Preface

This Master's thesis is part of the Sustainable Energy Master's program at the Norwegian University of Science and Technology (NTNU). The research work was conducted for the Department of Energy during the spring of 2022, and the thesis was completed in the autumn of 2023.

I would like to thank my supervisor, Armin Hafner, for giving me the opportunity to undertake this research project. His expert advice and encouragement have been of great help from start to finish. Further, I would like to express my great gratitude to my co-supervisor, Mihir Mouchum Hazarika, for his unwavering patience and guidance throughout the process.

I am also thankful to Ángel Álvarez Pardiñas for teaching me how to run the test rig and for always being present whenever help was needed.

Lastly, I would like to thank my fellow students Jan Bengsch and Lukas Köster for helping me perform the necessary experiments and for providing me with a good study environment. It has been a great pleasure working with all of you.

Trondheim, October 2023

Johan Hafsås

Abstract

This thesis presents experimental investigations on a novel two-stage evaporator integrated into a transcritical CO_2 refrigeration system. The motivation behind this research is to improve the energy efficiency of commercial heat pump chillers using CO_2 as a refrigerant. The objective of the thesis is to evaluate the performance of the two-stage evaporator in single-stage and two-stage modes, as well as to assess the pipe configurations attached to the gravity stage of the evaporator.

The experimental investigations were conducted using a test facility at the Norwegian University of Science and Technology (NTNU). The results obtained from the experiments were compared to the original design criteria of the two-stage evaporator.

The experimental results showed that the gravity loop and two-stage evaporator concept is feasible and has the potential to yield positive results on CO_2 refrigeration systems. However, there were some areas where further improvement is needed. The gravity loop only achieved a cooling capacity of approximately half of the expected capacity. This is attributed to the high pressure drop across the loop causing an insufficient CO_2 mass flow in the evaporator.

The two-stage evaporator concept was shown to be functional, with higher water mass flow rates providing a more even distribution of cooling capacity in the two stages. Overall, the results of this thesis demonstrate the feasibility of the gravity loop and two-stage evaporator concept for CO_2 refrigeration systems, but further optimization is required to improve its performance.

Sammendrag

I denne masteroppgaven er det blitt utført eksperimentelle forsøk på en ny totrinns fordamper integrert i et transkritisk CO_2-kjøleanlegg. Motivet for forsøkene er å se på muligheten for å forbedre energieffektiviteten i kommersielle varmepumpe-/kjøleanlegg ved bruk av CO_2 som kjølemedie. Oppgavens mål er å evaluere ytelsen i en totrinns fordamper, både ved bruk av kun ett fordampningstrinn, og ved bruk av begge fordampningstrinnene. Rørkonfigurasjonen til den selvsirkulerende gravitasjonsloopen blir også vurdert.

De eksperimentelle forsøkene ble utført ved et testanlegg ved Norges Teknisk-Naturvitenskapelige Universitet (NTNU). Resultatene fra forsøkene ble sammenlignet med de opprinnelige designkriteriene for totrinnsfordamperen.

De eksperimentelle resultatene viste at både gravitasjonsloopen og totrinns fordamperkonseptet er gjennomførbart og har potensiale til å gi positive resultater. Før man kommer dit kreves ytterligere forbedringer på konseptet. Gravitasjonsloopen oppnådde omtrent halvparten av forventet kjølekapasitet. Dette var hovedsaklig grunnet det høye trykktapet over kretsen som førte til utilstrekkelig CO_2-massestrøm i fordamperen.

Konseptet med totrinns fordamper viste spesielt positive resultater ved høye vannmengder gjennom fordamperen. Dette ga en jevnere fordeling av kjølekapasitet over de to fordampningstrinnenen. Samlet sett viser resultatene i denne avhandlingen at gravitasjonsloopen og totrinns fordampning er gjennomførbart for CO_2-kjølesystemer, men videre optimalisering er nødvendig for å forbedre ytelsen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The Intergovernmental Panel on Climate Change (IPCC) reported in their sixth assessment report that the amount of greenhouse gases in the atmosphere is steadily increasing, and is currently at its highest level for the last two million years [1]. To combat the potentially devastating effects of climate change, authorities are ordering new laws and regulations to reduce emissions. One such regulation is the EU F-Gas Regulation from 2015 [2], which will effectively ban the use of refrigerants with a global warming potential (GWP) of 2,500 or above by 2020 for certain refrigeration units, as well as initiate the phase-out of hydrofluorocarbons (HFCs) within 2030 [3].

This incentive has led researchers and engineers to revive their interest in natural refrigerants such as Ammonia (NH_3), Carbon Dioxide (CO_2) and Hydrocarbons. These refrigerants are cheap, easily available, have a close to negligible GWP, have good thermophysical properties, and have no risks of future restrictions or environmental taxation [4].

CO_2 is often preferred above other natural refrigerants in sectors such as the retail sector and hotel sector [5, 6]. This is mainly due to its superior safety features, being not acutely toxic, non-flammable, and odourless. It also has excellent thermophysical properties in terms of high-temperature heat rejection and low refrigeration temperatures [4].

CO_2 has a very low critical temperature of approximately 31 °C, making it ill-suitable in areas with ambient temperatures frequently above this point. However, due to recent advancements in the fields of CO_2 systems for high-performance buildings, solutions for overcoming this hurdle have been developed. The most notable effects of improvement have come from the introduction of parallel compressors and multi-ejectors to the system layout [6, 7].

The MultiPACK project is a project funded by the EU with the aim of building confidence in integrated heating, ventilation, air conditioning and refrigeration (HVAC&R) solutions based on new CO_2 technologies [8]. The project is primarily focused on Southern Europe where high ambient temperatures pose a challenge to HVAC&R technologies.

The MultiPACK reversible chiller/heat pump is one of the solutions included in the Multi-PACK project. It is an integrated unit providing heating, air conditioning (AC), and domestic hot water production (DHW) for high energy-demanding buildings such as hotels, gyms, and spas. One of the special features included in the reversible chiller/heat pump is the utilization of two evaporators where the first evaporator is driven by a self-circulating gravity loop, and the second is driven by a two-phase multiejector. The evaporators are connected in series on the waterside to cool down water for AC chilling in two different temperature stages. A detailed explanation of the concept can be read in Section 2.

Currently, the capacity of such heat pump chillers is limited due to the space requirements for the plate heat exchangers. A new concept of a single compact two-stage plate heat exchanger would allow to significantly increase the volume specific cooling capacity of CO_2 heat pump chillers built in a modular way.

1.2 Research question and objective

In the following thesis, a novel two-stage evaporator able to integrate a natural circulation loop, and an ejector-assisted circulation loop, is fitted to an existing CO_2-rig and tested to further improve the concept of two-stage evaporation.

The main objective of the thesis is to run experimental investigations on the novel two-stage evaporator. The experimental investigations will involve testing the performance of the evaporator in single-stage mode and in two-stage mode. Evaluations will also be performed on the configuration of the self circulating gravity loop. The results will be compared to the original design criteria of the two-stage evaporator.

Parallel to the writing of this thesis, fellow student Jan Bengsch is writing a thesis on the simulation of the very same experimental setup called "Investigation and analysis of a CO_2 heat pump chiller with novel two-stage evaporator" [9]. The simulation study aims to provide a theoretical analysis and complement the experimental results obtained in this thesis. The combination of experimental and simulation studies will allow for a comprehensive understanding of the performance of the two-stage evaporator.

Successful experimental results have the potential to encourage manufacturers to produce and promote the product. This, in turn, will contribute to enhancing the energy efficiency of commercial heat pump chillers utilizing CO_2 as an environmentally friendly refrigerant.

2 Theory

In this section, important facts and theories will be provided and explained for the reader to fully comprehend all relevant concepts.

2.1 Refrigeration principles

The second law of thermodynamics states that heat is naturally transferred in the direction of decreasing temperature. Therefore, in order to cool a reservoir using a warmer source, energy must be added to the process. One such process is the closed vapor-compression refrigeration process, commonly known as the heat pumping process.

The heat pumping process is a system configured with four main components: a compressor, a condenser, an expansion valve and an evaporator. These components are coupled together in a closed loop circuit as illustrated in Figure 1.

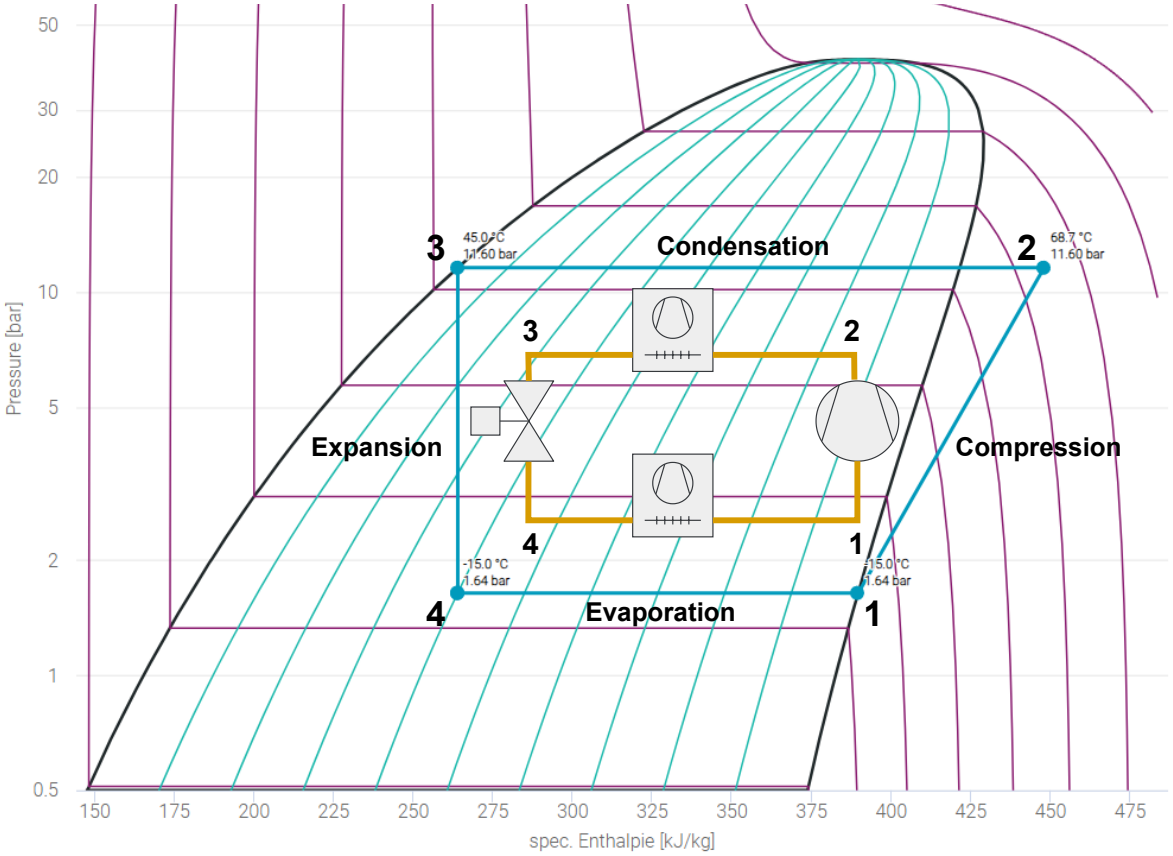


Figure 1: A conventional heat pump configuration with a corresponding log(p)-h diagram.

Inside the circuit, a refrigerant is used to extract, carry, and release heat from the cold side to the warm side of the system. Figure 1 represents a simple conventional refrigeration cycle and its corresponding log(p)-h diagram. With Figure 1 in mind, the four stages of an ideal heat pumping process are explained as follows:

- **1 → 2 Compression:**
  Low-pressure refrigerant gas is sucked into a compressor in the form of superheated

vapor. The vapor is compressed, increasing the pressure and saturation temperature of the refrigerant. The vapor exits the compressor with a saturation temperature exceeding the ambient temperature.

- **2 → 3 Condensation:**
  The high-pressure and high-temperature vapor enters a condenser in a superheated state. The superheated vapor is first cooled down to saturation temperature before starting the condensation process. During condensation, the refrigerant releases large amounts of latent heat to the ambient air. The refrigerant leaves the condenser in a liquid state.
- **3 → 4 Expansion:**
  After condensation, the high-pressure liquid refrigerant flows through a high-pressure expansion valve, which reduces its pressure and allows it to expand into a mixture of liquid and low-pressure gas.
- **4 → 1 Evaporation:**
  The mixture of low-pressure and low-temperature liquid and gas is fed into an evaporator at a pressure corresponding to a saturation temperature lower than the ambient temperature surrounding the evaporator. Heat is transferred from the ambient to the refrigerant as it evaporates from low-pressure liquid to low-pressure gas.

To easily evaluate a heat pumping cycle, the log(p)-h diagram is often used. Figure 1 represents a log(p)-h diagram corresponding to the heat-pumping configuration.. The specific enthalpy (kJ/kg), the "energy content" of the refrigerant, is plotted on the x-axis, and the logarithmic pressure (bar) is plotted on the y-axis.

The efficiency of a refrigeration cycle can be calculated using the Coefficient of Performance (COP). The measure of calculating the efficiency depends on the use case of the unit. It can be used for heating, cooling, or a combination of both. The corresponding efficiency is COP_{HP}, COP_R, and COP_{comb} respectively, and they are calculated as following [10]:

$$COP_{HP} = \frac{useful\ heatpumping\ capacity}{total\ input\ power} = \frac{q_H}{w_{net,in}} = \frac{h_2 - h_3}{h_2 - h_1} \tag{1}$$

$$COP_R = \frac{usefull\ refrigerating\ capacity}{total\ input\ power} = \frac{q_L}{w_{net,in}} = \frac{h_1 - h_4}{h_2 - h_1} \tag{2}$$

$$COP_{comb} = COP_{HP} + COP_R = \frac{q_H + q_L}{w_{net,in}} = \frac{(h_2 - h_3) + (h_1 - h_4)}{h_2 - h_1} \tag{3}$$

where q_H and q_L is the specific heating and cooling capacity (kJ/kg) respectively, $w_{net,in}$ is the required work input (kJ/kg) to the system, and h_n is the enthalpy, where n refers to the state points in Figure 1.

2.1.1 Subcritical

The conventional heat pump usually operates in a subcritical process which is a thermodynamic process that occurs within the two-phase region (between the critical pressure point and the triple point). In this process, the refrigerant is in the liquid or gas phase, and there is a clear distinction between the phases. The behavior of the refrigerant in a subcritical process is similar to an ideal gas, and the properties can be described using the ideal gas law [11].

2.1.2 Transcritical

A transcritical process occurs in a system where the heat rejection of the refrigerant happens at a pressure higher than the critical pressure. In this process, the fluid does not undergo a phase change, and there is no clear distinction between the liquid and gas phases. Since the vapor does not condense, a transcritical system use a gas cooler instead of a condenser. The heat is rejected at gliding temperatures, causing a big difference in inlet and outlet temperature in the gas cooler. The gliding temperatures can be beneficial when used to heat water or liquids from lower to higher temperatures. By using a counter flowing heat exchanger to heat tap water from 10 °C to 70 °C, the temperature difference (ΔT) between the refrigerant and the water stays relatively constant in a transcritical system compared to a subcritical system [11].

Figure 2 illustrates a subcritical and a transcritical process for a CO_2 cycle in a log(p)-h diagram.

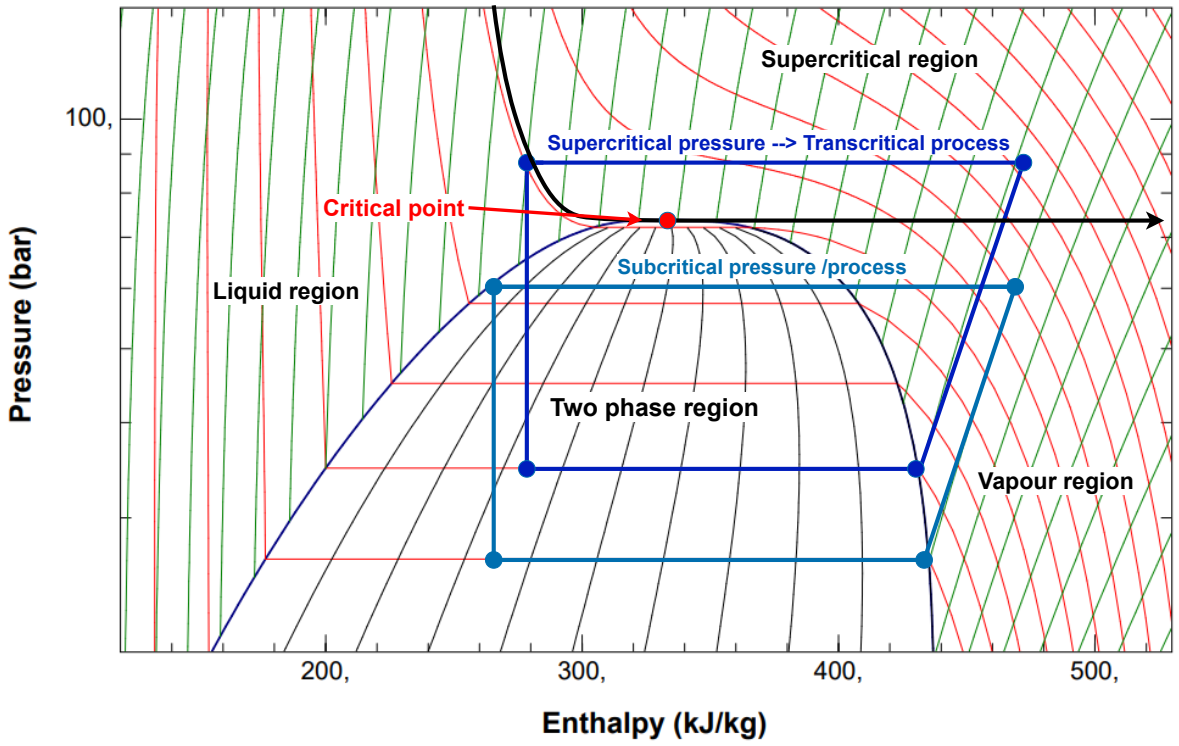


Figure 2: A log(p)-h diagram of CO_2 where the light blue line illustrate a subcritical process and the dark blue line illustrates a transcritical process.

2.2 Carbon dioxide as a refrigerant

CO_2 is among the safest refrigerants in use in relation to both environmental and human safety. As a natural substance, it is known to be harmless to the biosphere. It does not contribute to ozone depletion (ODP) and its global warming potential (GWP) is per definition equal to 1. As the CO_2 used in refrigeration is not actively produced but rather extracted from waste products in various industrial processes, it does not directly contribute to increasing the existing amount of CO_2 in the atmosphere. It can therefore be argued that CO_2 as a refrigerant has a net GWP = 0 [4].

CO_2 is non-flammable and not acutely toxic, earning it the highest possible safety classification

of A1 as a refrigerant. As it is 1,5 times heavier than air, the main safety concern is for large amounts to accumulate in enclosed areas, replacing the oxygen. Contrary to popular beliefs, CO_2 can be highly toxic in high concentrations. A volumetric concentration higher than 10 % can potentially be lethal [4]. However, simple measures like ensuring good ventilation and adding CO_2-detection devices will drastically lower the safety risk.

The properties of CO_2 are unique and very different compared to other conventional refrigerants. A comparison of pressure- and temperature-related properties between carbon dioxide, ammonia, and the traditional HFC refrigerant R134a is displayed in Table 1. The low critical temperature in particular has been considered a hindrance compared to competing refrigerants. If the condensing temperature rises above the critical temperature (30,98 °C [12]) due to high ambient temperatures, the system will start operating in transcritical mode, drastically reducing the efficiency of the system if operated in a conventional configuration. The high operating pressures have also previously been considered a drawback, with arguments such as higher potential for leakages and increased hazard levels.

Table 1: Comparison of pressure and temperature properties for CO_2, NH_3, and R134a

Fluid	Critical Pressure $[bar]$	Critical temperature $[°C]$	Triple point pressure $[bar]$	Triple point temperature $[°C]$	Temperature at atmospheric pressure $[°C]$
CO_2	73,77	30,98	5,18	-56,56	-78,46[1]
NH_3	113,33	132,25	0.0609	-77,65	-33,33
R134a	40,59	101,06	0,0039	-103,3	-26,07

On the other hand, high pressure levels also have their benefits. High pressure levels for relatively small temperature changes mean that the system will tolerate higher internal pressure drops without considerably affecting the internal temperatures. Thus, smaller pipe dimensions for a cheaper and more compact system can be used.

The high vapor density ensures a very large volumetric refrigeration capacity (presented in Table 2). This allows for smaller compressor stroke volumes and mass flow rates, further increasing the compactness of the CO_2 refrigeration system. Even though the compressors work at high pressure levels, the pressure ratio is still considered to be very low compared to systems with traditional refrigerants operating under similar conditions [13].

Table 2: Properties at 3,3 °C evaporation temperature

Fluids	Pressure $[bar]$	Vapor density $[kg/m^3]$	Latent heat $[kJ/kg]$	Volumetric capacity $[kJ/m^3]$
CO_2	38,00	108,50	220,57	23936
NH_3	4,85	3,88	1250,32	4852
R134a	3,30	16,17	196,07	3171

Properties such as the high specific heat capacity, low kinematic viscosity, relatively high thermal conductivity, and low surface tension lead to CO_2 having significantly higher heat transfer coefficients on the liquid refrigerant side of heat exchangers compared to traditional working fluids [14]. To fully profit on this quality, it is recommended to redesign and adapt the heat exchanger to take advantage of its full potential.

[1] Dry ice

2.3 Ejectors

An ejector is a device that uses a high-pressure fluid (motive fluid) to create suction by exploiting the relationship between pressure and velocity, commonly known as the Venturi effect. With Figure 3 in mind, the motive fluid enters the ejector and expands through the motive nozzle. As the motive nozzle is converging, the velocity of the expanded refrigerant increases vastly, lowering the surrounding pressure in the suction chamber. The low pressure in the suction chamber causes the suction fluid to be drawn into the ejector. The motive and suction fluid reaches a common velocity as they mix in the mixing section. As the ejector diverges in the diffuser section, the velocity decrease, and the static pressure is recovered [15].

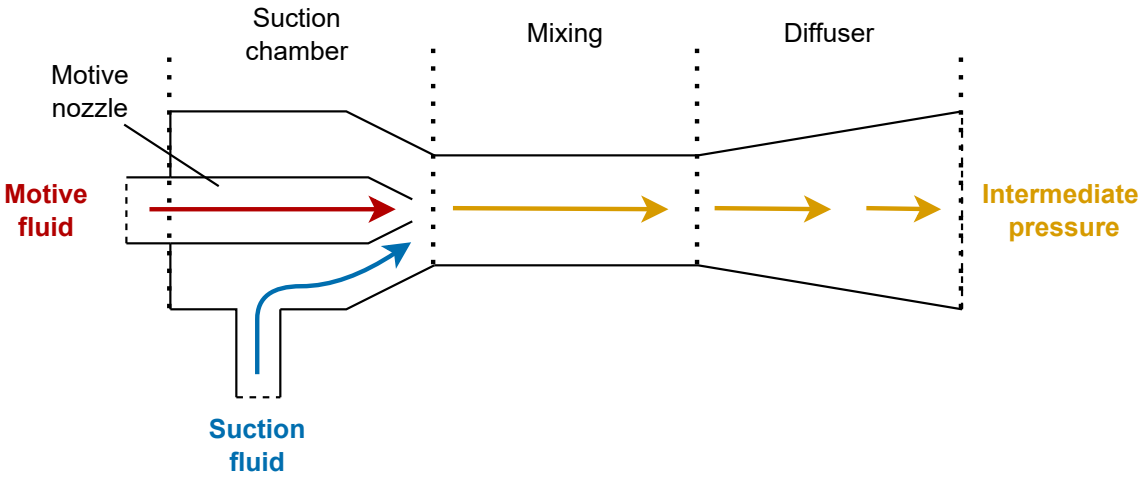


Figure 3: The working principle of an ejector illustrated

When evaluating the performance of an ejector, the following indicators are commonly used; the mass entrainment ratio (Φ), pressure lift (ΔP_{lift}), and ejector isentropic efficiency ($\eta_{ejector}$) [16].

The mass entrainment ratio indicates the ejector's capability to entrain or pump mass by comparing the suction mass flow rate ($\dot{m}_{ej,suc}$) to the motive mass flow rate ($\dot{m}_{ej,mot}$) using the following equation:

$$\Phi = \frac{\dot{m}_{ej,suc}}{\dot{m}_{ej,mot}} \tag{4}$$

The pressure lift indicates the difference in pressure by which the refrigerant at the suction side is lifted. It is expressed as following:

$$\Delta P_{lift} = P_{ej,out} - P_{ej,suc} \tag{5}$$

where $P_{ej,out}$ is the pressure at the ejector outlet, and $P_{ej,suc}$ is the pressure at the suction inlet.

The ejector work recovery efficiency is commonly calculated using the method presented by Elbel and Hrnjak [17]. It describes the ratio of the actual amount of work recovered by the

ejector to the maximum amount that could theoretically be recovered. It is determined by calculating the power required to compress the suction stream isentropically from suction inlet pressure to ejector outlet pressure. It is then divided by the theoretical maximum amount that could be recovered by an isentropic expansion of the motive stream from the motive inlet pressure to ejector outlet pressure [15]. The ejector work recovery efficiency is described by equation 6:

$$\eta_{ejector} = \Phi \cdot \frac{h(P_{ej,out}, s_{ej,suc}) - h_{ej,suc}}{h_{ej,mot} - h(P_{ej,out}, s_{ej,mot})} \tag{6}$$

where h is the specific enthalpy, and s is the specific entropy.

2.3.1 Multi Ejector

The discharge pressure control is crucial for achieving high energy efficiency in transcritical CO_2 systems. However, a single constant-geometry ejector is unable to control the compressor discharge pressure. To fix this, the concept of a Multi Ejector was introduced. A Multi Ejector module consists of multiple ejectors of different sizes connected in parallel. Figure 4 shows the anatomy of a commercial Danfoss Multi Ejector. The configuration allows for better control of the gas cooler pressure. By varying the number of employed ejectors, the Multi Ejector system can match the required capacity in any operating condition within its rated capacity.

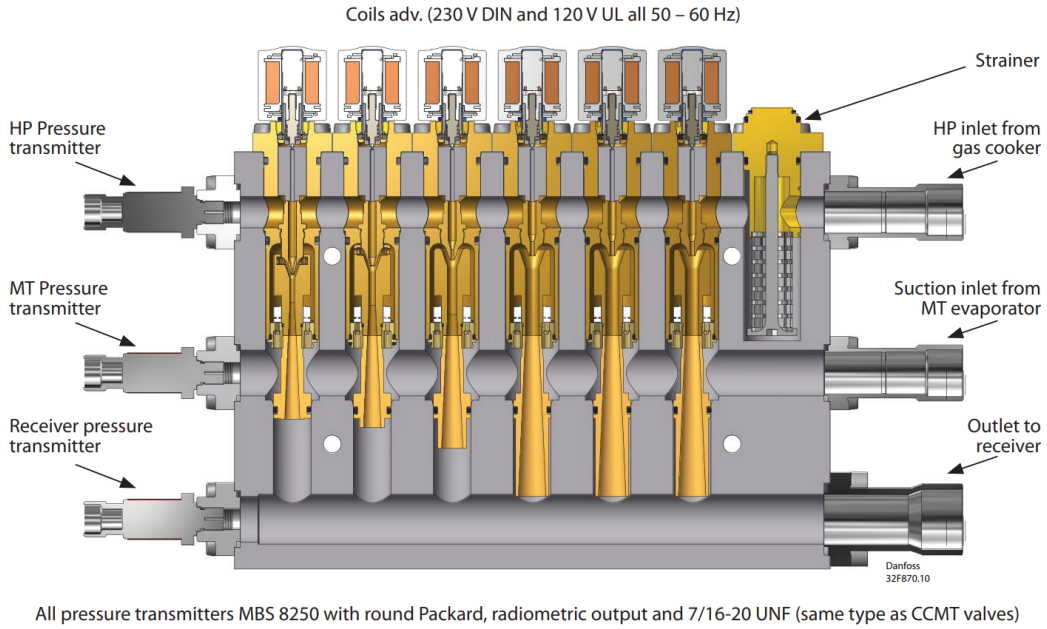


Figure 4: The anatomy of a commerical Danfoss Multi Ejector [18]

Figure 5 illustrates how the Multi Ejector regulates its capacity by deploying different numbers and combinations of ejector modules. Each ejector module has a different capacity, and by switching on or off modules, the overall capacity of the Multi Ejector can be adjusted. As shown in the illustration, this particular model switches between five different sized ejectors to create 32 levels of capacities, allowing for precise control and modulation of the refrigeration output [18].

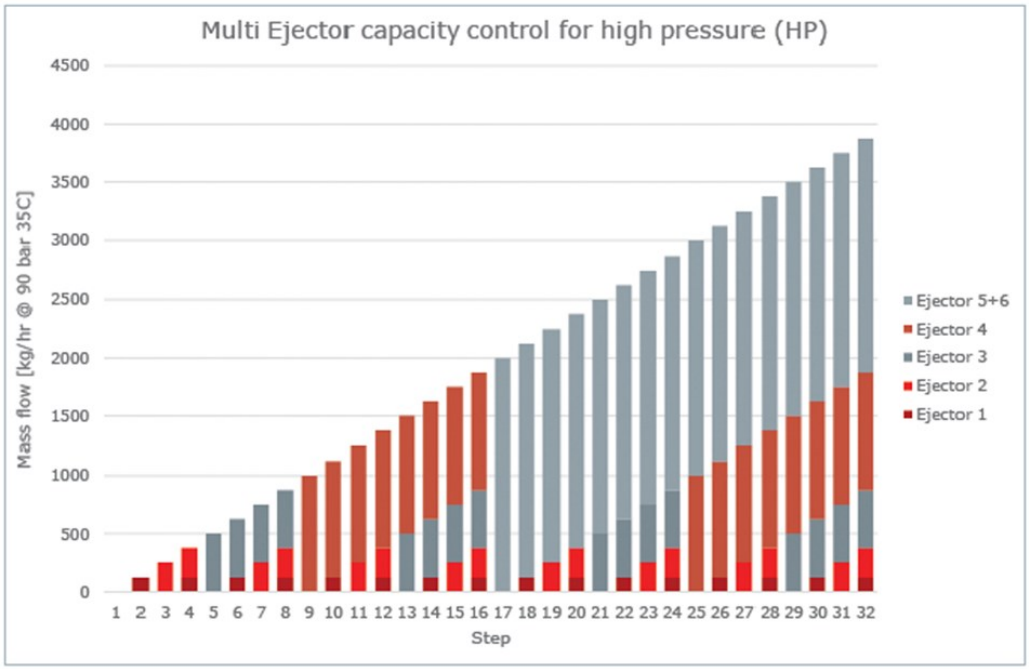


Figure 5: Illustration of the modulating steps of a Danfoss Multi Ejector [18]

2.3.2 The Multi Ejector in a heat pump chiller

Ejectors can be used to directly lift the pressure of the vaporized refrigerant at the evaporator outlet to an intermediate pressure level, or to remove liquid from the evaporator outlet, enabling overfeeding. Overfeeding the evaporators allows for higher evaporation pressures for the same cooling effect. In both cases, energy savings are achieved by increasing the suction pressure at the compressor inlet, thus reducing the required compressor pressure ratio.

