=(),
53         fprime=None, full_output=0, col_deriv=0,
54         xtol=1.49012e-08, maxfev=0, band=None, epsfcn=None,
55         factor=100, diag=None)
56     NTU, res_NTU, flag_ntu = float(out_fsolve[0]),
57         float(out_fsolve[1].get('fvec')), out_fsolve[2]
58     if abs(res_NTU)< 1e-2 and flag_ntu > 0:
59         if R==1:
60             F=P/NTU/(1-P)
61         elif P <= 1e-04:
62             F = 1
63         else:
64             F = epsilon*math.log((Pbis-1)/(Rbis*Pbis
65                 -1))/NTU/Pbis/(Rbis-1);
66         flag = flag_ntu
67     else:
68         #disp(['Prob compute NTU with F_lmt2(' num2str(R) ', '
69             num2str(P) ')'])
70         F = 1
71         flag = flag_ntu
72     return F, flag
73
74 #=====
75 def calcUA(self, su_state_h, ex_state_h, m_dot_h_su, su_state_c,
76     ex_state_c, m_dot_c_su):
77     self.m_dot_h = m_dot_h_su
78     self.m_dot_c = m_dot_c_su
79     self.su_state_h = su_state_h
80     self.ex_state_h = ex_state_h
81     self.su_state_c = su_state_c
82     self.ex_state_c = ex_state_c
83     #=====
84     #Calculate Q from inputs:

```

```

77     self.Q_dot_h = m_dot_h_su*(su_state_h.h - ex_state_h.h)
78     self.Q_dot_c = m_dot_c_su*(ex_state_c.h - su_state_c.h)
79     #Raise Error if the heat balance does not match
80     if not (self.Q_dot_h - self.Q_dot_c) < 1e-06:
81         raise ValueError("Hot and Cold side Heat Exchange does
82                             not match")
83
84     Thi = su_state_h.T
85     Tci = su_state_c.T
86     Tho = ex_state_h.T
87     Tco = ex_state_c.T
88     DTA = max(Thi - Tco, 1e-02)
89     DTB = max(Tho - Tci, 1e-02)
90
91     #-----
92     if DTA == DTB:
93         LMTD = DTA
94     else:
95         try:
96             LMTD = (DTA-DTB)/log(abs(DTA/DTB))
97         except ValueError as VE:
98             print(VE)
99
100    #-----
101    # F correction factor for LMTD method:
102    if self.typeHX == "CrossFlow":
103        if Tco == Tci:
104            self.R = 101
105            self.F = 1
106        else:
107            self.R = (Thi - Tho) / (Tco - Tci)
108            self.P = (Tco - Tci) / (Thi - Tci)
109            self.F = self.F_lmt2(self.R, self.P)[0]
110    else:
111        self.F = 1
112
113    #-----
114    self.UA = self.Q_dot_h/(self.F*LMTD)
115    self.LMTD = LMTD

```

## DeltaP

```

1  """
2  Title: DeltaP
3  """
4

```

```

5 import CoolProp.CoolProp as CP
6 import math
7 import numpy as np
8
9 class DeltaP():
10
11     def __init__(self, fluid):
12         self.fluid = fluid
13
14     def calcul_lambda_turb(self, Re, k, D_h):
15         lambda_turb = 0.05 #first guess
16         delta = 1 #initializing the iterations
17         tol = 1e-8
18         while (abs(delta)>tol):
19             lambda_tmp = lambda_turb
20             lambda_turb =
21                 1/(np.sqrt(-2*math.log10((2.51/(Re*np.sqrt(lambda_turb)))
22                 + k/(3.7*D_h))))
23             # Source of the formula: Go to report
24
25             delta = lambda_turb - lambda_tmp
26         return lambda_turb
27
28     def calcul_delta_P_T(self, T, P, M_dot, N_pipe, l, h, L):
29         M_dot_per_pipe = M_dot/N_pipe #Kg/s
30         rho_r = CP.PropsSI('D', 'T', T, 'P', P, self.fluid)
31         V_dot = M_dot_per_pipe/rho_r
32         section = l*h #m^2
33
34         "Hydraulic diameter"
35         D_h = (2*l*h)/(h+l) #Assumption of rectangular tubes
36
37         "Computation of the Reynolds number"
38         V = V_dot/section #m/s
39         dynamic_viscosity_r = CP.PropsSI('V', 'T', T, 'P', P,
40             self.fluid)
41         Re = (V*D_h*rho_r)/dynamic_viscosity_r
42         if Re<2320:
43             lambda_lam = 0.316*Re**(-0.25)
44             delta_P = lambda_lam*L/D_h*rho_r*(V**2)/2
45         else:
46             k = 0.001/1000
47             lambda_turb = self.calcul_lambda_turb(Re, k, D_h)

```

