A numerical study of a wastewater-air source hybrid heat pump for domestic use

by

Ugnė Bunikytė

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday November 18th, 2022 at 09:00 AM.

Student number:	5371929	
Project duration:	January 1st, 2022 – October 30th, 2022	
Thesis committee:	Prof. dr. ir. R. Pecnik,	TU Delft, supervisor
	Prof. dr. ir. K. Hooman,	TU Delft
	Dr. ir. J.W.R. Peeters,	TU Delft
	Ir. S. Wapperom,	DeWarmte, supervisor

This thesis is confidential and cannot be made public until November 13th, 2023.

Acknowledgements

I would like to express my gratitude to all those who have supported me during this project and have helped me develop as an engineer.

In particular, I would like to thank my supervisor at TU Delft, Prof.Dr.ir.R. Pecnik, for his engineering expertise and assistance in maintaining the scope and clarity of my work.

I would like to thank my supervisor at DeWarmte, Sander Wapperom, for his many sacrificed hours and for helping me develop as a professional engineer by always striking balance between guidance and assistance.

I would like to thank Daniel Mensink for his help in structuring my numerical model and his ideas in model development.

I would like to thank Dennis van Heerwaarden for his expertise on heat pump systems. Without his guidance, I would have spent many hours searching for relevant material, and without his humour, these hours would have felt far longer.

Finally, I would like to express my gratitude to all those who have helped and supported me during this project and my master's degree: my father Jonas Bunikis, my mother Elena Bunikienė, my sisters Ieva and Vaiva Bunikytė, my boyfriend Francisco Onofre and my friend and fellow student Chiara Lucia Tregnago.

Ugnė Bunikytė

Delft, November 2022

Abstract

To face the ongoing climate crisis, the Dutch government adopted a number of policies and measures to limit its greenhouse gas emissions. It is estimated that 10% of Dutch annual emissions are caused by the built environment, a large portion of which come from two sources: space heating and domestic hot water.

The use of heat pumps is said to meaningfully contribute to the electrification of domestic heat supply. However, heat pumps face intermittency issues during unfavorable climate conditions when heating is most needed, thus limiting their potential of being standalone heating systems.

A hybrid heat pump - meaning a single heat pump operating with multiple renewable heat sources - could efficiently provide heat all year round with maximised performance. However, literature on the subject is sparse and non-systematic. Therefore, conclusions on the system performance and its variability cannot be drawn.

Thus, the aim of this project was to create a modular numerical model of a hybrid heat pump system to analyse its performance. This was achieved by individually modelling and experimentally validating each component.

The chosen heat sources were wastewater and air. The numerical model showed satisfactory correlation with experimental results for each component, particularly the condenser of the heat pump, air source and wastewater storage bag. Improvements could be made in the evaporator of the heat pump, boiler and spiral heater.

After analysis of its yearly performance, it may be concluded that the modelled hybrid heat pump is not currently a worthwhile investment for a typical Dutch home. However, with a higher capacity heat pump, alongside other component and control optimisations, the system can be an all-electric alternative to many gas-heated homes.

Contents

Nomenclature

Constants

σ	Stefan Boltzmann constant = $5.67 \times 10^{-8} (W/m^2K^4)$
g	Gravitational acceleration = 9.81 (m/s^2)

Abbreviations

$ASHP$	Air source heat pump
COP	Coefficient of performance
DHW	Domestic hot water
DX	Direct exchange
HEX	Heat exchanger
HP	Heat pump
IX	Indirect exchange
NTU	Number of transfer units
RH	Relative humidity
SH	Space heating
SPF	Seasonal performance factor

Dimensionless numbers

Nu	Nusselt
Pr	Prandtl
Ra	Rayleigh
Re	Reynolds

Material Properties

α	Thermal diffusivity (m^2/s)
β	Thermal expansion (1/K)
ϵ	Emissivity (-)
μ	Dynamic viscosity (Pa s)
ν	Kinematic viscosity (m^2/s)
ρ	Density (kg/m^3)
C_p	Specific heat (J/kgK)
F	Fouling factor (m^2K/W)
h	Enthalpy (J/kg)
k	Thermal conductivity (W/mK)

Symbols

$\dot{m}$	Mass flow (kg/s))
ΔT	Temperature difference (K)
Δx	Length in x-direction (m)
Δy	Length in y-direction (m)
Δz	Length in z-direction (m)
η	Efficiency
A	Area (m^2)
d	Diameter (m)
f	Frequency (Hz)
h	Heat transfer coefficient (W/m^2K)
P	Pressure (Pa)
Q	Heat (W)
R	Thermal resistance (m^2K/W)
s	Spacing (m)
T	Temperature (K)
t	Thickness (m)
U	Overall heat transfer coefficient (W/m^2K)
V	Volume (m^3)
v	Velocity (m/s)
W	Power (W)

List of Figures

List of Tables

Part I: Project introduction

1

Motivation

Governments have recognized the significance of climate change and its acceleration due to human activity. The Paris agreement, held during the Paris climate conference in December 2015, was the first legally binding international treaty on the topic of climate change mitigation, with the goal of limiting global temperature rise to below 2°C, and preferably to 1.5°C, in comparison to pre-industrial times. Since then, countries have adopted policies and set future goals to limit their greenhouse gas emissions (GGEs). In the Dutch Climate Act of 2019, the Netherlands has pledged to decrease their GGEs by 49% in 2030 and by 95% in 2050, in comparison to 1990 levels [1]. Policies to reach these goals target electricity, industry, built environment, transport and agriculture sectors [12]. Strategies in the agreement include scaling up the production of electricity from renewable sources to 84 TWh, efforts for the integration of renewables into the electricity grid and introducing of a carbon levy on industry GGEs. Despite the gas shortage and rapid increases in energy prices in Europe, the Dutch government still aims to sease the production of natural gas in the Groningen gas field by 2024. The current state of affairs emphasizes the requirement for countries to decrease their dependence on natural gas and its suppliers. The primary method of achieving this is by investing in large-scale research and infrastructure of sustainable electricity and heat generation.

The primary focus within the built environment section of the Climate Act is to improve home insulation and replace gas boilers with renewable heating solutions or district heating. This is to be expected; it has been estimated that the built environment released 24.6 MT of GGEs in 2017, contributing to approximately 10% of the total Dutch GGEs that year [63]. Reducing these emissions will be achieved by means of subsidized relevant personal renovations, such as implementation of renewable heating solutions or higher grade thermal insulation in homes, district heating projects using geothermal energy and energy rating requirements for new buildings [16].

According to the Climate Agreement, the use of heat pumps meaningfully contributes to the electrification of heat supply to the built environment. The Netherlands aims to increase large-scale heat production using geothermal heat pumps. The transition to sustainable heating networks is paramount, however there is a market for independent heating solutions. Before the war in Ukraine, it was estimated that between 2014 and 2018 air source heat pump purchases increased from 4000 to 24000 per year [36]. The trend can only be expected to grow. Recently, the Dutch government announced the plan for homes to switch to a heating network or independent hybrid heat pump (note: hybrid in this context meaning in combination with fossil fuels) by banning the installation of gas boilers by 2026. Furthermore, the current subsidy for purchasing a heat pump has increased to approximately 30% of its price [62].

The European Heat Pump Association has stated that in 2020, 14.84 million heat pumps were installed in the EU, an increase of 6% or 1.6 million units from 2019 [2]. With the increase in market demand, academia has shifted its focus on maximising the performance of heat pump systems. Countless studies have been conducted on optimisation of their components, control and refrigerants, all with the aim of maximising heat extraction capability [48], [34], [65]. However, the potential of providing heat demand still heavily relies on the climate of the region in which the heat pump is installed. Furthermore, heat pump systems face intermittency of heat generation, seen on both a daily and seasonal scale, with sources such as solar power being unavailable during the night and temperatures of air and shallow ground sources dropping during heating season. These issues limit the potential of heat pumps being used as standalone heating systems, with no auxiliary heat being provided by natural gas. Therefore, academic efforts have recently turned towards the integration of multiple renewable heat sources operating with one heat pump, called a hybrid heat pump (note: hybrid in this context meaning with multiple renewable heat sources, henceforth this type of heat pump will always be referred to as hybrid). With a methodically justified choice of heat

sources, system design, control strategy and addition of thermal storage options, it is possible to maximise their performance and decouple the production and usage of heat. However, literature on the subject is considerably sparse, non-systematic and consists of a majority of numerical models with little experimental validation.

Thus, the aim of this research project is to contribute to the identified knowledge gap by creating a numerical performance model of a hybrid heat pump system. To achieve this goal, the model must be modular, able to predict performance for various households and climates and be experimentally validated.

This project is being conducted at DeWarmte. In Section 2, the company and it's wastewater heat pump are introduced. The reasoning behind creating a wastewater-air source hybrid heat pump is substantiated. In Section 3, a review of relevant theory and literature on air source heat pumps, wastewater recuperation technologies and hybrid heat pumps is presented. In Section 4, the hybrid heat pump system is described. Sections 5-9 describe the numerical modelling, experimental validation and sensitivity analysis of individual components in the system. Results of the performance model are presented in Section 11. In Section 12, further work in model improvement and recommendations for system improvement are outlined.

1.1. Research questions

This project will answer or give recommendations towards the following research questions:

- **How can the hybrid heat pump system be physically modelled?**
- **How can the system be modelled in a way that may be translated to different climates, homes and user profiles?**
- **What is the predicted performance of the system (in terms of COP, heat output, work input, monetary savings)?**
- **What affects the performance of the system?**
- **How can the system be optimised?**

2

Background

This thesis is being completed at DeWarmte, a scale-up located in Delft, the Netherlands. The company aims to reduce CO_2 emissions caused by the built environment. This is done with the HeatCycle, a heat pump (HP) system that recovers heat from a home's wastewater. For a detailed description of the HeatCycle, refer to Appendix A. Roughly 40% of the heat demand of an average Dutch household can be supplied by the HeatCycle, accounting for all domestic hot water (DHW) demand and a portion of space heating (SH). To compensate for the remaining heat demand, DeWarmte seeks to combine the HeatCycle with an air source, creating a hybrid heat pump.

DeWarmte conducted a preliminary feasibility study to test whether a hybrid wastewater heat pump would perform favorably and be economically advantageous. The internal document may be obtained upon request [67]. The HeatCycle was numerically tested in combination with the following heating technologies: air source heat pump (ASHP), exhaust air heat pump (EAHP), ground source heat pump (GSHP) and solar collectors (SCs) (flat plate and evacuated tube types).

Initially, the DHW demand profile over a year of a 4-person, typical Dutch home was created. The HP demand was modelled by creating a fictitious usage profile of a person throughout the day, based on average daily hot water consumption, the appliances it is used for and the time of day these appliances are most commonly used. The SH demand of a home was created by assuming a constant temperature at which the home is kept and calculating heat gains and losses by the home. For a more detailed description of the method, refer to Section 10.3.

Based on the SH and DHW demand profiles and calculated temperatures of the sources, the Coefficients of Performance (COPs, see Equation (3.2)) of the HeatCycle, ASHP, EAHP and GSHP are calculated. The COPs are considered to be their Carnot COP (see Equation (3.3)) multiplied by an efficiency of 0.5. For each timestep in a year, given a certain HeatCycle and source combination, the COP of the HeatCycle and the chosen source is compared. The system whose COP is higher supplies the respective demand in that timestep. The average COP over the year was calculated.

After running the simulation, the results in terms of gas and monetary savings may be obtained. These are seen below in Table 2.1. Single source heating technologies are compared to hybrid variations with the HeatCycle in terms of the installation cost, gas savings, monetary savings and payback time. The results indicate that the combination of the studied sources with a HeatCycle always increase the yearly monetary savings. The payback time decreases for combinations with SCs and an EAHP, but increases in combination with an ASHP or GSHP.

These results should be viewed critically, as many assumptions were made when calculating them; the largest of which is the assumption that the COP of a system is always half of its Carnot COP. This is unsubstantiated and will lead to both an overestimate or underestimate of the performance of the system, depending on the season. Furthermore, the study was conducted using 2020 energy prices. Therefore, the monetary savings are now very different.

Nevertheless, based on the preliminary study, it may be concluded that combining an additional heat source to the HeatCycle, effectively creating a hybrid wastewater HP, has potential in being advantageous for both the system's performance and its monetary savings. From the results, it was concluded that the combination of an ASHP to the HeatCycle would be most advantageous due to its ability to fulfill the full heat demand of a home (albeit with help from an electric heater), low upfront

cost and better marketability due to non-invasive installation.

Thus, the necessity for this thesis project is defined. The aim of the project from the company's perspective is to create a performance model of a hybrid air-wastewater source HP. Using this model, it will be possible to verify and modify the results of the preliminary study and to test the performance, savings and marketability of the system. Furthermore, by creating a modular numerical model, it may be used for performance calculations of the HeatCycle as a standalone system for customers. The necessity and role the thesis project from a research perspective has been introduced in Section 1 and will be further elaborated upon in Section 3.3. First, a relevant literature review on ASHPs, wastewater heat recuperation and hybrid HPs must be presented.

Table 2.1: Results of DeWarmte's techno-economic feasibility study of a hybrid wastewater HP

Heating System	**Cost (€)**	**Gas savings (m^3/yr)**	**Monetary savings (€/yr)**	**Payback time (yr)**
HeatCycle	3500	697	340	10.3
ASHP	8350	1405	612	13.6
GSHP	13305	1405	985	13.4
EAHP	2200	951	140	15.7
Evacuated tube SC	3500	235	207	16.9
Flat plate SC	3500	279	245	14.3
ASHP & HeatCycle*	11850	1405	740	16.0
GSHP & HeatCycle*	16805	1405	1004	16.7
EAHP & HeatCycle	5700	985	404	14.1
Evacuated tube SC & HeatCycle	7000	879	580	12.1
Flat plate SC & HeatCycle	7000	909	612	11.4

* The two systems have the same gas savings but different monetary savings, as a GSHP can fulfill the full demand of a home and does not require auxiliary electrical heating. An ASHP requires auxiliary electrical heating.

3

The wastewater-air source hybrid heat pump: a literature review

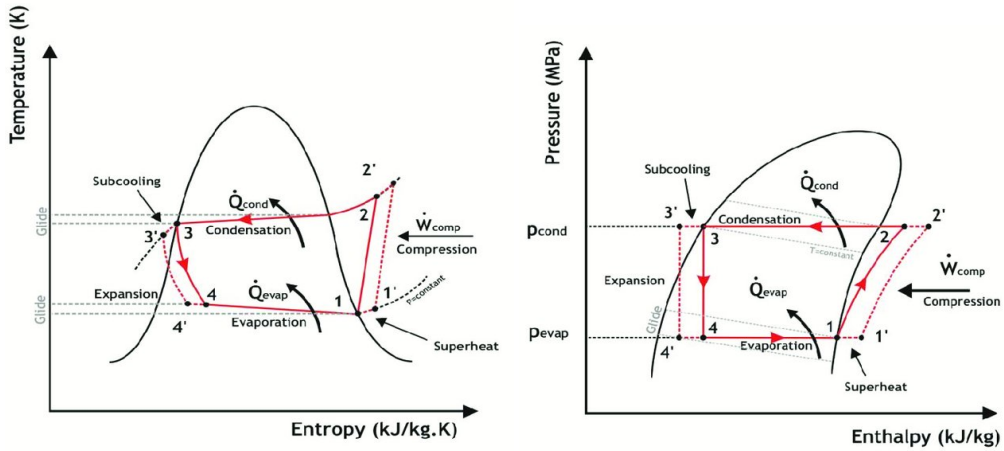
3.1. Air source heat pumps

In this research project, one of the heat sources of the hybrid heat pump will be air. Thus, in this section, a review of relevant topics on air source heat pumps (ASHPs) will be presented. Their operation, underlying thermodynamics and performance will be discussed.