An example of a heat pump chiller integrated with a vapor Multi Ejector is illustrated in Figure 6. The Multi Ejector serves as a replacement for the high pressure expansion valve (HPV) and its main function is to maintain the optimal compressor discharge pressure. The ejector outlet is connected to the low pressure receiver which operates as a separator by separating the two-phased flow entering from the ejector outlet. The vapor flows to the compressor, while the liquid is gathered in the receiver. An expansion valve is positioned between the receiver and the evaporator to maintain a pressure difference between the ejector suction inlet and ejector outlet by controlling the suction mass flow rate. The liquid is then evaporated at a lower pressure level and enters the ejector suction inlet as vapor. The ejector recovers expansion work from the high pressure motive gas, which then elevates the vapor pressure back to the pressure of the receiver tank.

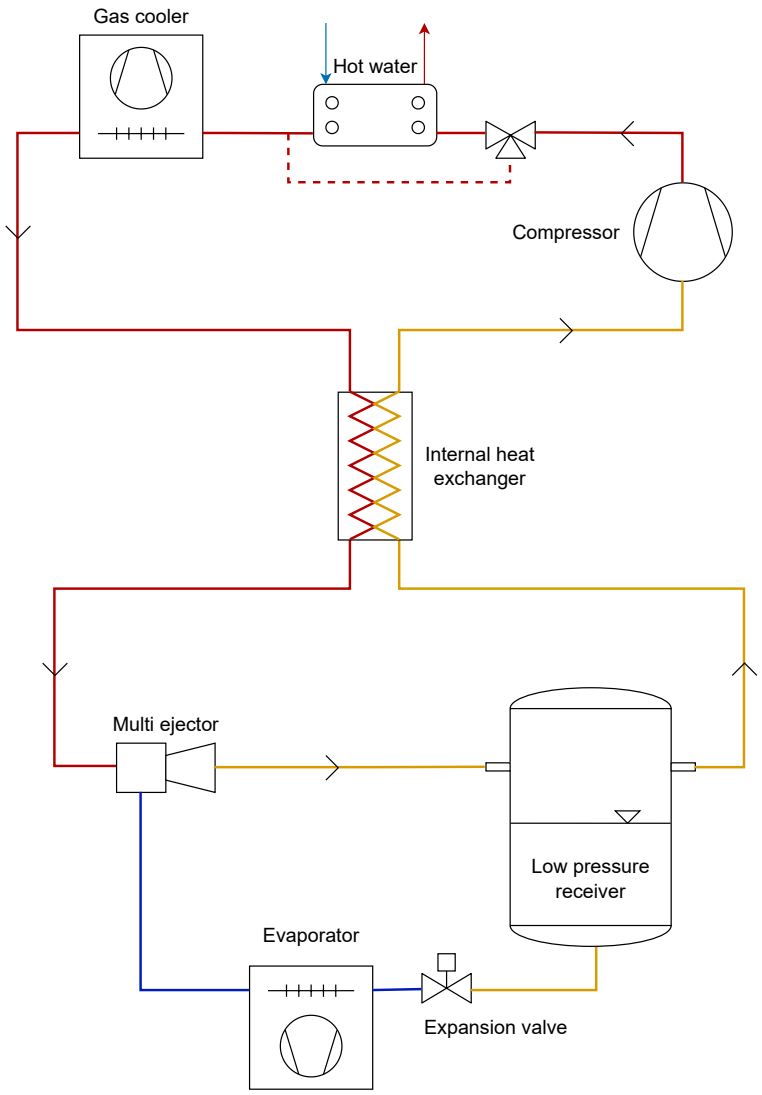


Figure 6: Simplified vapor ejector heat pump chiller with hot water production

2.4 Natural circulation loop

As depicted in Figure 7, a natural circulation loop consists of four main components; a separator tank, a downcomer, an evaporator, and a riser. A mixture of liquid and vapor enters the separator tank from the high-pressure side of a heat pump. The excess flash gas is passed on to the compressors to re-enter the high-pressure state, while all liquid is collected in the bottom half of the separator. The liquid runs down through the downcomer into the evaporator at the lowermost inlet. The evaporator consists of a brazed plate heat exchanger using counterflowing water as brine. The refrigerant in the evaporator starts to evaporate as it absorbs heat from the higher temperature brine and exits in a two-phased state through the evaporator outlet. The two-phase refrigerant flows through the riser and enters back into the separator where the process is repeated.

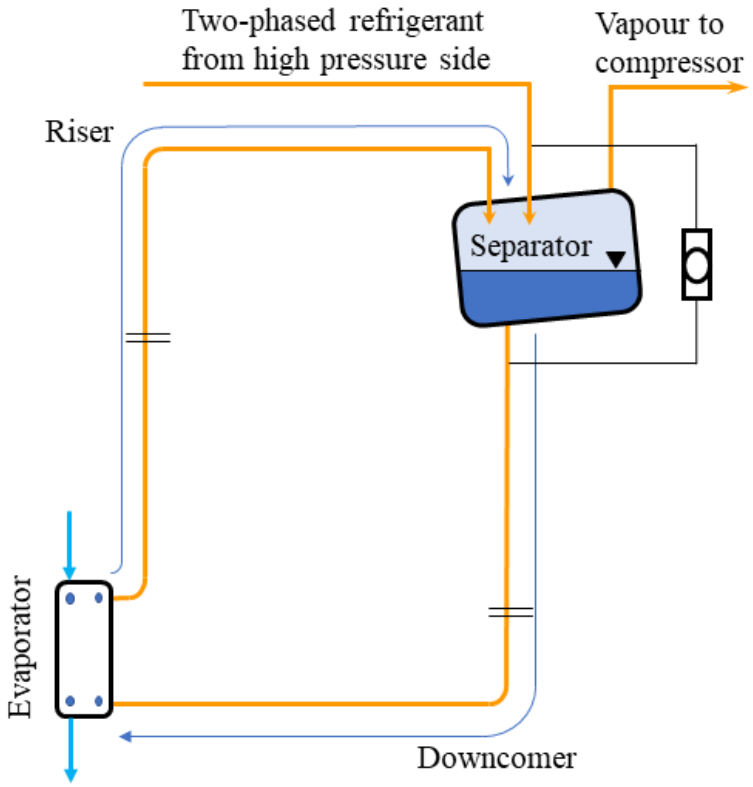


Figure 7: Simplified drawing of the gravity-fed evaporator loop

Liquid height is the most important variable in a two-phase thermosyphon system like this. The pressure at the evaporator inlet is higher than in the separator tank due to the liquid leg in the downcomer. This pressure difference, also known as the pumping pressure, is calculated by the following equation:

$$\Delta P = \rho \cdot g \cdot H \tag{7}$$

where ρ is the density of the refrigerant measured in kg/m^3, g is the gravitational acceleration ($9,81\ m/s^2$), and H is the liquid height measured in meters [19].

The driving force is caused by the heat exchange between the water and the CO_2. The water is hotter than the boiling point of the CO_2, causing the CO_2 to evaporate. As more liquid CO2 evaporates, the average density of the CO_2 in the evaporator and riser becomes much lower than the density of the liquid CO_2 in the downcomer and separator. This density difference creates a buoyant force, which pushes the two-phase CO_2 upwards through the riser and back into the separator tank. This cycle continues until the liquid level in the separator and downcomer becomes too low, or the heat flux from the water in the evaporator becomes insufficient.

The mass flow rate in the thermosyphon loop depends on the same factors. To increase the mass flow rate, one must either increase the pumping pressure or increase the heat flux in the evaporator, causing the liquid to boil more rapidly.

2.5 Combined natural circulation and ejector

By combining the principles of Section 2.3 and 2.4, a configuration illustrated in Figure 8 can be developed to utilize the benefits of both natural circulation and the reclaim of expansion work through an ejector. Figure 8 displays a simplified drawing of a CO_2 heat pump chiller integrated with a gravity loop, a two-stage evaporator, and a Multi Ejector. An illustrational log(p)-h diagram is shown in Figure 9 to visualize the behavior of the gravity loop in combination with the Multi Ejector.

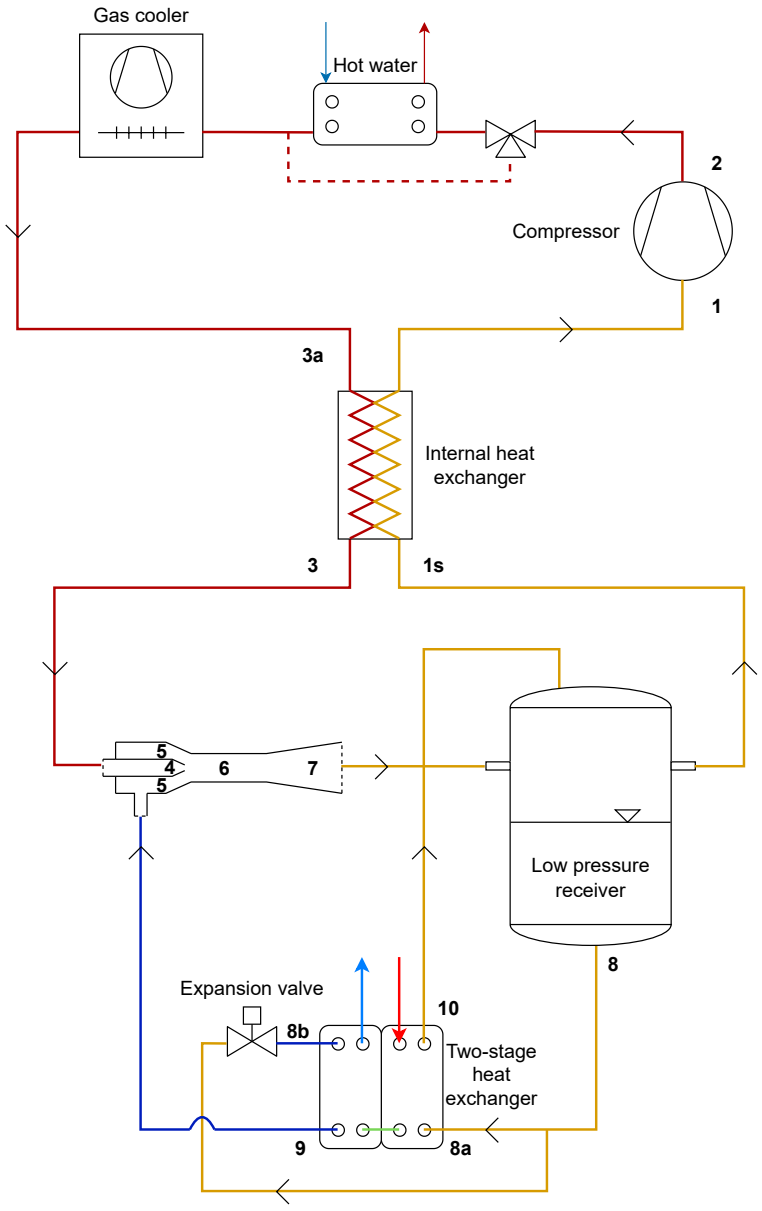


Figure 8: Conceptual drawing of a CO_2 heat pump chiller integrated with a two-staged evaporator, a natural circulation loop, and an ejector driven loop.

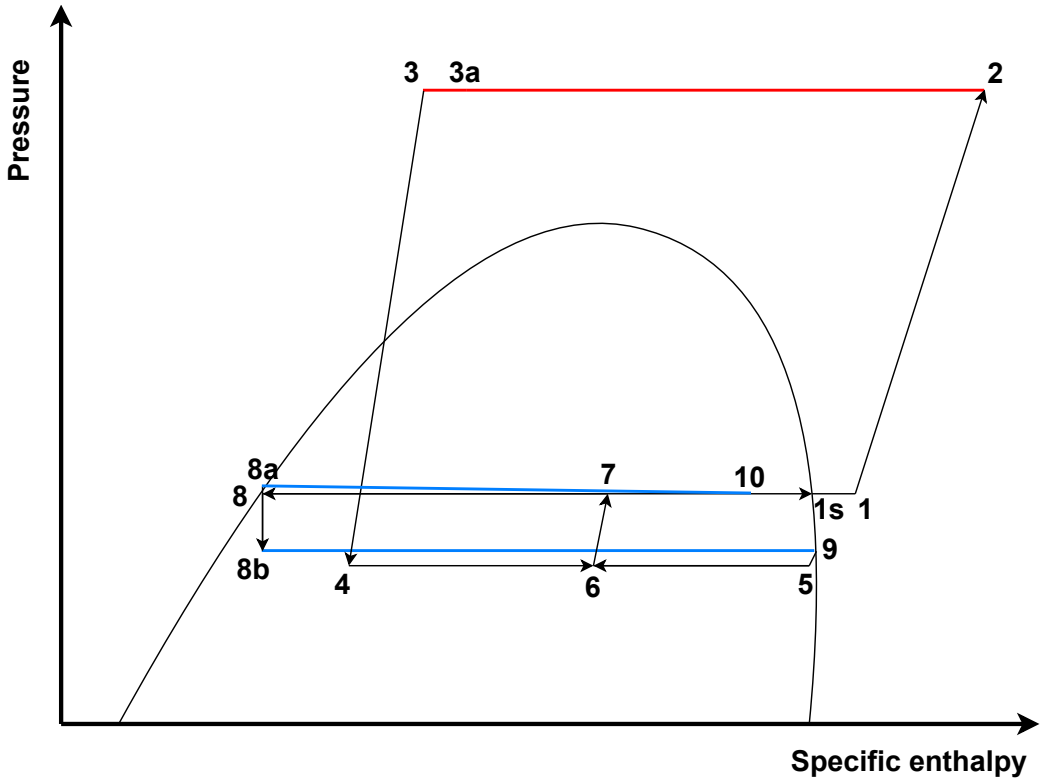


Figure 9: Illustrational log(p)-h diagram of a CO_2 heat pump chiller integrated with a two-staged evaporator, a natural circulation loop, and an ejector driven loop.

2.6 Brazed plate heat exchangers

A heat exchanger is a device that transfers energy in the form of heat from a warm to a cold medium. The brazed plate heat exchanger (BPHX) consists of several metal plates of high thermal conductivity brazed together in layers. Between each layer are thin gaps, creating isolated channels for the medium to pass through. The plates are connected to form separate circuits with the warm and cold medium flowing in alternating layers as depicted in Figure 10.

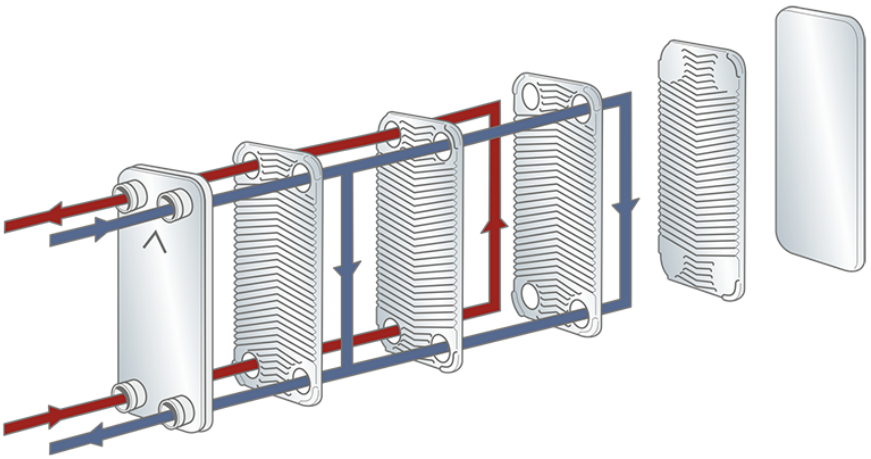


Figure 10: Configuration of a brazed plate heat exchanger [20].

As the warm and cold medium passes through the layers, energy is transferred from the warm medium to the cold medium through the plates in accordance with the second law of thermodynamics. The transfer of heat in the heat exchanger can be described by the heat transfer equation:

$$\dot{Q} = U \cdot A \cdot \Delta T \tag{8}$$

where $\dot{Q}$ is the energy flow (W) in the form of heat, U is the overall heat transfer coefficient ($W/(m^2 \cdot K)$), A is the heat transfer area (m^2), and ΔT is the temperature difference (K) between the warm and cold medium.

When the temperature difference between the hot and cold fluids varies throughout the heat exchanger, the arithmetic temperature difference (ΔT) is no longer a sufficiently accurate representation of the overall temperature difference. Instead, the logarithmic mean temperature difference ($LMTD$) is introduced to provide a more representative value. Equation 8 can therefore be expressed as

$$\dot{Q} = U \cdot A \cdot LMTD \tag{9}$$

where the $LMTD$ (K) is calculated using the following formula:

$$LMTD = \frac{(\Delta T_a - \Delta T_b)}{ln\left(\frac{\Delta T_a}{\Delta T_b}\right)} \tag{10}$$

where ΔT_a is the temperature difference (K) between the warm inlet and the cold outlet, and ΔT_b is the temperature difference (K) between the warm outlet and the cold inlet.

For fluids not undergoing any phase change during the heat transfer, the change in energy, $\dot{Q}$, can be calculated with the following formula:

$$\dot{Q} = \dot{m} \cdot C_p \cdot \Delta T \tag{11}$$

where $\dot{m}$ is the fluid mass flow (kg/s), C_p is the specific heat capacity of the fluid ($J/(kg \cdot K)$), and ΔT is the temperature difference (K) of the fluid at the inlet and the outlet of the heat exchanger.

If a refrigerant is undergoing phase change during the heat exchange, the amount of energy transferred, $\dot{Q}$, is calculated using the equation:

$$\dot{Q} = \dot{m}_R \cdot \Delta h \tag{12}$$

where, $\dot{m}_R$ is the refrigerant mass flow (kg/s), Δh is the enthalpy difference (J/kg) of the refrigerant at the inlet and outlet of the evaporator.

By assuming no loss to the surroundings, equation 9, 11 and 12 can be combined to the following expression:

$$\dot{Q} = \dot{m} \cdot C_p \cdot \Delta T = U \cdot A \cdot LMTD = \dot{m}_R \cdot \Delta h \tag{13}$$

The BPHX works perfectly well as an evaporator. The refrigerant is heated by the secondary medium until the refrigerant reaches its boiling point and starts to evaporate. The total transfer of energy from the secondary fluid is the sum of the sensible heat and latent heat added to the refrigerant, minus heat losses to the environment.

2.7 Two-stage evaporation

As mentioned in Section 2.3, lowering the pressure ratio by increasing the suction pressure decreases the energy input required by the compressors. Raising the evaporation pressure also increases the density of the vaporized refrigerant. Therefore, the compressor will transport a greater quantity of refrigerant with each stroke. Reducing electricity usage and increasing refrigeration capacity enhances the overall efficiency of the system.

Two-stage evaporation is desirable as it enables cooling at lower temperatures without lowering the suction pressure. It can be achieved by utilizing the pressure lift from an ejector as illustrated in Figure 8 and 9.

A novel two-stage evaporator has been made by Alfa Laval for this project. A principal sketch of the evaporator configuration is shown in Figure 11. It is made from two Alfa Laval AXP82-40M fused together back to back, with a special middle plate for only the water/brine to flow through, separating the two evaporation stages.

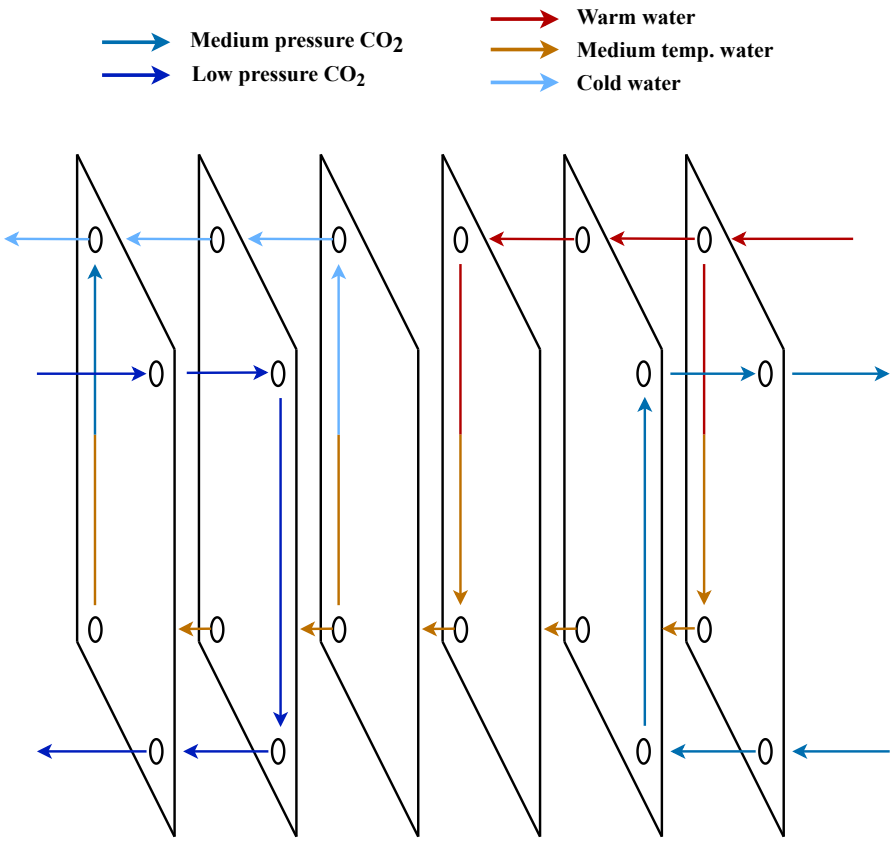


Figure 11: Plate configuration of a two-stage plate heat exchanger.

2.7.1 Design criteria of the novel two-stage evaporator

The novel two-stage evaporator provided for the experiment is design in accordance with the values stated in Table 3.

Table 3: Design requirements of the novel two-stage evaporator.

CO_2 side					
Gravity fed loop			Ejector assisted circulation loop		
Inlet		Exit	Inlet		Exit
P (bar)	Vapour fraction	Vapour fraction	P (bar)	Vapour fraction	Vapour fraction
42	0 = liquid	0.7	37	0.1	0.95
Capacity (kW)			Capacity (kW)		
20			20		
CO2 mass flow rate (kg/s)			CO2 mass flow rate (kg/s)		
0.1378			0.1051		
Secondary fluid (water = H_2O)					
Gravity fed loop			Ejector assisted circulation loop		
Inlet		Exit	Inlet		Exit
T (°C)		T (°C)	T (°C)		T (°C)
13		10	10		7
Capacity (kW)			Capacity (kW)		
20			20		
mass flow rate (kg/s)					
1.589					

A theoretical P&ID of the low-pressure side of the experimental setup including the gravity loop, ejector-assisted loop, and the two-stage evaporator is displayed in Figure 12.

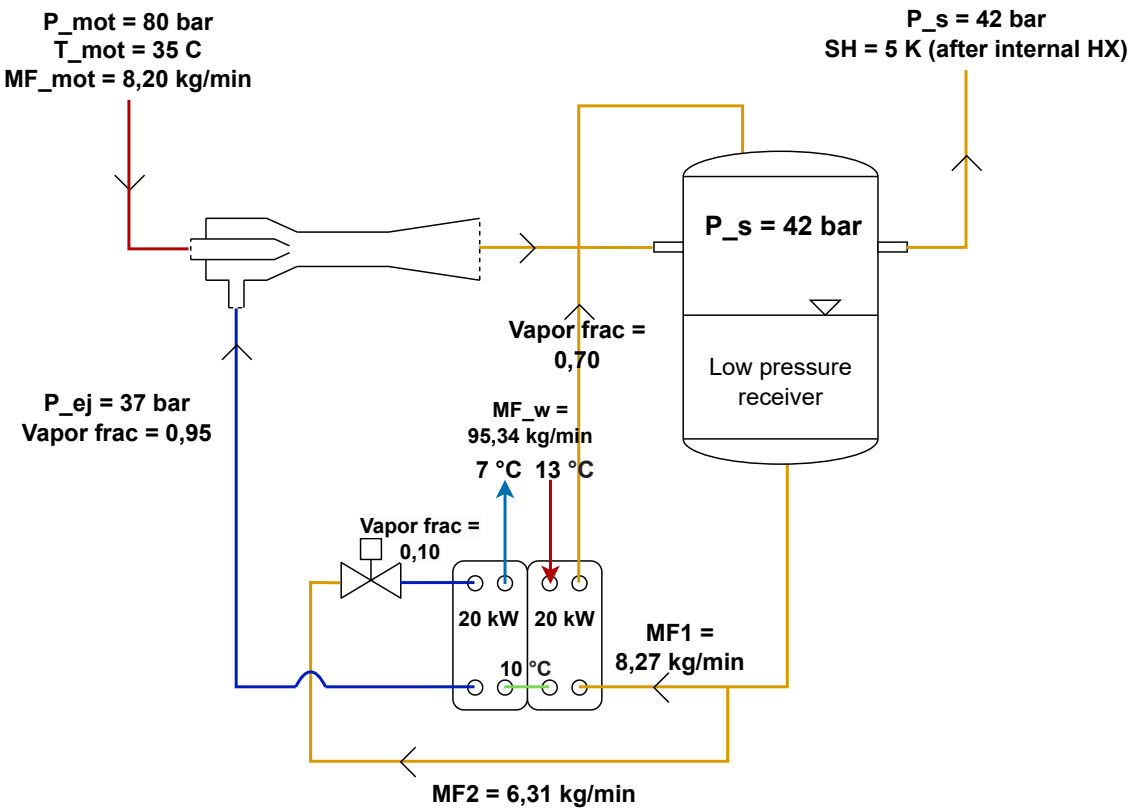


Figure 12: P&ID of the low-pressure side of the experimental setup including design values.

The values of pressure and temperature at the motive nozzle inlet was agreed upon by the participants of the experiment previous to the start of the tests. They were chosen as they were believed to reflect a realistic outcome in a warm temperature climate, or in a poorly sized but all too common space heating system.

The ejector has previously been tested on the same experimental machine rack by Banasiak et al. [21], but in a slightly different setup. The experiments indicated an ejector efficiency of approximately $\eta_{ejector} = 0,25$. By applying equation 4 and 6, and assuming an ejector efficiency of $\eta_{ejector} = 0,25$, the motive nozzle mass flow rate (MF_mot) is calculated to be 8,20 kg/min.

The right-hand side of the evaporator is connected to a gravity loop, making it a gravity-fed evaporator. The evaporation pressure in the gravity loop is at 42 bar, corresponding to an evaporation temperature of 7,2 °C. 13 °C water enters the gravity stage and cools down to 10 °C before entering the ejector stage. The ejector stage is at an evaporation pressure of 37 bar, yielding an evaporation temperature of 2,3 °C. The now 10 °C water is further cooled down to 7 °C before exiting the evaporator. Both sides of the evaporator are designed to produce 20 kW each, making it a 40 kW evaporator. The full set of design requirements is stated in Table 3.

A theoretical system COP can be calculated according to the formula presented by Gullo et al. [22] and illustrated by Pardiñas et al. [23]. First, the global efficiency of the compressor is calculated as follows:

$$\eta_{global,comp} = -0,0788 \cdot \left(\frac{p_{disch,comp}}{p_{suc,comp}} \right)^2 + 0,3708 \cdot \left(\frac{p_{disch,comp}}{p_{suc,comp}} \right) + 0,2729 \tag{14}$$

The enthalpy of the compressor discharge can then be calculated using this equation:

$$h_{disch,comp} = h_{suc,comp} + \frac{h_{disch(is),comp} - h_{suc,comp}}{\eta_{global,comp}} \tag{15}$$

The COP of the heat pump can finally be calculated thus:

$$COP_H = \frac{h_{disch,comp} - h_{GC,out}}{h_{disch,comp} - h_{suc,comp}} \tag{16}$$

Following the steps mentioned above, the COP of the system is calculated to be $COP_H = 3,28$

3 Methodology

3.1 System description

At the test facility of NTNU/SINTEF at Gløshaugen, the Multifunctional test-rack (Figure 13) is built as an experimental setup to test innovative components for integrated CO_2 refrigeration systems at freezing (LT), chilling (MT) and air conditioning (AC) temperatures [24]. All experimental results presented in this thesis are obtained from a total of 61 individual tests, carried out in two intervals on the Multifunctional test rack during the months of March and April 2022.

Figure 13: The Multifunctional test-rack

The MultiTest-Rack comes with three compressors, one for each level (MT, LT, and parallel). All three compressors have variable speed drive capabilities. The design cooling capacity is 30 kW at the MT level and 5 kW at the LT level. The rack features an integrated glycol loop to adjust for accurate operating conditions and demands through several brazed plate heat exchangers (BPHX) operating as evaporators and condensers. Various measuring equipment is placed throughout the rack for extensive logging of data. A piping and instrumentation diagram (P&ID) of the total system can be viewed in Appendix B

3.1.1 Test configuration

In addition to the original test rack, a configuration of a grey water (GW) tank, a multi-ejector, and a novel two-stage brazed plate heat exchanger was fitted and connected to a water loop system. The GW tank was utilized as the receiver tank, and the two-stage BPHX served as a two-stage evaporator. The water loop system was fitted with variable speed pumps

and electric heating elements to accurately adjust the temperature and mass flow rate of the water at the evaporator inlet. A picture of the configuration is shown in Figure 14.