```

45     delta_P = lambda_turb*L/D_h*rho_r*(V**2)/2
46     return delta_P
47
48 def calcul_delta_P_X(self, X, P, M_dot, N_pipe, l, h, L):
49     M_dot_per_pipe = M_dot/N_pipe
50     rho_r = CP.PropsSI('D', 'Q', X, 'P', P, self.fluid)
51     V_dot = M_dot_per_pipe/rho_r
52     section = l*h #m^2
53
54     "Hydraulic diameter"
55     D_h = (2*l*h)/(h+l)
56
57     "Computation of the Reynolds number"
58     V = V_dot/section
59     dynamic_viscosity_r = CP.PropsSI('V', 'Q', X, 'P', P,
60     self.fluid)
61     Re = (V*D_h*rho_r)/dynamic_viscosity_r
62     if Re<2320 :
63         lambda_lam = 0.316*Re**(-0.25)
64         delta_P = lambda_lam*L/D_h*rho_r*(V**2)/2
65     else:
66         k = 0.001/1000
67         lambda_turb = self.calcul_lambda_turb(Re, k, D_h)
68         delta_P = lambda_turb*L/D_h*rho_r*(V**2)/2
69     return delta_P

```

## CalcU

```

1  """
2  Title: CalcU
3  """
4
5  def CalcU(Mdot_n, Mdot, U_n):
6      return U_n*(Mdot/Mdot_n)**0.65
7      #return U_n

```

## Accumulator

```

1  """
2  Title : Accumulator
3  """
4  from CoolProp.CoolProp import PropsSI

```



```

5 import math
6 from Modules.ThermoState import ThermoState
7
8
9 class Accumulator :
10 def __init__(self, fluid):
11     self.fluid = fluid
12
13 def Pressure_drop_sudden(self, rho, v_1, v_2):
14     #Lacking data about the architecture of the accumulator
15     #Gives very high pressure drops so the function is NOT used
16     K = 1.84
17     DP = K*0.5*rho*(v_1-v_2)**2
18     return DP
19
20 def Pressure_drop_bend(self, mu, rho, v, L, D):
21     #https://neutrium.net/fluid-flow/pressure-loss-from-fittings-excess-head-k-
22     Re = rho*v*D/mu
23     if Re < 10e5:
24         f = 0.316*Re**(-0.25)
25     else:
26         f = 0.0032 + 0.221/Re**(0.237)
27     K = f*L/D
28     DP = K*0.5*rho*v**2
29     return DP
30
31 def Pressure_drop_friction(self, mu, rho, v, L, D):
32     Re = rho*v*D/mu
33     if Re < 10e5:
34         f = 0.316*Re**(-0.25)
35     else:
36         f = 0.0032 + 0.221/Re**(0.237)
37     DP = f*0.5*rho*v**2*L/D
38     return DP
39
40 def Pressure_drop(self, m_dot_r, point_A):
41     D = 1.6e-2
42     #D_acc = 8e-2 Value used for the Pressure_drop_bend function
43     pi = math.pi
44     fluid = 'R1234yf'
45     rho = 1100 #kg/m^3
46
47     #-----flux 1-----

```