3.1.1. Operational principles

An ASHP is a system used to transfer heat from ambient air to an indoor environment. An ASHP system contains an evaporator, compressor, condenser and expansion valve. The evaporator of the HP is installed in an outdoor unit, thus being the direct connection to the heat source of the system. The outdoor unit contains a fan, increasing the flow and therefore heat transfer of the air over the evaporator. Heat is transferred from the air to the refrigerant, causing it to boil. By undergoing the vapour compression refrigeration cycle, the refrigerant condenses in the condenser, releasing its latent heat to the inside of the house. The vapour compression refrigeration cycle is displayed in Figure 3.1. With ideal heat pump control, the steps are:

- **1'→2':** Refrigerant vapor exits the evaporator at a slightly superheated state. The vapor is compressed to a higher pressure and temperature state, where $_{in}$ represents the electric input into the compressor. The process is non-isentropic.
- **2'→3':** Refrigerant enters the condenser at a high pressure, superheated state and rejects heat to surroundings as it condenses, noted by $_{out}$. The process is non-adiabatic, but at constant pressure. It is released from the condenser at a slightly subcooled state.
- **3'→4':** The subcooled refrigerant passes through an expansion valve, passing to a lower pressure and temperature state. It exits the expansion valve with a low quality value.
- **4'→1':** Refrigerant passes through the evaporator, boiling, as it reaches a saturated vapor state. The heat input to the refrigerant is denoted as $_{in}$. The process is non-adiabatic, but at constant pressure.



(a) T-s diagram (b) P-h diagram

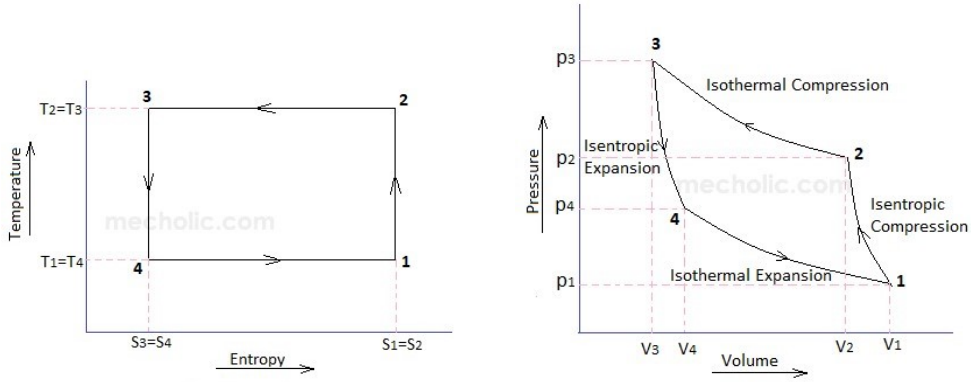
Figure 3.1: Refrigeration cycle [26]

Two primary types of ASHP systems exist; an air-to-air and air-to-water HP. The outside of the condenser of an air-to-air system is in contact with the indoor environment that is being heated. The condenser of an air-to-water system releases heat to an auxiliary working fluid or is used to directly heat water. The choice of ASHP is dependent on the needs of a home. For example, an air-to-air ASHP may be reversed to act as an air conditioner, however this means it requires a ventilation network to distribute heated/cooled air. This literature review and project will focus on air-to-water ASHPs, as the goal of the hybrid heat pump is to be able to fullfill both space heating and domestic hot water demands in a home. Furthermore, it will focus on the evaporator of the ASHP, as this component is in direct contact with the heat source. Thus, when considering the feasibility of a hybrid heat pump with an air source component, the operation and recent advances in optimisation of the performance of the evaporator are most relevant. The advancements in performance of other heat pump components and its control are considered to be beyond the scope of this review.

3.1.2. Thermodynamic analysis

The operation of a heat pump is based on a reverse Carnot cycle, displayed in Figure 3.2. The stages of the reverse Carnot cycle are as follows:

- **1→2:** Isentropic, adiabatic compression
- **2→3:** Isothermal compression
- **3→4:** Isentropic, adiabatic expansion
- **4→1:** Isothermal expansion



(a) T-s diagram (b) P-v diagram

Figure 3.2: Reverse Carnot cycle [14]

The Carnot efficiency of a heat engine is defined as the maximum efficiency a heat engine can have when operating between two temperatures. It is given by

$$\eta_{Carnot} = \frac{W_{in}}{Q_{out}} = 1 - \frac{T_C}{T_H}, \tag{3.1}$$

where η_{Carnot} is the Carnot efficiency, W_{in} is the work input to the engine in (W), Q_{out} is the heat output of the engine (W), T_C is the temperature of the cold source (K) and T_H is the temperature of the hot sink (K).

The COP of a heat pump is a measure of its efficiency. It is defined as the ratio of heat output to work input of the heat pump, given by

$$COP_{HP} = \frac{Q_{out}}{W_{in}}, \tag{3.2}$$

Substituting the expression for η_{Carnot} found in Equation 3.1, it is found that

$$COP_{Carnot} = \frac{1}{\eta_{Carnot}} = \frac{T_H}{T_H - T_C}, \tag{3.3}$$

where COP_{Carnot} may be seen as the ideal heat pump COP. As η_{Carnot} may never be above 1, it is noted that COP_{Carnot} will always remain above 1, meaning it will produce a larger heat output than the required work input. Furthermore, it is noted that a heat pump has a higher COP, and therefore operates better, under smaller temperature differences between the cold source and hot sink.

The heat output, Q_{out} of a heat pump is given by

$$Q_{out} = \dot{m}_{ref}\,(h_{in} - h_{out}), \tag{3.4}$$

where $\dot{m}_{ref}$ is the mass flow rate of the refrigerant in the condenser (kg/s), h is enthalpy (J/kg), and subscripts in and out correspond to condenser inlet (point 2 in Figure 3.1) and outlet (point 3 in Figure 3.1) respectively.

Similarly, the heat input, Q_{in} to a heat pump is given by

$$Q_{in} = \dot{m}_{ref}\,(h_{out} - h_{in}), \tag{3.5}$$

where $\dot{m}_{ref}$ is the mass flow rate of the refrigerant in the evaporator in (kg/s) and subscripts in and out correspond to evaporator inlet (point 4 in Figure 3.1) and outlet (point 1 in Figure 3.1) respectively.

Although measured in practice, the work input, W_{in} to a heat pump may be calculated with

$$W_{in} = \dot{m}_{ref}\,(h_{out} - h_{in}), \tag{3.6}$$

where $\dot{m}_{ref}$ is the mass flow rate of the refrigerant in the compressor in (kg/s) and subscripts in and out correspond to compressor inlet (point 1 in Figure 3.1) and outlet (point 2 in Figure 3.1) respectively.

Therefore, using Equation 3.2, the COP of a real heat pump is given by

$$COP_{HP} = \frac{\dot{m}_{ref,cond}\,(h_{in,cond} - h_{out,cond})}{\dot{m}_{ref,comp}\,(h_{out,comp} - h_{in,comp})}, \tag{3.7}$$

with subscripts $cond$ and $comp$ corresponding to condenser and compressor respectively.

3.1.3. Evaporator performance-affecting factors

In this section, the factors affecting the efficiency and performance of ASHP evaporators are discussed. They will provide insight on what is important when modelling the air source for the hybrid heat pump. The factors are presented in the context of recent research for a more comprehensive review.

3.1.3.1. Outside unit design

The outside unit is arguably the most important component of the ASHP, due to it being the direct connection to the heat source of the system. The design of the outside unit consists of a heat exchanger, fan and outer casing. The heat exchanger (HEX) is a collection of winded finned tubes with refrigerant flowing on the inside and air flowing on the outside. The purpose of the fins is to increase the effective surface area for heat transfer. The number of tubes and types of fins are designed accordingly to the heat requirements and noise allowance of the location. A fan is used to blow air over the finned tubes, promoting turbulence and therefore increasing the heat transfer between air and refrigerant. This design is well established and no recent developments have been made in its improvement.

3.1.3.2. Climate

ASHPs operating in mild or cold climates face the disadvantage of frost formation on the evaporator during cooler months. When the surface temperature of the evaporator coils drops to the dew point temperature of water, the moisture in the air condenses onto their surfaces. Due to cold refrigerant flowing through the tubes, the surface temperature of the evaporator may drop to the freezing point of water, freezing the condensed drops. Prolonged exposure to these conditions causes a layer of frost to form, acting as an insulation barrier to the heat transfer between air and refrigerant, decreasing the conductance, air flow and therefore performance of the ASHP. It is reported that the climate condition at which frost will form is at an ambient air temperature of approximately 5°C and a relative humidity (RH) of 65% [48], [59]. The effects of the frost formation on the COP of the system are said to be a decrease of up to 20% [37], [66].

3.1.4. Defrosting technologies

In cooler climates, the formation of frost is inevitable on the outside unit. However, it can be minimized and/or removed. Defrosting solutions in ASHPs may be classified into three primary groups:

- **Frost prevention**: solutions that aim to minimize the frost formation-prone conditions in the evaporator's surroundings.
- **Frost growth mitigation**: technologies that aim to prevent frost attachment and minimize its growth once the conditions for frost formation are met.
- **Defrosting techniques**: techniques in which already formed frost is removed from the evaporator.

3.1.4.1. Frost prevention

Reduction of inlet air humidity

To mitigate the effects of high relative humidity in ambient air, incoming air passing over the evaporator may initially be dehumidified. Although this technique is more commonly used in closed refrigeration units [48], studies have additionally been conducted for the outdoor units of ASHPs. Su et al. [60] studied the application of a system in which air initially passes through a liquid Lithium-Chloride desiccant dehumidifier before entering the evaporator. The sensible heat lost from the air to the desiccant circuit is recovered with an additional desiccant-to-refrigerant heat exchanger. It was found that the proposed system was superior to the conventional reverse cycle defrosting in sub-zero temperatures at a RH of 65%.

Preheating inlet air

Preheating incoming cold air is another technology used to prevent frost formation. Studies focus on an additional heat source, such as exhaust air or wastewater, to preheat the air. Nourozi et al. [49] studied the application of a wastewater-to-air heat exchanger to reduce frost formation on the heat exchanger in mechanical ventilation heat recovery units in the cold climate of Sweden. It was found that the system reduced defrosting needs by 30 to 50 % during heating season.

3.1.4.2. Frost growth mitigation

Heat exchanger and fin geometry

The number of tubes, tube alignment, fin type and density are design factors that affect the performance of the evaporator. Studies have been performed to find optimal values of these factors to minimize frost growth and its effect on heat transfer in the evaporator. Yang et al. [69] conducted an optimization study for values of fin spacing in a heat exchanger under frosting conditions, increasing the heat transfer rate and operating time of the heat exchanger by 5.5% and 12.9% respectively. Lee at al. [40] studied the air-side heat transfer coefficient under frosting conditions for varying fin spacing, tube numbers and tube alignment. It was found that the heat transfer coefficient increased with the number of tube rows, decrease of fin pitch and the use of aligned, instead of staggered, tube arrangement.

Fin coating

Frost growth rate and frost density on a metal surface is highly dependent on the its surface characteristics. An area of interest is frost growth on hydrophobic and hydrophilic surfaces, or varying the solid-liquid contact angle between frost and evaporator surface. A study by Mahvi et al. [45] investigated the effects of coating aluminium heat exchanger fins of an electric vehicle heat pump with hydrophobic and hydrophilic materials under frosting conditions. It was found that hydrophobic coating delayed frost formation by 55 min under the lowest temperature (-0.7 °C) tested. Under the same conditions, frost began forming on the uncoated and hydrophilically coated heat exchanger after the first few minutes of operation. Jhee et al. [35] seconds these results, finding that a hydrophilic coating on a heat exchanger under frosting conditions increased frost formation and density.

3.1.4.3. Defrosting techniques

Off-cycle

A simple defrosting technique is the shutdown of the heat pump compressor, stopping the flow of the refrigerant to the evaporator. Power to the fan is maintained. The fan blows outside air onto the frost, melting it. As expected, this technique requires the ambient air temperature to be above 0 °C. Although the simplicity and little energy consumption of the technique is attractive, the defrosting duration is longer, preventing the ASHP from being used for extended periods of time [48], [59].

Electric heating

Electric heating defrost consists of electric heaters being integrated into the outside unit of an air source heat pump. During defrosting, the electric heaters are turned on, allowing cool ambient air to heat up as it flows over the heaters and the evaporator coils, melting the frost. However, the efficiency of this process is very low, with most heat produced by the heaters being lost to ambient surroundings. To prevent this issue, Zhao et al. [72] investigated the performance of electric heater defrosting in the evaporators of large scale refrigerators and designed a fan cover that opens during cooling load and closes when defrosting is turned on, thus reducing thermal losses. Furthermore, an additional electric heater was placed in the evaporator unit to ensure uniform defrosting. This technique reduced energy usage by 1.2%.

Hot gas bypass

Hot gas bypass defrosting is traditionally applied to commercial ASHPs [48]. Instead of passing through the heat pump's condenser and expansion valve, superheated refrigerant vapor is directly passed through the evaporator. Here it condenses, releasing its latent heat and therefore melting frost formed on the outside of the evaporator. Currently, research is being done in control strategies for the hot gas bypass system to increase its efficiency. Xi et al. [68] proposed a new control strategy based on the maximum heating capacity of the ASHP. It was found that with this control, the defrosting cycle was 4.06% more energy efficient and its heating capacity was 10.17% higher.

Reverse cycle

Much like with hot gas bypass, reverse cycle defrosting directs superheated refrigerant gas to the evaporator of the ASHP, condensing the gas and releasing its latent heat. This is done through an addition of a four-way valve to the system, allowing the cycle to be reversed. Ye et al. [70] compared the performance of the two defrosting methods in a CO_2 heat pump water heater. It was found that reverse cycle defrost used 17.5% of the total electricity consumption of hot gas bypass and had a shorter defrosting time, however it had more complex parameter variations.

3.1.5. Air source heat pump performance

In Section 3.1.2 the theoretical performance of a HP was discussed. When quantifying the performance of an installed, working HP system, different performance parameters are introduced. In this section, these parameters and their relevance will be reviewed.

Carnot efficiency

The Carnot efficiency, $\eta_{Carnot,HP}$ of a HP is defined as the ratio between working HP COP and its Carnot COP, as defined in Equations 3.3 and 3.7 respectively. It is an indicator of how closely the HP is operating to its maximum theoretical performance and is given by

$$\eta_{Carnot,HP} = \frac{COP_{HP}}{COP_{Carnot}}. \tag{3.8}$$

Seasonal performance factor

The seasonal performance factor (SPF, also known as seasonal COP or SCOP) is a metric used to quantify the annual performance of a HP. The COP of an ASHP is influenced by both the weather of its installed location and the season at which it is being measured, meaning it is lower in the winter for a heating load. Therefore, SPF is used to more accurately represent its overall heating potential. The SPF is commonly included in manufacturer's specifications and is an average based on measured values over a year at various operational conditions. The required operational conditions are set in country-specific standards and are used to mimic the country's climate. There are 4 types of SPFs.

SPF_{H1} is given by

$$SPF_{H1} = \frac{Q_{out}}{W_{comp}}, \tag{3.9}$$

Where Q_{out} is the heat output of the HP, taking into account both space heating and domestic water, W_{comp} is the electrical work done by the compressor of the HP.

SPF_{H2} is given by

$$SPF_{H2} = \frac{Q_{out}}{W_{HP}}, \tag{3.10}$$

where the term W_{HP} is the full electrical work input required for the HP system and its working fluid circulation, including the compressor, control circuit and circulation pumps in the HP.

SPF_{H3} is given by

$$SPF_{H3} = \frac{Q_{out} + Q_{BH}}{W_{HP} + W_{BH}}, \tag{3.11}$$

where the Q_{BH} represents the heat provided by a backup heater system and W_{BH} its electrical input requirement. The source of the backup heaters is not significant; it may be gas, district heating, electrical or another renewable source.

SPF_{H4} is given by

$$SPF_{H4} = \frac{Q_{out} + Q_{BH}}{W_{HP} + W_{BH} + W_D}, \tag{3.12}$$

where W_D is the electrical input required to distribute the heat from all sources through the building using fans or pumps.

3.2. Wastewater heat utilization

In this section, existing wastewater heat recuperation technologies will be presented. The aim is to give a comprehensive review on wastewater as a heat source, its drawbacks and uncertainties. Furthermore, the novelty of using wastewater as a source for a domestic HP is emphasized.

It has been estimated that in a city, 40% of produced heat is lost via wastewater [73]. With a high specific heat and mass density, high temperature, large flow rate and near-constant production, wastewater holds a significant potential for thermal recuperation. Wastewater heat recuperation may be classified into the following subgroups: recovery from within a building, outside a building and outside a city [32]. The former two may be considered small-scale applications of heat recovery, as they concern heat recovery for households or non-residential buildings, such as a spa or gym. The latter is a large-scale application, with waste heat recuperation taking place in wastewater treatment plants. Thus, it will not be discussed as it is not considered relevant for the project.