Figure 14: Gravity loop configuration

A section of the P&ID from Appendix B is shown in Figure 15. It corresponds to Figure 14 and displays the placements of each measuring equipment used to monitor the gravity loop, ejector-driven loop, and water loop.

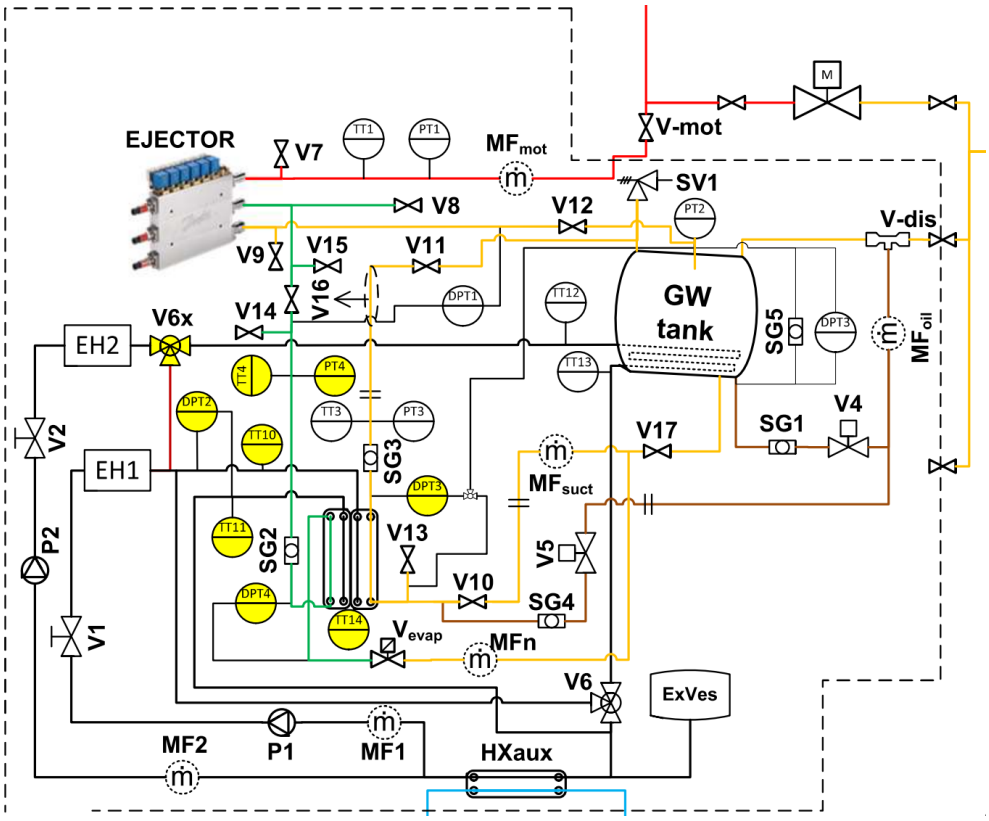


Figure 15: P&ID of the gravity loop configuration

3.1.2 Measuring equipment

Temperature sensors [25]
PT-100 and PT-1000 sensors were used to measure the temperature on both the water and CO_2 side. Both temperature sensors are rated Class A by the IEC 60751 standard. The operational range is between -100 °C to 450 °C [25]. Each specific sensor and its corresponding tag and accuracy are stated in Table 4.

Table 4: Temperature sensors

Component	TAG	Medium	Measurement setup area
Pt100	TT3	CO_2	Riser gravity loop
Pt100	TT4	CO_2	Ejector suction side
Pt100	TT14	Water	Middle junction of two-stage evaporator
Pt1000	TT1	CO_2	Ejector motive nozzle
Pt1000	TT10	Water	Inlet of the two-stage evaporator
Pt1000	TT11	Water	Outlet of the two-stage evaporator

Pressure sensors [26]
The pressure was measured using the E&H CERABAR S PMP71 pressure transmitter. It is a digital transmitter with a piezoresistive sensor and a welded metallic membrane. It has a measuring range from 0 mbar to 700 bar, and it can operate within temperatures ranging from -40 to +125 °C [26]. The pressure sensors were set to measure absolute pressure. The corresponding tag and accuracy are stated in Table 5.

Table 5: Pressure sensors

Component	TAG	Medium	Measurement setup area
PMP71	PT1	CO_2	Ejector motive nozzle
PMP71	PT2	CO_2	Receiver tank
PMP71	PT3	CO_2	Riser gravity loop
PMP71	PT4	CO_2	Ejector suction side

Differential pressure sensors [27]
The differential pressure was measured using the E&H DELTABAR S PMD75 differential pressure transmitter. It has an operational range from 10 mbar to 250 bar, with 1 mbar being the smallest calibratable span. The process temperature range is -40 to +85 °C [27]. Tags and accuracy values are stated in Table 6.

Table 6: Differential pressure sensors

Component	TAG	Medium	Differential pressure between
PMD75	DPT1	CO_2	Ejector suction inlet and ejector outlet
PMD75	DPT2	Water	Inlet and outlet of two-stage evaporator, water side
PMD75	DPT3	CO_2	Inlet and outlet of two-stage evaporator, gravity side
PMD75	DPT4	CO_2	Inlet and outlet of two-stage evaporator, ejector side

Mass flow meters [28, 29]
Rheonik RHM 08 and 06 were used to measure the CO_2 and water mass flow rate. The RHM 06 can operate in temperatures ranging from -40 °C - 210 °C, and at a maximum pressure of 150 bar. Its operational range is between 0,1 kg/min - 36 kg/min. The RHM 08 has an operational range between -45 °C - 210 °C and a maximum pressure of 220 bar. Its operational range is between 1,0 kg/min - 50 kg/min. Tags and accuracy values are stated in Table 7. The Rheonik RHM uses the Coriolis effect to accurately measure the massflow. A pressure drop chart for the RHM 06 is provided in Appendix **??** is provided

Table 7: Mass flow meter

Component	TAG	Medium	Measurement setup area	Operating range
RHM 08	MFmot	CO_2	Ejector motive nozzle	0,3-50 kg/min
RHM 08	MFn	CO_2	Ejector suction side	0,3-50 kg/min
RHM 08	MF1	Water	Evaporator water circuit 1	0,3-50 kg/min
RHM 08	MF2	Water	Evaporator water circuit 2	0,3-50 kg/min
RHM 06	MFsuct	CO_2	Gravity loop	0,1-36 kg/min

3.1.3 Pipe dimensions

The piping is made of stainless steel and has an internal diameter of 14 mm and an external diameter of 18 mm. The pipes are completely insulated with 19 mm thick insulation. All pipe lengths were measured manually using a tape measure after the installation of the rig. The measured pipe lengths are as follows:

- L1 = 530 mm
- L2 = 958 mm

- L3 = 205 mm
- L4 = 960 mm
- L5 = 651 mm
- L6 = 335 mm
- H_{evap} = 420 mm
- H = L1 + measured liquid height in receiver

The placement of each length and height is visualized in Figure 16.

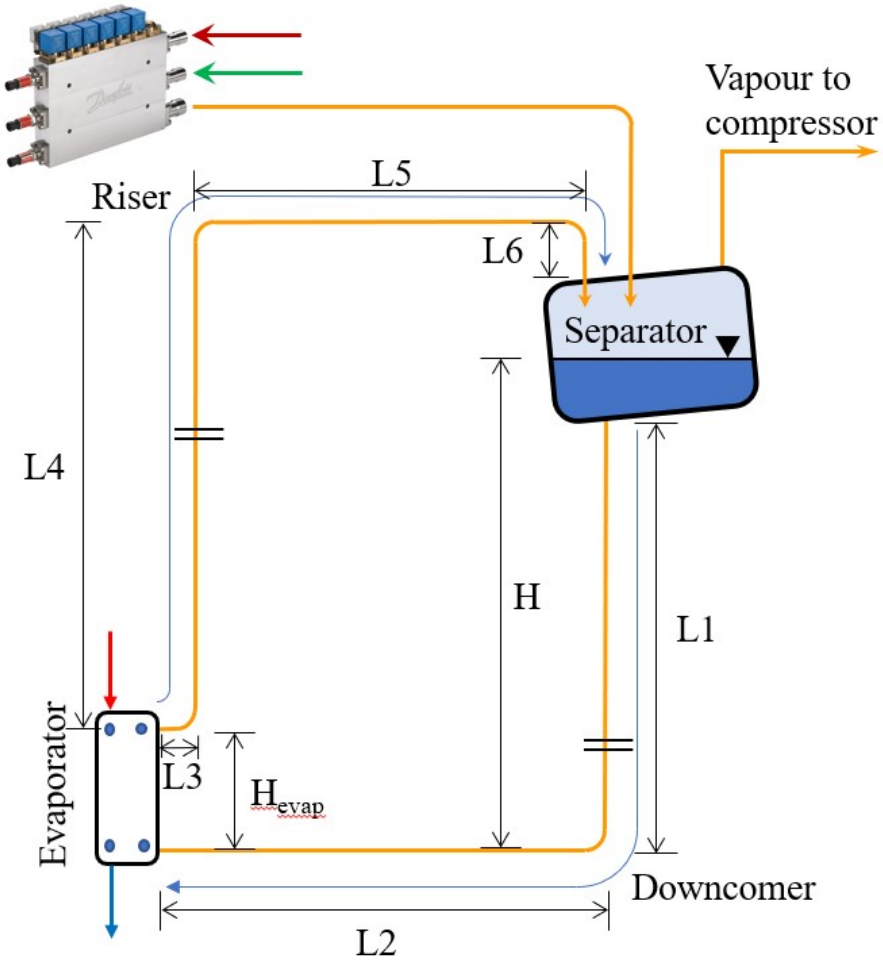


Figure 16: Length of piping and liquid height in the gravity loop

3.1.4 Two-stage BPHX

The novel two-stage evaporator is produced and delivered by Alfa Laval. It is made from two Alfa Laval AXP82-40M fused together back to back with a special middle plate for the water to pass through.

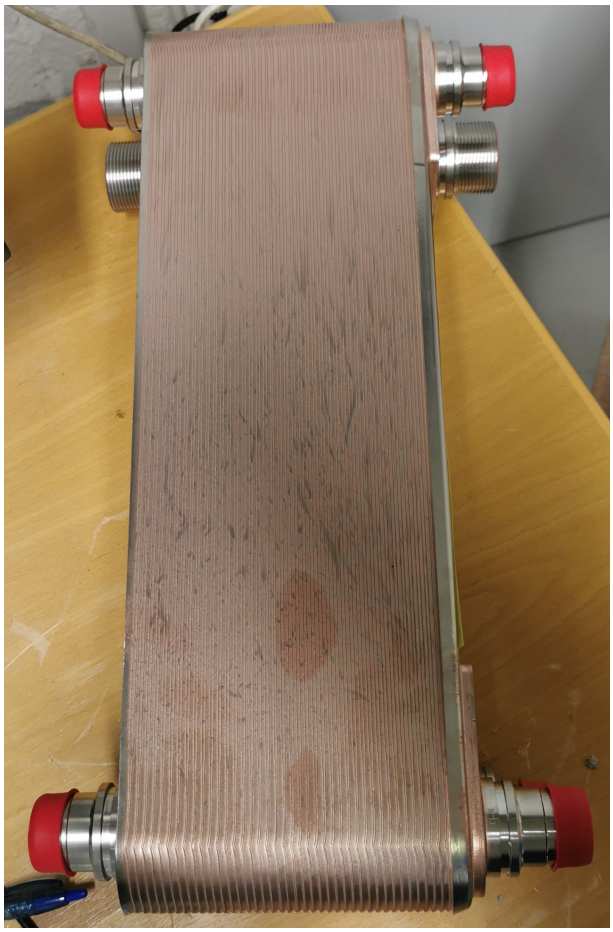


Figure 17: The two-stage evaporator before installation

3.1.5 Receiver tank

The receiver tank (Figure 18) was a repurposed 290 liter graywater tank used in a previous experiment on the same test rig. The evaporation coil within the receiver was turned off during all tests.

Figure 18: Gray water tank functioning as the receiver

The liquid height was a crucial part of the experiment. To read the liquid level inside the tank, a sight glass was fitted and tagged SG5. The height difference between the inlet of the evaporator and the bottom of the sight glass was measured. An improvised solution of simply taping a ruler onto the sight glass was done as shown in Figure 19. The liquid height could then be calculated by simply adding the constant height between the evaporator and the sight glass with the height read on the ruler during tests.

Figure 19: Sight glass to read the liquid level within the receiver tank

3.2 Test procedures

A test matrix was set up in cooperation with co-supervisor Mihir Mouchum Hazarika and co-student Jan Bengsch. The initial proposal comprised three separate tests: one focusing only on the operation of the gravity loop, another on the ejector-assisted loop, and a final combined test in which both the gravity loop and the ejector-assisted loop would operate simultaneously.

Due to time constraints during the course of testing, it was decided to deprioritize the ejector-assisted loop test and instead concentrate on the gravity loop and the combined test only. The duration of each test run ranged from 10 to 20 minutes, with an average duration of approximately 15 minutes. To minimize measurement uncertainty, data was recorded at one-second intervals throughout the entire duration of each test. All performed test runs are listed in Table 8.

Table 8: Tests performed on gravity loop only and combined ejector assisted loop and gravity loop

Gravity loop test			Combined test		
Receiver tank pressure	Water side mass flow rate	Inlet water temperature	Ejector suction side pressure	Water side mass flow rate	Inlet water temperature
[bar]	[kg/min]	[°C]	[bar]	[kg/min]	[°C]
35	24	20	35	16	20
35	24	15	35	16	18
35	24	12	35	16	15
35	24	10	38	12	20
38	12	15	38	12	15
38	12	12	38	12	15
38	12	10	38	16	20
38	16	20	38	16	18
38	16	15	38	16	15
38	16	12	38	16	12
38	16	10	38	18	20
38	18	15	38	18	15
38	18	10	38	18	12
38	24	15	38	24	20
38	24	12	38	24	18
38	24	10	38	24	15
41	16	20	38	24	14
41	16	15	38	24	12
41	16	12	41	16	20
41	16	10	41	16	18
41	24	20	41	16	15
41	24	15	41	24	20
41	24	12	41	24	18
41	24	10	41	24	17
42	16	14	41	24	15
42	16	13			
42	16	12			
42	16	11			
42	16	10			
42	16	12			
42	18	12			
42	20	12			
42	22	12			
42	24	12			
44	16	20			
44	16	15			
44	16	12			

Before recording each test, the desired parameters in the test matrix needed to be stabilized.

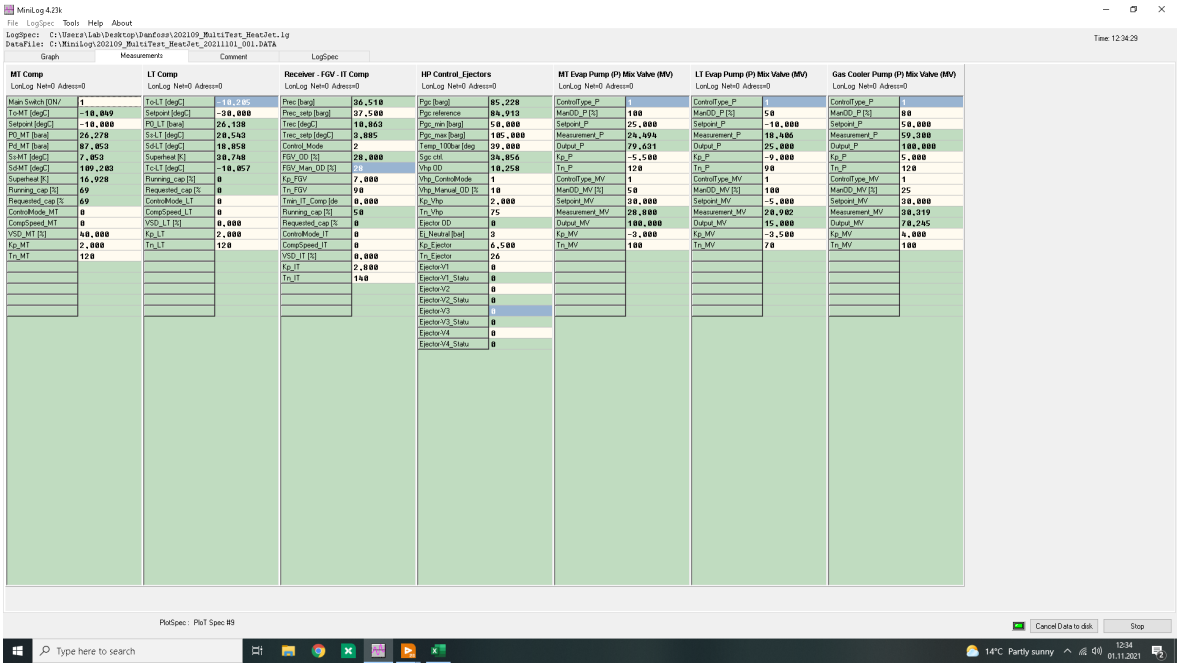


Figure 20: Interface for configuring machine-related set points

Figure 20 displays the interface used to control the set points related to the original MultiTest-rack, such as suction and discharge pressure of the compressor, and receiver tank pressure.

Figure 21 is a screenshot of the interface built to control the additional test configuration added to the MultiTest-rack, including the gravity loop and ejector-assisted loop.

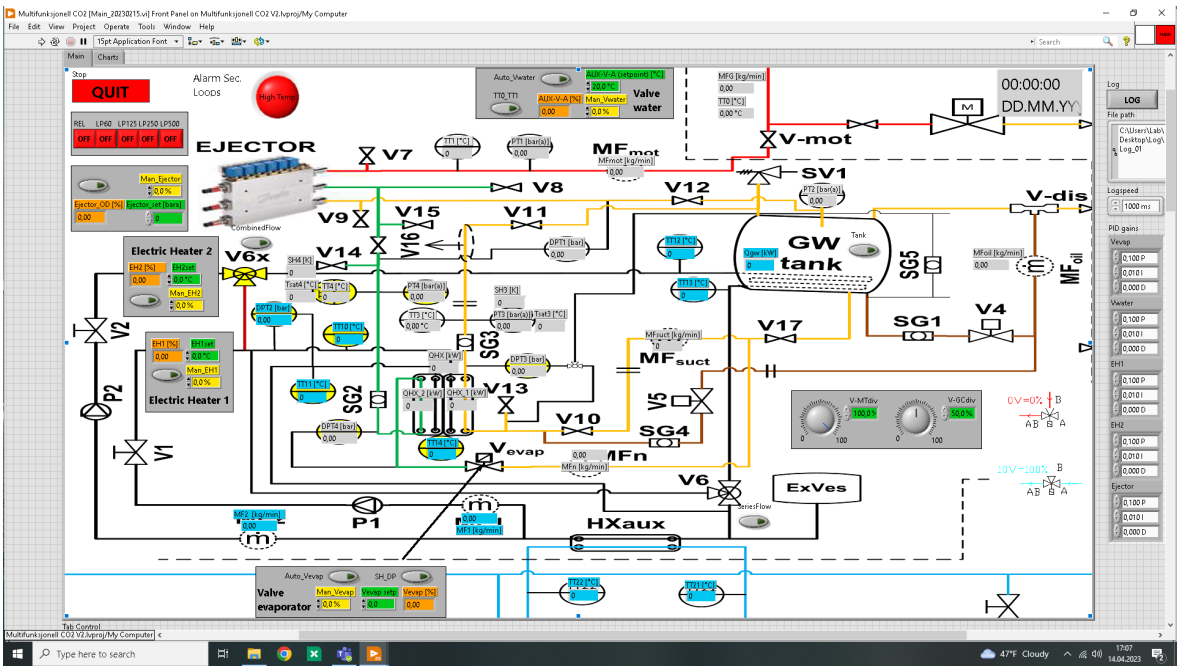


Figure 21: Interface for adjusting test-related set points

During the first tests, the parallel compressor would automatically shut off when the required

capacity dropped below 50 % of its rated capacity. This meant that the MT compressor had to be used instead, as it had a wider range of capacities.

The water pumps P1 and P2 had to be adjusted manually on site to the desired mass flow rate. The inlet water temperature entering the two-stage evaporator was regulated by the heat from the gas cooler through heat exchanger HXaux as depicted in Figure 21. Electric heater 1 and Electric heater 2 (EH1 and EH2) were used to further control and fine tune the inlet water temperature using a PID controller. The refrigerant temperature entering the ejector motive nozzle was controlled by adjusting the mass flow rate of cold tap water flowing through a gas cooler. The desired set point was set at AUX-V-A in Figure 21, then a PID controller would keep the temperature constant throughout the experiments. The ejector was operated manually by switching on and off the desired ejector modules.

3.2.1 Test procedures gravity loop

While running tests on the gravity loop only, the electronic expansion valve tagged V_{evap} was closed to prevent any refrigerant from flowing through the ejector-assisted loop. The receiver tank pressure was adjusted using the interface shown in Figure 20. As the ejector simply worked as an expansion valve, a specific pressure or temperature at the inlet motive nozzle was not required as long as the receiver tank pressure and the inlet water temperature into the evaporator were constant. Changing the pressure and temperature entering the motive nozzle was mainly done to ensure sufficient flash gas for the compressors to run smoothly even at low evaporation capacities.

3.2.2 Test procedures combined gravity loop and ejector assisted loop

For the tests conducted on the combined configuration, constant pressure on the suction side of the ejector was of greater interest than the receiver tank pressure. The resulting lift generated by the ejector would dictate the pressure in the receiver tank. In the tests conducted on the combined configuration, it was additionally important to maintain a pressure of 80 bar and a temperature of 35 °C at the entrance of the motive nozzle of the ejector. This ensured similar testing conditions for the ejector for each test run. V_{evap} was set to a constant opening degree of 25 %.

3.3 Flaws and drawbacks

As always when conducting an experiment on a provisional test configuration, flaws and shortcomings are to be expected. For instance, the effect of the liquid height on the performance of the gravity loop was supposed to be a major part of the thesis. This would include performing tests with the receiver tank placed at different heights relative to the evaporator to compare and analyze mass flow rates, pressure drop in the downcomer and riser, and evaporator performances. Changing the height turned out to be a time-consuming constructional operation, so the plan was scrapped due to time limitations.

Another proposal was to regulate the liquid level in the receiver tank. The tank has an internal radius of approximately 40 cm, which could have yielded a realistic height difference of $\approx$ 25 - 30 cm. During testing, it was noticed that the ejector interfered with the liquid level displayed in the sight glass of the receiver (SG5). This was due to the high velocity refrigerant blowing into the upper part of the sight glass, pushing the liquid down and making the liquid level in

the receiver tank appear to be lower than it was. The liquid level seemed to immediately sink or rise according to the opening degree of the ejector.

Figure 22 shows how P2 and the upper inlet of SG5 were coupled together on the receiver tank. The red arrow leading from the ejector is the ejector outlet blowing into the receiver tank.

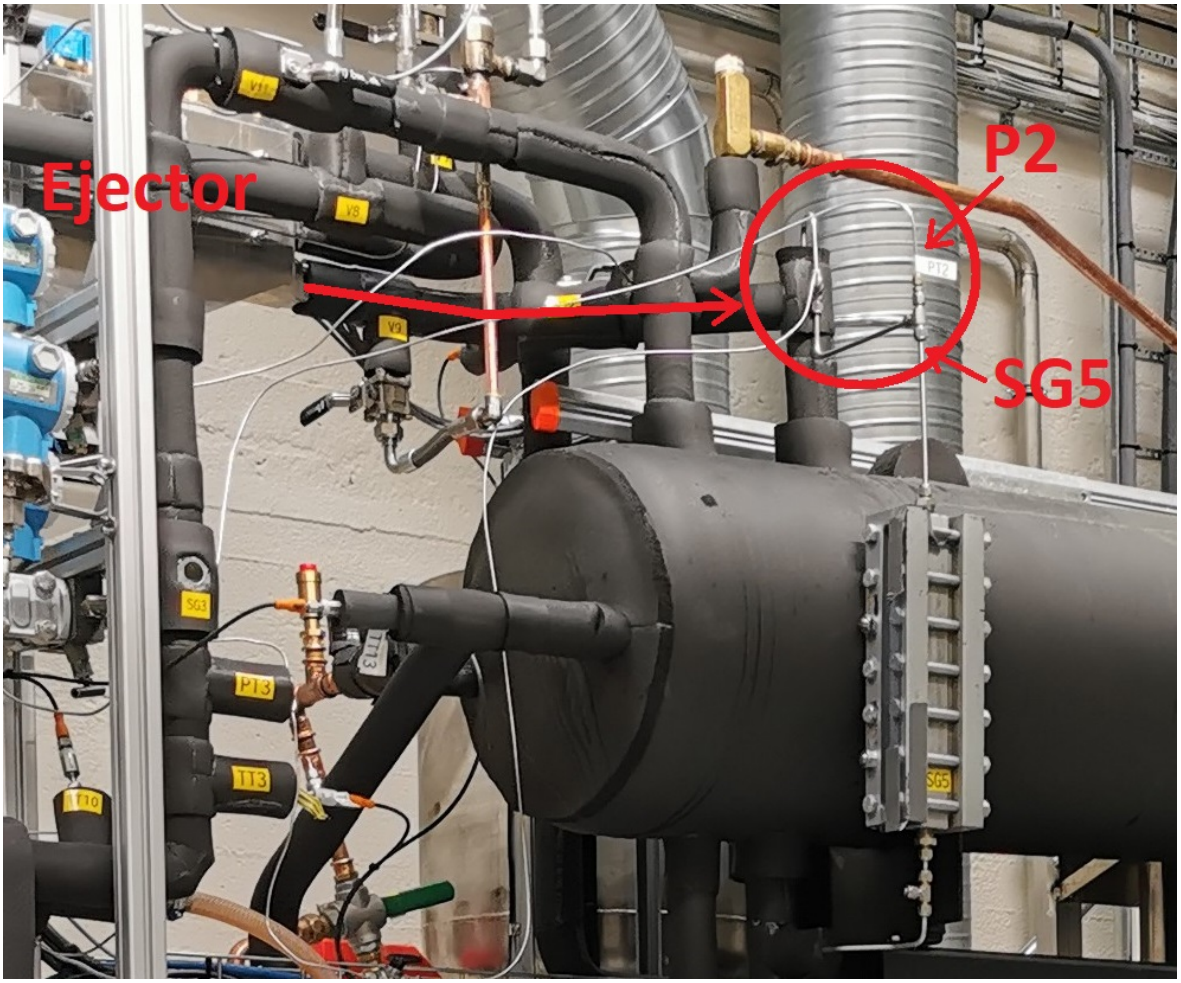


Figure 22: The placement of P2 and upper inlet of SG5

As the inlet of SG5 proved to display a higher pressure than the receiver tank, and P2 was coupled at the same exact coupling point (see Fig. 22), it follows logically that P2 also would register a slightly higher pressure than the actual static pressure in the receiver tank. Results viewed in Section 5.2 support the hypothesis.

An attempt to obtain an approximate liquid height was still carried out. Each time the test rig was switched on in the morning, the first priority was to relocate all liquid CO_2 to the receiver tank and the test configuration. This was done by evaporating all liquid in the low pressure (LP) receiver of the MultiTest-rack using the MT-evaporator. Once there was no more liquid in the lower sight glass on the LP receiver, and no change in glycol temperature over the MT-evaporator, the ejector would be turned off, and the liquid level in the receiver tank was measured through SG5. The glycol loop was constantly running through the MT-evaporator to immediately evaporate any liquid entering the LP receiver and returning it to the compressor. The liquid height was measured a second time when turning off the test rig in the evening to see if the liquid height had changed during tests. The liquid height in the

receiver tank turned out to be very consistent, with results ranging from 26 - 28 cm and an average height of $\approx$ 27 cm.

3.4 Data collection and calculations

The collected data points were stored in two separate files. All values displayed in Figure 20 were stored in a TDMS file throug LabVIEW, and all sensor values from Figure 21 were stored in a CSV file using Minilog.

Data points were logged every second for the whole duration of the test rigs run-time. To retrieve the correct data, the date, time, and duration of each test were noted in the test matrix. Three python scripts were used to compile the relevant data and to perform the desired calculations. All python scripts were originally made by Ángel Álvarez Pardiñas for previous projects conducted on the MultiTest-rig. The files were edited to fit the layout and needs of the current test configuration. All scripts are available in Appendix D.

3.4.1 Main.py

Main.py extracts and merges the requested data points from the Minilog and LabVIEW files into several new CSV files, one for each test run. The newly created CSV files are processed through accuracyCalculations.py to add the accuracy calculations of each sensor, and system-Performance.py where additional calculations and accompanying accuracy calculations are made. The average values for each test run are calculated and merged into a final Excel file where the average values and errors of all test runs are displayed.

3.4.2 accuracyCalculations.py

The error in the measuring equipment is calculated in accuracyCalculations.py. It is assumed that the sensors are correctly installed and calibrated according to the operation manual. The calculation method and stated accuracy of each sensor according to the manufacturer is displayed in Section 4.

3.4.3 systemPerformance.py

The system performance calculations and their accompanying errors are calculated in system-Performance.py. Thermodynamic values and properties provided by CoolProp [30] were used in addition to the measured values.

4 Uncertainty and sensitivity

Most results presented in this thesis will be calculated using one or more of the measured values attained by the measuring equiptment described in Section 3.1.2. What must be kept in mind when processing these values is that the *measured value* is not necessarily equal to the *true value*.

In general, an error will always arise due to miscellaneous imperfections in the measurement. The error represents the difference between the true value and the measured value. We have no means of knowing the exact error, but we can calculate an estimated interval based on available knowledge in which the error is likely to be. This estimate is referred to as the *uncertainty*, and can be defined as a "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed a measurand" [31]. It is generally understood that given correct installation and handling of the measuring equipment, the measured result is the best estimate of the value of the measurand.

The uncertainty in measurements will in this thesis be calculated using the methods described in the Guide to the Expression of Uncertainty in Measurement, also known as the GUM [31].