```

48 L_AB = 5e-2
49 L_bend = pi*9e-2/2 #Length thanks to the bend radius
      (assumption)
50
51 v_A = m_dot_r/rho/(pi*D**2/4)
52 #v_AA = m_dot_r/rho/(pi*D_acc**2/4) Speed normally used for the
      Pressure_drop_bend function
53 mu_A = PropsSI('VISCOSITY','P',point_A.P,'T',point_A.T,fluid)
54
55 P_dyn_A = 0.5*rho*v_A**2
56 P_stag_A = P_dyn_A + point_A.P
57
58 DP1 = self.Pressure_drop_bend(mu_A, rho, v_A, L_bend, D)
59 + self.Pressure_drop_friction(mu_A,rho,v_A,L_AB,D)
60 + self.Pressure_drop_bend(mu_A, rho, v_A, L_bend,D)
61 #+ self.Pressure_drop_sudden(point_A.rho,v_A,v_AA)
62 #+ self.Pressure_drop_sudden(point_A.rho,v_AA,v_A)
63
64 P_stag_B = P_stag_A - DP1
65 P_B = P_stag_B
66
67 point_B = ThermoState(self.fluid)
68 point_B.pT_ThermoState(P_B, point_A.T)
69
70 #-----flux 2-----
71 L_BC = 20e-2
72 v_B = m_dot_r/rho/(pi*D**2/4)
73 mu_B = PropsSI('VISCOSITY','P',point_B.P,'T',point_B.T,fluid)
74
75 DP2 =self.Pressure_drop_friction(mu_B, rho, v_B, L_BC, D)
76 + self.Pressure_drop_bend(mu_B, rho, v_B, L_bend,D)
77 # + self.Pressure_drop_sudden(point_B.rho, v_AA,v_B)
78
79 P_stag_C = P_stag_B - DP2
80
81 P_dyn_C = 0.5*rho*v_B**2
82 P_C = P_stag_C - P_dyn_C
83
84 point_C = ThermoState(self.fluid)
85 point_C.pT_ThermoState(P_C, point_A.T)
86
87 #-----flux 3-----

```

```

88 #The height of the fluid in the accumulator usefull to compute
    the rise
89 #of pressure at point C is not known. Thus this is not used
    here.
90
91 #-----flux 4-----
92 #Not used due to lack of data
93 "DP4 = Pressure_drop_sudden()"
94 "P_D = P_B -DP4"
95
96 "Point_D = ThermoState(self.fluid)"
97 "Point_D.pT_ThermoState(P_D, Point_A.T)"
98
99 #-----flux 5-----
100 L_CD = L_BC
101 v_C = m_dot_r/rho/(pi*D**2/4)
102 mu_C = PropsSI('VISCOSITY','P',point_C.P,'T',point_C.T,fluid)
103
104 DP5 = self.Pressure_drop_bend(mu_C, rho, v_C, L_bend,D)
105 + self.Pressure_drop_friction(mu_C,rho, v_C, L_CD, D)
106
107 P_D = P_C - DP5
108
109 point_D = ThermoState(self.fluid)
110 point_D.pT_ThermoState(P_D, point_A.T)
111
112 #-----flux 6-----
113 L_DE = L_AB
114 v_D = m_dot_r/rho/(pi*D**2/4)
115 mu_D = PropsSI('VISCOSITY','P',point_D.P,'T',point_D.T,fluid)
116
117 P_dyn_D = 0.5*rho*v_D**2
118 P_stag_D = P_D + P_dyn_D
119 DP6 = self.Pressure_drop_friction(mu_D,rho,v_D,L_DE,D)
120 + self.Pressure_drop_bend(mu_D, rho, v_D , L_bend,D)
121
122 P_stag_E = P_stag_D - DP6
123 P_dyn_E = 0.5*rho*v_D**2
124 P_E = P_stag_E - P_dyn_E
125
126 point_E = ThermoState(self.fluid)
127 point_E.pT_ThermoState(P_E, point_A.T)
128

```

```
return point_E
```

## IdealExpValve

```
1 """
2 Title: IdealExpValve
3 """
4
5 from Modules.ThermoState import ThermoState
6
7 class IdealExpValve:
8     def __init__(self, fluid):
9         self.fluid = fluid
10
11     def IdealExp(self, su_state, P_exp, Mdot_su):
12         h_ex = su_state.h
13         ex_state = ThermoState(self.fluid)
14         ex_state.ph_ThermoState(P_exp, h_ex)
15         self.ex_state = ex_state
16         self.Mdot_ex = Mdot_su
```