3.2.1. Household-level

For household-level recovery from wastewater, current technologies extract heat from the pipes of individual hot-water consuming devices, before the wastewater reaches the household's sewage pipe, seen in Figure 3.3. A few variations of such technologies are described below.

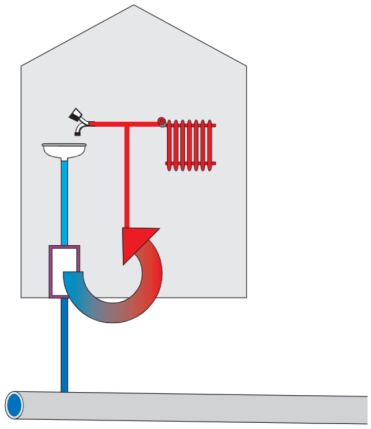


Figure 3.3: Household-level wastewater heat recovery [55]

Due to its high temperature, shower water is generally used for household-level heat recovery systems. The drained shower water passes through a heat exchanger before entering the sewage. The secondary heating fluid in the HEX is passed to a HEX in a water heater or boiler, preheating the water within. The type of HEX used to retrieve the heat varies.

A common type of HEX used in drain recovery systems is a falling-film HEX, seen in Figure 3.4. It consists of copper helical coils around a drain pipe, recuperating heat from the hot water flowing through the drain. Salama et al. [54] experimentally and numerically investigated the performance of this type of HEX. It was found that under fully-wetted conditions of the drain pipe, the system recuperated 29% to 46% of the available heat. The system lost 28.5% of its efficiency when operating under partially-wetted conditions, indicating the importance of flow rate and proper sizing of the HEX. Slys et al. [57] conducted a financial analysis of such systems, reporting that the optimal installation type is one in which the preheated water flows to a water heater and then to a shower mixing valve. Drain water heat recovery is functional, however its yield is low and hence are its savings. This is due to low water volume and fast flowing conditions. However, its yield may be increased with more home occupants and longer shower duration. Therefore the system must be implemented taking into consideration the number of home occupants and their hot water consumption pattern. Due to their relatively low cost, the payback time for these types of systems under appropriate operating conditions are estimated to be between 5-10 years [57], [52].

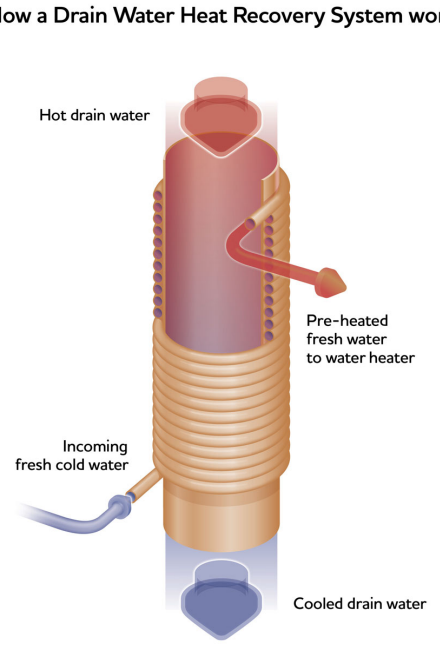


Figure 3.4: Falling-film HEX for drain heat recovery [6]

3.2.2. Building-level

For building-level recovery from wastewater, heat must be extracted from a main sewage pipe, as displayed in Figure 3.5.

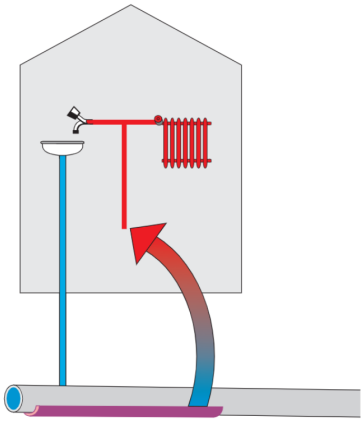


Figure 3.5: Building-level wastewater heat recovery [55]

Systems for building-level recuperation were found to exchange heat with an auxiliary fluid circuit via a HEX, with none passing wastewater directly through a HP. Varying HEXs have been found for this purpose. Sewage pipes may be constructed with inbuilt metal pipes on their bottom, allowing for heat exchange between sewage flow in the concrete pipe and auxiliary fluid in the metal pipe. Swiss company Rabtherm AG, produces such HEXs in sewage systems, supplying heat to large buildings or portions of cities [13] using a heat pump. Their system design is displayed below in Figure 3.6. Similarly, a project for recuperation from Toronto's hospital's wastewater has recently been approved [7], using a heat pump and HUBER RoWin HEXs, seen in Figure 3.7. A branch of Hungarian company Thermowatt specializes in wastewater HP installations for projects focused on large buildings. In comparison to the previously mentioned systems, Thermowatt filters wastewater before passing it to a HEX, which is in turn connected to a HP [15]. It is assumed that the additional step of filtering the wastewater is done to prevent the HEX from clogging and to subsequently decrease the constraints in HEX choice.

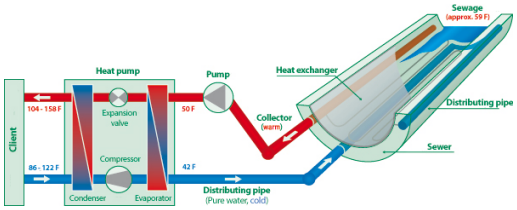


Figure 3.6: Wastewater heat pump system using inbuilt HEX in sewage pipe [13]

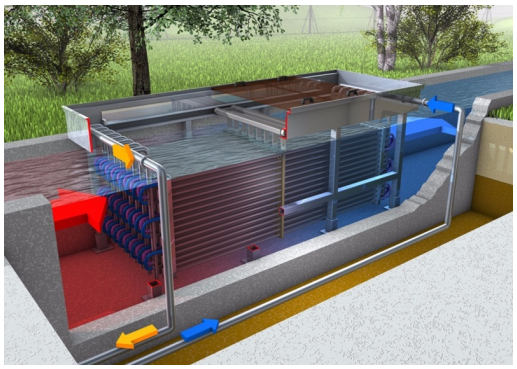


Figure 3.7: Wastewater HEXs to be used in hospital heat recuperation project [8]

With company specialization in and governmental funding of such projects, it is clearly feasible to extract heat from wastewater in large building complexes. However, the impact of local recuperation technologies on existing infrastructure must be examined. A study by Golzar et al. [29] investigated the impact of increasing the number of building-level wastewater heat recuperation technologies on the central heating system in Sweden. It was concluded that such local technologies were disruptive towards the current centralized heating network and wastewater plant recovery system. Conversely, a study by Hadengue et al. [30] developed a model to predict the effect of implementing household and multiple-household wastewater heat recovery on the recovery processes in wastewater treatment plants in Switzerland. It was found that due to significant thermal damping, an implementation of heat retrieval devices in 50% of the wastewater treatment plant's catchment area would decrease the incoming wastewater temperature by only 0.3 K. A Dutch feasibility study of small scale heat recovery investigated recuperating heat from the sewage of 20-50 houses, where all produced wastewater was taken into consideration [11]. Having estimated the heat recuperation potential of the system, the study focused on its economic feasibility, concluding that using a gas fired boiler was cheaper for inhabitants. However, the Dutch feasibility study was conducted in 2014, using 0.65 €/m^3 and 0.23 €/kWh for gas and electricity prices respectively. It was estimated that assuming constant electricity price, gas prices would have to rise to above 1.10 €/m^3 to make the investment worthwhile. With the highly unpredictable variability of these prices over time, a comparison between costs may not be the optimal metric for feasibility.

3.3. Hybrid heat pumps

In this section, hybrid heat pumps will be discussed. The use of wastewater as a source for a heat pump is a novel concept with no relating literature, therefore hybrid heat pumps containing an air source will be focused upon. Studies on hybrid ASHPs focus on its combination with solar energy, called a solar assisted ASHP (SA-ASHP). In this section, the discovered system designs and operating modes of SA-ASHPs are introduced. Recent studies and their reported performance are presented. The aim of this section is to better understand the types of hybrid heat pump systems and how they are designed. In turn, this will aid in the design of the hybrid wastewater-air source HP.

3.3.1. System design

System designs of hybrid heat pumps may be classified into two sections; direct exchange (DX) and indirect exchange (IX). This affects how the heat sources are connected and what their working fluid is.

3.3.1.1. Direct exchange

In DX systems, the ASHP outside HEX and the solar collector of the system act as an evaporator to a heat pump. There is no auxiliary heat exchanger or working fluid to transfer heat to the evaporator, simplifying the system. However, the installation of such a system is more complex and controlled, due to the increased and direct usage of refrigerant. Furthermore, the probability of refrigerant leakage is increased.

A DX hybrid heat pump design is feasible for the combination of the HeatCycle and air source. In such a system, the refrigerant circuit of an ordinary ASHP evaporator would be extended to a HEX used to exchange heat between wastewater and refrigerant. The benefits of this design would be reduced heat loss and possibility for reverse defrost of the air source evaporator. However, the HeatCycle heat pump would have to be redesigned, as it would no longer require a water-refrigerant evaporator.

3.3.1.2. Indirect exchange

In IX systems, the air source and solar collector circuits are separate. An auxiliary working fluid is collecting heat from the solar collector. The separation of the two circuits leads to a simpler installation than that of a DX system, but the additional HEX leads to more losses.

The combination of the HeatCycle with an air source in an IX system has the benefit of no redesign of the HeatCycle heat pump. In this arrangement, an auxiliary working fluid would gain heat from wastewater through an additional HEX. The air source evaporator would remain the same, however it would no longer act as an evaporator for the refrigerant. Instead it would transfer heat from the ambient to the auxiliary fluid.

3.3.2. Operating modes

In this section, the operating modes and their respective control will be presented. Three relevant operating modes may be distinguished.

3.3.2.1. Single source heat pump

In this operating mode, the heat pump uses one source at time. For the combination of wastewater and air, the heat pump can prioritize the source with the highest temperature, thus maximising COP. If no wastewater is present, air may be used as a source to fulfill heat demand.

3.3.2.2. Dual source heat pump

Both sources are used together to provide heat to the HP. This mode requires a redesigned evaporator to account for the three-fluid heat exchange. It is able to achieve the simplest control: the HP is simply turned on or off when demand is present. However, this lack of complexity leads to sub-optimal control. The temperatures of each source must be taken into account to test whether it is more beneficial to run the system as single or dual source.

3.3.2.3. Thermal energy charging

In this mode, an auxiliary buffer vessel is used to store heat. This is most commonly done with direct solar heating due to its high temperatures and low electricity consumption. However, it is also possible to store the heat produced by a HP when no demand is present.

3.3.3. Reported performance

A review of relevant hybrid SA-ASHP studies and their results are presented in Table 3.1. Only one conclusion can be drawn; all the studies that compared the performance of SA-ASHP to that of an ASHP or solar HP found that the performance of the hybrid system was enhanced. These conclusions may be found under the "Other findings" column in Table 3.1 for all studies excluding [41] and [23].

The reported performance, particularly when taking the COP or SPF as the primary performance indicator, varies greatly between studies, with COP/SPF values ranging from 2.25 to 6.3. The reason for these major differences in performance is twofold: the vast variability between system designs and testing conditions. All research found during this literature review focused on the study of a single chosen SA-ASHP system. No comparison studies were found between a few system types. Furthermore, the test conditions between studies were highly varying. Test conditions included testing the system under a certain temperature and irradiance level, a typical heating/cooling season day in a certain location/country and throughout the heating season in a certain location/country. These factors make the studies incomparable, and therefore no further conclusions may be drawn about SA-ASHP systems as whole.

Thus, a knowledge gap is identified: there is no benchmark to compare or test hybrid heat pump systems by. This may be because the relative novelty of SA-ASHPs leads to studies testing their performance and potential on a high-over, system level, with the purpose of market-availability for the energy transition. This translates to test conditions, both numerical and experimental, being highly variant, with studies focusing only on the location in which it is being conducted.

Therefore, the aim of this thesis from a research perspective is defined. The goal of the project is to aid in filling the identified knowledge gap in hybrid heat pump systems by creating a modular numerical model of the wastewater-air source hybrid heat pump. Each component modelled will be experimentally validated. By varying a set of input conditions, namely weather data and hot water usage characteristics, the model will be able to predict yearly performance of the system in various households and countries. Where possible, the inputs will be defined by European standards. Different control of the system will be easily implementable. This will allow for deeper analysis of the performance-affecting variables. Finally, due to the model's modularity, each component may be modified or replaced. For example, the wastewater source may be replaced by an additional model of a solar collector. Then, the model will be able to test the yearly performance of a SA-ASHP.

Table 3.1: Review of relevant studies on SA-ASHP

Study	Authors	System	Num/Exp	Test conditions	COP/SPF	Other findings
Performance analysis of a novel air source hybrid solar assisted heat pump [22]	Cai et al.	Series DX-SA-ASHP with SCs	Num	I=100 W/m^2, T_a=10°C	COP = 2.71	Irreversibility of the system analysed by calculating exergy loss ratio of each component. Highest irreversibility in compressor and condenser. Performance compared to that of ASHP and DX-SAHP. Proposed design yields highest COP.
Design and performance simulation of a novel hybrid PV/T-air dual source heat pump system based on a three-fluid heat exchanger [71]	Zhang et al.	Parallel IX-SA-ASHP with PV/T panels	Num	I=545 W/m^2, T_a=-3.3°C	COP = 5.9	Compared to ASHP, energy consumption of the system during heating season decreased by 13.1%. Compared to PV panel, electricity output increased by 14.7%.
A solar-air hybrid source heat pump for space heating and domestic hot water [53]	Ran et al.	Parallel IX-SA-ASHP with SCs	Num	Weather data over 3 days in mid-January in Chengdu, Beijing, Shenyang	SPF = 3.61, 3.27, 2.45 respectively	When operating under low irradiance during heating season, the heat produced accounted for a large portion of total heat production, signifying system is still useful under worst conditions. Defrosting energy consumption was lower in proposed system than in ASHP and SC+ASHP system.
Simulation analysis on operation performance of a hybrid heat pump system integrating photovoltaic/thermal and air source [42]	Li et al.	Parallel DX-SA-ASHP with PV/T panels	Num	I=100 W/m^2, T_a=0°C, I=400 W/m^2, T_a=10°C, I=700 W/m^2, T_a=20°C	COP = 3.7, 4.4, 6.3 respectively	Under first environmental condition, average COP of proposed system is 29.7% and 19.8% higher than that of SAHP and ASHP respectively.
Experiment Study on Heating Performance of Solar-Air Source Heat Pump Unit [44]	Liu et al.	Series IX-SA-ASHP with SCs	Exp	T_a=-5°C, SC water T=20°C	COP = 2.4	In comparison to ASHP performance increased by 13.9%.
Study on the performance of a solar assisted air source heat pump system for building heating [43]	Liang et al.	Series IX-SA-ASHP with SCs	Both	Sunny day during heating season in Nanjing	COP = 4.3	COP increases proportionally with SC area. In comparison to ASHP, COP increases by 11.22%.
Study on performance of solar assisted air source heat pump systems for hot water production in Hong Kong [41]	Li et al.	Parallel IX-SA-ASHP with SCs	Num	15th of July and 15th of November weather data in Hong Kong	COP = 3.86, 3.5 respectively	Effect of circulation flow rate, solar collector area, tilt angle on COP investigated.
Energy and exergy analysis of a novel solar-air composite source multi-functional heat pump [24]	Cai et al.	Parallel IX-SA-ASHP with SCs	Both	October 6th, 2019 weather data in Shanghai	COP = 3.26	Found refrigerant distribution ratio is sensitive to T_a and I, changing COP value. Proposed system has a higher saved operation cost than ASHP.
Analysis and optimization on the performance of a heat pump water heater with solar-air dual series source [23]	Cai et al.	DX-SA-ASHP with SCs or PV/T panels	Num	I=100 W/m^2, T_a = 10°C	COP = 2.25, 2.25 respectively	COPs remain same for SCs and PV/T. COP rises from 2.23 to 2.4 with I = 100 to 500 W/m^2, 2.23 to 2.63 with T_a = 10 to 20°C

Part II: Numerical model development

4

System description

In this section, the chosen system design and control is described and substantiated. Table 4.1 displays an overview of the chosen system. Figure 4.1 displays the modelled system.