4.1 Uncertainty

Uncertainty in measurements is expressed as the standard deviation of the measurand. A measurand is a well defined physical quantity intended to be measured. An example of a measurand from the experiments is the temperature of the CO_2 in °C at the outlet of the evaporator on the gravity loop side, given a receiver pressure of 41 bar, a water mass flow rate of 24 kg/min, and an evaporator inlet water temperature of 15 °C.

The GUM supplies two different approaches to calculating the uncertainty of the measurand. The first approach is called Type A, and is a method for evaluating the uncertainty by the statistical analysis of a series of observations. The other approach, Type B, is a method used to evaluate the uncertainty by means *other* than the statistical analysis of a series of observations. It is instead evaluated by scientific judgement based on all of the available information on the possible variability of a measurand.

4.1.1 Type A

During the experimental runs, measurements were logged every second for approximately 10-15 minutes for each measurand. Some tests were logged for a longer period, while others for a shorter period depending on the stability of the system. These measured points formed the intervals to which the Type A method was applied.

The formula for calculating the experimental standard deviation, $u(x_k)$, of a series of N measurements in the same measurand is given as:

$$u(x_k) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{17}$$

where x_k is the result of the kth measurement and $\overline{x}$ the arithmetic mean of the N results. This formula was applied to each measurand.

4.1.2 Type B

All sensors used to collect the measured data provide a certain accuracy in their measurements. The reported accuracy is provided in the datasheet of each component and is noted in their respective tables. To calculate the effect on the uncertainty provided by these inaccuracies, the Type B model was applied.

Absolute pressure sensors and differential pressure sensors
The pressure sensors are calibrated according to their set span. The set span ranges from its lower range value (0 bar) to its upper range value. Since the lower range value is 0 bar, the set span is represented as the upper range value. The set span for the absolute pressure sensors (PT) and the differential pressure sensors (DPT) are presented in Table 9 and Table 10 respectively.

Table 9: Accuracy and set span of the absolute pressure sensors

Component	TAG	Accuracy [% of set span]	Set Span [bar]
Cerabar PMP71	PT1	0,075	70
	PT2	0,075	50
	PT3	0,075	60
	PT4	0,075	60

Table 10: Accuracy and set span of the differential pressure sensors

Component	TAG	Accuracy [% of set span]	Set Span [bar]
Deltabar S PMD71	DPT1	0,035	10
	DPT2	0,035	5
	DPT3	0,035	5
	DPT4	0,035	5

The accuracy is represented as a percentage of the set span of the sensors. To convert the provided data to uncertainty represented as the estimated standard deviation, $u(x_k)$, the following equation is used:

$$u(x_{PT}), u(x_{DPT}) = \frac{\frac{accuracy}{100} \cdot setspan}{k_p} \tag{18}$$

The *accuracy* and *setspan* for PT and DPT are provided in Table 9 and Table 10 respectively. k_p is the coverage factor representing our level of confidence that the true measured value is within the chosen distribution. A $k_p = \sqrt{3}$ represent a confidence of 100 % for a rectangular distribution. This is standard for sensors and measuring equipment unless anything else is stated.

Mass flow meters
The uncertainty of the mass flow meters is calculated in a similar fashion to the pressure sensors. The difference is that the reported accuracy is a percentage of the reading in the normal operating range. To avoid unnecessary iterations, the arithmetic mean value, $\overline{x}$, of the measurand is used to calculate the estimated standard deviation $u(x_k)$ through the formula:

$$u(x_{MF}) = \frac{\frac{accuracy}{100} \cdot \overline{x}}{k_p} \tag{19}$$

$k_p = \sqrt{3}$ for the same reasons as mentioned in Equation 18. The *accuracy* of the two mass flow meters used in the experiments are specified in Table 11. Uncertainties outside the mass flow meters' respective operational rang are neglected, as such mass flows are not expected to occur during testing.

Table 11: Accuracy and operational range of the mass flow meters

Component	TAG	Accuracy [% flow rate]	Operational range [kg/min]
Rheonik RHM 08L	MFmot	0,2	1 - 50
	MFn	0,2	1 - 50
	MFg	0,2	1 - 50
	MF1	0,2	1 - 50
	MF2	0,2	1 - 50
Rheonik RHM 06L	MFsuct	0,5	0,1 - 36

Temperature sensors
The temperature sensors have a base accuracy of $\pm 0,15$ °C at 0 °C, with an added $\pm 0,002$ °C for each degree in a positive or negative direction relative to 0 °C. This is specified in Table 12.

Table 12: Accuracy of the temperature sensors

Component	TAG	Accuracy [°C]
Pt100	TT3	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$
	TT4	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$
	TT14	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$
Pt1000	TT1	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$
	TT10	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$
	TT11	$\pm(0,15 + 0,002 \cdot \lvert t \rvert)$

The estimated standard deviation, $u(x_k)$, is expressed as following:

$$u(x_{TT}) = \frac{0,15 + 0,002 \cdot |t|}{k_p} \tag{20}$$

where $|t|$ is the measured absolute value of the temperature in °C, and $k_p = \sqrt{3}$ for the same reason as mentioned in Equation 18.

4.2 Sensitivity

Before combining the Type A and Type B uncertainties for each sensor, an overview on how to combine uncertainties in general is required.

The values measured from the sensors are used in functions and equations to calculate other values such as heat transfer and temperature differences. To calculate the uncertainties in these values, a sensitivity analysis has to be implemented. The sensitivity analysis is done by calculating a sensitivity coefficient (SC) for every input in a function or equation related to the output. This is done to define how the change in each parameter impacts the calculated result.

$$y = f(x_1, x_2, x_3, ...x_N) \tag{21}$$

Equation 21 describes an output y as a function of x_N parameters. To calculate the magnitude of change in y produced by a change in x_i, the partial derivative of x_i is applied using the following equation:

$$\Delta y_i = \frac{\partial f}{\partial x_i} \cdot \Delta x_i \tag{22}$$

where f is the function described in Equation 21, Δx_i is the change in the ith input variable, and Δy_i is the corresponding change in the output. If this change is generated by the standard uncertainty of x_i, the corresponding change in y can be expressed as:

$$\Delta y_i = \frac{\partial f}{\partial x_i} \cdot u(x_i) \tag{23}$$

where $u(x_i)$ is the standard uncertainty of x_i. The combined uncertainty $u_c(y)$ of a function is viewed as a sum of terms, each representing the uncertainty associated with the output y generated by the uncertainty associated with each input x_i. The equation for the combined uncertainty can thus be expressed as:

$$u_c(y) = \pm\sqrt{\left(\frac{\partial f}{\partial x_1} \cdot u(x_1)\right)^2 + \left(\frac{\partial f}{\partial x_2} \cdot u(x_2)\right)^2 + ... + \left(\frac{\partial f}{\partial x_N} \cdot u(x_N)\right)^2} \tag{24}$$

where $u_c(y)$ is the combined uncertainty of the function or equation, $\partial f/\partial x_i$ represents the sensitivity coefficient for each x_i, and $u(x_i)$ is the standard uncertainty of x_i.

4.2.1 Combined uncertainty of the sensors

To estimate the combined uncertainty of the values measured by a sensor, the uncertainties from Type A and Type B needs to be added. As the sources of uncertainty are quantified in the same units of measurement and simply added together, the partial derivative will be equal to 1 for both types. Applying the uncertainties and SC = 1 for both types to Equation 24 gives:

$$u_c(y) = \sqrt{(1 \cdot u_A(x_k))^2 + (1 \cdot u_B(x_k))^2} \tag{25}$$

where $u_c(y)$ is the combined uncertainty of output y, $u_A(x_k)$ is the Type A uncertainty for input x_k, and $u_B(x_k)$ is the Type B uncertainty for input x_k.

Although the combined uncertainty of the measurand has been estimated by applying Equation 25, there is still no guarantee that the error lies within the confines of the estimated uncertainty. An incomplete knowledge of the contributors to the error may cause the estimated uncertainty to deviate significantly from the error. Faults such as imperfect calibrations of sensors and incomplete knowledge of the correct placement of the sensors may cause the uncertainty to deviate from the error in either direction. However, if not stated otherwise, it is assumed that the correct procedures regarding calibration and installation has been followed for all test results.

5 Results

This section presents the results collected from the experiments conducted on the gravity-loop, ejector-supported loop, and the two-stage evaporator.

5.1 Gravity loop

All experiments presented in this section were conducted in gravity mode. The inlet of the ejector stage of the evaporator was closed, preventing any refrigerant from flowing through the second stage.

5.1.1 Effect of water temperature

The following graphs illustrate the impact of the evaporator inlet water temperature at a constant water mass flow rate of 16 kg/min.

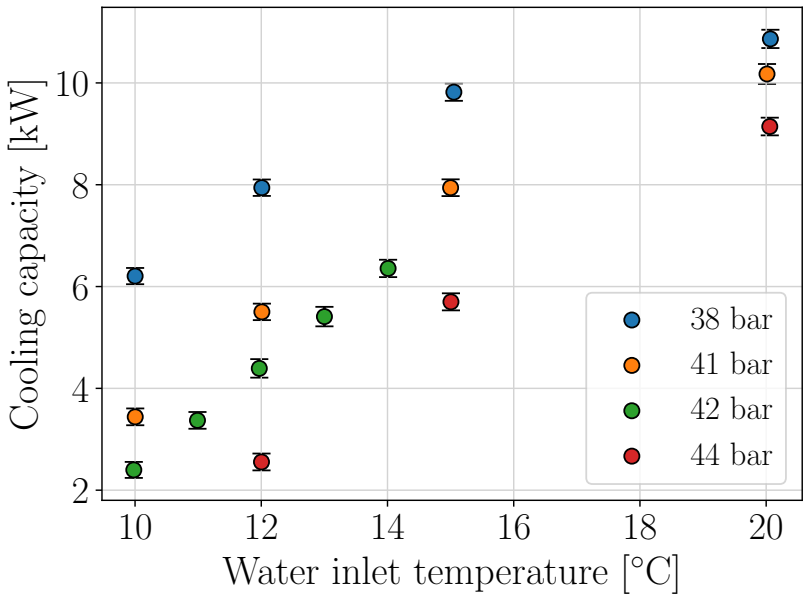


Figure 23: Cooling capacity in relation to the inlet water temperature at 16 kg/min water mass flow rate.

Figure 23 presents the resulting cooling capacity of the gravity-loop at evaporation pressures of 38, 41, 42 and 44 bar. The tests were performed with a constant water mass flow rate of 16 kg/min and temperatures ranging from 10-20 °C. Equation 11 was used to calculate the cooling capacity on the water side of the evaporator. Equation 11 shows that the cooling capacity is proportional to the difference in inlet and outlet water temperature, ΔT. The outlet water temperature is highly dependent on the evaporation temperature of the refrigerant. A lower refrigerant evaporation temperature leads to a lower outlet water temperature, resulting in a higher water temperature difference. The impact of the different evaporation pressures can be interpreted by looking at the difference in cooling capacity measured at the same inlet water temperature depicted in Figure 23.

The impact of the inlet water temperature is read by looking at the gradient of the slopes

between each measured point. A steeper slope means a higher increase in cooling capacity for each degree increase in inlet water temperature. As indicated by Figure 23, the rate of increased cooling capacity is higher at the lower temperatures than at the higher temperatures. Before examining the reason, it is important to remember the different characteristics of the heat transfer coefficient for liquid and vapor flow. A higher degree of superheat would imply that a larger area of the heat exchanger is used to heat vapor instead of liquid. The correlation between the degree of superheat and the reduction in the positive gradient is noticeable when comparing Table 13 to the results in Figure 23.

Table 13: Calculated and measured CO_2 vapor fraction and superheat at evaporator outlet at a constant 16 kg/min water mass flow rate

Pressure [bar]	Inlet water temp. [°C]	Vapor fraction [-]	Superheat [°C]
38	10	0,86	-
38	12	-	-0,01
38	15	-	10,86
38	20	-	16,59
41	10	0,64	-
41	12	0,83	-
41	15	-	0,22
41	20	-	13,50
42	10	0.59	-
42	11	0.65	-
42	12	0.74	-
42	13	0.83	-
42	14	0.93	-
44	12	0,59	-
44	15	0,86	-
44	20	-	10,44

Figure 24 illustrates the measured approach temperature of the system. The approach temperature is the difference between the water outlet temperature and the evaporation temperature. A high approach temperature indicates a rising outlet water temperature, leading to a reduction in the increase of ΔT. The continued increase in cooling capacity in Figure 23 demonstrates that the increase in outlet water temperature is not higher than the increase in inlet water temperature. The rising approach temperatures are caused by the lack of capacity on the refrigerant side of the evaporator, making the refrigerant superheat.

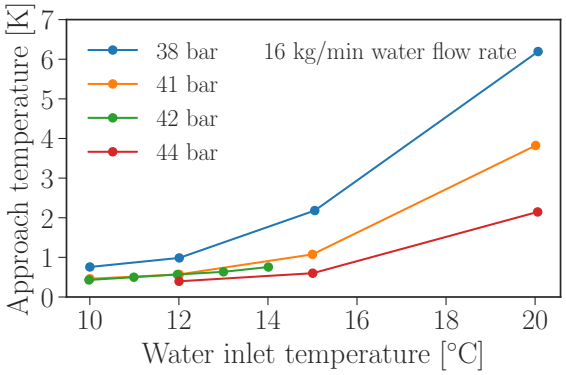


Figure 24: The approach temperature in relation to the inlet water temperature at 16 kg/min water mass flow rate.

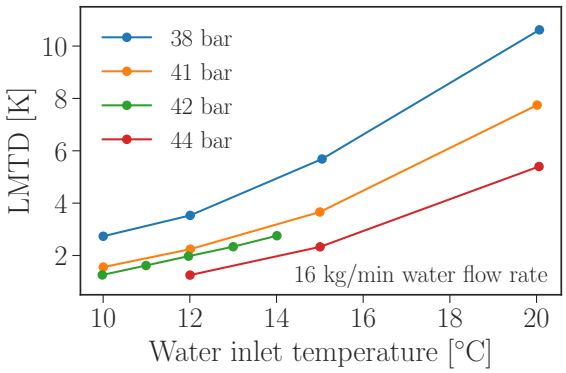


Figure 25: LMTD in relation to the inlet water temperature at 16 kg/min water mass flow rate.

Figure 25 depicts the increasing LMTD for each pressure level. Equation 10 shows that the rise in LMTD can be explained by the increase of inlet water temperature in relation to the outlet water temperature and the refrigerant evaporation temperature.

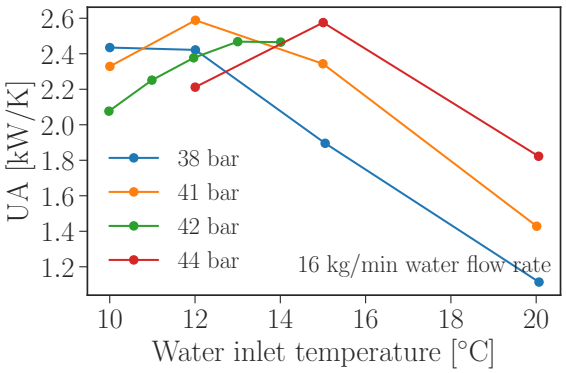


Figure 26: UA in relation to the inlet water temperature at 16 kg/min water mass flow rate.

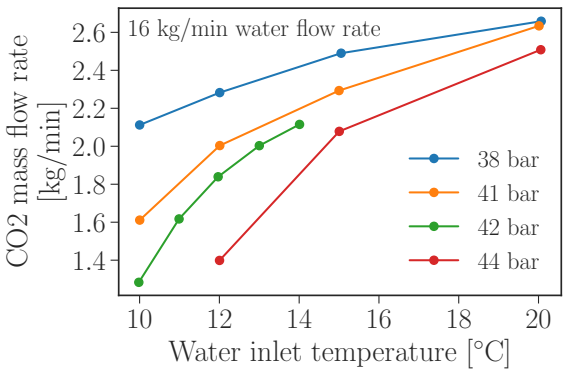


Figure 27: Refrigrant mass flow rate in relation to the inlet water temperature at 16 kg/min water mass flow rate.

The UA (W/K) is a parameter used to determine the heat-transferring abilities of the evaporator. It is a combination of the heat transfer coefficient, U, and the heat transferring area, A, of the BPHX. It was chosen as the parameter to show the heat-transferring abilities as the area of the heat exchanger was unknown during calculations. By assuming no heat loss to the surroundings, it can be calculated using Equation 13. The main factors influencing the UA are the temperature difference between the two heat-exchanging media, the internal turbulence, and the phase of the media (liquid, two-phased, or gas).

Figure 26 illustrates the change in UA in relation to the increased water inlet temperature. Comparing the results to Table 13 suggests that the optimal UA independent of evaporation pressure is achieved at a vapour fraction in the range of 0,8 - 1, no superheat. Once the refrigerant starts superheating, the UA drops drastically. The presence of superheat indicates some of the heat transferring area is being used to heat gas instead of boiling liquid. The heat transfer coefficient for gas is significantly lower than the heat transfer coefficient for liquid, reducing the overall heat transfer coefficient of the evaporator.

Figure 27 presents the refrigerant mass flow rate. The mass flow rate rises with the increase of water inlet temperature. Figure 27 is fairly similar in shape to Figure 23, which can be explained through Equation 12, $\dot{Q} = \dot{m}_R \cdot \Delta h$. As shown by the equation, the refrigerant mass flow rate is directly related to the cooling capacity and the change in enthalpy of the refrigerant.

5.1.2 Effect of water flow rate

The following graphs illustrates the impact of the water mass flow rate in the evaporator at a constant inlet water temperature.

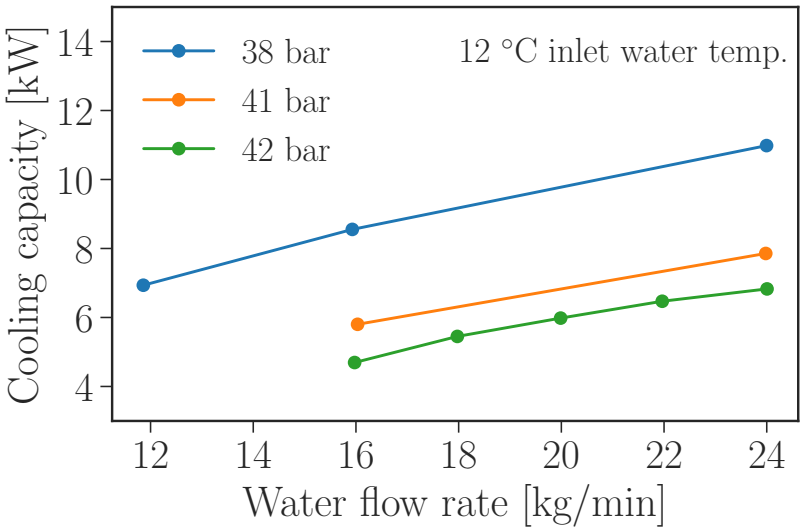


Figure 28: Cooling capacity in relation to the water mass flow rate at 12 °C inlet water temperature.

Figure 28 presents the cooling capacity of the gravity-loop at 38, 41, and 42 bar evaporation pressure. The tests were performed with a constant 12 °C inlet water temperature and water flow rates ranging from 12-24 kg/min. The cooling capacity was calculated using Equation 11 on the water side of the evaporator. Equation 11, $\dot{Q} = \dot{m} \cdot C_p \cdot \Delta T$, shows that the difference in cooling capacity between the pressure levels at the same mass flow is caused by the difference in ΔT. The difference in ΔT is due to the different evaporation temperatures between the respective evaporation pressures.

As illustrated in Figure 28, increasing the water mass flow rate, $\dot{m}$, leads to an increased cooling capacity. As $\dot{m}$ increases, the outlet water temperature also rises, causing a reduction in ΔT. Figure 29 depicts the change in the approach temperature. As the evaporation temperature is constant, the increasing approach temperature clearly illustrates the rising outlet water temperature. The reduction of ΔT contributes to slowing down the initial increase of cooling capacity caused by the increasing $\dot{m}$.

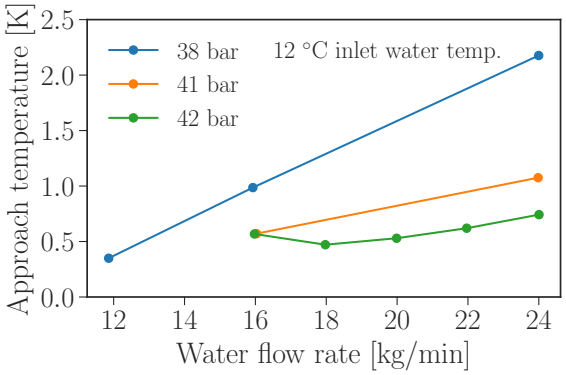


Figure 29: The approach temperature in relation to the water mass flow rate at 12 °C inlet water temperature.

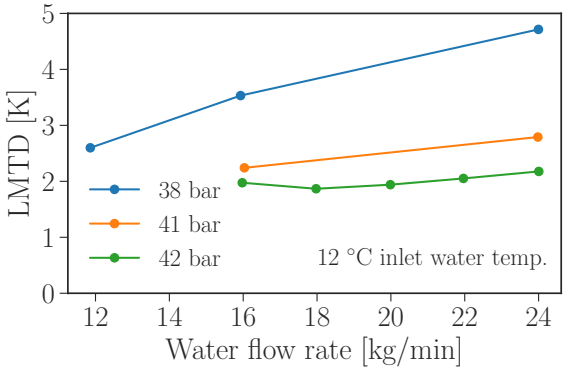


Figure 30: LMTD in relation to the water mass flow rate at 12 °C inlet water temperature.

Figure 30 depicts the increasing LMTD for each pressure level. By looking at Equation 10, the rise in LMTD can be explained as a consequence of the increased outlet water temperature in relation to the constant inlet water temperature and refrigerant evaporation temperature.

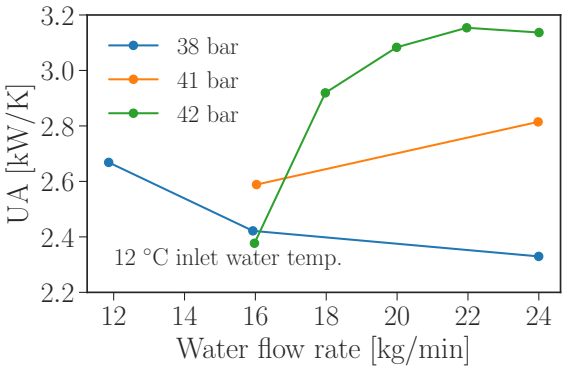


Figure 31: UA in relation to the water mass flow rate at 12 °C inlet water temperature.

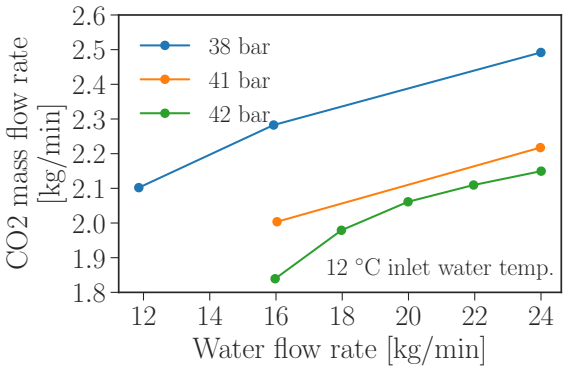


Figure 32: Refrigrant mass flow rate in relation to the water mass flow rate at 12 °C inlet water temperature.

Figure 31 illustrates the change in UA in relation to the increased water mass flow rate. At 38 bar evaporation pressure, the slope shows a negative gradient. The steep gradient between 12-16 kg/min water flow rate is likely to be caused by the transition from two-phased refrigerant flow to superheated vapour at the evaporator outlet for the respective mass flow rates. As stated earlier, the transition from two-phased refrigerant flow to superheated refrigerant flow has a significant effect on the heat transfer abilities of the evaporator.

At 42 bar evaporation pressure, the slope shows a positive gradient between each interval. The highest gradient slope lies between 16-18 kg/min water mass flow rate. A plausible explanation might be the increase in refrigerant mass flow rate depicted in Figure 32. Increasing mass flow increases the heat transfer rate in an evaporator by introducing more turbulence.

Another explanation might be found in the fact that the measured point of 42 bar evaporation pressure, 12 °C inlet water temp and 16 kg/min water flow rate was measured on a separate day than the following four points of 18, 20, 22 and 24 kg/min water flow rate. Minor changes in one or more conditions might have occurred, causing the measured result at 16 kg/min to

deviate somewhat from the expected trend set by the other four measurements. A similar deviation at the same measured point can be seen in Fig. 28, 29, 30 and 32.

Table 14: Calculated and measured CO_2 vapor fraction and superheat at evaporator outlet for constant 12 °C inlet water temperature

Pressure [bar]	Water mass flow rate [kg/min]	Vapor fraction [-]	Superheat [°C]
38	12	0,90	-
38	16	-	-0,01
38	24	-	7,88
41	16	0,83	-
41	24	-	-0,01
42	16	0.74	-
42	18	0.80	-
42	20	0.84	-
42	22	0.89	-
42	24	0.92	-

Table 14 provides an overview of the amount of vapor fraction or superheat for each measurement provided in section 5.1.2 at a constant water temperature of 12 °C.

5.1.3 Running the two-stage evaporator in gravity mode only

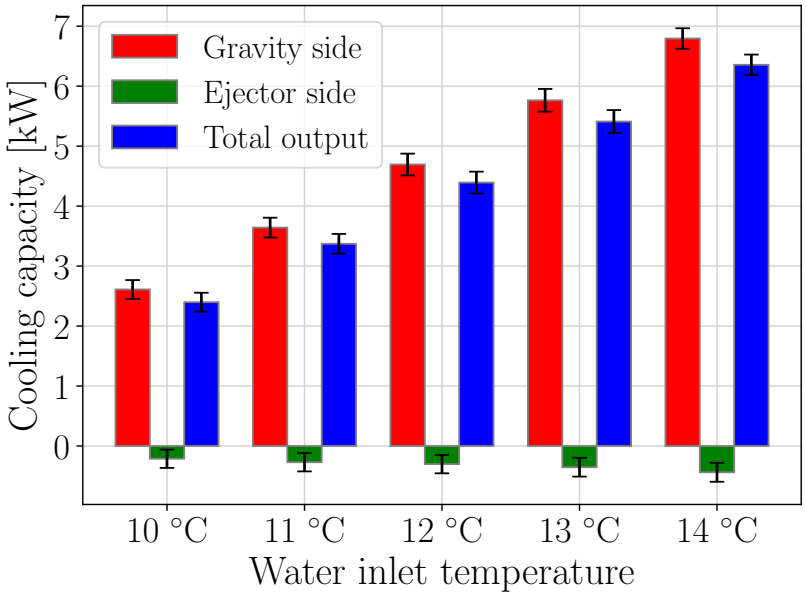


Figure 33: Cooling capacity at 42 bar evaporation pressure and 16 kg/min water mass flow rate. The capacities are measured on the gravity side, ejector side, and combined. The ejector assisted loop is inactive.

The cooling capacity was measured on the water side for each stage in the evaporator using temperature and water mass flow rates. The water temperature was measured at the inlet, middle, and outlet of the evaporator by the temperature sensors TT10, TT14, and TT11 respectively. Figure 36 displays the cooling capacities measured during a gravity only-test at 42 bar evaporation pressure and 16 kg/min water mass flow rate.

5.2 Measured differential pressure in the downcomer and riser

Figure 34 shows the measured differential pressure in the downcomer and riser for tests performed during only gravity mode.

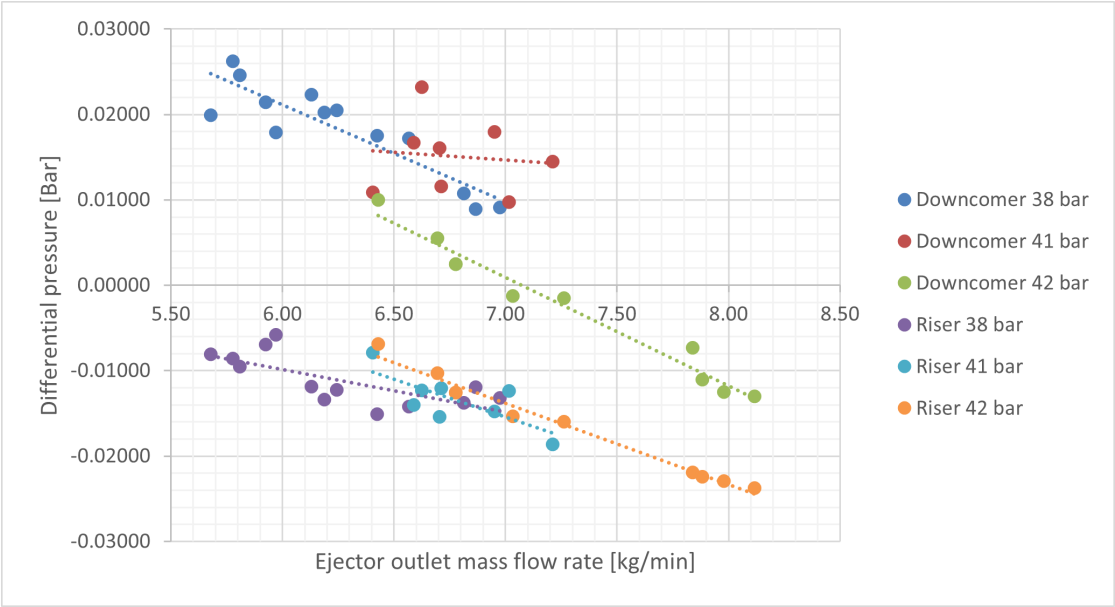


Figure 34: Differential pressure in the downcomer and the riser as a function of ejector outlet mass flow rate at gravity mode.

Figure 35 shows the measured differential pressure in the downcomer and riser for tests performed during combined mode.

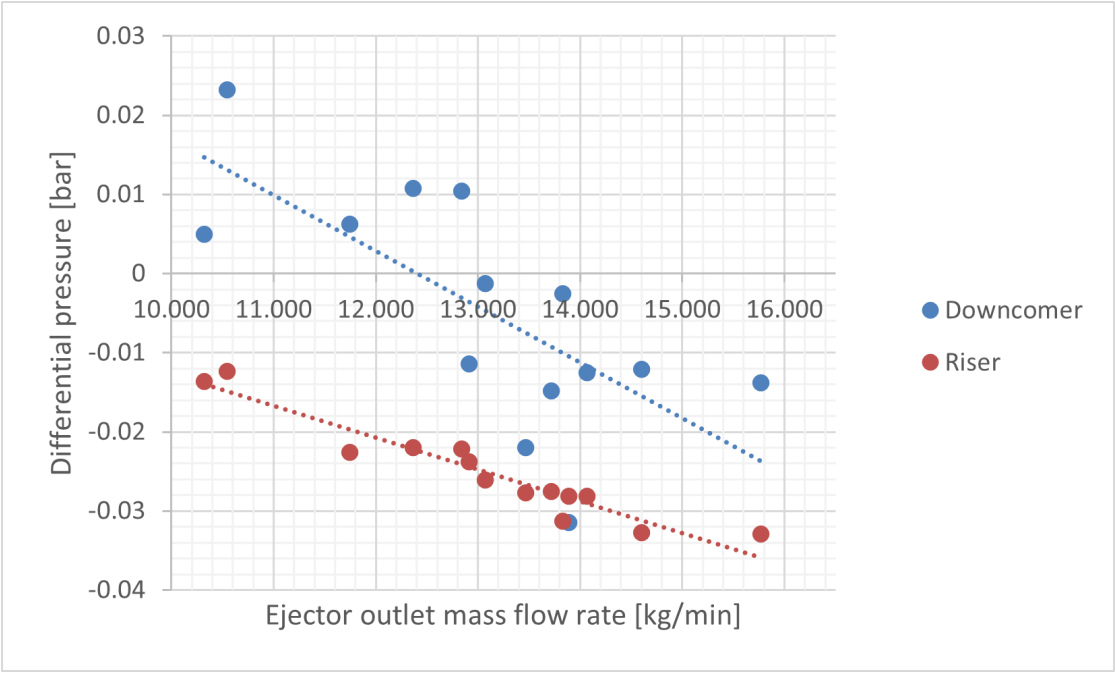


Figure 35: Differential pressure in the downcomer and the riser as a function of ejector outlet mass flow rate at combined mode. The receiver tank pressure is $\approx$ 42,2 bar.

The differential pressure over the riser was calculated as follows:

$$dP_{\text{riser}} = P3 - P2 \tag{26}$$

where $P3$ is the pressure at the evaporator outlet, and $P2$ is the pressure in the receiver tank.

The pressure at the evaporator inlet is simply the pressure at the evaporator outlet, $P3$, plus the differential pressure between the evaporator inlet and outlet, $DP3$. The differential pressure over the downcomer is therefore calculated as:

$$dP_{\text{downcomer}} = (P3 + DPT3) - P2 \tag{27}$$

According to equation 26 and 27, the resulting differential pressure in both the riser and downcomer should always be positive, as both $P3$ and $P3 + DP3$ should in theory always be greater than $P2$.

As mentioned in Section 3.3, it is suspected that the placement of pressure sensor P2 is such that the dynamic pressure from the ejector outlet flow affects the measured values. During the gravity test runs, the ejector motive mass flow rate was equal to the outlet ejector mass flow rate, as the suction side of the ejector was shut.

Correlations in Figure 34 and 35 seem to support the claim that the ejector outlet mass flow rate affects the differential pressure measurements, with the exception of the results for the downcomer at 41 bar for gravity mode, where the results seem to be randomly scattered.

5.3 Ejector and gravity loop combined

All experiments presented in this section were conducted in combined mode, meaning both the gravity loop and ejector assisted loop were active. The evaporation pressure at the ejector stage was 38 bar, while the evaporation pressure in the gravity stage was approximately 42.2 bar.

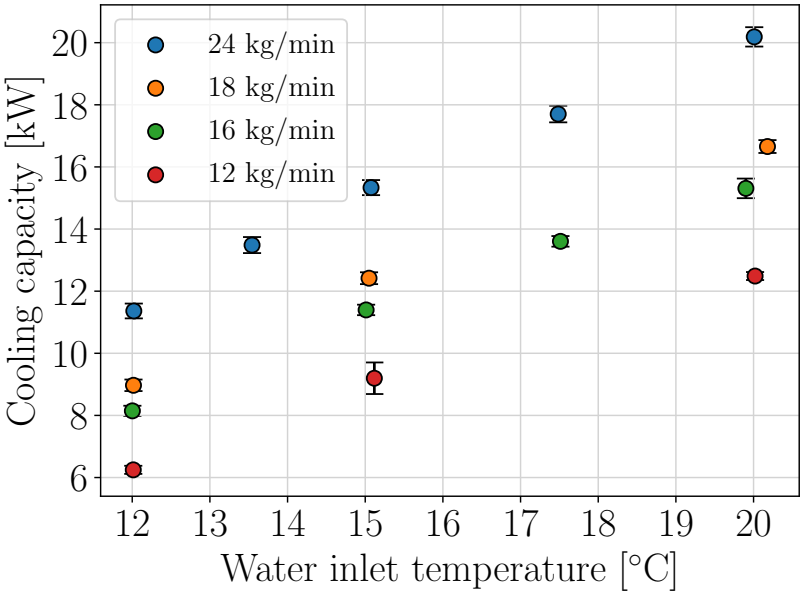


Figure 36: Combined cooling capacity in relation to inlet water temperature at four different water mass flow rates.

Figure 36 displays the overall cooling capacity at various water mass flow rates as a function of water inlet temperature. The different mass flow rates are 12, 16, 18, and 24 kg/min. The temperatures were measured at the evaporator's inlet and outlet.

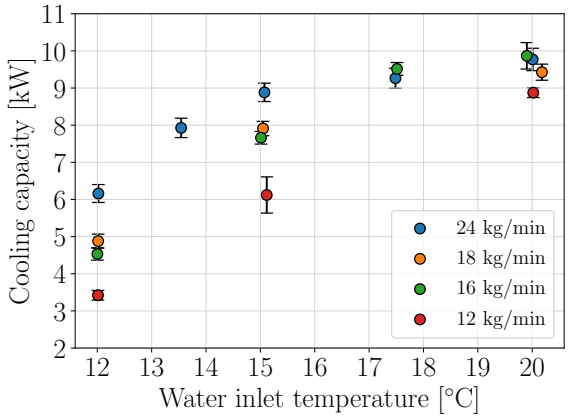


Figure 37: Gravity stage cooling capacity in relation to inlet water temperature at four different water mass flow rates.

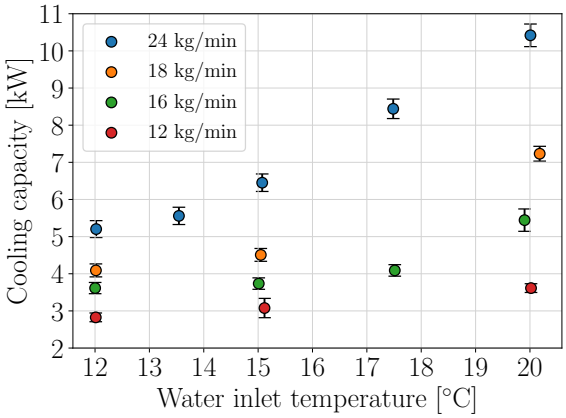


Figure 38: Ejector stage cooling capacity in relation to inlet water temperature at four different water mass flow rates.

Figur 37 og Figur 38 shows the distribution of cooling capacity in each of the two evaporation stages.

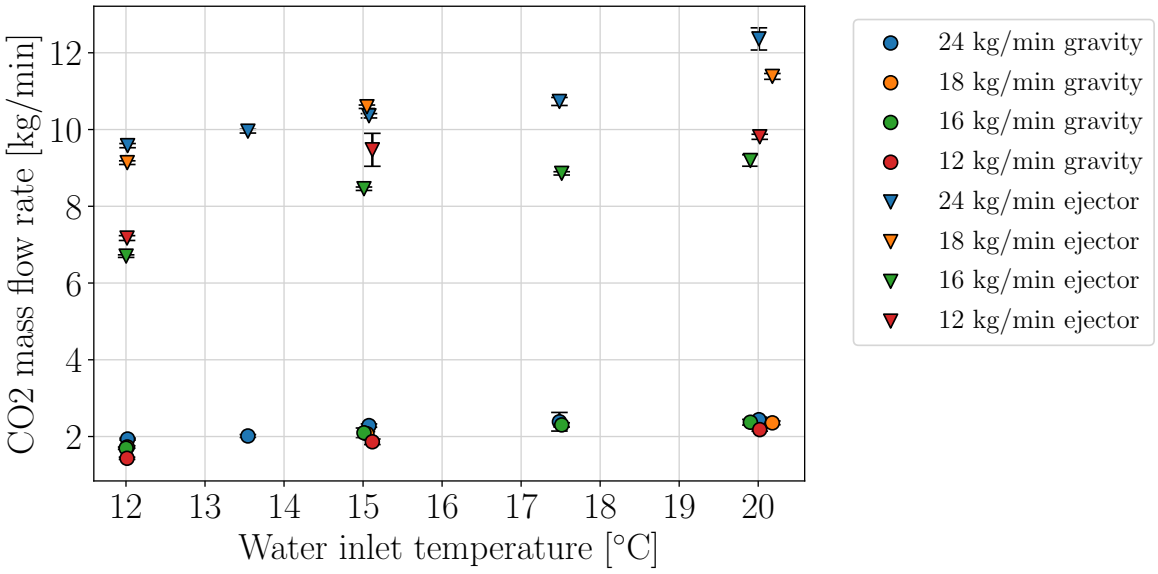


Figure 39: The CO_2 mass flow rate through the gravity stage and the ejector stage of the two-stage evaporator at their respective inlet water temperatures.

Figure 39 shows the distribution of CO_2 mass flow rate in the gravity loop and the ejector-assisted loop.

6 Discussion

In this section, the results will be presented, compared to, and discussed, in relation to the reference values provided in Table 3 in Section 2.7.

6.1 Gravity loop

The initial phase of the experiments focused solely on the gravity loop. As outlined in Table 3, the gravity loop was designed with specific parameters: 20 kW cooling capacity, 42 bar evaporation pressure, a CO_2 mass flow rate of 0.1378 kg/s (8.27 kg/min), an outlet vapor fraction of 0.70 kg/kg CO_2 , a water mass flow rate of 1.589 kg/s (95.34 kg/min), an inlet water temperature of 13 °C, and an outlet water temperature of 10 °C.

6.1.1 Low flow rate and high pressure losses

When examining the test results, the most notable observation is that the highest achieved cooling capacity is only about half of the expected cooling capacity based on reference values. Both Fig. 23 and Fig. 28 illustrate a maximum cooling capacity of approximately 11 kW at an evaporation pressure of 38 bar. In these instances, a superheat of 16.59 °C and 7.88 °C is recorded, as per Table 13 and Table 14, respectively. Low cooling power and high superheat suggest that the CO_2 mass flow rate is insufficient to meet the cooling demand. This is further confirmed in Fig. 27 and Fig. 32, where the highest measured mass flow is 2.66 kg/min, 68 % lower than the designed value.

The cause of the low CO_2 mass flow rate is attributed to excessive pressure drop compared to the pumping pressure provided by the liquid leg in the downcomer. The solution lies in increasing pumping pressure and/or reducing the pressure drop in the loop. To boost pumping pressure, one would need to increase the elevation difference between the receiver tank and the evaporator. However, since one of the primary objectives of implementing this type of evaporator is to maintain system compactness, this solution may need to be considered after exploring other alternatives. A more viable approach is to attempt to reduce the pressure drop. Aside from the pressure drop across the evaporator, it is likely that there is some unnecessary pressure drop through the riser and the CO_2 mass flow meter (MFsuct). Early assumptions suggested that the riser's pipe dimension is too small, a notion validated by simulations conducted by Jan Bengsch in his master's thesis [9]. Figure 40 in Appendix A indicates that the CO_2 mass flow meter is expected to contribute to an increase in pressure drop. The figure demonstrates that an increasing mass flow results in an exponentially rising pressure drop across the measuring instrument, which would, in turn, pose greater resistance to potential improvements at other points in the loop. Removing the mass flow meter would mean sacrificing the ability to collect valuable data, but may be necessary to achieve the desired effect.

6.1.2 The effect of vapor fraction

The experiments conducted at 42 bar primarily aimed to achieve a design vapor fraction of 0.70 at the design pressure and to investigate how vapor fraction influences heat transfer internally within the evaporator. The experiment was initially carried out by maintaining a constant water flow rate of 16 kg/min while varying the incoming water temperature into the

evaporator between 10-14 °C. The subsequent experiment involved maintaining a constant water temperature of 12 °C at the evaporator inlet and varying the water flow rate between 16-24 kg/min into the evaporator.

A vapor fraction of 0.74 was achieved at a mass flow rate of 16 kg/min and an incoming water temperature of 12 °C. Under these conditions, a CO_2 mass flow rate of 1.84 kg/min and a cooling capacity of 4.7 kW were recorded. The highest achieved cooling capacity at 42 bar evaporation pressure and a vapor fraction < 1.0 was 6.8 kW. This was attained with an incoming water temperature of 12 °C, a water mass flow rate of 24 kg/min, a CO_2 mass flow rate of 2.15 kg/min, and a vapor fraction of 0.92. The cooling capacity would likely have been higher if the water mass flow rate had been allowed to increase further.

The highest measured UA value was 3.15 kW/K and was achieved at 42 bar evaporator pressure, a water inlet temperature of 12 °C, a water flow rate of 22 kg/min, and a vapor fraction of 0.89. Comparing Fig. 26 and Table 13 reveals that UA starts to decrease rapidly once the evaporator produces superheat. This is expected since heat transfer in a plate heat exchanger between a gas and a liquid is significantly lower than that between two liquids.

6.1.3 Differential pressure in the downcomer and riser

An essential aspect of the experiment was the measurement of pressure drops across the downcomer, the evaporator, and the riser. These values would have provided valuable insights for optimizing the loop itself and simulating two-phase CO_2 flow in pipes. Unfortunately, due to the unfavorable placement of pressure sensor P2 and sight glass SG5 at the ejector outlet, the measurements based on these parameters carry significant uncertainty. In practice, P2 likely measures the total pressure, including static, hydrostatic, and dynamic pressure from the ejector, whereas the remaining pressure sensors only measure static and hydrostatic pressure in the pipelines.

In theory, the differential pressure measured across the riser and the downcomer should have a positive sign due to the arrangement of equations 26 and 27. In the downcomer, the total pressure should be higher at the evaporator inlet than the static pressure in the receiver tank. This is attributed to the additional hydrostatic pressure at the evaporator inlet provided by the liquid leg. A negative sign in the downcomer indicates that the dynamic pressure from the ejector exceeds the hydrostatic pressure of a 0.7 m liquid leg.

For the refrigerant to flow from the evaporator outlet, up through the riser, and back to the receiver tank, the total pressure must be slightly higher at the evaporator's exit than at the receiver tank's inlet. According to Figures 34 and 35, the measured pressure in the receiver (P2) is consistently higher than the measured pressure at the evaporator's exit (P3) in all measurements. This pressure difference appears to increase with the rise in ejector outlet mass flow rate. As long as P3 only measures static and hydrostatic pressure in the riser, and P2 includes the dynamic pressure from the ejector, these measurements will remain inconclusive.

6.1.4 Loss of cooling capacity in the ejector stage of the evaporator

Figure 36 shows an unexpected loss in cooling capacity through the ejector stage during tests with the gravity loop. As the ejector stage of the evaporator was inactive and closed off on the CO_2 side, the expected outcome was for the total cooling capacity to roughly equal the cooling capacity produced by the gravity stage. However, the graph shows a considerable loss

in cooling capacity through the ejector stage. This trend applied to all tests conducted in gravity mode. The reason for the losses is that the water heats up as it flows through the inactive stage. The temperature increase ranges from 0.16 K to as high as 0.87 K, which far exceeds the expected difference between the temperature sensors.

One plausible explanation is that heat may be "leaking" through the central divider plate within the heat exchanger. Figure 11 shows how hot water enters the heat exchanger from the top and gradually cools while descending towards the bottom. Within this second stage, water ascends and exits from the top on the opposite side of the entrance. As cooling is not active in the second stage, the water is solely subjected to warming by the central divider plate.

The placement of the temperature sensors might also be a contributing factor, especially the sensor placed in the middle of the evaporator. Given that this sensor is installed in such a manner that it cannot be visually inspected, as opposed to the entrance and exit temperature sensors, it remains challenging to determine its influence on the registered temperature difference.

6.2 Two-stage evaporator

The results of the combined test show that the concept works. Figures 36, 37 and 38 show that a higher water mass flow rate provides a more even distribution of cooling capacity in the two stages of the evaporator. Figure 39 clearly emphasizes how the gravity loop does not achieve a high enough CO2 mass flow rate.

7 Conclusion

The main objective of the thesis was to run experimental investigations on a novel two-stage evaporator in a transcritical CO_2 system. The thesis aimed to investigate the performance of the evaporator in single-stage and in two-stage, as well as evaluating the performance of the gravity loop configuration.

This is the first time that this type of testing has been conducted on the rig with an integrated two-stage evaporator. Due to the lack of experience, several significant obstacles were encountered along the way, greatly affecting the scope of the thesis.

The tests that were planned and carried out did not prove to be the most suitable for the thesis. Several of the results from specific experiments have been excluded from discussion due to their limited relevance. They are, however, included in the attachments.

The gravity loop was able to achieve a cooling capacity of approximately 11 kW at an evaporation pressure of 38 bar. This is approximately half of the expected cooling capacity. The low cooling capacity is attributed to low CO_2 mass flow rate through the gravity loop, caused by the high pressure drop across the loop compared to the pumping pressure provided by the liquid leg.

The unfortunate placement of the P2 pressure sensor made the measurement of pressure drop across the downcomer and the riser invalid. This made it hard to identify what part of the gravity loop contributed most to the pressure loss.

Another challenge is the loss of cooling capacity in the ejector stage of the evaporator during gravity mode tests. This is likely due to heat leakage through the central divider plate or the positioning of the temperature sensors.

The two-stage evaporator concept was shown to be functional. Figures 36, 37 and 38 show that a higher water mass flow rate provides a more even distribution of cooling capacity in the two stages of the evaporator.

Overall, the results of this thesis show that the gravity loop and two-stage evaporator concept is feasible for CO2 transcritical refrigeration. However, further work is needed to optimize the design and improve the performance of the system.

8 Further Work

The most critical part of making the concept work optimally is to reduce the pressure loss in the gravity loop. This can be done by increasing the pipe diameter in the riser and by removing the mass flow meter in the downcomer.

The second thing that needs to be done is to relocate the pressure sensor P2 and the connection to SG5 to a new point at the top of the receiver tank. This will allow for the measurement of the liquid height in the tank, as well as accurate pressure measurements. With accurate pressure measurements, it will be possible to see the pressure difference between the riser and the downcomer, and to analyze the effect of the measures taken to reduce pressure loss.

Additional experiments should be conducted to investigate the liquid distribution in the ejector stage and the gravity stage more closely. A more even distribution equal to the mass flow in the design criteria should be a goal.

The experiments must be carefully planned in advance, and the goal of the task must be clearly defined from start to finish.

References

[1] *FNs klimapanel: Alvorlige klimaendringer er i full gang - Miljødirektoratet*. https://www.miljodirektoratet.no/aktuelt/nyheter/2021/august-2021/fns-klimapanel-alvorlige-klimaendringer-er-i-full-gang/. (Accessed on 12/16/2021). Aug. 2021.

[2] *EU legislation to control F-gases*. https://ec.europa.eu/clima/eu-action/fluorinated-greenhouse-gases/eu-legislation-control-f-gases_en. (Accessed on 12/14/2021).

[3] The Association of Convenience Stores. *F-Gas Regulations.pdf*. https://www.acs.org.uk/sites/default/files/f_gas.pdf. (Accessed on 12/14/2021). 2019.

[4] Hans T. Haukås and Alexander Cohr Pachai. *Information sheets on natural refrigerants*. eng. Nordiske Arbejdspapirer. Copenhagen: Nordisk Ministerråd, 2014. DOI: 10.6027/NA2014-908.

[5] P. Gullo, L. Fusini, and A. Hafner. *D2.8 Educational e-book about MultiPACK No 1*. https://www.ntnu.edu/documents/1272037961/0/723137_Deliverable_16_%28Educational+e-book+about+MULTIPACK+No+1%29.pdf/37332fd4-da00-55d8-ab6f-ff7282949ec8?t=1616766346730. (Accessed on 12/16/2021). June 2017.

[6] Nour ABDIN et al. "CO2 commercial refrigeration system in Jordan". In: *Proceedings of the 25th IIR International Congress of Refrigeration: Montréal , Canada, August 24-30, 2019*. 25. IIF-IIR. Iif-iir - France/France, Aug. 2019.

[7] Armin Hafner, Hagar Elarga, and Prem Kumar Sherman. *D2.10 Educational e-book about MultiPACK No 3*. https://www.ntnu.edu/documents/1272037961/0/723137_Deliverable_17_Educational+e-book+about+MULTIPACK+No+3+%283%29.pdf/bdf62beb-47ac-0999-6006-fe067c100fc2?t=1638259340299. (Accessed on 12/16/2021). Nov. 2021.

[8] *MultiPACK - NTNU*. https://www.ntnu.edu/multipack/multipack. (Accessed on 05/20/2023).

[9] Jan Bengsch. *Investigation and analysis of a CO2 heat pump chiller with novel twostage evaporator*. Master's thesis. Available at https://hdl.handle.net/11250/3030759. 2022.

[10] Yunus A. Cengel and Michael A. Boles. *Thermodynamics: An Engineering Approach, Eight Edition in SI Units*. 2015. ISBN: 978-981-4595-29-2.

[11] Tryge M. Eikevik. *Compendium for Heat Pumping Processes and Systems*. 2015.

[12] E. W. Lemmon et al. *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 10.0, National Institute of Standards and Technology*. 2018. DOI: https://doi.org/10.18434/T4/1502528. URL: https://www.nist.gov/srd/refprop.

[13] Alberto Cavallini and Claudio Zilio. "Carbon dioxide as a natural refrigerant". In: *International Journal of Low-carbon Technologies* 2 (July 2007), pp. 225–249. DOI: 10.1093/ijlct/2.3.225.

[14] Armin Hafner and Ángel Álvarez Pardiñas. "CO2 refrigeration technology: possible innovations". In: *8th Conference on Ammonia and CO2 Refrigeration Technology* (Apr. 2019). DOI: 10.18462/iir.nh3-co2.2019.0024.

[15] Stefan Elbel and Neal Lawrence. "Review of recent developments in advanced ejector technology". In: *International Journal of Refrigeration 62* (Feb. 2016), pp. 1–18. DOI: 10.1016/j.ijrefrig.2015.10.031.

[16] Knut Emil Ringstad et al. "A detailed review on CO2 two-phase ejector flow modeling". In: *Thermal Science and Engineering Progress* 20 (2020), p. 100647. ISSN: 2451-9049. DOI: https://doi.org/10.1016/j.tsep.2020.100647. URL: https://www.sciencedirect.com/science/article/pii/S2451904920301670.

[17] Stefan Elbel and Pega Hrnjak. "Experimental validation of a prototype ejector designed to reduce throttling losses encountered in transcritical R744 system operation". In: *International Journal of Refrigeration* 31.3 (2008), pp. 411–422. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2007.07.013. URL: https://www.sciencedirect.com/science/article/pii/S0140700707001508.

[18] Przemyslaw Kalinski. *The Danfoss Multi Ejector range for CO refrigeration: design, applications and benefits.* (accessed: 18.06.2023). URL: https://www.danfoss.com/en/service-and-support/case-stories/dcs/the-danfoss-multi-ejector-range-for-co2-refrigeration/.

[19] Hans T. Haukås. "Design and optimization of flooded evaporators with natural circulation". In: *The joint Israeli - Norwegian symposium on refrigeration* (Jan. 1986).

[20] SWEP. *Refrigeration handbook.* https://www.swep.net/refrigerant-handbook/refrigerant-handbook/. (Accessed on 07/30/2022).

[21] Krzysztof Banasiak et al. "Development and performance mapping of a multi-ejector expansion work recovery pack for R744 vapour compression units". In: *International Journal of Refrigeration* 57 (2015), pp. 265–276. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2015.05.016. URL: https://www.sciencedirect.com/science/article/pii/S0140700715001553.

[22] Paride Gullo, Brian Elmegaard, and Giovanni Cortella. "Energy and environmental performance assessment of R744 booster supermarket refrigeration systems operating in warm climates". In: *International Journal of Refrigeration* 64 (2016), pp. 61–79. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2015.12.016. URL: https://www.sciencedirect.com/science/article/pii/S0140700715003941.

[23] Ángel Á. Pardiñas et al. "Two-stage evaporator for R744 heat pumps using greywater as heat source". In: *Energy and Buildings* 289 (2023), p. 113047. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2023.113047. URL: https://www.sciencedirect.com/science/article/pii/S0378778823002773.

[24] *onepager_multitest.pdf.* https://www.sintef.no/globalassets/project/highefflab/onepager_multitest.pdf. (Accessed on 06/26/2022).

[25] *IEC 60751:2022 - the main points and updated tolerance classes.* https://www.senmatic.com/sensors/knowledge/iec-60751-sensor-tolerance-classes. (Accessed on 07/05/2022).

[26] *Cerabar S PMC71, PMP71, PMP75 (Technical Information).* http://www.merteh.lv/eh/pdf/TI383PEN.PDF. (Accessed on 07/01/2022).

[27] *Deltabar S PMD75, FMD77, FMD78.* https://portal.endress.com/wa001/dla/5000557/7176/000/23/TI00382PEN_2716.pdf. (Accessed on 07/01/2022).

[28] *RHM06 Data Sheet.* https://www.rheonik.com/fileadmin/Documents/PDF/Datasheets/RHM/Rheonik-RHM06-Coriolis-Mass-Flow-Meter-Datasheet.pdf. (Accessed on 07/01/2022).

[29] *RHM08L Data Sheet.* https://www.rheonik.com/wp-content/uploads/pdf/Rheonik_Coriolis_RHM08_Datasheet.pdf. (Accessed on 07/01/2022).

[30] 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. eprint: http://pubs.acs.org/doi/pdf/10.1021/ie4033999. URL: http://pubs.acs.org/doi/abs/10.1021/ie4033999.

[31] *Evaluation of measurement data — Guide to the expression of uncertainty in measurement*. Standard. Joint Committee for Guides in Metrology (JCGM), Sept. 2008.

Appendix

A Pressure drop over the CO_2 mass flow meter

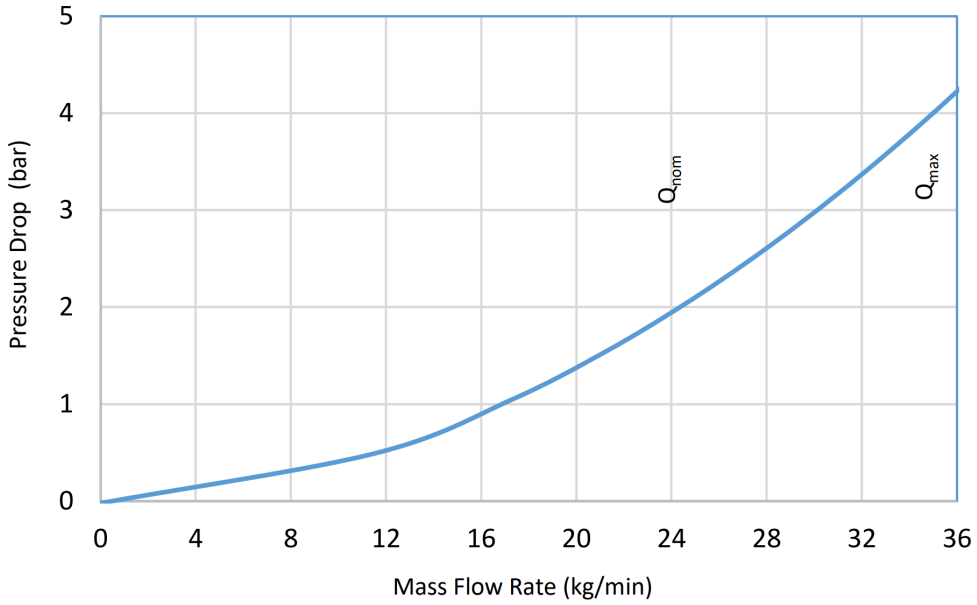
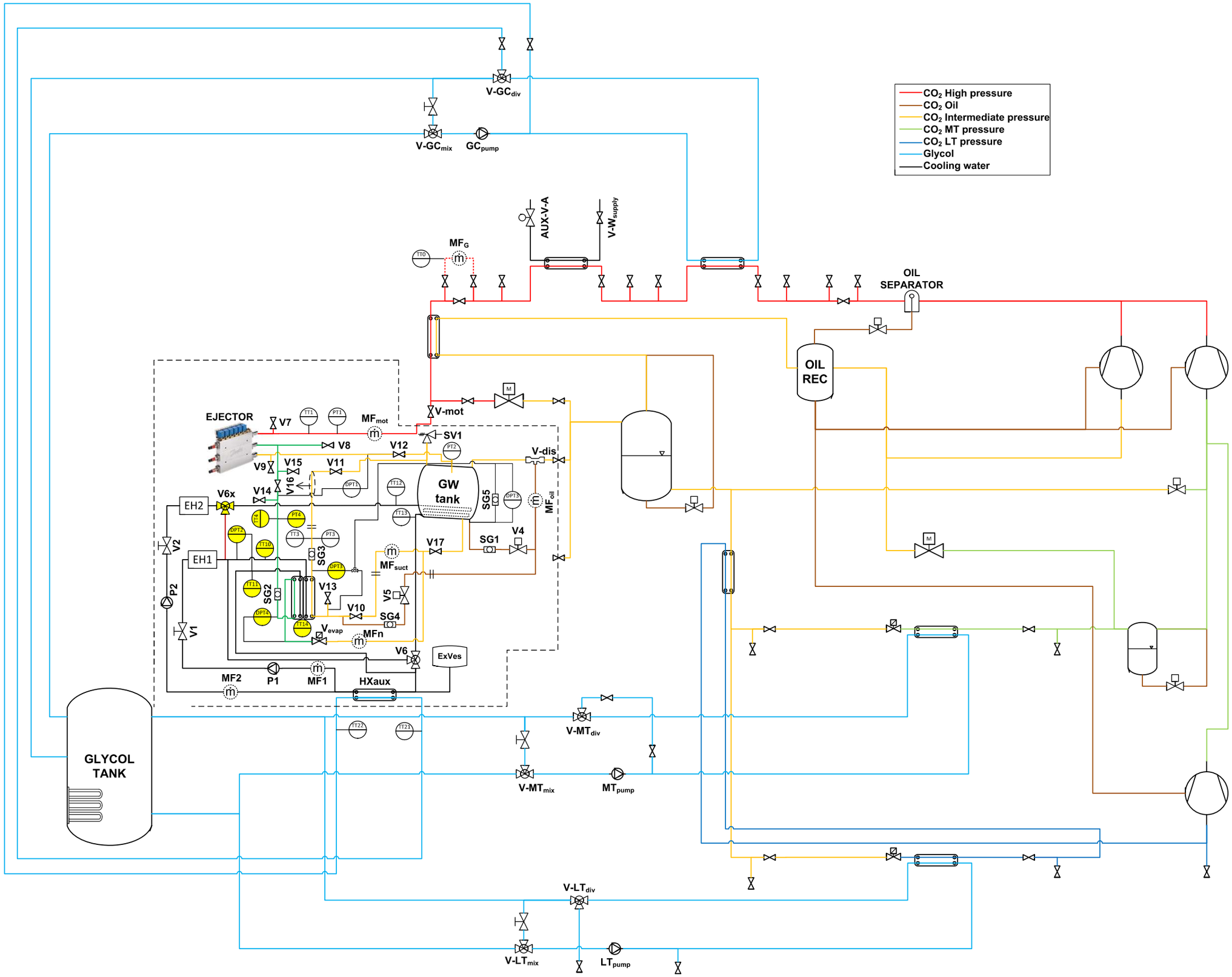


Figure 40: Pressure drop for water provided by the manufacturer [28]

Figure 40 is the pressure drop over the RHM06 mass flow meter provided by the manufacturer.