An indirect exchange system was chosen because it is simpler to implement than a direct exchange system. The handling and loading of refrigerant requires hiring of specialised equipment and personnel. Furthermore, the current HeatCycle HP would no longer function in the system, as it has a shell and tube evaporator which would no longer be needed. The working fluid is glycol solution, due to its low freezing point. As the glycol will be used outside during winter, it must be able to withstand subzero temperatures. Heat extraction from the wastewater bag will be done with a plate HEX, with wastewater and glycol as the working fluids. A plate HEX was chosen as it transfers heat very efficiently. Heat extraction from the air will be done by a dry cooler. The difference between a dry cooler and an ASHP evaporator is the working fluid. In a dry cooler, glycol will only absorb heat, but will not evaporate. The chosen operating mode is single source heat extraction by the HP, with the source being chosen by its temperature. The reason for choosing this mode and control is to maximise the COP of the system. The chosen defrost method is heating by an electric element. This method is chosen due to its easy implementation, however it is sub-optimal. In further work, reversed cycle defrosting should be modelled and compared. Any additional heat for DHW or SH is also provided by electrical elements to test the viability of the system as a fully electric alternative to a gas-heated home.

Table 4.1: Modelled system description

System type	Indirect Exchange
Working fluid	Glycol solution
Heat extraction from wastewater source	Plate HEX
Heat extraction from air source	Dry cooler
Operating mode	Single source
Control	Highest temperature source chosen
Defrost method	Electric heating
Auxiliary DHW heating method	Electric heating
Auxiliary SH method	Electric heating

When a DHW or SH demand is registered (the demand calculations and prioritization is described in Section 10.1 and 10.3), the control checks whether the dry cooler or wastewater bag has the highest temperature. This source is chosen. Glycol solution is circulated from the HEX of the source to the evaporator of the HP. If during the cycle a higher temperature is logged in the other source, the system switches to using the latter. When defrosting is required (the control of which can be found under Section 6.2.1.1), the HP's compressor shuts off and the defrosting electric heater turns on. Heated glycol solution is circulated in the dry cooler until the frost melts.

On the supply side, heat is released by condenser. Depending on whether the HP is on DHW or SH heating mode, the heat will be released into the DHW circuit or SH circuit. In the DHW circuit, heated water circulates to the spiral heater, heating up

a boiler. When hot water is needed, the boiler is drained. If needed, this water is additionally heated with an electric heater. The same mass of cold water enters the boiler as has been drained. In the SH circuit, heated water circulates through the home and its radiators, returning to the condenser. Any additional required heat is provided by the electric element. The requirement for additional heat is described in Sections 10.1 and 10.3 for DHW and SH circuits respectively.

In the upcoming sections, the calculation methods within the individual components displayed in Figure 4.1 will be described. An experimental validation and a sensitivity analysis of relevant components will be presented.

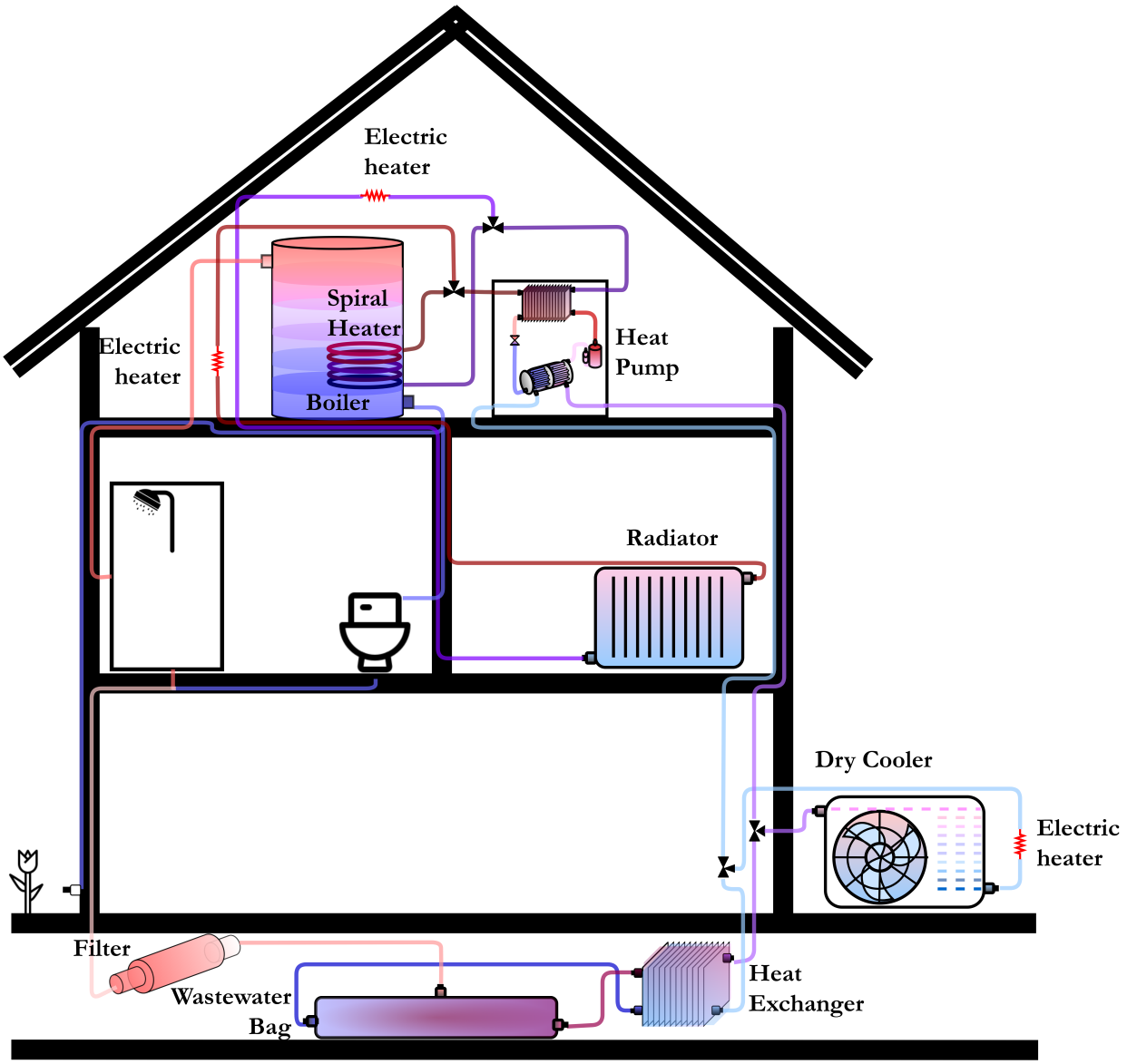


Figure 4.1: Modelled hybrid heat pump system

5

Heat pump model

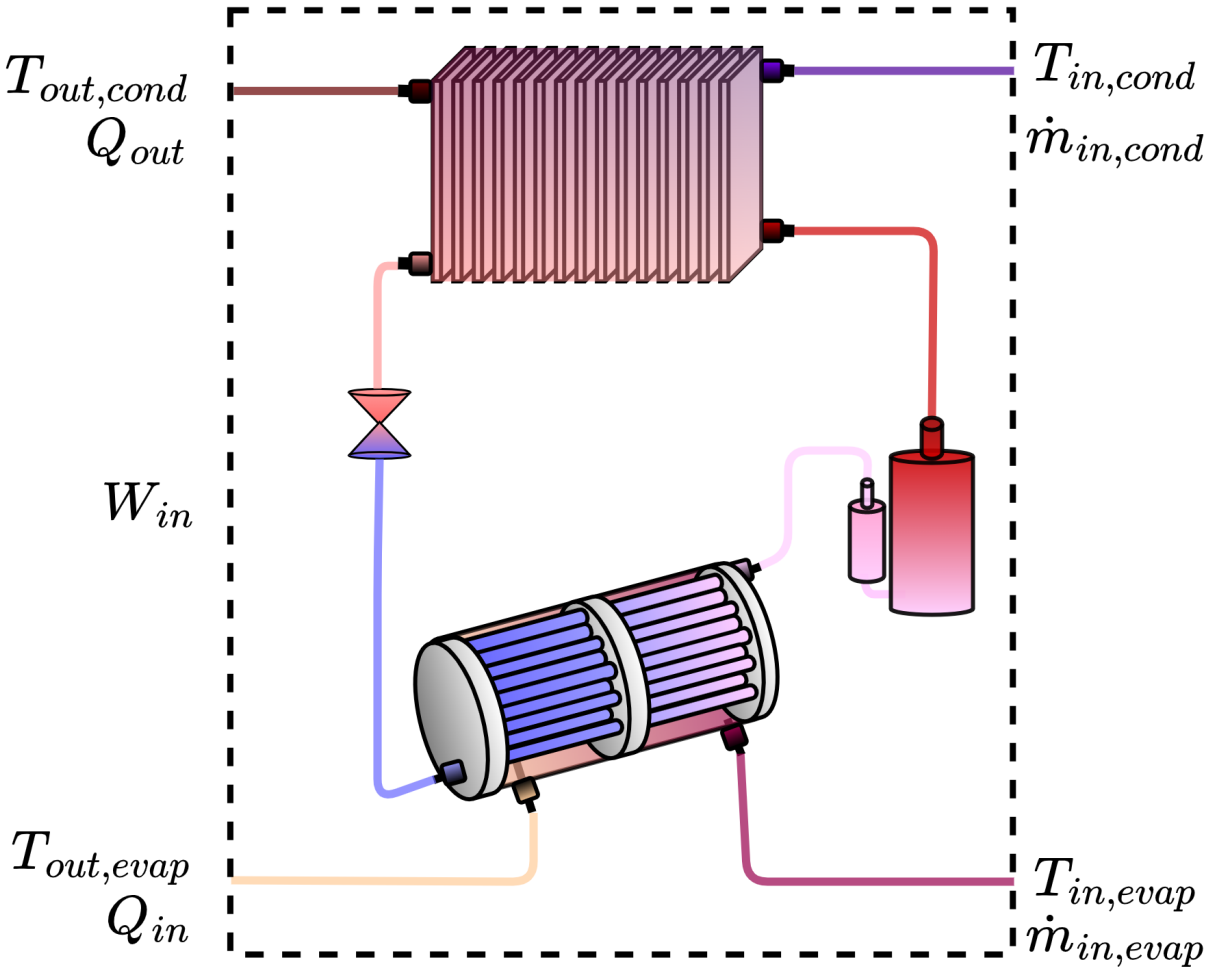


Figure 5.1: Heat pump class

Table 5.1: Inputs and outputs of heat pump class

Variable	Unit	Type	Description	Source
$T_{in,evap}$	K	Input	Temperature of incoming glycol to evaporator	HEX class
$T_{in,cond}$	K	Input	Temperature of incoming water to condenser	Spiral heater class
$\dot{m}_{in,evap}$	kg/s	Input	Mass flow rate of incoming glycol to evaporator	HEX class
$\dot{m}_{in,cond}$	kg/s	Input	Mass flow rate of incoming water to condenser	Spiral heater class
$T_{out,evap}$	K	Output	Temperature of outgoing glycol from evaporator	-
$T_{out,cond}$	K	Output	Temperature of outgoing water from condenser	-
Q_{in}	W	Output	Heat input to evaporator	-
Q_{out}	W	Output	Heat output by condenser	-
W_{in}	W	Output	Work input to compressor	-

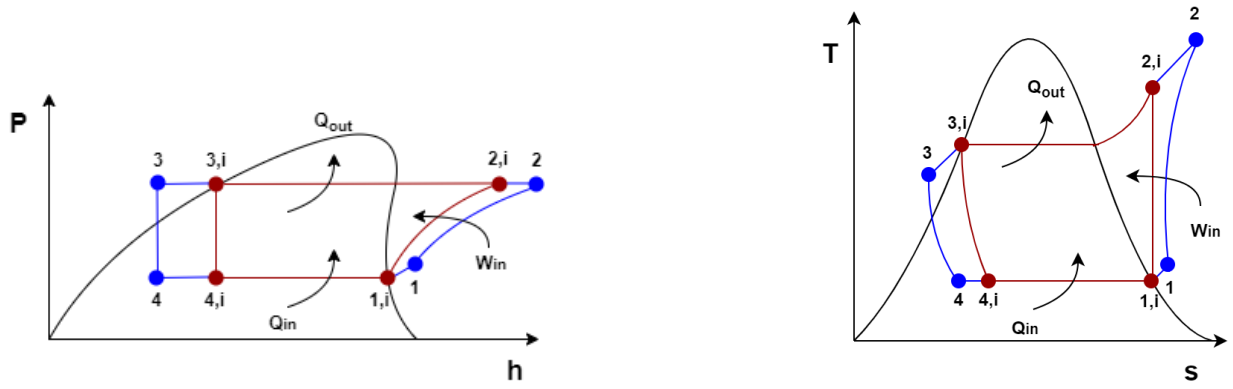
5.1. Numerical modelling approach

State of the art HP models split the evaporator and condenser into control volumes, modelling refrigerant's phase change within the system. By doing so, the model is able to capture dynamic changes in heat transfer coefficient throughout the HEX, and thus, in theory, is able to predict the HP's performance accurately. However, this method is exceedingly complex, as correct modelling of the heat transfer in two-phase flow requires modelling mass transfer. Thus, the method relies on empirical correlations for heat transfer at various qualities and two-phase flow regimes. With such complexity, potential source of error and computational requirements are increased. Thus, for the HP model presented in this work, a simpler approach is taken.

The HP of the system is modelled using a simplified thermodynamic model, a plate and shell-and-tube HEX model and a look-up table provided by its compressor's manufacturer. The benefits of the approach are twofold; a lower computational power requirement and a tailor-made model for the specific HP being used in the system. Notably, a different HP would be simple to implement in the model; a new compressor datasheet would be required.

5.1.1. Thermodynamic model of modelled heat pump

The HP model calculates condenser capacity, Q_{out} (W) at varying refrigerant evaporating temperatures, T_{evap} and condensing temperatures, T_{cond}. This is done using a semi-ideal vapour-compression refrigeration thermodynamic cycle. Its ideal and non-ideal pressure-enthalpy (Ph) and temperature-entropy (Ts) diagrams are displayed in Figure 5.2. Points ending with *i* signify stages in the ideal cycle. Note it is assumed that no pressure drop occurs in the evaporator and condenser.



(a) Ph diagram

(b) Ts diagram

Figure 5.2: Vapour compression refrigeration cycle

Q_{out} is given by equation

$$Q_{out} = \dot{m}_{ref} (h_3 - h_2), \tag{5.1}$$

where h is specific enthalpy (J/kg) and numeric subscripts signify the corresponding stage in the vapour-compression cycle.

In an ideal cycle, refrigerant enters the condenser at a quality of 1. However, in reality this is difficult to achieve due to limits of compressor control and dynamic changes in temperature. Thus, in the evaporator, refrigerant is superheated as a safety precaution. In HPs with complex control, superheat may be varied to increase efficiency. For a water-to-water HP, the degree of superheat is set low, due to less dynamic changes in temperature in comparison to that of air. Therefore, one assumption will be made to define the full thermodynamic system for the model of the HP: point 1 is equal to point 1,i.

The model uses the following relations:

$$Q_{in} = \dot{m}_{ref} (h_1 - h_4), \tag{5.2}$$

$$W_{in} = \dot{m}_{ref} (h_2 - h_1), \tag{5.3}$$

Values of Q_{in}, W_{in} and $\dot{m}_{ref}$ are known from the compressor datasheet given a T_{evap} and T_{cond} combination, as described in Section 5.1.2. Values of enthalpy are found using the Coolprop 6.4.1 library in Python. $h_{1,i}$ is defined with its respective quality and temperature: 1 at T_{evap}. This is set to h_1. h_4 is found using Equation 5.2 and is equal to h_3. h_2 is found using Equation 5.3. Using h_2 and P_{cond}, the refrigerant temperature at the exit of the compressor, $T_{exhaust}$, may be found. This parameter will be further used for safety control of the HP, described in Section 5.2.

5.1.2. Compressor look-up table

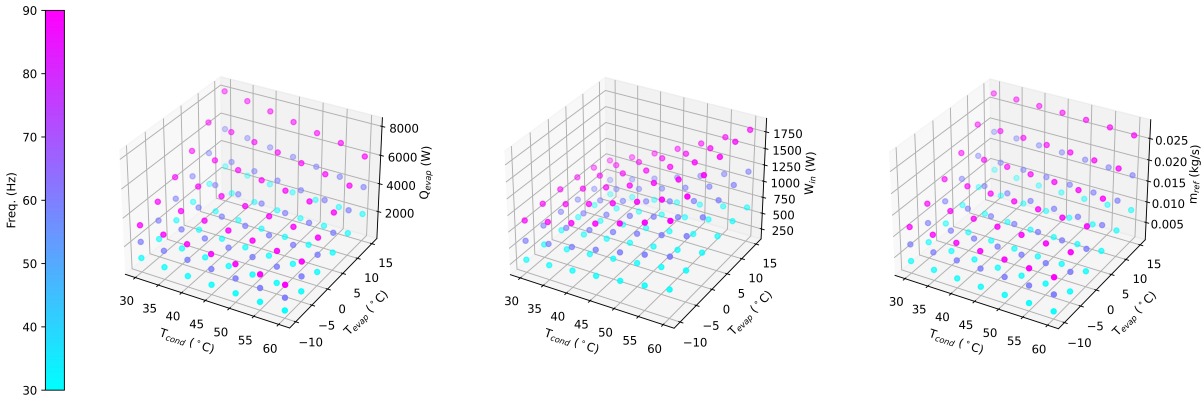
The model of the rotary compressor in the heatpump is GMCC KSN98D22UFZ. Its datasheet outlines its performance in various conditions. The required input variables to define these conditions and the given outputs are shown in Table 5.2. An example of the datasheet is seen in Figure 5.3. The full datasheet may found in Appendix B.2. The data displays the input power required for the compressor to run at 30 Hz for varying evaporation and condensation temperatures. The same tables are given for compressor speeds of 60 and 90 Hz, for all output variables seen in Table 5.2. Figure 5.4 displays the plotted data. Using the thermodynamic model described in Section 5.1.1, the same graphs are obtained for condenser capacity, Q_{out}, COP and compressor exhaust temperature $T_{exhaust}$, shown in Figure 5.5.

Table 5.2: Input and output variables of compressor datasheet

Output variables	**Required input variables**	**Acceptable range of input variables**
Evaporator capacity (W)	Compressor frequency (Hz)	30 - 90
Input power (W)	Refrigerant condensing temperature (°C)	30 - 60
Mass flow of refrigerant (kg/s)	Refrigerant evaporating temperature (°C)	-10 - 15

Input Power(W)		**Evaporating Temp.(℃)**					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	530.1	574.3	607.2	627.5	634.0	625.6
	55.0	473.3	510.8	537.3	551.5	552.3	538.3
	50.0	425.7	456.6	476.8	485.0	480.1	460.7
	45.0	385.3	409.7	423.7	426.1	415.5	390.9
	40.0	350.3	368.3	376.3	372.9	356.9	327.0
	35.0	318.7	330.6	332.6	323.5	302.1	267.2
	30.0	288.9	294.5	290.7	276.1	249.5	209.7

Figure 5.3: Example of table in compressor datasheet, given for compressor speed of 30 Hz

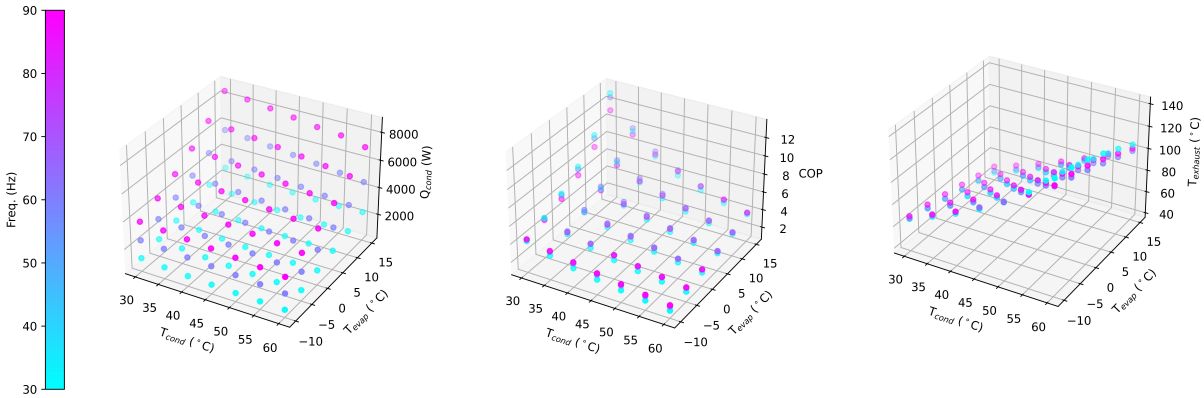


(a) Evaporator capacity, Q_{in} (W)

(b) Compressor work input, W_{in} (W)

(c) Refrigerant mass flow rate, $\dot{m}_{ref}$, (kg/s)

Figure 5.4: Defined outputs at varying refrigerant evaporation and condensation temperatures, T_{evap} and T_{cond}, (°C) and compressor speeds f, (Hz).



(a) Condenser capacity, Q_{out} (W)

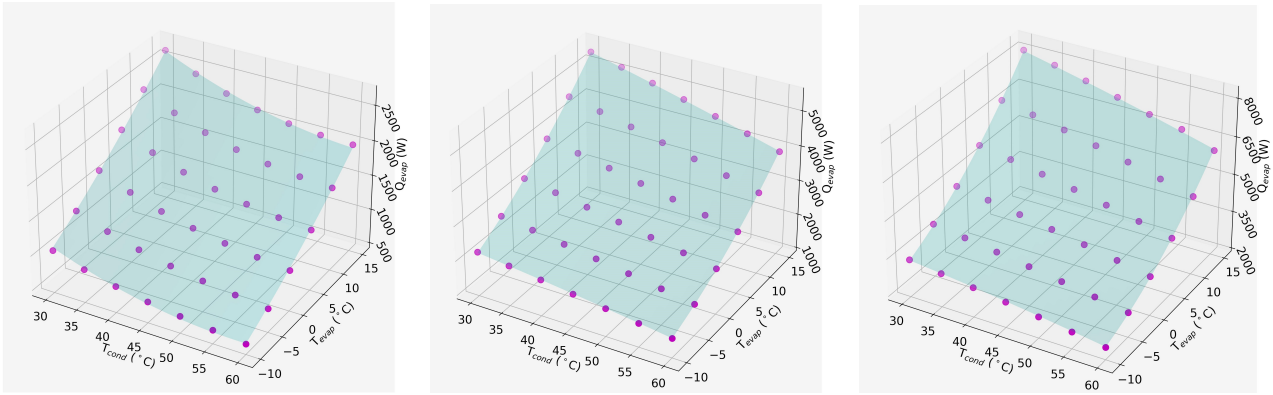
(b) COP

(c) Exhaust temperature at compressor outlet, $T_{exhaust}$ (°C)

Figure 5.5: Calculated outputs at varying refrigerant evaporation and condensation temperatures, T_{evap} and T_{cond}, (°C) and compressor speeds f, (Hz).

5.1.3. Interpolation

To fully define the HP model at any frequency, T_{evap} and T_{cond} combination, a method to interpolate to this combination must be defined. 3rd and 4th order surfaces are fit to the datasheet-defined (Q_{in}, W_{in}, $\dot{m}_{ref}$) and calculated (Q_{out}) output variables at each defined frequency (30, 60, 90 Hz). Equation coefficients may be found in Appendix B.2. Figure 5.6 displays an example of a fit surface and its real value for Q_{in}.



(a) 30 Hz (b) 60 Hz (c) 90 Hz

Figure 5.6: Example of fitted surface of Q_{in} at varying compressor speeds.

Thus, the numerical model is able to calculate an output for any T_{evap} and T_{cond}, given a compressor speed of 30, 60 or 90 Hz. To obtain the output for a frequency between these speeds, values for the output are calculated at the corresponding lower and upper frequency (f.e. the lower and upper frequencies for a frequency of 45 Hz would be 30 and 60 Hz respectively). The values are linearly interpolated to obtain the final output. Extrapolation is avoided by assuming that the HP will not operate outside of the allowable input ranges.

5.1.4. Heat exchanger model

The HP's T_{evap} and T_{cond} depend on inlet conditions on the water-side of the evaporator and condenser. Thus, the two HEXs must be modelled to obtain these temperatures and subsequently the water-side outlet temperatures. Note water-side refers to the side of the HEX that does not contain refrigerant.

5.1.4.1. Calculation method

The Number of Transfer Units (NTU) method is used to model the evaporator and condenser of the HP. NTU is given by

$$NTU = \frac{UA}{min(\dot{m}C_p)}, \tag{5.4}$$

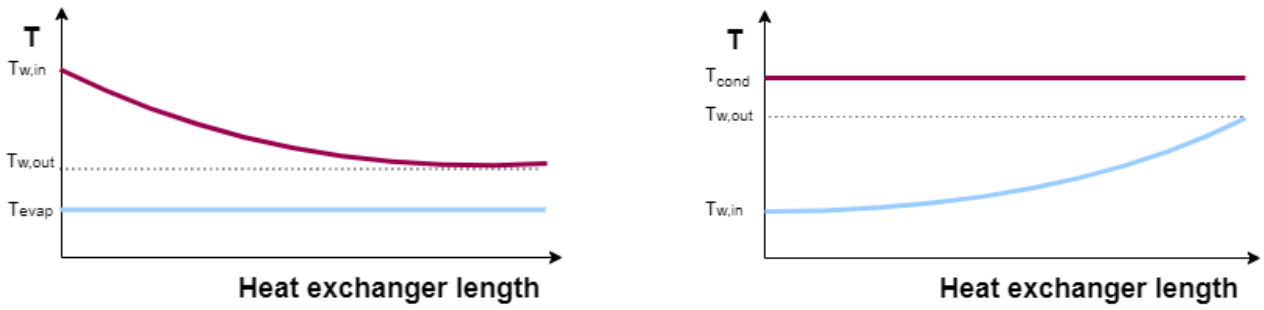
where U is the overall heat transfer coefficient in the HEX (W/m^2K), outlined in Section 5.1.4.2, A is total heat exchange area (m^2), $\dot{m}_w$ is the mass flow rate of heat exchanging fluid (kg/s) and C_p is its specific heat (J/kgK). The product $\dot{m}$ and C_p is calculated for both heat exchange fluids and the minimum value is chosen to calculate NTU. For a refrigerant undergoing phase change, the product is infinite. Thus, the NTU method is defined with the water-side properties and given by

$$NTU = \frac{UA}{\dot{m}_w C_{p,w}}, \tag{5.5}$$

Figure 5.7 displays the temperature profiles on the refrigerant and water sides in the evaporator and condenser. Thus, using the NTU method

$$T_{evap} = T_{in,evap} - \frac{T_{in,evap} - T_{out,evap}}{1 - e^{-NTU}}, \tag{5.6}$$

$$T_{cond} = T_{in,cond} + \frac{T_{out,cond} - T_{in,cond}}{1 - e^{-NTU}}. \tag{5.7}$$



(a) Evaporator

(b) Condenser

Figure 5.7: Temperature profiles over heat exchanger length.

However, there exists and interdependence between $T_{w,out}$ and T_{evap} or T_{cond}. $T_{w,out}$ cannot be known without knowing the heat removed or added by refrigerant, which depends on T_{evap} or T_{cond} respectively. Thus, a convergence loop is required. The following steps outline the procedure to calculate the temperatures:

Initially, a $T_{w,out}$ is assumed based on $T_{w,in}$. T_{evap} and T_{cond} are calculated using Equations 5.6-5.7 with their corresponding pairs of water temperatures. Having calculated the refrigerant temperatures, Q_{evap} and Q_{cond} are obtained as described in Sections 5.1.1-5.1.3. Using

$$Q_{out} = \dot{m}_{in,cond} C_{p,water} \left(T_{out,cond} - T_{in,cond}\right), \tag{5.8}$$

$$Q_{in} = \dot{m}_{in,evap} C_{p,glycol} \left(T_{in,evap} - T_{out,evap}\right), \tag{5.9}$$

new values for $T_{w,out}$ may be obtained. This process is repeated until the difference between newly calculated and previous $T_{w,out}$ values is negligible.

In the HP system, T_{evap} and T_{cond} are dependent on one another. Therefore, a 3rd and final convergence loop is used to ensure that their values converge to a feasible solution. The convergence loop stores the previously calculated values of T_{evap} and T_{cond}. It then re-runs the convergence loop used to calculate T_{evap} and T_{cond} until the differences between newly and previously calculated refrigerant temperatures are negligible.

5.1.4.2. Heat transfer coefficients

The NTU method described in Section 5.1.4.1 requires an overall heat transfer coefficient. Appendix B.2 displays the HEX dimensions used in these calculations.

Evaporator

The evaporator of the HP is a shell and tube HEX (with evaporation occuring on the tube-side). Thus, the overall heat transfer coefficient is given by

$$\frac{1}{U_{evap}} = \frac{1}{h_{conv,glycol}} + \frac{d_{outer}}{d_{inner}} \frac{1}{h_{conv,ref}} + \frac{d_{outer} ln\left(\frac{d_{outer}}{d_{inner}}\right)}{2k_{al}}, \tag{5.10}$$

where U_{evap} is the overall heat transfer coefficient (W/m^2K), $h_{conv,glycol}$ is the shell-side heat transfer coefficient (W/m^2K), $h_{conv,ref}$ is the tube-side heat transfer coefficient (W/m^2K), d_{outer} is the tube outer diameter (m), d_{inner} is the tube inner diameter and k_{al} is the thermal conductivity of aluminium, of which the HEX is made.

Tube-side heat transfer

Within the tube-side of the HEX, refrigerant is evaporating. The quality of the refrigerant varies throughout the length of the evaporator, thus does the heat transfer coefficient. To avoid an in-depth, computationally expensive simulation of two-phase flow in the evaporator, it is assumed that the heat transfer coefficient remains constant throughout the evaporator.

Note this is a very crude approximation. However, if the temperature at the outlet of the evaporator is well-predicted by the model, the approximation can be considered valid.

Empirical correlations are used to estimate heat transfer coefficient $h_{conv,ref}$. Sinnot and Towler [56] describe Chen's method in calculating the heat transfer due to forced convective boiling. It is assumed that the heat transfer coefficient is made up of two components, nucleate boiling and forced convection, given by

$$h_{conv,ref} = h'_{boiling} + h'_{conv} = f_s h_{boiling} + f_c h_{conv} \tag{5.11}$$

where the heat transfer coefficient due to forced convection, h'_{conv} may be estimated by using correlations for single phase flow with the multiplication of a factor, f_c to account for two phase flow. f_c is a function of the Lockhart-Martinelli parameter, given by

$$\frac{1}{X_{tt}} = \left(\frac{x}{1-x}\right)^{0.9} \left(\frac{\rho_L}{\rho_v}\right)^{0.5} \left(\frac{\mu_v}{\mu_l}\right)^{0.1}, \tag{5.12}$$

where X_{tt} is the Lockhart-Martinelli parameter, x is the quality of the fluid, ρ_L is the liquid density (kg/m^3), ρ_v is the vapor density (kg/m^3), μ_L is the liquid viscosity (Pas) and μ_v is the vapor viscosity (Pas). To avoid modelling two-phase flow, the quality is varied from 0 to 1. The mean value of X_{tt} is used for further calculations. Figure 5.8a displays the relationship between the X_{tt} and f_c.

The heat transfer coefficient in a tube, h'_{conv} is given by

$$\frac{h_{conv} d_{inner}}{k_{ref}} = j_h Re_{ref} Pr_{ref}^{0.33} \left(\frac{\mu_{ref}}{\mu_{ref,w}}\right)^{0.14}, \tag{5.13}$$

where d_{inner} is the inner tube diameter (m), k_{ref} is the thermal conductivity of refrigerant (W/mK), j_h is a factor accounting for laminar and turbulent flow in the tube, equal to 0.004 [56], Re_{ref} is the Reynolds number, Pr_{ref} is the Prandtl number and $\frac{\mu_{ref}}{\mu_{ref,w}}$ is the ratio between mean viscosity and viscosity at the tube wall, assumed to be equal to 1 due to lack of complexity in HEX model.