B P&ID of the Multifunctional test-rig


CO2 High pressure
CO2 Oil
CO2 Intermediate pressure
CO2 MT pressure
CO2 LT pressure
Glycol
Cooling water
V-GCdiv
V-GCmix
GCpump
AUX-V-A
V-Wsupply
MFG
TT0
OIL SEPARATOR
OIL REC
EJECTOR
V7
TT1
PT1
MFmot
V-mot
M
SV1
PT2
V8
V12
V-dis
V9
V15
V11
V16
V6x
V14
EH2
GW tank
SG5
DPT5
MFoil
TT12
TT13
DPT1
PT4
TT3
PT3
DPT2
TT10
SG3
V4
SG1
V17
MFsuct
EH1
V2
DPT3
TT11
SG2
V13
V5
P2
V10
SG4
DPT4
TT14
Vevap
MFn
V1
V6
ExVes
MF2
P1
MF1
HXaux
TT22
TT21
V-MTdiv
V-MTmix
MTpump
GLYCOL TANK
V-LTdiv
V-LTmix
LTpump

C Risk assessment

The following risk assessment was written by Krzysztof Banasiak and Helene Langeng for SINTEF Energi AS, and was available in the lab by the Multifunctional test-rack. It was thoroughly read before the first test runs to gain a proper understanding of the possible risks, their consequences, and the actions to reduce their likelihood or prevent them from happening.

Additionally, a practical demonstration of emergency procedures was performed by Ángel Álvarez Pardiñas from SINTEF.

SINTEF

Enhet: SINTEF Energi AS
Bygning: VATL
Rom eller rigg / utstyr: MultiTest-Rack, Kuldelab 1. etasje

Dato: 2021-10-05
Deltagere: Krzysztof Banasiak og Helen Langeng
Linjeleder: Petter Røkke

Id	Aktivitet	Mulig hendelse	Mulig konsekvens	Eksisterende barrierer	Risikoverdi med eksisterende tiltak				Nye barrierer / risikoreduserende tiltak (handlingsplan)	Risikoverdi med nye tiltak				Ansvarlig Frist	Status
					Menneske	Ytre miljø	Omdømme	Økonomi / materiell		Menneske	Ytre miljø	Omdømme	Økonomi / materiell		
1	Vurdering av hele riggen i ett														
2	Ingen strømning/No flow	Glemmer å åpne isvannstilførsel	Kjører uten kjøling og da stiger glykol temperatur i tanken. Trykkoppbygging på kompressor utløpssiden.	Regulator slår av kompressor. Presostat slår av kompressor. Sikkerhetsventil åpner ved 120bar. Åpning av isvannstilførsel står i sjekkliste for kjøring.	A1	A1	A1	A1	Glykol-temperatur alarm.						
3	Reversert strømning/Reverse flow	Reversert strøming i CO2-kretsen kan skje på grunn av trykkdifferanse ved at ejektor(e) yter for dårlig	Riggen får dårlig virkningsgrad	Riggen har to-trinns tilbakeslagsventil	A1	A1	A1	A1							
4	For stor strømning/More flow	Alle kretser er dimensjonert slik at det ikke er mulig å få for stor strøming. I alle kretser, vann, glykol og CO2. Ikke mulig	Ingen fare		v	v	v	v							
5	For lav strømning/Less flow	Glykolpumper kan kjøres med lav massestrøm	Riggen fungerer ustabilt. Får ikke stabile driftsforhold. Kompressorer slår på/av hele tida.Ingen fare	Sjekkliste inneholder info om instillinger for å oppnå stabile driftsforhold.	A1	A1	A1	A1							
6	Høyere nivå/More level	I glykol-, vannkrets og CO2-krets kan en få for høyt nivå av væske.	Vann: spruter ut av kretsen (er ikke en lukket krets) Glykol: en lukket krets med akkumuleringstank CO2: Tilfører for mye væske til kompressorer. De kan i verste fall ødelegges. Også, for mye væske i tankene kan gi for høyt trykknivå ved høye omgivelsestemperaturer (> 31 varmegrader)	Erfaren riggoperatør regulerer væskenivået i alle kretsene. Vanntanken fylles ut kun delvis slik at det er hele tida et ledig volum for å kompensere forandringer i tetthet. I glykolkretsen samme oppgave utføres av preinstallert kompensasjonstank med 'gassputo' for volumkompensering. Barriere i CO2 kretsen er gitt i form av avblåsningsventiler tilkoblet till felles utblåsninglinje (SuperSmart-Rack rigg). Fylling av medier utføres kun av utstyrsansvarlig	A2	B2	A2	A2							
7	Lavere nivå/Less level	Glykol-, vann, og CO2-krets kan ha for lavt nivå	Riggen fungerer ustabilt. Får ikke stabile driftsforhold. Verst I verste fall pumpene kan ødelegges.		A1	A1	A1	A1							
8	Høyere trykk/More pressure	Romstemperaturer over 31 varmegrader kan gi forhøyet trykk på CO2-kretsen og i flasken.	Trykkoppbygging i kretsen helt til sikkerhetsventil utløses. Dette kan skje bare når riggen ikke er i bruk eller om riggen brukes uten å kjøle den ned med kjølevann.	Følge med på værvarsel og romstemperatur og ha kjølevannet på for riggen. Da vil ikke temperaturen stige. Vernebriller. Avblåsningsventiler (sikkerhetsventiler) er tilkoblet till felles avblåsningslinje.	A2	A2	A2	A2							
	Høyere trykk/More pressure	Forhøyet trykk på glykol- eller vannkrets. Stenger ventiler mens pumpa er på	Trykkoppbygging i krets	Sikkerhetsventiler på glykolkretsen. Sjekkliste som beskriver at avstengingsventil skal være åpne. Vernebriller.	A1	A1	A1	A1	Tanke for å samle glykolen om sikkerhetsventil åpnes.						
9	Lavere trykk/Less pressure	For lav trykk i CO2-kretsen. For liten mengde CO2 i systemet.	Ustabil drift; redusert kjøleytelse.	Se-glass i medium- og lavtrykkbeholdere for visuell overvåking av væskenivå	A1	A1	A1	A1							
10	Høyere temperatur/More temperature	Forhøyet temperatur i CO2-krets på kompressorutløp; forhøyet temperatur i glykolkretsen	Nedbryting av smøreolje i kompressoren og kompressorhavari; smelting av pakningspasta i glykolkretsen - glykollekkasje	Termisk bryter (termistor montert på elektrisk motor til kompressoren). Sjekkliste som inneholder info om nødvendighet av isvannsirkulasjon og glykolsirkulasjon	B2	B2	A1	A2							
11	Lavere temperatur/Less temperature	For lav temperatur i CO2 kretsen pga. for lav sugetrykk	Iørrisdannelse i CO2-kretsen; frysning av vannkrystaler i glykol- og vannkretsen	For CO2-kretsen: Regulator slår av kompressor; lavtrykkpresostat slår av kompressor. For glykol- og vannkretsen: måling og oppfølging av mediumtemperatur (spesiell lavtemperaturalarmfunksjon per krets opprettes)	A2	A2	A2	A2							
12	Høyere viskositet/More viscosity	For høy viskositet til glykol (se: for lave temperaturer i CO2-kresten)	Lav massestrøm i glykolkretsen	Regulator slår av kompressor. Lavtrykkpresostat slår av kompressor.	A2	A2	A2	A2							

SINTEF

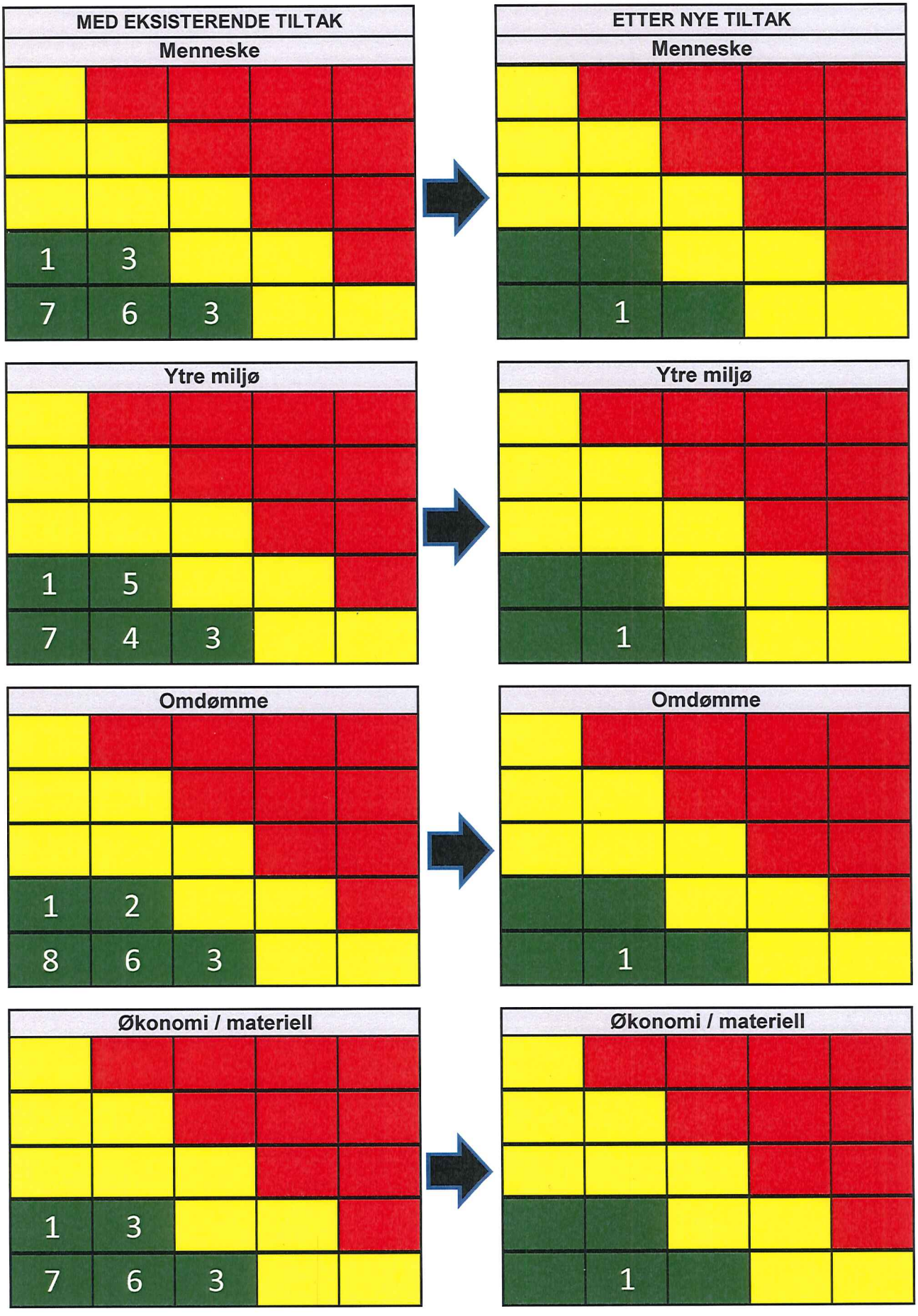
Enhet: SINTEF Energi AS
Bygning: VATL
Rom eller rigg / utstyr: MultiTest-Rack, Kuldelab 1. etasje

Dato: 2021-10-05
Deltagere: Krzysztof Banasiak og Helen Langeng
Linjeleder: Petter Røkke

Id	Aktivitet	Mulig hendelse	Mulig konsekvens	Eksisterende barrierer	Risikoverdi med eksisterende tiltak				Nye barrierer / risikoreduserende tiltak (handlingsplan)	Risikoverdi med nye tiltak				Ansvarlig Frist	Status
					Menneske	Ytre miljø	Omdømme	Økonomi / materiell		Menneske	Ytre miljø	Omdømme	Økonomi / materiell		
13	Lavere viskositet/Less viscosity	gjelder ikke	gjelder ikke	gjelder ikke	v	v	v	v							
14	Endring i sammensetning/Composition Change	Oljenedbryting pga. for høy oljetemperatur	Kompressorhavari pga. friksjon	Termisk bryter (termistor montert på elektrisk motoren til kompressoren). Sjekkliste som inneholder info om nødvendighet av isvannsirkulasjon og glykolsirkulasjon.	A2	A2	A2	A2	Glykol-temperatur alarm. Vann også.						
15	Forurensning/Contamination	Fuktighet i CO2-kretsen	dannelse av iskrystaller i CO2-kretsen	Filter-avfukter montert i CO2-kretsen; servicing må gjennomføres i jevne mellomrom ifølge manuale fra produsenten). Rustfri stål	A2	B2	A2	B2							
16	Lekkasje/ leakage	CO2-lekkasje Glykol-lekkasje , vann-lekasje	Tap av CO2 sammen med olje Tap av glykol, spyll av vann	Begrenset antall skrutilkoblinger. Hyppige testprøver og inspeksjoner av rør.	A3	A3	A3	A3							
17	Instrumentering/Instrumentation	feilt signal; strømbrudd	Feilregulering; tap av registrert data;	Hyppige dataanalyser av måleutstyret for datakvalitet. Temiske-beskyttelser electric-heaters	A3	A3	A3	A3	Validering at sensorene fortsatt måler godt nok. To ganger i året.	A2	A2	A2	A2		
18	Prøvetaking/Sampling	gjelder ikke	gjelder ikke	gjelder ikke	v	v	v	v							
19	Korrosjon/Corrosion	korrosjon av rørmateriale	lekkasje/tap av glykol/CO2	rørmaterialer som ikke korroderer med medier brukt (rustfritt stål, kobber, plasmaterialer). Glykol ble byttet til en med stoffer for å unngå korrosjon.	B2	B2	B2	B2							
20	Erosjon/erosion	gjelder ikke	gjelder ikke	Filter på CO2 siden.	v	v	v	v							
21	Driftsavvik/Abnormal operation	gjelder ikke	gjelder ikke	gjelder ikke	v	v	v	v							
22	Vedlikehold/Maintenance	Vedlikehold ikke gjennomført	riggen klarer ikke å oppnå normale driftsforhold	Hyppige kontroller og vedlikehold aktiviteter	A1	A1	A1	A1	To ganger i året (automatisk påminnelse på e-post). Inkludere sensor-validering. Sjekke dokumentasjonen av leverandøren.						
23	Antennelse/Ignition	CO2 er ikke brannbart; glykol tennes kun i høgere konsentrasjoner ved bruk av flamme	gjelder ikke	gjelder ikke	v	v	v	v							
24	Reservedeler/Spare equipment	Havari av komponenter	rigg i ustand	God kommunikasjon med riggleverandøren.	A3	A3	A3	A3							
25	Svikt i kritisk infrastruktur (strøm, vann, kjøling, ventilasjon)/ Failure of critical infrastructure	Ved svikt i isvanntilførsel når riggen er i drift stiger høytrykk (se 'høyere trykk')	Trykkoppbygging i kretsen helt til sikkerhetsventil utløses. Dette kan skje bare når riggen ikke er i bruk eller ved misbruk av riggen	Se 'høyere trykk'.	B2	B2	B2	B2							
26	Sikkerhet/Safety	diverse ulykker	helseskader	ryddig arbeidsplass	B1	B1	B1	B1							

RISIKOBILDE FOR: #REF!

Dato: 2021-10-05



D Python codes

D.1 Main

```
"""
Created on Wed October 13 2021

@author: Angel Alvarez Pardinas

Edited by Johan Hafsaas and Mihir Hazarika
"""

import numpy as np
import pandas as pd

import accuracyCalculations as aC
import systemPerformance as sP

import os
import matplotlib.pyplot as plt

import CoolProp
from CoolProp.CoolProp import PropsSI

from nptdms import TdmsFile


# ============================================================================
# Functions
# ============================================================================


def importLabVIEW(filename):
    #Function that imports a LabVIEW file (tdms) and formats it.

    tdms_file = TdmsFile.read(filename)
    group = tdms_file["Data"]
    selected_data = tdms_file['Data'].channels()[:]
    df = pd.DataFrame(selected_data[0].data, columns=[selected_data[0].name])
    for j in range(1,len(group)):
        df[j]=pd.DataFrame(selected_data[j].data)
        df=df.rename(columns={j:selected_data[j].name})

    # df['Time'] = df['Time'].apply(lambda x: pd.to_datetime(x.tz_localize('UTC').tz_convert('CET'), errors='coerce', format='%Y-%m-%d %H:%M:%S'))
    df['Time'] = df['Time'].apply(lambda x: pd.to_datetime(x.tz_localize('UTC').tz_convert('CET'), errors='coerce'))
    df['Time'] = pd.to_datetime(df.Time).dt.tz_localize(None)

    return df # df = data frame

def importMinilog(filename):
    #Function that imports a Minilog file and formats it.
    # df =  pd.read_csv(filename, sep = ';', decimal= '.', dtype = str)
    df =  pd.read_csv(filename, sep = ';', decimal= ',')
    df['Time'] = pd.to_datetime(df['Time'], errors='coerce')
    # df['Time'] = df['Time'].apply(lambda x: pd.to_datetime(x.tz_localize('UTC').tz_convert('CET'), errors='coerce'))

    return df

def tableStats(df):
```

```
    #Creates dataframes with statistical values
    all_columns = df.columns[1:].to_numpy(dtype = str) #all column names except 'Time'
    df = df[all_columns].astype('float64') #all columns except 'Time' is now converted to floats (objects -> string -> float)
    print('All columns: \n',all_columns)

    #Dataframe with mean, standard deviation, maximum and minimum
    df_stats = pd.DataFrame([df.mean(numeric_only=True), df.std(numeric_only=True), df.max(numeric_only=True), df.min(numeric_only=True)],index=['Mean','Std', 'Max', 'Min'], columns=all_columns)
    df_stats = aC.accuracySensors(df_stats)
    return df_stats

def writeExcel(df, df_stats, filename):
    #Converts all interesting information from one measurement to a excel file.
    #df = merged df, df1 = labVIEW, df2 = minilog, df_stats, df_corrections, df_corrected from function tableStats.
    with pd.ExcelWriter(filename) as writer:
        df.to_excel(writer, sheet_name='Synchronized data')
        # df1.to_excel(writer, sheet_name='LabVIEW data')
        # df2.to_excel(writer, sheet_name='Minilog data')
        df_stats.to_excel(writer, sheet_name='Statistical data')
        # df_corrections.to_excel(writer, sheet_name='Corrections')
        # df_corrected.to_excel(writer, sheet_name='Corrected statistical data')
    return

def createExcel(filename,df_sP,df_sP_u):
    #Creates a new file for systemPerformance values.
    with pd.ExcelWriter(filename) as writer:
        df_sP.to_excel(writer, sheet_name='main', index=False)
        df_sP_u.to_excel(writer, sheet_name='uncertainty', index = False)
    return

def appendToExcel(filename,df_sP,df_sP_u):
    #Appends systemPerformance to an existing excel file.
    df_old = pd.read_excel(filename,sheet_name='main')
    df_old_u = pd.read_excel(filename,sheet_name='uncertainty')

    df_new = df_old.append(df_sP,ignore_index=True)
    df_new_u = df_old_u.append(df_sP_u,ignore_index=True)

    with pd.ExcelWriter(filename) as writer:
        df_new.to_excel(writer, sheet_name='main', index = False)
        df_new_u.to_excel(writer, sheet_name='uncertainty', index = False)
    return


measurements_gravity = {'filename_LabVIEW': ['Log_01_24.03.2022_08.33.44.tdms','Log_01_24.03.2022_08.33.44.tdms','Log_01_24.03.2022_08.33.44.tdms','Log_01_24.03.2022_08.33.44.tdms',
                                              'Log_01_24.03.2022_08.33.44.tdms','Log_01_25.03.2022_08.39.58.tdms','Log_01_25.03.2022_08.39.58.tdms','Log_01_25.03.2022_08.39.58.tdms',
                                              'Log_01_25.03.2022_08.39.58.tdms','Log_01_28.03.2022_11.05.05.tdms','Log_01_28.03.2022_11.05.05.tdms','Log_01_28.03.2022_11.05.05.tdms',
                                              'Log_01_29.03.2022_08.12.36.tdms','Log_01_29.03.2022_08.12.36.tdms','Log_01_29.03.2022_08.12.36.tdms',
                                              'Log_01_29.03.2022_08.12.36.tdms'
```

```
    ,'Log_01_29.03.2022_08.12.36.tdms','Log_01_29.03.2022_08.12.36.tdms','
    Log_01_30.03.2022_08.25.36.tdms',
104                                         'Log_01_30.03.2022_08.25.36.tdms'
    ,'Log_01_30.03.2022_08.25.36.tdms','Log_01_30.03.2022_08.25.36.tdms','
    Log_01_30.03.2022_08.25.36.tdms',
105                                         'Log_01_30.03.2022_08.25.36.tdms'
    ,'Log_01_31.03.2022_08.29.07.tdms','Log_01_31.03.2022_08.29.07.tdms','
    Log_01_31.03.2022_08.29.07.tdms'],
106                 'filename_MiniLog': ['202109
    _MultiTest_HeatJet_20220324_001.csv','202109
    _MultiTest_HeatJet_20220324_001.csv','202109
    _MultiTest_HeatJet_20220324_001.csv',
107                                       '202109
    _MultiTest_HeatJet_20220324_001.csv','202109
    _MultiTest_HeatJet_20220324_001.csv','202109
    _MultiTest_HeatJet_20220325_001.csv',
108                                       '202109
    _MultiTest_HeatJet_20220325_001.csv','202109
    _MultiTest_HeatJet_20220325_001.csv','202109
    _MultiTest_HeatJet_20220325_001.csv',
109                                       '202109
    _MultiTest_HeatJet_20220325_001.csv','202109
    _MultiTest_HeatJet_20220328_002.csv','202109
    _MultiTest_HeatJet_20220328_002.csv',
110                                       '202109
    _MultiTest_HeatJet_20220328_002.csv','202109
    _MultiTest_HeatJet_20220329_001.csv','202109
    _MultiTest_HeatJet_20220329_001.csv',
111                                       '202109
    _MultiTest_HeatJet_20220329_001.csv','202109
    _MultiTest_HeatJet_20220329_001.csv','202109
    _MultiTest_HeatJet_20220329_001.csv',
112                                       '202109
    _MultiTest_HeatJet_20220329_001.csv','202109
    _MultiTest_HeatJet_20220330_001.csv','202109
    _MultiTest_HeatJet_20220330_001.csv',
113                                       '202109
    _MultiTest_HeatJet_20220330_001.csv','202109
    _MultiTest_HeatJet_20220330_001.csv','202109
    _MultiTest_HeatJet_20220330_001.csv',
114                                       '202109
    _MultiTest_HeatJet_20220330_001.csv','202109
    _MultiTest_HeatJet_20220331_001.csv','202109
    _MultiTest_HeatJet_20220331_001.csv','202109
    _MultiTest_HeatJet_20220331_001.csv'],
115                 'start': ['2022-03-24 11:27:00','2022-03-24 10:47:00','
    2022-03-24 13:25:00','2022-03-24 15:21:00','2022-03-24 15:54:00','
    2022-03-25 14:56:19','2022-03-25 14:14:00',
116                           '2022-03-25 10:44:00','2022-03-25 11:20:00','
    2022-03-25 13:15:00','2022-03-28 12:51:00','2022-03-28 13:51:00','
    2022-03-28 15:01:00','2022-03-29 09:55:00',
117                           '2022-03-29 10:42:00','2022-03-29 11:40:00','
    2022-03-29 12:45:00','2022-03-29 14:12:00','2022-03-29 14:52:00','
    2022-03-30 11:41:00','2022-03-30 12:10:00',
118                           '2022-03-30 13:50:00','2022-03-30 14:18:00','
    2022-03-30 15:13:00','2022-03-30 16:18:00','2022-03-31 11:08:00','
    2022-03-31 12:17:00','2022-03-31 13:10:27',],
119                 'end': ['2022-03-24 11:42:00','2022-03-24 11:02:00','
    2022-03-24 13:45:00','2022-03-24 15:36:00','2022-03-24 16:09:00','
    2022-03-25 15:22:38','2022-03-25 14:29:00',
120                           '2022-03-25 10:59:00','2022-03-25 11:35:00','
    2022-03-25 13:30:00','2022-03-28 13:06:00','2022-03-28 14:06:00','
    2022-03-28 15:16:00','2022-03-29 10:10:00',
```

```
121                             '2022-03-29 10:57:00','2022-03-29 11:55:00','
    2022-03-29 13:00:00','2022-03-29 14:27:00','2022-03-29 15:07:00','
    2022-03-30 11:57:00','2022-03-30 12:25:00',
122                             '2022-03-30 14:05:00','2022-03-30 14:34:00','
    2022-03-30 15:28:00','2022-03-30 16:33:00','2022-03-31 11:23:00','
    2022-03-31 12:32:00','2022-03-31 13:17:02'],
123                     'Mode': 'Gravity',
124                     'T_gw,in [ C ]': ['15','15','15','12','10','20','10','10'
    ,'10','15','20','15','10','15','12','12','12','12','10','20','15','12','10
    ','20','15','20','15','12'],
125                     'p_evap [bar]': ['38','38','38','38','38','35','35','38',
    '38','38','41','41','41','35','35','38','41','41','41','38','38','38','38'
    ,'41','41','44','44','44'],
126                     'flow_water [kg/min]': ['36','24','12','12','12','24','24
    ','24','18','18','24','24','24','24','24','24','24','16','16','16','16','
    16','16','16','16','16','16','16']}
127
128 measurements_combined = {'filename_LabVIEW': ['Log_01_07.04.2022_08.26.23.
    tdms','Log_01_07.04.2022_08.26.23.tdms','Log_01_07.04.2022_08.26.23.tdms',
    'Log_01_06.04.2022_14.52.32.tdms',
129                                                 'Log_01_07.04.2022_08.26.23.
    tdms','Log_01_07.04.2022_08.26.23.tdms','Log_01_05.04.2022_09.41.40.tdms',
    'Log_01_06.04.2022_14.52.32.tdms',
130                                                 'Log_01_06.04.2022_08.53.24.
    tdms','Log_01_08.04.2022_09.22.47.tdms','Log_01_08.04.2022_09.22.47.tdms',
    'Log_01_08.04.2022_09.22.47.tdms',
131                                                 'Log_01_08.04.2022_09.22.47.
    tdms','Log_01_08.04.2022_09.22.47.tdms','Log_01_12.04.2022_08.39.28.tdms',
    'Log_01_08.04.2022_09.22.47.tdms',
132                                                 'Log_01_11.04.2022_08.22.10.
    tdms','Log_01_11.04.2022_08.22.10.tdms','Log_01_11.04.2022_08.22.10.tdms',
    'Log_01_11.04.2022_08.22.10.tdms',
133                                                 'Log_01_11.04.2022_08.22.10.
    tdms','Log_01_11.04.2022_08.22.10.tdms'],
134                     'filename_MiniLog': ['202109
    _MultiTest_HeatJet_20220407_001.Csv','202109
    _MultiTest_HeatJet_20220407_001.Csv','202109
    _MultiTest_HeatJet_20220407_001.Csv',
135                                                 '202109
    _MultiTest_HeatJet_20220406_003.Csv','202109
    _MultiTest_HeatJet_20220407_001.Csv','202109
    _MultiTest_HeatJet_20220407_001.Csv',
136                                                 '202109
    _MultiTest_HeatJet_20220405_001.Csv','202109
    _MultiTest_HeatJet_20220406_003.Csv','202109
    _MultiTest_HeatJet_20220406_002.Csv',
137                                                 '202109
    _MultiTest_HeatJet_20220408_001.Csv','202109
    _MultiTest_HeatJet_20220408_001.Csv','202109
    _MultiTest_HeatJet_20220408_001.Csv',
138                                                 '202109
    _MultiTest_HeatJet_20220408_001.Csv','202109
    _MultiTest_HeatJet_20220408_001.Csv','202109
    _MultiTest_HeatJet_20220412_001.Csv',
139                                                 '202109
    _MultiTest_HeatJet_20220408_001.Csv','202109
    _MultiTest_HeatJet_20220411_001.Csv','202109
    _MultiTest_HeatJet_20220411_001.Csv',
140                                                 '202109
    _MultiTest_HeatJet_20220411_001.Csv','202109
    _MultiTest_HeatJet_20220411_001.Csv','202109
    _MultiTest_HeatJet_20220411_001.Csv',
141                                                 '202109
```

```
    _MultiTest_HeatJet_20220411_001.Csv'],
                    'start': ['2022-04-07 13:43:00','2022-04-07 14:14:00','
    2022-04-07 11:48:00','2022-04-06 16:04:00','2022-04-07 9:55:00','
    2022-04-07 10:22:00','2022-04-05 15:52:00',
                              '2022-04-06 15:32:00','2022-04-06 11:15:00','
    2022-04-08 11:25:00','2022-04-08 10:45:00','2022-04-08 12:10:00','
    2022-04-08 13:28:00','2022-04-08 13:01:00',
                              '2022-04-12 14:13:00','2022-04-08 15:25:00','
    2022-04-11 11:38:00','2022-04-11 10:45:00','2022-04-11 12:36:00','
    2022-04-11 14:25:00','2022-04-11 13:31:00',
                              '2022-04-11 15:50:00'],
                    'end': ['2022-04-07 13:58:00','2022-04-07 14:29:00','
    2022-04-07 12:03:00','2022-04-06 16:25:00','2022-04-07 10:10:00','
    2022-04-07 10:41:00','2022-04-05 16:10:00',
                            '2022-04-06 15:47:00','2022-04-06 11:30:00','
    2022-04-08 11:45:00','2022-04-08 11:05:00','2022-04-08 12:30:00','
    2022-04-08 13:44:00','2022-04-08 13:21:00',
                            '2022-04-12 14:33:00','2022-04-08 15:40:00','
    2022-04-11 11:53:00','2022-04-11 11:00:00','2022-04-11 12:51:00','
    2022-04-11 14:45:00','2022-04-11 13:51:00',
                            '2022-04-11 16:10:00'],
                    'Mode': 'Combined',
                    'T_gw,in [ C ]': ['20','17.7','15','20','17.7','15','12',
    '20','15','20','20','20','15','15','12','12','17.5','15','12','20','17.5',
    '15'],
                    'p_evap [bar]': ['35','35','35','38','38','38','38','41',
    '41','38','38','38','38','38','38','38','38','38','38','41','41','41'],
                    'flow_water [kg/min]': ['16','16','16','16','16','16','16
    ','16','16','12','18','24','12','18','12','18','24','24','24','24','24','
    24']}


# ============================================================================
# Which measurement or measurements the for-loop runs
# ============================================================================
measurements = measurements_gravity
filename_exportExcel = 'SystemPerformance_gravity.xlsx'
write_excel = True
new_excel = True
SM = False

# ============================================================================
# For-loop analysing the measurements
# ============================================================================

for i in range(len(measurements['start'])):

    ############################################################################
    #
    #
    # Import .csv-files, merge and interpolate with nearest value.
    #
    #
    ############################################################################

    filename1 = measurements['filename_LabVIEW'][i]
    filename2 = measurements['filename_MiniLog'][i]

    os.chdir(r'C:\Users\johan\Desktop\Experimental_Python')

    df1 = importLabVIEW(filename1)
    df2 = importMinilog(filename2)