The heat transfer coefficient due to boiling, $h'_{boiling}$ is estimated using correlations for nucleate boiling and is multiplied by a suppression factor, f_s which accounts for the decrease in nucleate boiling in a flowing liquid. f_s is a function of f_c and the liquid Reynolds number, given by

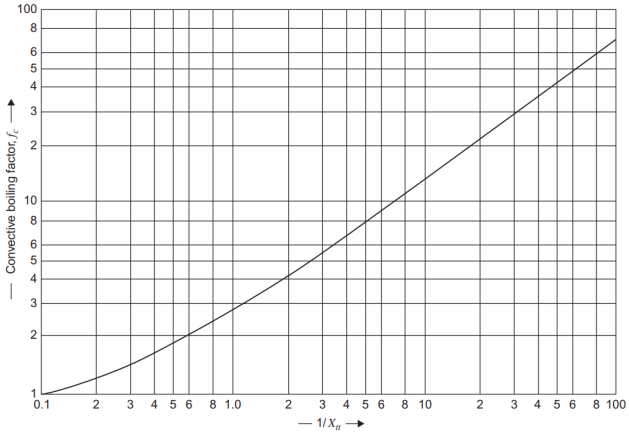
$$Re_{ref,L} = \frac{(1-x) G d_{inner}}{\mu_{ref,l}}, \tag{5.14}$$

where G is the mass flow per unit area (kg/sm^2). Figure 5.8b displays the relationship between f_s, f_c and $Re_{ref,L}$.

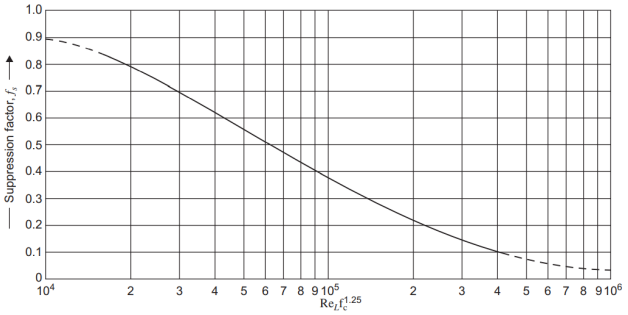
The heat transfer coefficient due to nucleate boiling, $h_{boiling}$ is described by the Mostinski correlation [56]:

$$h_{boiling} = 0.104 (P_c)^{0.69} (h_{boiling} (T_w - T_s))^{0.7} \left(1.8 \left(\frac{P}{P_c}\right)^{0.17} + 4 \left(\frac{P}{P_c}\right)^{1.2} + 10 \left(\frac{P}{P_c}\right)^{10}\right), \tag{5.15}$$

where P_c is the critical pressure of refrigerant (bar), P is the pressure of the refrigerant (bar), T_w is the wall temperature (K), assumed to be equal to fluid temperature in mode and T_s is saturation temperature of refrigerant (K).



(a) f_c as a function of the Lockhart-Martinelli parameter [56]



(b) Suppression factor as a function of Reynolds number and f_c [56]

Shell-side heat transfer

The heat transfer coefficient on the HEX's shell side, $h_{conv,glycol}$ is defined by

$$\frac{h_{conv,glycol} d_e}{k_{glycol}} = j_h Re_{glycol} Pr_{glycol}^{0.33} \frac{\mu_{glycol}}{\mu_{glycol,w}}, \tag{5.16}$$

where d_e is equivalent diameter (m), $\frac{\mu_{glycol}}{\mu_{glycol,w}}$ is assumed to be equal to 1 due to lack of complexity in the HEX model and j_h is the heat transfer factor.

j_h is a function of Reynolds number and HEX geometry. This relationship is presented in Figure 5.9. Note baffle cut is not a known quantity, therefore an average value is taken.

The Reynolds number is defined as

$$Re_{glycol} = \frac{\rho_{glycol} v_{glycol} d_e}{\mu_{glycol}}, \tag{5.17}$$

where v_{glycol} is the velocity (m/s) and is defined by

$$v_{glycol} = \frac{\dot{m}_{glycol}}{\rho_{glycol} A_e}, \tag{5.18}$$

where $\dot{m}_{glycol}$ is the mass flow (kg/s) and A_e is the cross-flow area (m^2), defined as

$$A_e = \frac{\pi}{4} d_e^2. \tag{5.19}$$

Equivalent diameter d_e is defined as

$$d_e = \frac{(d_{shell}^2 - d_{tube}^2)}{(d_{shell} - d_{tube})}, \tag{5.20}$$

d_{shell} is the HEX's inner diameter (m) and d_{tube} is the HEX's tube bundle diameter (m).

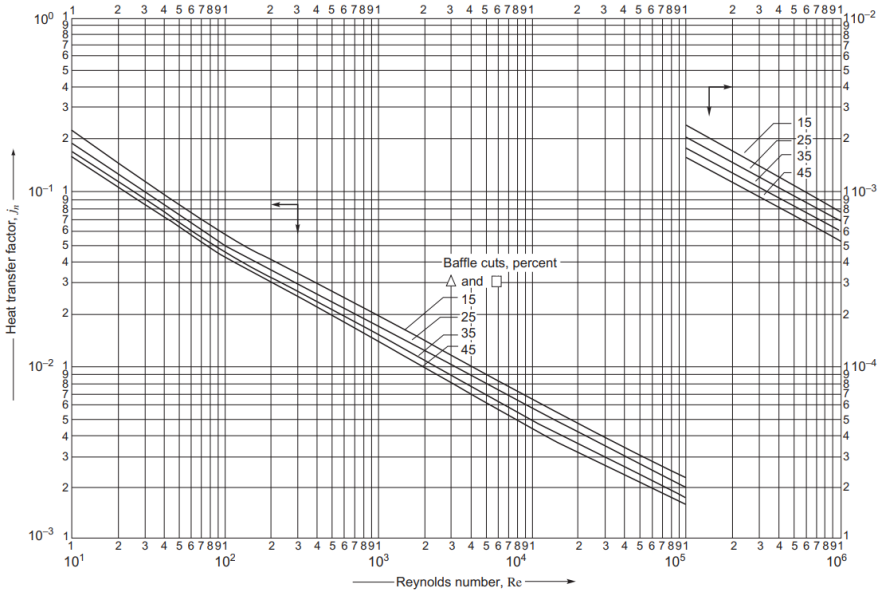


Figure 5.9: Heat transfer factor as a function of Reynolds number [56]

Condenser

As described in Section 5.1.4.1, an overall heat transfer coefficient is required for the condenser. The condenser of the HP is a plate HEX. Thus, the overall heat transfer coefficient, U_{cond} is given by

$$\frac{1}{U_{cond}} = \frac{1}{h_{ref}} + \frac{1}{h_{water}} + \frac{t_p}{k_{copper}}, \tag{5.21}$$

where h_{ref} is the refrigerant heat transfer coefficient (W/m^{2}K), h_{water} is the water heat transfer coefficient (W/m^{2}K), t_p is the plate thickness, k_{copper} is the thermal conductivity of copper of which the plates are made.

The water heat transfer coefficient is calculated using the following empirical relation:

$$\frac{h_{water} d_e}{k_{water}} = 0.26 Re_{water}^{0.65} Pr_{water}^{0.4} \left(\frac{\mu_{water}}{\mu_{w,water}}\right), \tag{5.22}$$

where $\frac{\mu_{water}}{\mu_{w,water}}$ is again assumed to be equal to 1.

Equivalent diameter is given by

$$d_e = 2s, \tag{5.23}$$

where s is the spacing between two plates (m).

Reynolds number is calculated using Equations 5.17 and 5.18 with water properties. Furthermore, cross sectional area in Equation 5.18 is the product of plate spacing s and plate width w_p.

Refrigerant in the condenser undergoes phase change. Thus, a different correlation is required to quantify its heat transfer coefficient, h_{ref}.

Cascales et al. [27] define an empirical correlation for h_{ref} as

$$h_{ref} = 0.023 Re_{L,ref}^{0.8} Pr_{L,ref}^{0.4} \frac{k_{L,ref}}{d_e} \left((1-x)^{0.8} + \frac{3.8 x^{0.76} (1-x)^{0.04}}{\left(\frac{P}{P_{cr}}\right)^{0.38}} \right), \tag{5.24}$$

Where Re_L is defined as in Equation 6.9, Pr_L is the liquid Prandtl number, k_L is the refrigerant's liquid thermal conductivity (W/mK), d_e is defined as in Equation 5.23. Quality x is varied from 0 to 1 in steps of 0.1. The mean value is calculated and taken to be h_{ref}.

5.2. Heat pump control

As described in the chapter thus far, with an inlet water-side evaporator and condenser temperature and mass flow rate, the HP is able to calculate the evaporator and condenser capacities and refrigerant and water-side outlet temperatures.

However, three control methods must be defined. Firstly, the HP must choose which source to draw heat from, thus defining the inlet temperature and mass flow on the water-side of the evaporator. This is done by choosing the highest temperature source. The control checks whether the temperature in the wastewater bag or the outlet of the air source (the calculations of which are defined in Sections 7.1.1 and 6.1.1 respectively). The highest temperature circuit is chosen to pass through the evaporator.

A control must also be defined for the condenser side of the HP. Here, a choice between providing DHW or SH is made. The control prioritises DHW. Thus, if a DHW demand is logged (as described in Section 10.1), the HP switches to DHW mode. Then, the spiral heater circuit (as described in Section 9.2), is chosen to pass through the water side of the condenser. If there is no DHW demand, but a SH demand is logged (as described in Section 10.3), the circuit is switched to the radiator circuit (as described in Section 9.3).

Finally, a control must be defined for the HP to choose a frequency to modulate to. The current HeatCycle HP control is implemented. Frequency is chosen based on the source temperature entering the water side of the evaporator. Table 5.3 displays the control.

Table 5.3: Heat pump frequency control

Source temperature (°C)	Frequency (Hz)
$\geq$ **30**	50
$\geq$ **20**	54
$\geq$ **12**	65
$\geq$ **4**	76
4 $\geq$	80

The HP has a safety protocol when its compressor outlet temperature becomes too high. Depending on the logged temperature, the frequency is either decreased or the HP is switched off. This control is displayed in Table 5.4

Table 5.4: Heat pump safety control

Exhaust temperature (°C)	Frequency (Hz)
$\geq$ **105**	0 for 4 min
97 $\geq$	f - 15
92 $\geq$	f - 7.5

5.3. Experimental validation

The two primary variables of interest in the HP are the outlet evaporator and condenser temperatures, $T_{evap,out}$ and $T_{cond,out}$, as they are the outputs of the model. Thus, these temperatures must be experimentally validated.

5.3.1. Methodology

Data was obtained from a previous experiment at DeWarmte. The experimental setup used was DeWarmte's HeatCycle setup, seen in Figure 7.6, with components as in Table 7.2, with additional temperature sensors placed at the inlet and outlet

of the condenser. A flow rate sensor was placed at the outlet of the condenser. The HP was run twice for each frequency of 30, 60 and 90 Hz. Between repeating experiments, the HP was turned off and the bag was refilled and allowed to heat up to room temperature, which fluctuated between 21-24 °C.

Having obtained the experimental data, $T_{evap,in}$, $T_{cond,in}$, $f_{evap,in}$, $f_{cond,in}$, and the corresponding frequencies of the experiment are used as inputs in the HP model. $T_{evap,out}$ and $T_{cond,out}$ are calculated and compared to their experimental counterparts.

5.3.2. Results

Figure 5.10 displays the measured evaporator and condenser temperatures, alongside the calculated evaporator and condenser outlet temperatures for all experiments. From initial observation, the numerical outlet temperatures follow closely the experimental outlet temperatures. A more in-depth analysis is presented in Section 5.3.2.1.

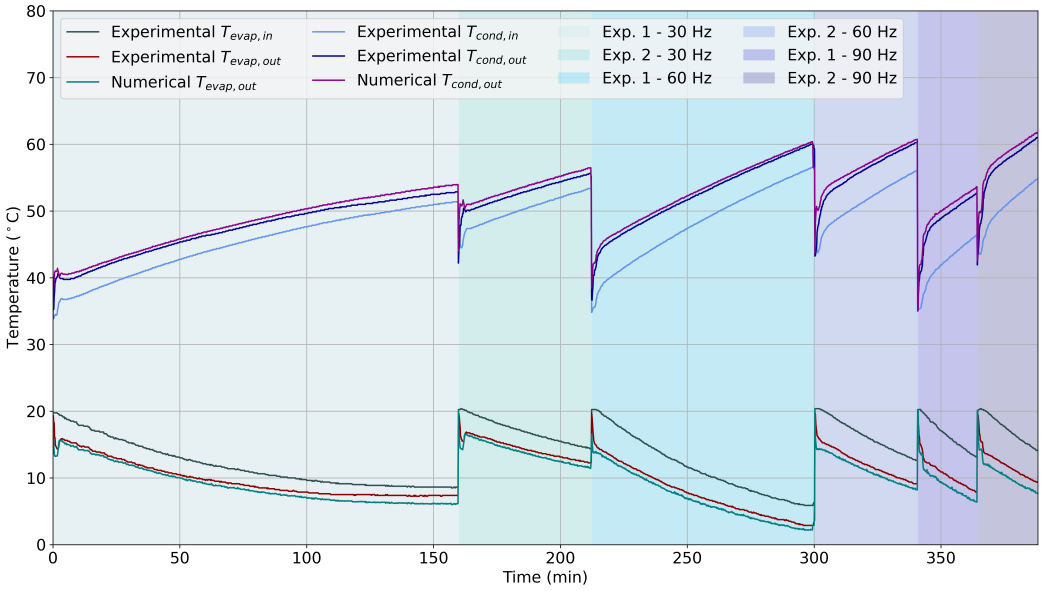


Figure 5.10: Measured and calculated evaporator and condenser temperatures at 30, 60 and 90 Hz

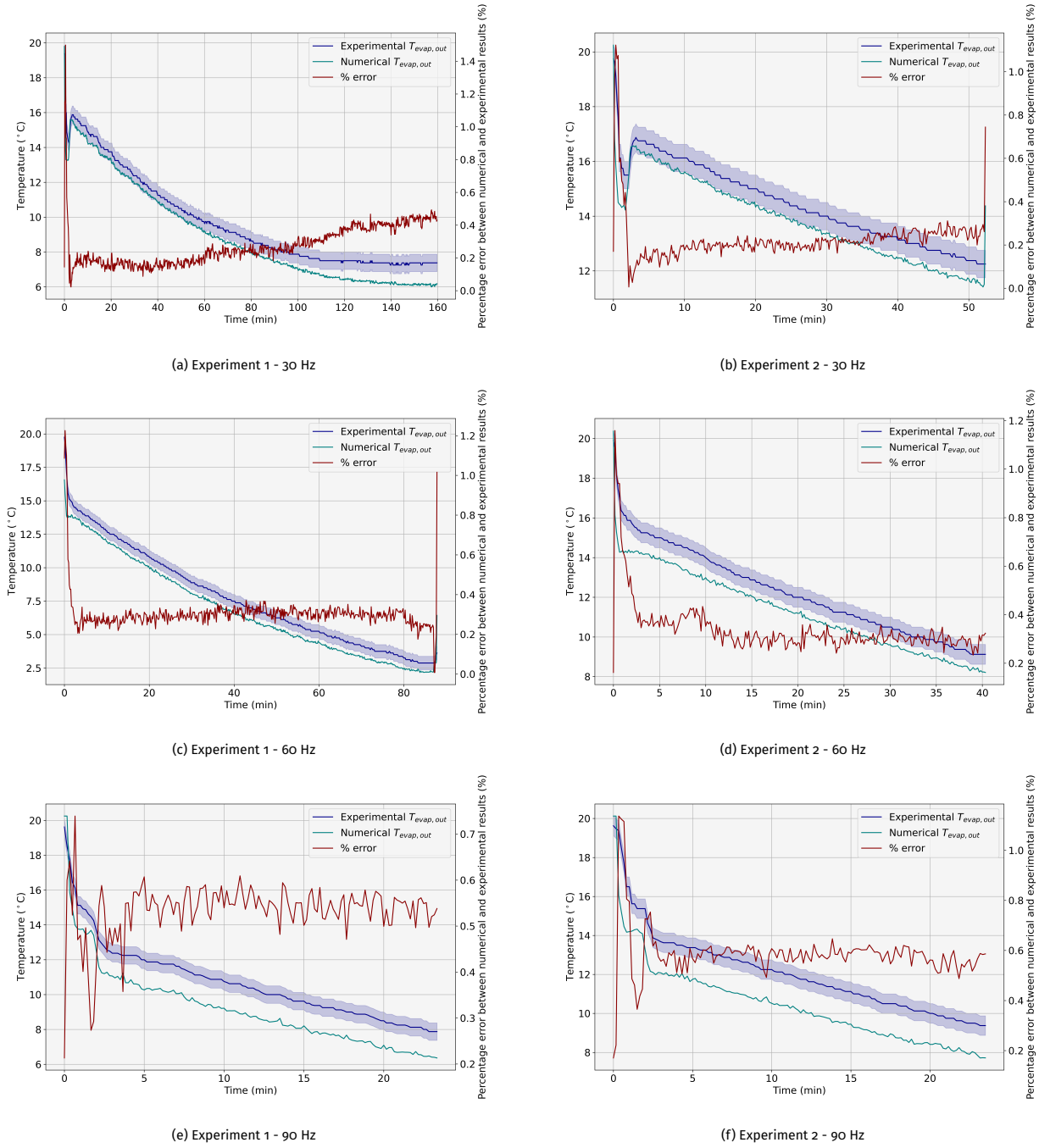
5.3.2.1. Error analysis

Figure 5.11 displays the comparison between experimental and numerical evaporator outlet temperature, alongside the percentage error between the two. Note the experimental outlet temperature is plotted with its sensor error of ±0.5 °C.