```

```
187     #Only use for the first files which did not have all the channels up to
        date
188     df1 = df1.rename(columns={'AI11': 'RHE_MFn', 'AI12': 'PT4', 'AI13': 'DPT3
        ', 'AI14': 'DPT4', 'RTD02':'TT4', 'RTD03':'TT14'})
189     df1 = df1.rename(columns={'RTD02':'TT4', 'RTD03':'TT14'})
190     # #Merge the two dataframes with respect to 'Time' column and fill NaN
        values with nearest value.
191     # #Should be similar to MATLAB's synchronize(table1, table2, 'union', '
        nearest') function.
192     # df = pd.merge_asof(df1, df2, on="Time", tolerance=pd.Timedelta("1ms"))
193     df = pd.merge_asof(df1, df2, on="Time", direction='nearest')
194
195     #Converts datatypes in dataframe to a prefered datatype. Convert_integer=
        False means it does not convert anything to integers.
196     #Used mainly to convert any objects to 'string' (and later to float).  (
        objects -> string -> float)
197     df = df.convert_dtypes(convert_integer=False)
198
199
200 #       ########################################################################
201 #       #
202 #       #
203 #       # Load calculations
204 #       #
205 #       #
206 #       ########################################################################
207
208     #Choose measurement part to look at:
209     start = pd.to_datetime(measurements['start'][i], errors='coerce')
210     end = pd.to_datetime(measurements['end'][i], errors='coerce')
211     condition = (df['Time'] > start) & (df['Time'] <= end)
212     measurement = df.loc[condition]
213
214     #Include uncertainties in the DataFrame
215     df_stats = tableStats(measurement)
216
217     measurement_info = {'Date': str(start.date()),
218             'Time_start': str(start.time()),
219             'Time_end': str(end.time()),
220             'Mode': measurements['Mode'],
221             'T_gw,in [ C ]': str(measurements['T_gw,in [ C ]'][i]),
222             'p_evap [bar]': str(measurements['p_evap [bar]'][i]),
223             'flow_water [kg/min]': str(measurements['flow_water [kg/min]'][i
        ])}
224
225
226     df_systemPerformance = pd.DataFrame(measurement_info,index=[0])
227     df_uncertainty_systemPerformance = pd.DataFrame(measurement_info,index
        =[0])
228
229     #Most important data
230     df_systemPerformance, df_uncertainty_systemPerformance = sP.getData(
        df_stats, df_systemPerformance, df_uncertainty_systemPerformance)
231
232     #Q evaporators Water Side / CO2 side / Velocity pipe / Superheat plate HX
         / Ejector calculations
233     df_systemPerformance, df_uncertainty_systemPerformance = sP.getLoadsWater
        (df_stats, df_systemPerformance, df_uncertainty_systemPerformance)
234     df_systemPerformance, df_uncertainty_systemPerformance = sP.
        getLoadCO2_ejector(df_stats, df_systemPerformance,
        df_uncertainty_systemPerformance)
235     df_systemPerformance, df_uncertainty_systemPerformance = sP.
        getLoadCO2_gravity(df_stats, df_systemPerformance,
```

```
    df_uncertainty_systemPerformance)
236     df_systemPerformance, df_uncertainty_systemPerformance = sP.getSuperheat(
    df_stats, df_systemPerformance, df_uncertainty_systemPerformance)
237     df_systemPerformance, df_uncertainty_systemPerformance = sP.getLMTD(
    df_stats, df_systemPerformance, df_uncertainty_systemPerformance)
238     df_systemPerformance, df_uncertainty_systemPerformance = sP.getUA(
    df_systemPerformance, df_uncertainty_systemPerformance)
239     df_systemPerformance, df_uncertainty_systemPerformance = sP.ejector_calc(
    df_stats, df_systemPerformance, df_uncertainty_systemPerformance)
240     df_systemPerformance, df_uncertainty_systemPerformance = sP.
    getVapourFraction_gravity(df_stats, df_systemPerformance,
    df_uncertainty_systemPerformance)
241     df_systemPerformance = sP.getDPT(df_stats, df_systemPerformance)
242 
243     ##########################################################################
244     #
245     #
246     # Export data to excel (if True).
247     #
248     #
249     ##########################################################################
250 
251     filename_export = r'{}_{}_{}_{}_{}C_{}_{}.xlsx'.format(measurement_info['
    Date'],measurement_info['Time_start'].replace(':','-'),measurement_info['
    Time_end'].replace(':','-'), measurement_info['Mode'], measurement_info['
    T_gw,in [ C ]'], measurement_info['p_evap [bar]'], measurement_info['
    flow_water [kg/min]'])
252 
253     if write_excel == True:
254         writeExcel(measurement, df_stats, filename_export)
255 
256         if new_excel == True and i == 0:
257             createExcel(filename_exportExcel,df_systemPerformance,
    df_uncertainty_systemPerformance)
258         else:
259             appendToExcel(filename_exportExcel,df_systemPerformance,
    df_uncertainty_systemPerformance)
```

D.2 accuracyCalculations

```
"""
Created on Wed October 13 2021

@author: Angel Alvarez Pardinas

Edited by Johan Hafsaas
"""

import numpy as np
import pandas as pd

def eliminatingMissing(df_corrected, variables, variables2 = []):
    #Find logical array with the values that are in the array
    notMissing = np.in1d(variables, df_corrected.columns.to_numpy())
    variables = variables[notMissing]
    # print(variables)
    if len(variables2) != 0:
        variables2 = variables2[notMissing]
        # print(variables2)
    return variables, variables2

def diffPressEH_PMD75(df_corrected, set_span, variables, rect):
    #Differential pressure Deltabar PMD75
    MME = 0.035  # Max. measurement error 0,035% of set span
    for i in range(len(variables)):
        typeA = df_corrected.loc['Std'].at[variables[i]]
        typeB = MME/100*set_span[i]/rect
        df_corrected.at['Uncertainty',variables[i]] = np.sqrt(typeA**2 +
    typeB**2)
    return df_corrected

def pressPMP71(df_corrected, set_span, variables, rect):
    #Endress+Hauser Absolute and gauge pressure Cerabar PMP71
    Accuracy = 0.075 #Percentage of the set_span for the standard version
    for i in range(len(variables)):
        typeA = df_corrected.loc['Std'].at[variables[i]]
        typeB = Accuracy/100*set_span[i]/rect
        df_corrected.at['Uncertainty',variables[i]] = np.sqrt(typeA**2 +
    typeB**2)
    return df_corrected

def massFlowRheonik08(df_corrected, variables, rect):
    #Mass flow meters Rheonik
    RHM_mass = 0.2 #Percentage of the reading in the normal operation range
    #We neglect the different uncertainty for very low values
    for i in range(len(variables)):
        typeA = df_corrected.loc['Std'].at[variables[i]]
        typeB = df_corrected.loc['Mean'].at[variables[i]]*RHM_mass/100/rect
        df_corrected.at['Uncertainty',variables[i]] = np.sqrt(typeA**2 +
    typeB**2)

    return df_corrected

def massFlowRheonik06(df_corrected, variables, rect):
    #Mass flow meters Rheonik
    RHM_mass = 0.5 #Percentage of the reading in the normal operation range
    #We neglect the different uncertainty for very low values
    for i in range(len(variables)):
        typeA = df_corrected.loc['Std'].at[variables[i]]
        typeB = df_corrected.loc['Mean'].at[variables[i]]*RHM_mass/100/rect
        df_corrected.at['Uncertainty',variables[i]] = np.sqrt(typeA**2 +
```

```
    typeB**2)

    return df_corrected

def tempRTD(df_corrected, variables, rect):
    #PT100 and PT1000 Class A = PT_factor*(a+b*TdegC)
    PT_factor = 1 #Due to class
    PT_a = 0.15
    PT_b = 0.002
    #We are conservative and not reduce the uncertainty due to correction
    for i in range(len(variables)):
        typeA = df_corrected.loc['Std'].at[variables[i]]
        typeB = PT_factor*(PT_a+PT_b*df_corrected.loc['Mean'].at[variables[i]])/rect
        df_corrected.at['Uncertainty',variables[i]] = np.sqrt(typeA**2 + typeB**2)
    return df_corrected

def accuracySensors(df_corrected):
    all_columns = df_corrected.columns
    #Add a new row to df_corrected with the uncertainty
    df_uncertainty = pd.DataFrame([np.zeros(len(all_columns))], index = ['Uncertainty'], columns = all_columns)
    df_corrected = df_corrected.append(df_uncertainty)

    #Rectangulat approximation of measurements. Uncertainty of sensor is 'error' divided by factor sqrt(3)
    rect = np.sqrt(3)

    #PMD75
    variables = np.array(['DPT1', 'DPT2', 'DPT3', 'DPT4'])
    set_span = np.array([10, 5, 5, 5]) #bar
    df_corrected = diffPressEH_PMD75(df_corrected, set_span, variables, rect)

    #Pressure sensors Endress+Hauser PMP21
    variables = np.array(['PT2', 'PT1', 'PT3', 'PT4'])
    set_span = np.array([50,70,60,60]) #bar
    df_corrected = pressPMP71(df_corrected, set_span, variables, rect)

    #Mass flow meters Rheonik RHL08
    variables = np.array(['RHE_26472_MFoil', 'RHE_24722_MF1', 'RHE_24721_MF2', 'RHE_33612_MFmot', 'RHE_33708_MFG', 'RHE_MFn'])
    variables, param = eliminatingMissing(df_corrected, variables)
    df_corrected = massFlowRheonik08(df_corrected, variables, rect)

    #Mass flow meters Rheonik RHL06
    variables = np.array(['RHE_09109_MFsuct'])
    variables, param = eliminatingMissing(df_corrected, variables)
    df_corrected = massFlowRheonik06(df_corrected, variables, rect)

    #Temperature sensors RTD
    variables = np.array(['TT0', 'TT3', 'TT1', 'TT10', 'TT11', 'TT12', 'TT13', 'TT22', 'TT21', 'TT4', 'TT14'])
    variables, param = eliminatingMissing(df_corrected, variables)
    df_corrected = tempRTD(df_corrected, variables, rect)


    return df_corrected
```

D.3 systemPerformance

```
"""
Created on Wed October 13 2021

@author: Angel Alvarez Pardinas

Edited by Johan Hafsaas and Mihir Hazarika
"""

import numpy as np
from CoolProp.CoolProp import PropsSI
import CoolProp.CoolProp as CP

# CP.set_config_string(CP.ALTERNATIVE_REFPROP_LIBRARY_PATH, r'C:\Programfiler (x86)\REFPROP\REFPRP64.dll')
# print(CP.get_global_param_string("REFPROP_version"))

fluid1 = 'CarbonDioxide'
fluid2 = 'Water'

#Example
#PropsSI('T','P',1e5*35,'Q',1,fluid1)

def tempDiff(temp1, u_temp1, temp2, u_temp2):
    #Difference of max two temperatures
    temp_diff = (temp1 - temp2)
    SC_temp = 1 #Sensitivity coefficient of temp1 and temp2
    u_temp_diff = np.sqrt(np.power(SC_temp*u_temp1,2) + np.power(SC_temp*u_temp2,2))
    return temp_diff, u_temp_diff

def addMassFlow(massflow1, u_massflow1, massflow2, u_massflow2):
    #Addition of two mass flowrates for water
    massflow = massflow1 + massflow2
    SC_massflow = 1
    u_massflow = np.sqrt(np.power(SC_massflow*u_massflow1,2) + np.power(SC_massflow*u_massflow2,2))
    return massflow, u_massflow

def tempAv(temp1, u_temp1, temp2, u_temp2):
    #Average of max two temperatures
    temp_av = (temp1 + temp2)/2
    SC_temp = 0.5 #Sensitivity coefficient of temp1 and temp2
    u_temp_av = np.sqrt(np.power(SC_temp*u_temp1,2) + np.power(SC_temp*u_temp2,2))
    return temp_av, u_temp_av

def densWater(temp, u_temp):
    #Properties water from CoolProp
    temp = temp + 273.15#To Kelvin
    dens_water = PropsSI('D','P',1e5,'T',temp,fluid2)
    #Uncertainty density due to temperature measurement type B
    u_dens_water_temp = PropsSI('T','P',1e5,'T',temp + u_temp,fluid2) - PropsSI('T','P',1e5,'T',temp,fluid2)
    SC_temp = 1 #Sensitivity coefficient of the temperature on the combined uncertainty
    #Percentage due to uncertainty of the properties of water from CoolProp.
    #Assumed same as REFPROP, i.e. 0.00001% at 1atm, 0.001% at other pressures
    correlation_diff = 0.001
    u_dens_water_corr = correlation_diff/100*dens_water #Uncertainty density due to correlation type B
```

```
    SC_corr = 1 #Sensitivity coefficienct is assumed to be 1
    #Uncertainty type B
    u_dens_water = np.sqrt(np.power(SC_temp*u_dens_water_temp,2) + np.power(SC_corr*u_dens_water_corr,2))
    return dens_water, u_dens_water

def cpWater(temp, u_temp):
    #Properties water from CoolProp
    temp = temp + 273.15 #To Kelvin
    cp_water = PropsSI('C','P',1e5,'T', temp, fluid2) #Actual cp in J/kgK;
    #Uncertainty cp due to temperature
    u_cp_water_temp = PropsSI('C','P',1e5,'T', temp + u_temp, fluid2) - PropsSI('C','P',1e5,'T', temp, fluid2) #Uncertainty cp due to temperature measurement
    SC_temp = 1 #Sensitivity coefficient of the temperature on the combined uncertainty
    #Percentage due to uncertainty of the properties of water from CoolProp
    #Assumed same as REFPROP, i.e. 0.1% in the liquid
    correlation_diff = 0.1 #Percentage for the uncertainty of the tabled data
    u_cp_water_corr = correlation_diff/100*cp_water #Uncertainty cp due to correlation
    SC_corr = 1 #Sensitivity coefficienct is assumed to be 1
    #Uncertainty type B
    u_cp_water = np.sqrt(np.power(SC_temp*u_cp_water_temp,2) + np.power(SC_corr*u_cp_water_corr,2))
    return cp_water, u_cp_water

#Calculate loads and uncertainties water side
def loadWaterSide(temp_in, u_temp_in, temp_out, u_temp_out, massflow, u_massflow):
    #Function for the load in the water side, any HX.
    #Assume properties at average temperature.
    #Call function of average temperature
    temp_av,u_temp_av = tempAv(temp_in,u_temp_in,temp_out,u_temp_out)
    #Call function of temperature difference
    temp_diff,u_temp_diff = tempDiff(temp_in,u_temp_in,temp_out,u_temp_out)
    #Call functions for the properties needed.
    cp_Water,u_cp_Water = cpWater(temp_av,u_temp_av) #cp is calculated with the average temperature, J/kgK
    #Calculate the load at the water side
    load_Water = temp_diff * cp_Water * massflow #mass flow originally in kg/s

    #Calculate the uncertainty of the load at the glycol side
    SC_temp_diff = cp_Water*massflow #Sensitivity coefficient of the temperature difference on the combined uncertainty
    SC_cp_Water = temp_diff*massflow #Sensitivity coefficient of the cp on the combined uncertainty
    SC_massflow = temp_diff*cp_Water #Sensitivity coefficient of the volflow on the combined uncertainty

    u_load_Water = np.sqrt(np.power(SC_temp_diff*u_temp_diff,2) + np.power(SC_cp_Water*u_cp_Water,2) + np.power(SC_massflow*u_massflow,2))

    return load_Water, u_load_Water


def getLoadsWater(df_stats, df_sP, df_sP_u):

    # Plate HX. 3 possibilities, only gravity loop, only ejector loop, gravity loop and ejector loop in series
    temp_in = df_stats.loc['Mean'].at['TT10']
    u_temp_in = df_stats.loc['Uncertainty'].at['TT10']
```

```
    temp_mid = df_stats.loc['Mean'].at['TT14']
    u_temp_mid = df_stats.loc['Uncertainty'].at['TT14']
    temp_out = df_stats.loc['Mean'].at['TT11']
    u_temp_out = df_stats.loc['Uncertainty'].at['TT11']

    # Mass flow for gravity loop and ejector loop in series
    massflow1 = df_stats.loc['Mean'].at['RHE_24722_MF1']/60 #kg/s
    u_massflow1 = df_stats.loc['Uncertainty'].at['RHE_24722_MF1']/60 #kg/s
    massflow2 = df_stats.loc['Mean'].at['RHE_24721_MF2']/60 #kg/s
    u_massflow2 = df_stats.loc['Uncertainty'].at['RHE_24721_MF2']/60 #kg/s
    massflow, u_massflow =  addMassFlow(massflow1, u_massflow1, massflow2,
    u_massflow2)

    load_Water_gravity, u_load_Water_gravity = loadWaterSide(temp_in,
    u_temp_in, temp_mid, u_temp_mid, massflow, u_massflow)
    df_sP.loc[0,'Q_Water_gravity [W]'] = load_Water_gravity
    df_sP_u.loc[0,'Q_Water_gravity [W]'] = u_load_Water_gravity

    load_Water_ejector, u_load_Water_ejector = loadWaterSide(temp_mid,
    u_temp_mid, temp_out, u_temp_out, massflow, u_massflow)
    df_sP.loc[0,'Q_Water_ejector [W]'] = load_Water_ejector
    df_sP_u.loc[0,'Q_Water_ejector [W]'] = u_load_Water_ejector

    load_Water_combined, u_load_Water_combined = loadWaterSide(temp_in,
    u_temp_in, temp_out, u_temp_out, massflow, u_massflow)
    df_sP.loc[0,'Q_Water_combined [W]'] = load_Water_combined
    df_sP_u.loc[0,'Q_Water_combined [W]'] = u_load_Water_combined

    return df_sP, df_sP_u


#Superheat PlateHX
def Superheat(temp, u_temp, pres, u_pres):
    #Properties CO2 from CoolProp
    temp = temp + 273.15 #To Kelvin
    #Calculate saturation temperature and uncertainty for the saturation
    temperature
    sat_temp = PropsSI('T','P',1e5*pres,'Q',1,fluid1) #In Kelvin;
    SC_sat_temp_CoolProp = 1
    sat_temp_CoolProp = 0.2 #In percentage, from REFPROP
    u_sat_temp_CoolProp = sat_temp_CoolProp/100*sat_temp
    SC_sat_temp_upres = 1
    u_sat_temp_upres = PropsSI('T','P',1e5*(pres + u_pres),'Q',1,fluid1) -
    PropsSI('T','P',1e5*pres,'Q',1,fluid1)
    u_sat_temp = np.sqrt(np.power(SC_sat_temp_CoolProp*u_sat_temp_CoolProp,2)
     + np.power(SC_sat_temp_upres*u_sat_temp_upres,2))
    #Calculate superheat and uncertainty of superheat
    superheat = temp - sat_temp
    SC_temp = 1
    u_superheat = np.sqrt(np.power(SC_temp*u_temp,2) + np.power(SC_temp*
    u_sat_temp,2))
    return superheat, u_superheat


def getSuperheat(df_stats, df_sP, df_sP_u):
    # Superheat for plate HX in ejector
    temp_out = df_stats.loc['Mean'].at['TT4']
    u_temp_out = df_stats.loc['Uncertainty'].at['TT4']
    pres_vap = df_stats.loc['Mean'].at['PT4']
    u_pres_vap = df_stats.loc['Uncertainty'].at['PT4']
    superheat_ejector, u_superheat_ejector = Superheat(temp_out, u_temp_out,
    pres_vap, u_pres_vap)
    df_sP.loc[0,'Superheat_ejector [K]'] = superheat_ejector
```

```
157     df_sP_u.loc[0,'Superheat_ejector [K]'] = u_superheat_ejector
158 
159     # Superheat for plate HX in gravity loop
160     temp_out_gravity = df_stats.loc['Mean'].at['TT3']
161     u_temp_out_gravity = df_stats.loc['Uncertainty'].at['TT3']
162     pres_vap_gravity = df_stats.loc['Mean'].at['PT3']
163     u_pres_vap_gravity = df_stats.loc['Uncertainty'].at['PT3']
164     superheat_gravity, u_superheat_gravity = Superheat(temp_out_gravity,
        u_temp_out_gravity, pres_vap_gravity, u_pres_vap_gravity)
165     df_sP.loc[0,'Superheat_gravity [K]'] = superheat_gravity
166     df_sP_u.loc[0,'Superheat_gravity [K]'] = u_superheat_gravity
167     return df_sP, df_sP_u
168 
169 # Pressure drop in riser and downcomer
170 def getDPT(df_stats, df_sP): # Missing uncertainty calculations
171 
172     PT2 = df_stats.loc['Mean'].at['PT2']
173     PT3 = df_stats.loc['Mean'].at['PT3']
174     DPT3 = df_stats.loc['Mean'].at['DPT3']
175     # Downcomer gravity side
176     p_evap = PT3 + DPT3
177     DPT_downcomer = p_evap - PT2
178     df_sP.loc[0,'dp_Downcomer_gravity [bar]'] = DPT_downcomer
179     # Riser gravity side
180     DPT_riser = PT3 - PT2
181     df_sP.loc[0,'dp_Riser_gravity [bar]'] = DPT_riser
182 
183     return df_sP
184 
185 
186 
187 # Estimate cooling load through ejector on CO2 side when superheated
188 def getLoadCO2_ejector(df_stats, df_sP, df_sP_u):
189     PT4 = df_stats.loc['Mean'].at['PT4']
190     u_PT4 = df_stats.loc['Uncertainty'].at['PT4']
191     DPT4 = df_stats.loc['Mean'].at['DPT4']
192     u_DPT4 = df_stats.loc['Uncertainty'].at['DPT4']
193     p_evap = PT4 + DPT4
194     SC_PT4 = 1
195     SC_DPT4 = 1
196     u_p_evap = np.sqrt(np.power(SC_PT4*u_PT4,2) + np.power(SC_DPT4*u_DPT4,2))
197 
198     #Enthalpy from REFPROP:
199     #Inlet evaporator
200     h_in = PropsSI('H','P', p_evap*1e5,'Q', 0,fluid1)
201     SC_h_in_pres = 1
202     u_h_in_pres = PropsSI('H','P', (p_evap + u_p_evap)*1e5,'Q', 0,fluid1) -
        PropsSI('H','P', p_evap*1e5,'Q', 0,fluid1)
203     SC_h_in_CoolProp = 1
204     h_in_CoolProp = 1.5 #% from REFPROP (heat capacity liquid)
205     u_h_in_CoolProp = h_in_CoolProp/100*h_in
206     u_h_in = np.sqrt(np.power(SC_h_in_pres*u_h_in_pres,2) + np.power(
        SC_h_in_CoolProp*u_h_in_CoolProp,2))
207 
208     #Outlet evaporator
209     pres_out = df_stats.loc['Mean'].at['PT4']
210     u_pres_out = df_stats.loc['Uncertainty'].at['PT4']
211     temp_out = df_stats.loc['Mean'].at['TT4'] + 273.15 # In Kelvin
212     u_temp_out = df_stats.loc['Uncertainty'].at['TT4']
213     h_out = max(PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1),PropsSI('
        H','P', pres_out*1e5,'Q', 1,fluid1))
214     SC_h_out_pres = 1
215     u_h_out_pres = PropsSI('H','P', (pres_out+u_pres_out)*1e5,'T', temp_out,
```

```
    fluid1) - PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1)
    SC_h_out_temp = 1
    u_h_out_temp = PropsSI('H','P', pres_out*1e5,'T', temp_out + u_temp_out,
    fluid1) - PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1)
    SC_h_out_CoolProp = 1
    h_out_CoolProp = 0.15 #% from REFPROP (heat capacity vapour)
    u_h_out_CoolProp = h_out_CoolProp/100*h_out
    u_h_out = np.sqrt(np.power(SC_h_out_pres*u_h_out_pres,2) + np.power(
    SC_h_out_temp*u_h_out_temp,2) + np.power(SC_h_out_CoolProp*
    u_h_out_CoolProp,2))

    #Enthalpy difference
    delta_h = h_out - h_in
    SC_h = 1
    u_delta_h = np.sqrt(np.power(SC_h*u_h_out,2) + np.power(SC_h*u_h_in,2))

    #Mass flow meter
    massflow = df_stats.loc['Mean'].at['RHE_MFn']/60 #kg/s
    u_massflow = df_stats.loc['Uncertainty'].at['RHE_MFn']/60 #kg/s

    #Calculation load
    load_CO2 = massflow * delta_h
    SC_load_CO2_dh = massflow
    SC_load_CO2_massflow = delta_h
    u_load_CO2 = np.sqrt(np.power(SC_load_CO2_dh*u_delta_h,2) + np.power(
    SC_load_CO2_massflow*u_massflow,2))
    df_sP.loc[0,'Q_CO2_ejector [W]'] = load_CO2
    df_sP_u.loc[0,'Q_CO2_ejector [W]'] = u_load_CO2

    #Outlet vapor frac
    Q_Water_ejector = df_sP.loc[0,'Q_Water_ejector [W]']
    u_Q_Water_ejector = df_sP_u.loc[0,'Q_Water_ejector [W]']
    SC_h_real_hin = 1
    SC_h_real_Q = 1/massflow
    SC_h_real_mass = Q_Water_ejector/np.power(massflow,2)
    h_out_real = Q_Water_ejector/massflow + h_in
    u_h_out_real = np.sqrt(np.power(SC_h_real_Q*u_Q_Water_ejector,2) + np.
    power(SC_h_real_mass*u_massflow,2) + np.power(SC_h_real_hin*u_h_in,2))
    vapour_frac_out = PropsSI('Q', 'P', p_evap*1e5, 'H', h_out_real, fluid1)
    u_vapour_frac_out = PropsSI('Q', 'P', (p_evap + u_p_evap)*1e5, 'H', (
    h_out_real + u_h_out_real), fluid1) - PropsSI('Q', 'P', p_evap*1e5, 'H',
    h_out_real, fluid1)
    df_sP.loc[0,'vapour_frac_ejector'] = vapour_frac_out
    df_sP_u.loc[0,'vapour_frac_ejector'] = u_vapour_frac_out

    return df_sP, df_sP_u

# Estimate cooling load through gravity loop on CO2 side when superheated
def getLoadCO2_gravity(df_stats, df_sP, df_sP_u):
    PT3 = df_stats.loc['Mean'].at['PT3']
    u_PT3 = df_stats.loc['Uncertainty'].at['PT3']
    DPT3 = df_stats.loc['Mean'].at['DPT3']
    u_DPT3 = df_stats.loc['Uncertainty'].at['DPT3']
    p_evap = PT3 + DPT3
    SC_PT3 = 1
    SC_DPT3 = 1
    u_p_evap = np.sqrt(np.power(SC_PT3*u_PT3,2) + np.power(SC_DPT3*u_DPT3,2))

    #Enthalpy from REFPROP:
    #Inlet evaporator
    h_in = PropsSI('H','P', p_evap*1e5,'Q', 0,fluid1)
    SC_h_in_pres = 1
    u_h_in_pres = PropsSI('H','P', (p_evap + u_p_evap)*1e5,'Q', 0,fluid1) -
```

```
    PropsSI('H','P', p_evap*1e5,'Q', 0,fluid1)
    SC_h_in_CoolProp = 1
    h_in_CoolProp = 1.5 #% from REFPROP (heat capacity liquid)
    u_h_in_CoolProp = h_in_CoolProp/100*h_in
    u_h_in = np.sqrt(np.power(SC_h_in_pres*u_h_in_pres,2) + np.power(
    SC_h_in_CoolProp*u_h_in_CoolProp,2))

    #Outlet evaporator
    pres_out = df_stats.loc['Mean'].at['PT3']
    u_pres_out = df_stats.loc['Uncertainty'].at['PT3']
    temp_out = df_stats.loc['Mean'].at['TT3'] + 273.15 # In Kelvin
    u_temp_out = df_stats.loc['Uncertainty'].at['TT3']
    h_out = max(PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1),PropsSI('
    H','P', pres_out*1e5,'Q', 1,fluid1))
    SC_h_out_pres = 1
    u_h_out_pres = PropsSI('H','P', (pres_out+u_pres_out)*1e5,'T', temp_out,
    fluid1) - PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1)
    SC_h_out_temp = 1
    u_h_out_temp = PropsSI('H','P', pres_out*1e5,'T', temp_out + u_temp_out,
    fluid1) - PropsSI('H','P', pres_out*1e5,'T', temp_out,fluid1)
    SC_h_out_CoolProp = 1
    h_out_CoolProp = 0.15 #% from REFPROP (heat capacity vapour)
    u_h_out_CoolProp = h_out_CoolProp/100*h_out
    u_h_out = np.sqrt(np.power(SC_h_out_pres*u_h_out_pres,2) + np.power(
    SC_h_out_temp*u_h_out_temp,2) + np.power(SC_h_out_CoolProp*
    u_h_out_CoolProp,2))

    #Enthalpy difference
    delta_h = h_out - h_in
    SC_h = 1
    u_delta_h = np.sqrt(np.power(SC_h*u_h_out,2) + np.power(SC_h*u_h_in,2))

    #Mass flow meter
    massflow = df_stats.loc['Mean'].at['RHE_09109_MFsuct']/60 #kg/s
    u_massflow = df_stats.loc['Uncertainty'].at['RHE_09109_MFsuct']/60 #kg/s

    #Calculation load
    load_CO2 = massflow * delta_h
    SC_load_CO2_dh = massflow
    SC_load_CO2_massflow = delta_h
    u_load_CO2 = np.sqrt(np.power(SC_load_CO2_dh*u_delta_h,2) + np.power(
    SC_load_CO2_massflow*u_massflow,2))
    df_sP.loc[0,'Q_CO2_gravity [W]'] = load_CO2
    df_sP_u.loc[0,'Q_CO2_gravity [W]'] = u_load_CO2

    #Outlet vapor frac
    Q_Water_gravity = df_sP.loc[0,'Q_Water_gravity [W]']
    u_Q_Water_gravity = df_sP_u.loc[0,'Q_Water_gravity [W]']
    SC_h_real_hin = 1
    SC_h_real_Q = 1/massflow
    SC_h_real_mass = Q_Water_gravity/np.power(massflow,2)
    h_out_real = Q_Water_gravity/massflow + h_in
    u_h_out_real = np.sqrt(np.power(SC_h_real_Q*u_Q_Water_gravity,2) + np.
    power(SC_h_real_mass*u_massflow,2) + np.power(SC_h_real_hin*u_h_in,2))
    vapour_frac_out = PropsSI('Q', 'P', p_evap*1e5, 'H', h_out_real, fluid1)
    u_vapour_frac_out = PropsSI('Q', 'P', (p_evap + u_p_evap)*1e5, 'H', (
    h_out_real + u_h_out_real), fluid1) - PropsSI('Q', 'P', p_evap*1e5, 'H',
    h_out_real, fluid1)
    df_sP.loc[0,'vapour_frac_gravity'] = vapour_frac_out
    df_sP_u.loc[0,'vapour_frac_gravity'] = u_vapour_frac_out

    return df_sP, df_sP_u