Experiments conducted at 30 and 60 Hz display similar error. The numerical temperature is underestimated by less than 1 °C. During experiment 1 at a frequency of 30 Hz, the error between the two values increases as the experiment progresses. The reason for this is unclear. This behaviour is not seen in the other experiments and may therefore be considered an outlier.

Experiments conducted at 90 Hz display a significantly larger error, with evaporator outlet temperature being underestimated by approximately 1.5 °C, meaning the heat transfer in the evaporator is over-predicted. The most likely reason for this is that the heat transfer coefficient is over-predicted at the higher refrigerant flow rate due to the high compressor frequency.

To conclude, the HP model predicts the temperature at the outlet of the evaporator sufficiently well. However, although the discrepancy is small at 0.5 - 1.5 °C, it overestimates the performance of the evaporator and therefore the amount of heat extracted from the source. This will slightly overestimate the evaporator's performance.



(a) Experiment 1 - 30 Hz

(b) Experiment 2 - 30 Hz

(c) Experiment 1 - 60 Hz

(d) Experiment 2 - 60 Hz

(e) Experiment 1 - 90 Hz

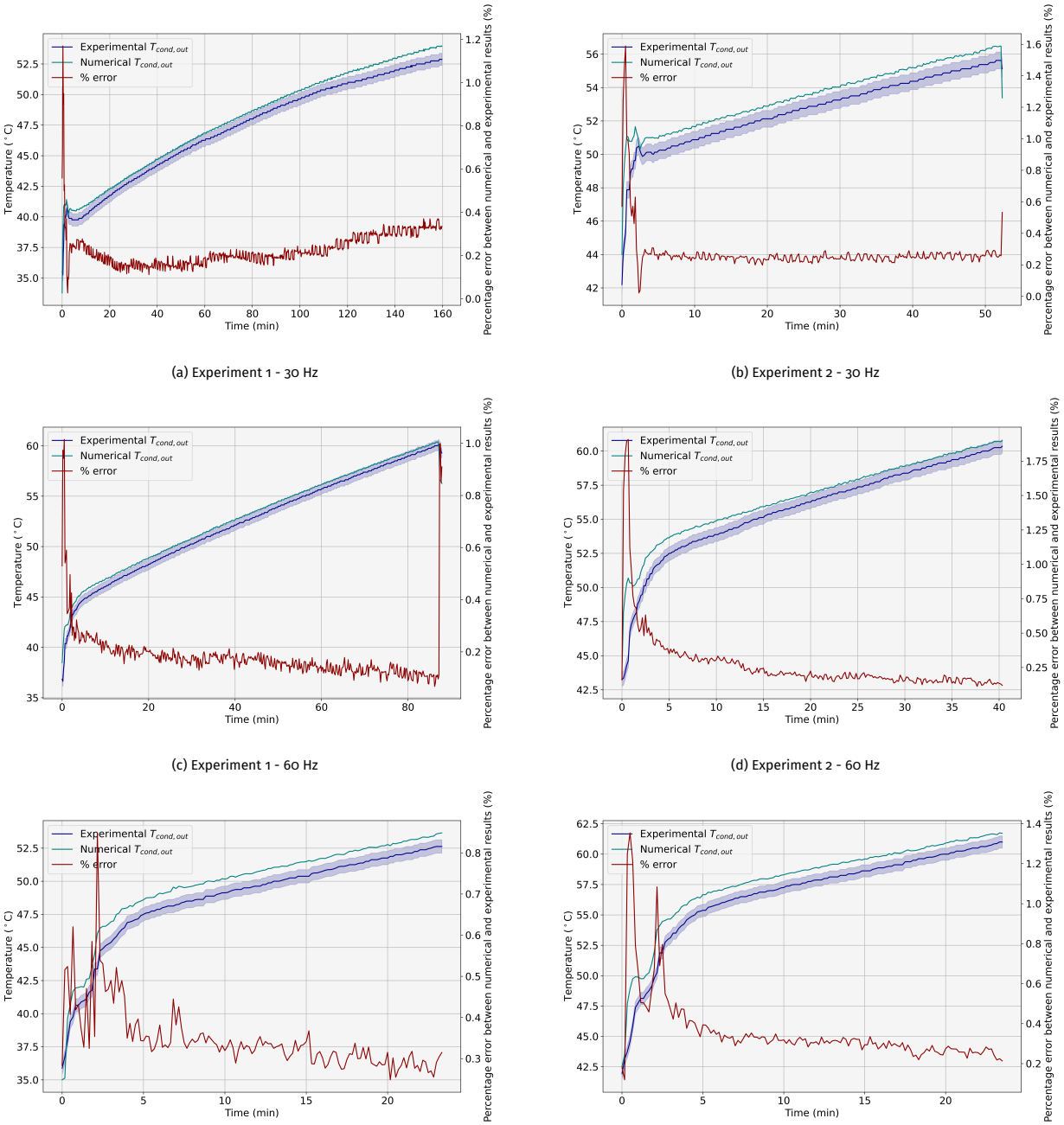
(f) Experiment 2 - 90 Hz

Figure 5.11: Comparison between experimental and numerical evaporator outlet temperature, $T_{evap,out}$ at 30, 60 and 90 Hz

Figure 5.12 displays the comparison between experimental and numerical outlet temperatures of the condenser for each experiment at frequencies 30, 60 and 90 Hz. Note the experimental outlet temperature is plotted with its sensor error of ±0.5 °C.

Similarly to that of the evaporator outlet temperature, condenser outlet temperature is well-predicted for a frequency of 30 and 60 Hz. The error is lower, at less than 1 °C. In experiment 1 of 30 Hz, a hysteresis is again seen as the experiment progresses. Again, the reason for this is unknown. However, as it is not seen in the other experiments, it is considered an outlier.

The error is larger for a frequency of 90 Hz than that of 30 and 60 Hz, at a difference of approximately 1 °C. This is smaller than that of the evaporator at 90 Hz. Again, this is theorised to be due to the heat transfer coefficient being overestimated at a higher flow rate.



(a) Experiment 1 - 30 Hz

(b) Experiment 2 - 30 Hz

(c) Experiment 1 - 60 Hz

(d) Experiment 2 - 60 Hz

(e) Experiment 1 - 90 Hz

(f) Experiment 2 - 90 Hz

Figure 5.12: Comparison between experimental and numerical condenser outlet temperature, $T_{cond,out}$ at 30, 60 and 90 Hz

The HP model predicts condenser performance better than evaporator performance. The condenser is a plate HEX, while the evaporator is a shell and tube HEX. Therefore, this could be due to the heat transfer coefficient correlation being better for the condenser than that of the evaporator. The well-predicted condenser temperature means the supply side of the performance model will be modelled correctly. The evaporator performance is slightly overestimated, meaning the model may predict that more heat is extracted from the source than is in reality. For the water bag, this effect may not be as consequential, as the source is limited. Therefore, the time taken to extract heat may be underestimated in the model, but the amount of heat extracted will remain the same as in reality. However, the air source is near-infinite, meaning the model will consistently predict more heat extracted from the source. Nevertheless, with relatively small discrepancies, the HP model is considered experimentally validated.

5.3.2.2. Sobol's variance-based sensitivity analysis

A Sobol analysis is a variance-based sensitivity analysis. Given a set of inputs and their variance, it determines their contribution to the variance of a given output. This contribution is reflected by Sobol indices. A Sobol index is calculated for each tested input. The sum of indices for each input must equal to 1. Thus, a Sobol index represents the fraction by which variance in the input causes variance in the output. Consequently, it gives insight on which inputs the model is most sensitive to.

A brief introduction to a Sobol sensitivity analysis will be presented. For a deeper insight into the method and its governing equations, refer to the paper published by I.M. Sobol [58]. There is presented the derivation of a function's output's variance and its decomposition into a fraction of each of its inputs. Using Monte Carlo sampling, the process is repeated for a user-set amount of randomized input sets. A higher number of sets results in smaller confidence intervals of the indices, resulting in very high computational cost. The possible bounds of values for each input is also set by the user. In the case of an experimental validation of a numerical model, these bounds are sensor error. The analysis is able to quantify both first order and second order effects of the inputs. Thus, the final result of the sensitivity analysis is a first order, second order and total Sobol index and corresponding confidence intervals for each input tested. In this project, only the total Sobol indices will be presented. The analysis was done using the SALib Python library.

This method is utilised for the validation of the HP model to gain insight on its calculation method and acts as a sanity check. Furthermore, it is used as a tool to critically assess whether the experimental design and the used sensors were satisfactory in its validation.

The tested output is the water-side outlet condenser temperature, $T_{out,cond}$, as it is considered the most important output of the model, quantifying the amount of heat the HP is able to provide. It's value depends on the following inputs: water-side evaporator inlet temperature $T_{in,evap}$, water-side condenser inlet temperature $T_{in,\,cond}$, water-side evaporator mass flow $\dot{m}_{evap}$ and water-side condenser mass flow $\dot{m}_{cond}$. A Sobol analysis was conducted for each experiment to test the sensitivity of the model at varying conditions. The bounds for the inputs were the sensor error from the test point, given in Table 6.2. 128 random samples were used to calculate Sobol indices at each experimental test point. This value is on the low side, resulting in larger confidence intervals. It was chosen due to very high computational costs of the method.

Figure 5.13 displays the results of the sensitivity analysis. Throughout all experiments, $T_{in,cond}$ remains the variable with the highest index, followed by $\dot{m}_{cond}$. $T_{in,evap}$ and $\dot{m}_{evap}$ have little-to-no influence on the output's variance. The index of $T_{in,cond}$ is the highest at experiments conducted at 30 Hz, decreasing for the experiments conducted at 60 and 90 Hz. With the decrease, the index of $\dot{m}_{cond}$ increases. This result coincides with the theoretical modelling of the HP. At lower compressor frequencies, and therefore lower mass flow rates, heat transfer due to convection is lower. Thus, the temperature at the outlet of the condenser is more dependent on its inlet temperature. At higher frequencies, the heat transfer increases, thus its dependence on mass flow will increase. Furthermore, as each experiment progresses, the dependence on $T_{in,cond}$ increases, while the dependence on $\dot{m}_{cond}$ decreases. This may be explained as follows: as the condenser inlet temperature increases throughout an experiment, the ratio of inlet-to-outlet temperature increases, while the ratio of an added ΔT due to convective heat transfer-to-outlet temperature decreases. Finally, the condenser outlet temperature has no dependence on the evaporator's inlet conditions. With a set frequency, as was done in this experiment, this result is obvious. However, due to the presence of the convergence loop linking evaporator and condenser conditions when calculating their capacities, as described in Section 5.1.4, a relationship may exist. The sensitivity analysis concludes that this relationship is near-negligible.

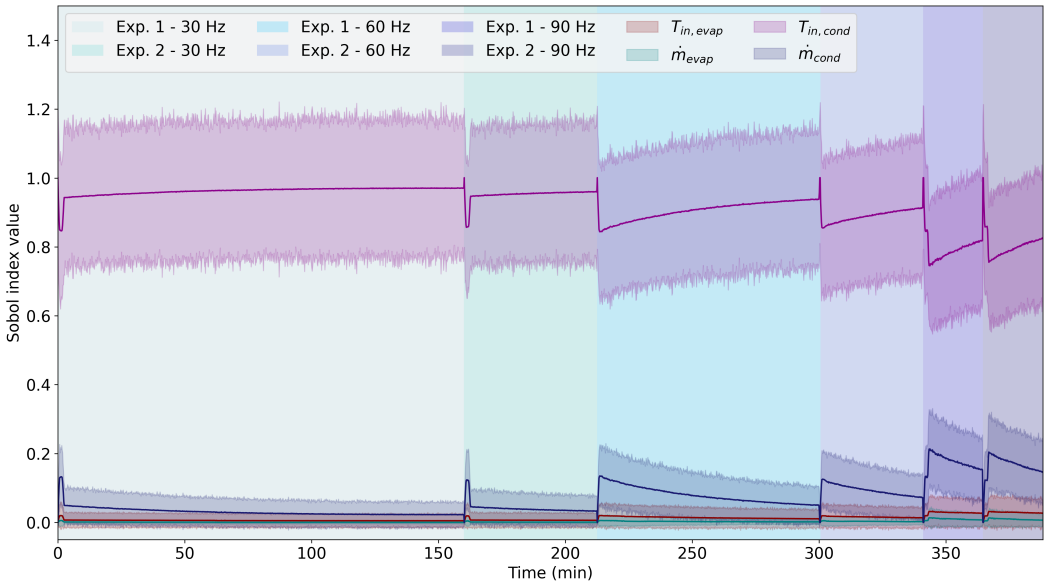


Figure 5.13: Sobol analysis of heat pump model

6

Air-source model

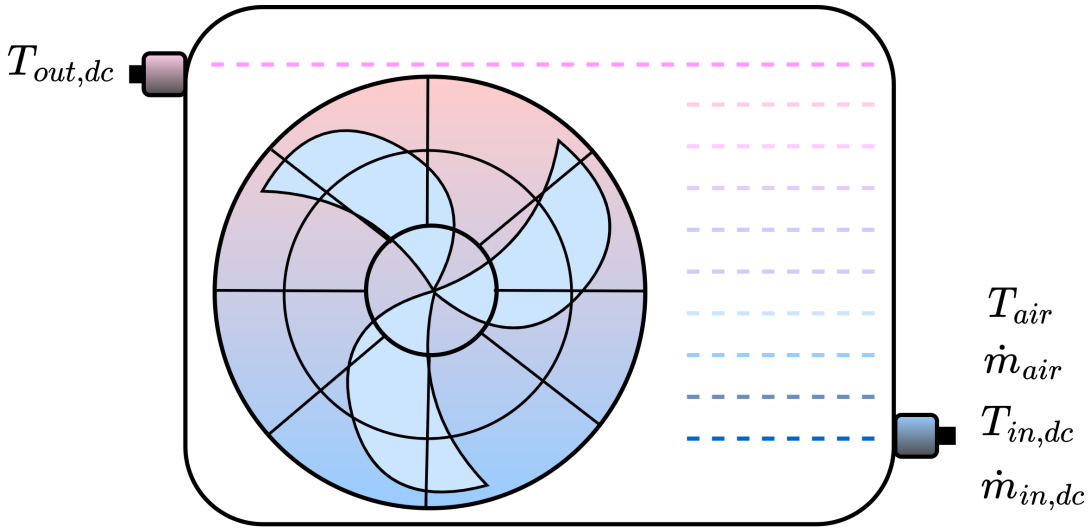


Figure 6.1: Dry cooler class

Table 6.1: Inputs and outputs of dry cooler class

Variable	Unit	Type	Description	Source
$T_{air,}$	K	Input	Temperature of ambient air	Weather data class
$\dot{m}_{air}$	kg/s	Input	Mass flow rate of ambient air	User input
$T_{in,dc}$	K	Input	Temperature of incoming glycol	HP class
$\dot{m}_{in,dc}$	kg/s	Input	Mass flow of incoming glycol	-
$T_{out,dc}$	K	Output	Temperature of outgoing glycol	-

6.1. Numerical modelling approach

The air-source being modelled is a dry cooler, consisting of a fin-tube HEX and fan. The inflow conditions to the dry cooler are known from the HP evaporator outflow conditions. Thus, the numerical model must contain a calculation method for obtaining the outflow conditions from the dry cooler, namely the fluid temperature at its exit.