```

```
323
324 # Estimate the the vapour fraction at outlet gravity HX
325 def getVapourFraction_gravity(df_stats, df_sP, df_sP_u):
326     PT3 = df_stats.loc['Mean'].at['PT3']
327     u_PT3 = df_stats.loc['Uncertainty'].at['PT3']
328     DPT3 = df_stats.loc['Mean'].at['DPT3']
329     u_DPT3 = df_stats.loc['Uncertainty'].at['DPT3']
330     p_evap = PT3 + DPT3
331     SC_PT3 = 1
332     SC_DPT3 = 1
333     u_p_evap = np.sqrt(np.power(SC_PT3*u_PT3,2) + np.power(SC_DPT3*u_DPT3,2))
334
335     #Enthalpy from REFPROP:
336     #Inlet evaporator
337     h_in = PropsSI('H','P',p_evap*1e5,'Q',0,fluid1)
338     h_lat = PropsSI('H','P',p_evap*1e5,'Q',1,fluid1)
339     SC_h_in_pres = 1
340     u_h_in_pres = PropsSI('H','P', (p_evap + u_p_evap)*1e5,'Q', 0,fluid1) -
    PropsSI('H','P', p_evap*1e5,'Q', 0,fluid1)
341     SC_h_in_CoolProp = 1
342     h_in_CoolProp = 1.5 #% from REFPROP (heat capacity liquid)
343     u_h_in_CoolProp = h_in_CoolProp/100*h_in
344     u_h_in = np.sqrt(np.power(SC_h_in_pres*u_h_in_pres,2) + np.power(
    SC_h_in_CoolProp*u_h_in_CoolProp,2))
345
346     #Mass flow meter
347     massflow = df_stats.loc['Mean'].at['RHE_09109_MFsuct']/60 #kg/s
348     u_massflow = df_stats.loc['Uncertainty'].at['RHE_09109_MFsuct']/60 #kg/s
349
350     #Outlet vapour fraction
351     Q_Water_gravity = df_sP.loc[0,'Q_Water_gravity [W]']
352     h_out = Q_Water_gravity/massflow + h_in
353     # vapour_frac_out = PropsSI('Q', 'P', p_evap*1e5, 'H', h_out, fluid1)
354     vapour_frac_out_lin = (h_out-h_in)/(h_lat-h_in)
355     # df_sP.loc[0,'vapour_frac,gravity'] = vapour_frac_out
356     df_sP.loc[0,'vapour_frac_lin,gravity'] = vapour_frac_out_lin
357
358     return df_sP, df_sP_u
359
360
361 def LMTD(temp1_a, u_temp1_a, temp1_b, u_temp1_b, temp2_a, u_temp2_a, temp2_b,
     u_temp2_b):
362     diff_a = temp1_a - temp2_a
363     SC_temp = 1
364     u_diff_a = np.sqrt(np.power(SC_temp*u_temp1_a,2) + np.power(SC_temp*
    u_temp2_a,2))
365     diff_b = temp1_b - temp2_b
366     u_diff_b = np.sqrt(np.power(SC_temp*u_temp1_b,2) + np.power(SC_temp*
    u_temp2_b,2))
367     LMTD = (diff_a - diff_b)/np.log(diff_a/diff_b)
368     SC_LMTD_diff_a = (np.log(diff_a/diff_b)-(diff_a-diff_b)/diff_a)/np.power(
    np.log(diff_a/diff_b),2)
369     SC_LMTD_diff_b = (-np.log(diff_a/diff_b)+(diff_a-diff_b)/diff_b)/np.power
    (np.log(diff_a/diff_b),2)
370     u_LMTD = np.sqrt(np.power(SC_LMTD_diff_a*u_diff_a,2) + np.power(
    SC_LMTD_diff_b*u_diff_b,2))
371     return LMTD, u_LMTD
372
373 def getLMTD(df_stats, df_sP, df_sP_u):
374     #Plate HX in gravity loop
375     temp_water_in = df_stats.loc['Mean'].at['TT10'] + 273.15 # In Kelvin
376     u_temp_water_in = df_stats.loc['Uncertainty'].at['TT10']
377     temp_water_out = df_stats.loc['Mean'].at['TT14'] + 273.15 # In Kelvin
```

```
    u_temp_water_out = df_stats.loc['Uncertainty'].at['TT14']
    p_evap = df_stats.loc['Mean'].at['PT3']
    u_p_evap = df_stats.loc['Uncertainty'].at['PT3']
    temp_evap = PropsSI('T','P',1e5*p_evap,'Q',1,fluid1) #In Kelvin;
    SC_temp_evap_CoolProp = 1
    temp_evap_CoolProp = 0.2 #In percentage, from REFPROP
    u_temp_evap_CoolProp = temp_evap_CoolProp/100*temp_evap
    SC_temp_evap_upres = 1
    u_temp_evap_upres = PropsSI('T','P',1e5*(p_evap + u_p_evap),'Q',1,fluid1)
     - PropsSI('T','P',1e5*p_evap,'Q',1,fluid1)
    u_temp_evap = np.sqrt(np.power(SC_temp_evap_CoolProp*u_temp_evap_CoolProp
    ,2) + np.power(SC_temp_evap_upres*u_temp_evap_upres,2))
    LMTD_HX_gravity, u_LMTD_HX_gravity = LMTD(temp_water_in, u_temp_water_in,
     temp_water_out, u_temp_water_out, temp_evap, u_temp_evap, temp_evap,
    u_temp_evap)
    df_sP.loc[0,'LMTD_HX_gravity [K]'] = LMTD_HX_gravity
    df_sP_u.loc[0,'LMTD_HX_gravity [K]'] = u_LMTD_HX_gravity

    #Plate HX in ejector loop
    temp_water_in = df_stats.loc['Mean'].at['TT14'] + 273.15 # In Kelvin
    u_temp_water_in = df_stats.loc['Uncertainty'].at['TT14']
    temp_water_out = df_stats.loc['Mean'].at['TT11'] + 273.15 # In Kelvin
    u_temp_water_out = df_stats.loc['Uncertainty'].at['TT11']
    p_evap = df_stats.loc['Mean'].at['PT4']
    u_p_evap = df_stats.loc['Uncertainty'].at['PT4']
    temp_evap = PropsSI('T','P',1e5*p_evap,'Q',1,fluid1) #In Kelvin;
    SC_temp_evap_CoolProp = 1
    temp_evap_CoolProp = 0.2 #In percentage, from REFPROP
    u_temp_evap_CoolProp = temp_evap_CoolProp/100*temp_evap
    SC_temp_evap_upres = 1
    u_temp_evap_upres = PropsSI('T','P',1e5*(p_evap + u_p_evap),'Q',1,fluid1)
     - PropsSI('T','P',1e5*p_evap,'Q',1,fluid1)
    u_temp_evap = np.sqrt(np.power(SC_temp_evap_CoolProp*u_temp_evap_CoolProp
    ,2) + np.power(SC_temp_evap_upres*u_temp_evap_upres,2))
    LMTD_HX_ejector, u_LMTD_HX_ejector = LMTD(temp_water_in, u_temp_water_in,
     temp_water_out, u_temp_water_out, temp_evap, u_temp_evap, temp_evap,
    u_temp_evap)
    df_sP.loc[0,'LMTD_HX_ejector [K]'] = LMTD_HX_ejector
    df_sP_u.loc[0,'LMTD_HX_ejector [K]'] = u_LMTD_HX_ejector
    return df_sP, df_sP_u


def UA (Qevap, u_Qevap, LMTD_evap, u_LMTD_evap):
    UA = Qevap/LMTD_evap
    SC_Qevap = 1/LMTD_evap
    SC_LMTD_evap = Qevap/np.power(LMTD_evap,2)
    u_UA = np.sqrt(np.power(SC_Qevap*u_Qevap,2)+np.power(SC_LMTD_evap*
    u_LMTD_evap,2))
    return UA, u_UA


def getUA(df_sP, df_sP_u):

    #Plate HX in ejector loop
    Qevap = df_sP.loc[0,'Q_Water_ejector [W]']
    u_Qevap = df_sP_u.loc[0,'Q_Water_ejector [W]']
    LMTD_evap = df_sP.loc[0,'LMTD_HX_ejector [K]']
    u_LMTD_evap = df_sP_u.loc[0,'LMTD_HX_ejector [K]']
    UA_HX, u_UA_HX = UA(Qevap, u_Qevap, LMTD_evap, u_LMTD_evap)
    df_sP.loc[0,'UA_HX_ejector [W/K]'] = UA_HX
    df_sP_u.loc[0,'UA_HX_ejector [W/K]'] = u_UA_HX

    #Plate HX in gravity loop
```

```
    Qevap = df_sP.loc[0,'Q_Water_gravity [W]']
    u_Qevap = df_sP_u.loc[0,'Q_Water_gravity [W]']
    LMTD_evap = df_sP.loc[0,'LMTD_HX_gravity [K]']
    u_LMTD_evap = df_sP_u.loc[0,'LMTD_HX_gravity [K]']
    UA_HX, u_UA_HX = UA(Qevap, u_Qevap, LMTD_evap, u_LMTD_evap)
    df_sP.loc[0,'UA_HX_gravity [W/K]'] = UA_HX
    df_sP_u.loc[0,'UA_HX_gravity [W/K]'] = u_UA_HX
    return df_sP, df_sP_u

def getData(df_stats, df_sP, df_sP_u):
    df_sP.loc[0,'p_motive [bar]'] = df_stats.loc['Mean'].at['PT1']
    df_sP_u.loc[0,'p_motive [bar]'] = df_stats.loc['Uncertainty'].at['PT1']
    df_sP.loc[0,'p_tank [bar]'] = df_stats.loc['Mean'].at['PT2']
    df_sP_u.loc[0,'p_tank [bar]'] = df_stats.loc['Uncertainty'].at['PT2']
    df_sP.loc[0,'p_HX_gravity [bar]'] = df_stats.loc['Mean'].at['PT3']
    df_sP_u.loc[0,'p_HX_gravity [bar]'] = df_stats.loc['Uncertainty'].at['PT3
    ']
    df_sP.loc[0,'p_HX_ejector [bar]'] = df_stats.loc['Mean'].at['PT4']
    df_sP_u.loc[0,'p_HX_ejector [bar]'] = df_stats.loc['Uncertainty'].at['PT4
    ']
    df_sP.loc[0,'p_receiver [bar]'] = df_stats.loc['Mean'].at['Prec [barg]']
    + 1
    df_sP_u.loc[0,'p_receiver [bar]'] = df_stats.loc['Uncertainty'].at['Prec
    [barg]']
    df_sP.loc[0,'dp_ej,lift [bar]'] = df_stats.loc['Mean'].at['DPT1']
    df_sP_u.loc[0,'dp_ej,lift [bar]'] = df_stats.loc['Uncertainty'].at['DPT1'
    ]
    df_sP.loc[0,'dp_wat,HX [bar]'] = df_stats.loc['Mean'].at['DPT2']
    df_sP_u.loc[0,'dp_wat,HX [bar]'] = df_stats.loc['Uncertainty'].at['DPT2']
    df_sP.loc[0,'dp_CO2_gravity,HX [bar]'] = df_stats.loc['Mean'].at['DPT3']
    df_sP_u.loc[0,'dp_CO2_gravity,HX [bar]'] = df_stats.loc['Uncertainty'].at
    ['DPT3']
    df_sP.loc[0,'dp_CO2_ejector,HX [bar]'] = df_stats.loc['Mean'].at['DPT4']
    df_sP_u.loc[0,'dp_CO2_ejector,HX [bar]'] = df_stats.loc['Uncertainty'].at
    ['DPT4']
    df_sP.loc[0,'MF_wat,MF1 [kg/min]'] = df_stats.loc['Mean'].at['
    RHE_24722_MF1']
    df_sP_u.loc[0,'MF_wat,MF1 [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_24722_MF1']
    df_sP.loc[0,'MF_wat,MF2 [kg/min]'] = df_stats.loc['Mean'].at['
    RHE_24721_MF2']
    df_sP_u.loc[0,'MF_wat,MF2 [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_24721_MF2']
    df_sP.loc[0,'MF_gascooler [kg/min]'] = df_stats.loc['Mean'].at['
    RHE_33708_MFG']
    df_sP_u.loc[0,'MF_gascooler [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_33708_MFG']
    df_sP.loc[0,'MF_oil [kg/min]'] = df_stats.loc['Mean'].at['RHE_26472_MFoil
    ']
    df_sP_u.loc[0,'MF_oil [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_26472_MFoil']
    df_sP.loc[0,'MF_ej,mot [kg/min]'] = df_stats.loc['Mean'].at['
    RHE_33612_MFmot']
    df_sP_u.loc[0,'MF_ej,mot [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_33612_MFmot']
    df_sP.loc[0,'MF_gravity [kg/min]'] = df_stats.loc['Mean'].at['
    RHE_09109_MFsuct']
    df_sP_u.loc[0,'MF_gravity [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_09109_MFsuct']
    df_sP.loc[0,'MF_ejector [kg/min]'] = df_stats.loc['Mean'].at['RHE_MFn']
    df_sP_u.loc[0,'MF_ejector [kg/min]'] = df_stats.loc['Uncertainty'].at['
    RHE_MFn']
    df_sP.loc[0,'T_wat,HX,in [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at['
```

```
    TT10']
475     df_sP_u.loc[0,'T_wat,HX,in [\N{DEGREE SIGN}C]'] = df_stats.loc['
    Uncertainty'].at['TT10']
476     df_sP.loc[0,'T_wat,HX,mid [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at[
    'TT14']
477     df_sP_u.loc[0,'T_wat,HX,mid [\N{DEGREE SIGN}C]'] = df_stats.loc['
    Uncertainty'].at['TT14']
478     df_sP.loc[0,'T_wat,HX,out [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at[
    'TT11']
479     df_sP_u.loc[0,'T_wat,HX,out [\N{DEGREE SIGN}C]'] = df_stats.loc['
    Uncertainty'].at['TT11']
480     df_sP.loc[0,'T_ej,mot [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at['TT1
    ']
481     df_sP_u.loc[0,'T_ej,mot [\N{DEGREE SIGN}C]'] = df_stats.loc['Uncertainty'
    ].at['TT1']
482     df_sP.loc[0,'T_ej,suct [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at['
    TT4']
483     df_sP_u.loc[0,'T_ej,suct [\N{DEGREE SIGN}C]'] = df_stats.loc['Uncertainty
    '].at['TT4']
484     df_sP.loc[0,'T_gravity,out [\N{DEGREE SIGN}C]'] = df_stats.loc['Mean'].at
    ['TT3']
485     df_sP_u.loc[0,'T_gravity,out [\N{DEGREE SIGN}C]'] = df_stats.loc['
    Uncertainty'].at['TT3']
486     return df_sP, df_sP_u
487
488 def ejector_calc(df_stats, df_sP, df_sP_u):
489     #All LabVIEW values
490     p_outlet = df_stats.loc['Mean'].at['PT2']
491     p_suction = df_stats.loc['Mean'].at['PT4']
492     p_motive = df_stats.loc['Mean'].at['PT1']
493     u_p_outlet = df_stats.loc['Uncertainty'].at['PT2']
494     u_p_suction = df_stats.loc['Uncertainty'].at['PT4']
495     u_p_motive =  df_stats.loc['Uncertainty'].at['PT1']
496
497 # ============================================================================
498 #     Pressure lift
499 # ============================================================================
500
501     p_lift_calc = p_outlet - p_suction
502     u_p_lift_calc = np.sqrt(np.power(u_p_outlet,2)+np.power(u_p_suction,2))
503     df_sP.loc[0,'p_lift_calc [bar]'] = p_lift_calc
504     df_sP_u.loc[0,'p_lift_calc [bar]'] = u_p_lift_calc
505
506     p_lift_dp = df_stats.loc['Mean'].at['DPT1']
507     u_p_lift_dp = df_stats.loc['Uncertainty'].at['DPT1']
508     df_sP.loc[0,'p_lift_dp [bar]'] = p_lift_dp
509     df_sP_u.loc[0,'p_lift_dp [bar]'] = u_p_lift_dp
510
511 # ============================================================================
512 #     Suction pressure ratio
513 # ============================================================================
514
515     sp_ratio = p_outlet/p_suction
516     SC_outlet = 1/p_suction
517     SC_suction = p_outlet/np.power(p_suction,2)
518     u_sp_ratio = np.sqrt(np.power(SC_outlet*u_p_outlet,2)+np.power(SC_suction
    *u_p_suction,2))
519     df_sP.loc[0,'suction_pressure_ratio [-]'] = sp_ratio
520     df_sP_u.loc[0,'suction_pressure_ratio [-]'] = u_sp_ratio
521
522 # ============================================================================
523 #     Mass entrainment ratio
524 # ============================================================================
```

```
525
526     mdot_motive = df_stats.loc['Mean'].at['RHE_33612_MFmot']
527     u_mdot_motive = df_stats.loc['Uncertainty'].at['RHE_33612_MFmot']
528     mdot_suction = df_stats.loc['Mean'].at['RHE_MFn']
529     u_mdot_suction = df_stats.loc['Uncertainty'].at['RHE_MFn']
530
531     me_ratio = mdot_suction / mdot_motive
532     SC_mdot_motive = mdot_suction/np.power(mdot_motive,2)
533     SC_mdot_suction = 1/mdot_motive
534     u_me_ratio = np.sqrt(np.power(SC_mdot_motive*u_mdot_motive,2)+np.power(
        SC_mdot_suction*u_mdot_suction,2))
535
536     df_sP.loc[0,'mass_entrainment_ratio [-]'] = me_ratio
537     df_sP_u.loc[0,'mass_entrainment_ratio [-]'] = u_me_ratio
538
539 # ============================================================================
540 #     Ejector efficiency
541 # ============================================================================
542
543     temp_suction = df_stats.loc['Mean'].at['TT4']
544     u_temp_suction = df_stats.loc['Uncertainty'].at['TT4']
545     temp_motive = df_stats.loc['Mean'].at['TT1']
546     u_temp_motive = df_stats.loc['Uncertainty'].at['TT1']
547
548     #Enthalpy motive
549     h_A = PropsSI('H','P', p_motive*1e5,'T', temp_motive+273.15,fluid1)
550     SC_h_A_pres = 1
551     u_h_A_pres = PropsSI('H','P', (p_motive+u_p_motive)*1e5,'T', temp_motive
        +273.15,fluid1) - PropsSI('H','P', p_motive*1e5,'T', temp_motive+273.15,
        fluid1)
552     SC_h_A_temp = 1
553     u_h_A_temp = PropsSI('H','P', p_motive*1e5,'T', temp_motive+u_temp_motive
        +273.15,fluid1) - PropsSI('H','P', p_motive*1e5,'T', temp_motive+273.15,
        fluid1)
554     SC_h_A_CoolProp = 1
555     h_A_CoolProp = 1.5 #% from REFPROP (heat capacity liquid)
556     u_h_A_CoolProp = h_A_CoolProp/100*h_A
557     u_h_A = np.sqrt(np.power(SC_h_A_pres*u_h_A_pres,2)+np.power(SC_h_A_temp*
        u_h_A_temp,2)+np.power(SC_h_A_CoolProp*u_h_A_CoolProp,2))
558
559     #Entropy motive nozzle
560     s_A = PropsSI('Smass','P', p_motive*1e5,'T', temp_motive+273.15,fluid1)
561     SC_s_A_pres = 1
562     u_s_A_pres = PropsSI('Smass','P', (p_motive+u_p_motive)*1e5,'T',
        temp_motive+273.15,fluid1) - PropsSI('Smass','P', p_motive*1e5,'T',
        temp_motive+273.15,fluid1)
563     SC_s_A_temp = 1
564     u_s_A_temp = PropsSI('Smass','P', p_motive*1e5,'T', temp_motive+
        u_temp_motive+273.15,fluid1) - PropsSI('Smass','P', p_motive*1e5,'T',
        temp_motive+273.15,fluid1)
565     SC_s_A_CoolProp = 1
566     s_A_CoolProp = 1.5 #% from REFPROP (heat capacity liquid). No better info
         for the entropy..
567     u_s_A_CoolProp = s_A_CoolProp/100*s_A
568     u_s_A = np.sqrt(np.power(SC_s_A_pres*u_s_A_pres,2)+np.power(SC_s_A_temp*
        u_s_A_temp,2)+np.power(SC_s_A_CoolProp*u_s_A_CoolProp,2))
569
570     #Enthalpy motive after isentropic expansion to outlet pressure
571     h_B = PropsSI('H','P', p_outlet*1e5,'Smass', s_A,fluid1)
572     SC_h_B_pres = 1
573     u_h_B_pres = PropsSI('H','P', (p_outlet+u_p_outlet)*1e5,'Smass', s_A,
        fluid1) - PropsSI('H','P', p_outlet*1e5,'Smass', s_A,fluid1)
574     SC_h_B_s = 1
```

```
    u_h_B_s = PropsSI('H','P', p_outlet*1e5,'Smass', s_A + u_s_A,fluid1) -
    PropsSI('H','P', p_outlet*1e5,'Smass', s_A,fluid1)
    SC_h_B_CoolProp = 1
    h_B_CoolProp = 1.5 #% from REFPROP (heat capacity liquid)
    u_h_B_CoolProp = h_B_CoolProp/100*h_B
    u_h_B = np.sqrt(np.power(SC_h_B_pres*u_h_B_pres,2)+np.power(SC_h_B_s*
    u_h_B_s,2)+np.power(SC_h_B_CoolProp*u_h_B_CoolProp,2))

    #Enthalpy difference isentropic expansion
    delta_h_exp = h_A - h_B
    SC_h = 1
    u_delta_h_exp = np.sqrt(np.power(SC_h*u_h_A,2) + np.power(SC_h*u_h_B,2))

    #Enthalpy suction
    h_D = max(PropsSI('H','P', p_suction*1e5,'T', temp_suction+273.15,fluid1)
    , PropsSI('H','P', p_suction*1e5,'Q', 1,fluid1))
    SC_h_D_pres = 1
    u_h_D_pres = PropsSI('H','P', (p_suction+u_p_suction)*1e5,'T',
    temp_suction+273.15,fluid1) - PropsSI('H','P', p_suction*1e5,'T',
    temp_suction+273.15,fluid1)
    SC_h_D_temp = 1
    u_h_D_temp = PropsSI('H','P', p_suction*1e5,'T', temp_suction+
    u_temp_suction+273.15,fluid1) - PropsSI('H','P', p_suction*1e5,'T',
    temp_suction+273.15,fluid1)
    SC_h_D_CoolProp = 1
    h_D_CoolProp = 0.15 #% from REFPROP (heat capacity vapour)
    u_h_D_CoolProp = h_D_CoolProp/100*h_D
    u_h_D = np.sqrt(np.power(SC_h_D_pres*u_h_D_pres,2)+np.power(SC_h_D_temp*
    u_h_D_temp,2)+np.power(SC_h_D_CoolProp*u_h_D_CoolProp,2))

    #Entropy suction nozzle
    s_D = max(PropsSI('Smass','P', p_suction*1e5,'T', temp_suction+273.15,
    fluid1), PropsSI('Smass','P', p_suction*1e5,'Q', 1,fluid1))
    SC_s_D_pres = 1
    u_s_D_pres = PropsSI('Smass','P', (p_suction+u_p_suction)*1e5,'T',
    temp_suction+273.15,fluid1) - PropsSI('Smass','P', p_suction*1e5,'T',
    temp_suction+273.15,fluid1)
    SC_s_D_temp = 1
    u_s_D_temp = PropsSI('Smass','P', p_suction*1e5,'T', temp_suction+
    u_temp_suction+273.15,fluid1) - PropsSI('Smass','P', p_suction*1e5,'T',
    temp_suction+273.15,fluid1)
    SC_s_D_CoolProp = 1
    s_D_CoolProp = 0.15 #% from REFPROP (heat capacity liquid). No better
    info for the entropy..
    u_s_D_CoolProp = s_D_CoolProp/100*s_D
    u_s_D = np.sqrt(np.power(SC_s_D_pres*u_s_D_pres,2)+np.power(SC_s_D_temp*
    u_s_D_temp,2)+np.power(SC_s_D_CoolProp*u_s_D_CoolProp,2))

    #Enthalpy suction after isentropic compression to outlet pressure
    h_C = PropsSI('H','P', p_outlet*1e5,'Smass', s_D,fluid1)
    SC_h_C_pres = 1
    u_h_C_pres = PropsSI('H','P', (p_outlet+u_p_outlet)*1e5,'Smass', s_D,
    fluid1) - PropsSI('H','P', p_outlet*1e5,'Smass', s_D,fluid1)
    SC_h_C_s = 1
    u_h_C_s = PropsSI('H','P', p_outlet*1e5,'Smass', s_D + u_s_D,fluid1) -
    PropsSI('H','P', p_outlet*1e5,'Smass', s_D,fluid1)
    SC_h_C_CoolProp = 1
    h_C_CoolProp = 0.15 #% from REFPROP (heat capacity vapour)
    u_h_C_CoolProp = h_C_CoolProp/100*h_C
    u_h_C = np.sqrt(np.power(SC_h_C_pres*u_h_C_pres,2)+np.power(SC_h_C_s*
    u_h_C_s,2)+np.power(SC_h_C_CoolProp*u_h_C_CoolProp,2))

    #Enthalpy difference isentropic compression
```

```
    delta_h_comp = h_C - h_D
    SC_h = 1
    u_delta_h_comp = np.sqrt(np.power(SC_h*u_h_C,2) + np.power(SC_h*u_h_D,2))

    #Ejector efficiency
    nj_ejector = me_ratio * delta_h_comp/delta_h_exp
    SC_nj_me_ratio = delta_h_comp/delta_h_exp
    SC_nj_delta_h_comp = me_ratio /delta_h_exp
    SC_nj_delta_h_exp = me_ratio * delta_h_comp/np.power(delta_h_exp,2)
    unj_ejector = np.sqrt(np.power(SC_nj_me_ratio*u_me_ratio,2)+np.power(
    SC_nj_delta_h_comp*u_delta_h_comp,2)+np.power(SC_nj_delta_h_exp*
    u_delta_h_exp,2))

    df_sP.loc[0,'ejector_efficiency [-]'] = nj_ejector
    df_sP_u.loc[0,'ejector_efficiency [-]'] = unj_ejector
    return df_sP, df_sP_u
```