6.1.1. Heat exchanger model

A control volume (CV) approach is taken to model the finned tube HEX. In the model, its length is split into 100 CVs. 100 CVs

has been found to be the minimum value to sufficiently approximate the temperature at the outlet of the dry cooler. It is assumed that within one CV, the temperature remains homogeneous. For each CV, the previous CV's exit temperature is taken as its inlet temperature. Heat flux from the air and subsequently the new CV temperature is calculated, acting as the input for the next CV. Figure 6.2 shows a visual representation of this method.

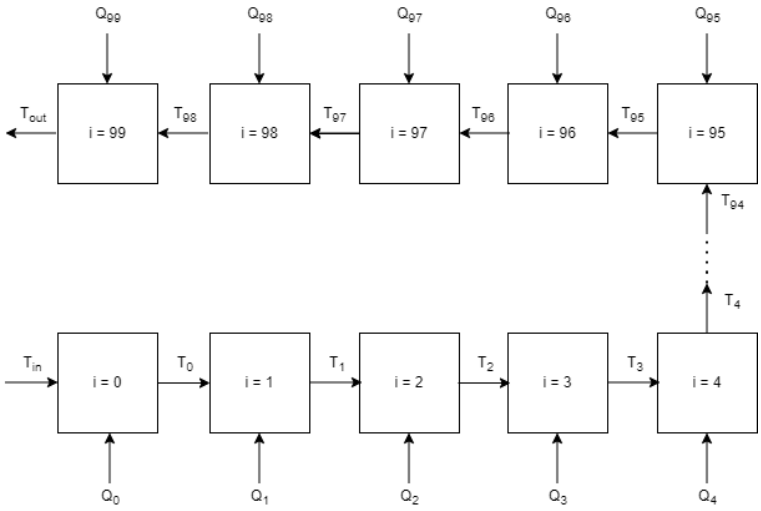


Figure 6.2: Control volume method used in numerical model of dry cooler

The method to calculate each CV's temperature is as follows; initially, T_i leaving CV *i* is assumed based on the value of T_{i-1}, the temperature entering the CV. Based on these values, the heat flux to a CV, Q_i (W) to the CV is calculated using

$$Q_i = \dot{m}_{in,dc} C_{p,glycol} \left(T_i - T_{i-1}\right), \tag{6.1}$$

where $\dot{m}_{in,dc}$ is the mass flow of glycol solution flowing in the dry cooler (kg/s), $C_{p,glycol}$ is the specific heat of glycol solution (J/kgK).

The newly calculated heat flux corresponds to the heat added to the water in CV *i* and therefore the heat removed from the air flowing over CV *i*. Therefore, the temperature of air leaving the dry cooler is calculated with

$$T_{air,i} = T_{air} - \frac{Q_i}{\dot{m}_{air} C_{p,air}}, \tag{6.2}$$

where $T_{air,i}$ corresponds to air temperature leaving the dry cooler after CV *i* (K), T_{air} is the ambient air temperature, assumed to be homogeneous over the whole HEX (K), $\dot{m}_a$ is the mass flow rate of air over the drycooler (kg/s), $C_{p,a}$ is the specific heat of air (J/kgK).

With all temperatures being defined, the logarithmic mean temperature difference, ΔT_{lm} may be calculated with

$$\Delta T_{lm} = \frac{(T_{air} - T_i) - (T_{air,i} - T_{i-1})}{In\left(\frac{T_{air} - T_i}{T_{air,i} - T_{i-1}}\right)}. \tag{6.3}$$

A new heat flux value is calculated, using

$$Q_i = U A_i \Delta T_{lm}, \tag{6.4}$$

where U is the HEX's overall heat transfer coefficient (W/m²K) (described in Section 6.1.2), A_i is the heat transfer area of CV *i* (m²).

Having obtained a new value of heat flux Q, temperature leaving CV *i* is recalculated with

$$T_i = T_{i-1} + \frac{Q}{\dot{m}_{in,dc} C_{p,glycol}} \tag{6.5}$$

This value is compared with previously calculated T_i from Equation 6.1. If the difference between the two is not sufficiently small, the initially assumed T_i is set to its newly calculated counterpart. A convergence loop is run until the difference between the two is negligible. The process is repeated for each CV until the output temperature leaving the dry cooler is obtained.

6.1.2. Heat transfer coefficient

An overall heat transfer coefficient, U is used to approximate the heat transfer to the dry cooler's HEX. It is calculated using

$$\frac{1}{U} = \frac{d_{outer}}{d_{inner}} F_{glycol} + \frac{d_{outer}}{d_{inner}} h_{conv,glycol} + \frac{1}{F_{air}} + \frac{1}{h_{conv,air}} + \frac{d_{outer} ln\left(\frac{d_{outer}}{d_{inner}}\right)}{2k_{copper}}, \tag{6.6}$$

where U is the overall heat transfer coefficient (W/m^2K), d_{outer} is the outer tube diameter of the HEX (m), d_{inner} is the inner tube diameter of the HEX (m), F_{glycol} is the fouling factor of glycol (m^2K/W), F_{air} is the fouling factor of air (m^2K/W) , k_{copper} is the thermal conductivity of copper (W/mK) of which the HEX is made, $h_{conv,\ glycol}$ is the convective heat transfer coefficient due to glycol circulation (W/m^2K) and $h_{conv,\ air}$ is the convective heat transfer coefficient due to air flow over the HEX (W/m^2K).

When frost accumulates on the HEX, an additional thermal resistance is added to the calculation of the overall heat transfer coefficient. Then, the expression becomes

$$\frac{1}{U} = \frac{d_{outer}}{d_{inner}} F_{glycol} + \frac{d_{outer}}{d_{inner}} h_{conv,glycol} + \frac{1}{F_{air}} + \frac{1}{h_{conv,air}} + \frac{d_{outer} ln\left(\frac{d_{outer}}{d_{inner}}\right)}{2k_{copper}} + \frac{(d_{outer} + t_{frost}) ln\left(\frac{d_{outer}+t_{frost}}{d_{outer}}\right)}{2k_{copper}}, \tag{6.7}$$

where k_{frost} is the thermal conductivity of frost (W/mK) and t_{frost} is the buildup of frost (m).
The convective heat transfer coefficients must be calculated separately using empirical correlations for finned HEXs.

6.1.2.1. Convective heat transfer due to glycol circulation

The tube-side heat transfer coefficient is calculated using [56]

$$\frac{h_{conv,glycol} d_{inner}}{k_{glycol}} = j_h Re_{air} Pr_{air}^{0.33} \left(\frac{\mu_{glycol}}{\mu_{glycol,w}}\right)^{0.14}, \tag{6.8}$$

where k_{glycol} is the thermal conductivity of glycol solution (W/mK), j_h is a factor accounting for laminar and turbulent flow in the tube, equal to 0.004 [56], Re is the Reynolds number, Pr is the Prandtl number and $\frac{\mu_{glycol}}{\mu_{glycol,w}}$ is the ratio between mean viscosity and viscosity at the tube wall, assumed to be equal to 1 due to the assumption of homogeneous temperature throughout one CV.

Reynolds number is calculated using

$$Re = \frac{v_{glycol} d_{inner}}{v_{glycol}}, \tag{6.9}$$

where v_{glycol} is the flow velocity of glycol (m/s) and v_{glycol} is the kinematic viscosity of glycol (m^2/s).

Fluid properties are obtained from REFPROP 10.0, for a predefined ethelyne-glycol solution. Mean values of properties between -10 °C to 10 °C are used.

6.1.2.2. Convective heat transfer due to air flow

The fin-side heat transfer coefficient is given by

$$h_{conv,air} = h_{conv,shell}\left(1 - (1 - \eta_{fin})\frac{A_{fin}}{A}\right), \tag{6.10}$$

where $h_{conv,shell}$ is the heat transfer coefficient neglecting the effect of the fins, η_{fin} is the fin efficiency, $\frac{A_{fin}}{A}$ is the ratio of fin area to heat transfer area.

$\frac{A_{fin}}{A}$ is given by

$$\frac{A_{fin}}{A} = \frac{2\frac{\pi}{4}(d_{fin}^2 - d_{outer}^2)}{\pi d_{outer}(s - t_{fin}) + 2\frac{\pi}{4}(d_{fin}^2 - d_{outer}^2)}, \tag{6.11}$$

where d_{fin} is the diameter of the fin (m), s is fin spacing (m) and t_{fin} is fin thickness (m).

Fin efficiency, η_{fin} is defined as the ratio of heat removed by the fin over the heat removed by the fin if it was at its root temperature. It is given by

$$\eta_{fin} = \frac{tanhX}{X}, \tag{6.12}$$

where X is given by

$$X = \phi \frac{d_{outer}}{2} \sqrt{\frac{2h_{conv,shell}}{k_{fin} t_{fin}}}, \tag{6.13}$$

where k_{fin} is the thermal conductivity of the fin material (W/mK) and ϕ is given by

$$\phi = \left(\frac{d_{fin}}{d_{outer}} - 1\right)\left(1 + 0.35In\left(\frac{d_{fin}}{d_{outer}}\right)\right). \tag{6.14}$$

Finally, $h_{conv,shell}$ is given by

$$\frac{h_{conv,shell} d_{outer}}{k_{air}} = 0.38 Re_{air}^{0.6} \left(\frac{A}{A_{to}}\right)^{-0.15} Pr_{air}^{0.33}, \tag{6.15}$$

where k_{air} is the thermal conductivity of air (W/mK), Re is the Reynolds number, given by

$$Re = \frac{v_{air} d_{outer}}{v_{air}}, \tag{6.16}$$

and ratio $\frac{A}{A_{to}}$ is given by

$$\frac{A}{A_{to}} = 1 + 2\frac{h_{fin}(h_{fin} + d_{outer} + \delta_{fin})}{s d_{outer}}, \tag{6.17}$$

where h_{fin} is the height of the fin.

Fluid properties are computed using REFPROP 10.0, using predefined mixture air. Mean values for properties between -10 °C and 10 °C are used. Appendix B.3 displays the HEX dimensions used in the calculation for the air heat transfer coefficient.

6.2. Frost growth model

To include the change in thermal resistance and time taken for fully frosting over the HEX, a frost growth model was developed.

6.2.1. Numerical modelling approach

From literature it may be concluded that ambient air temperature, its relative humidity and the time frosting conditions have been met are primary variables affecting frost thickness and its growth rate. A study by Nordsyn [50] investigated the disagreement between labelled performance and field test performance in air-to-water HPs in cold, humid climates. It was concluded that the difference between performance indicators was due to increased frost growth and therefore HP defrosting time. The study quantified the frost thickness as a function of time, ambient air temperature and relative humidity from a dataset provided by Haugerud et al [31]. This dataset is used as the basis for frost growth simulation in this numerical model. The graphs presented in the study are shown in Figures 6.3-6.4.

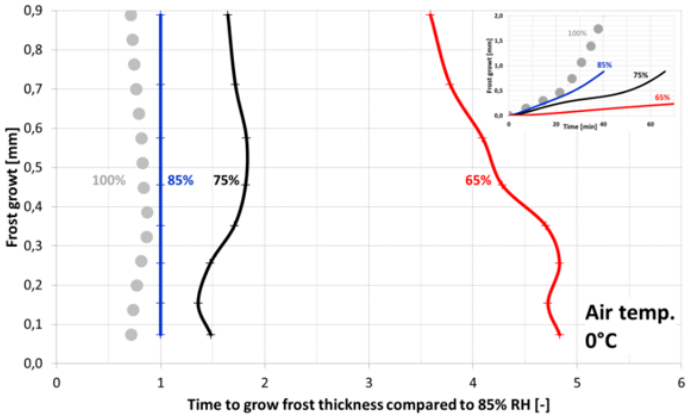


Figure 6.3: Frost thickness at varying relative humidities as a function of time, given for an ambient air temperature of 0 °C, [50] with dataset from [31]

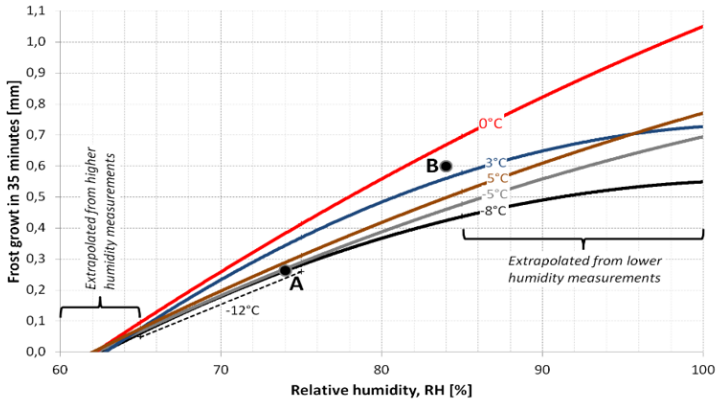


Figure 6.4: Frost thickness at 35 minutes at varying relative humidities and ambient air temperatures, [50] with dataset from [31]

The graph in the upper right corner of Figure 6.3 displays the variation of frost thickness over time at varying relative humidity levels. The main graph displays the relative time to grow frost thickness in comparison to the frost thickness growth over time at a relative humidity of 85%.

Best line fits were obtained for the 85% RH line in the upper right corner of Figure 6.3 and for the 100%, 75% and 65% RH lines in the main graph. The former line equation was divided by three latter line equations, to obtain the best fit lines of the same graph as in the upper right corner. This method was chosen to decrease inaccuracies, as the software used to obtain measured values from a graph performs better with larger graphs. Furthermore, not all data was available for the desired time

range (0-60 min). Figure 6.5 displays the fitted lines as data points. Plotting these points together gives a surface, seen in Figure 6.6.

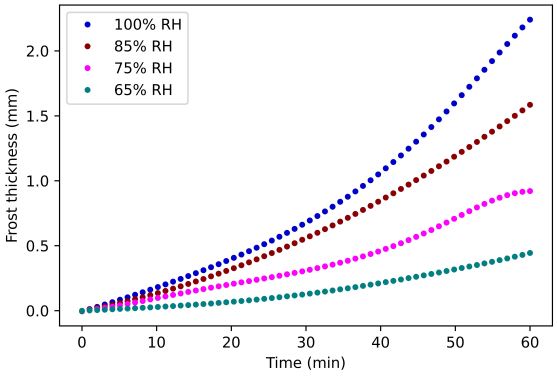


Figure 6.5: Fitted data of frost thickness over time at varying relative humidity for an ambient air temperature of 0 °C

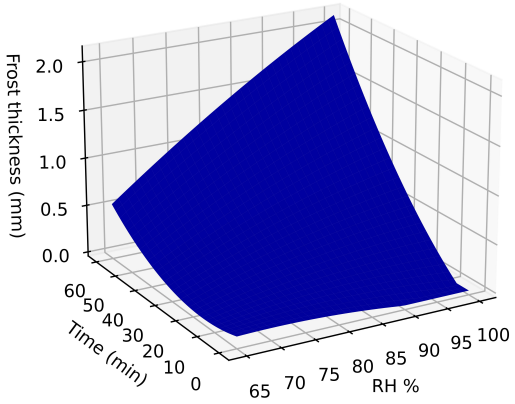


Figure 6.6: Fitted surface of frost thickness over time at varying relative humidity for an ambient air temperature of 0 °C

The data in Figure 6.4 was used to create equivalent surfaces for ambient air temperatures between -8 to 0 °C. The data is given for a frosting time of 35 min. It is assumed that the relationship between frost thickness and ambient air temperature will remain the same at all times. Using the line of T_a = 0 °C as a base case, the relative humidity to grow frost thickness compared to that of a T_a = 0 °C, similarly to that of Figure 6.3, was computed for each ambient temperature. The equation for the surface seen in 6.6 was multiplied by these best fit lines. The surfaces obtained are displayed in Figure 6.7. Appendix B.3 contains the equations of the surfaces. The same method of linear interpolation, as described in Section 5.1.3 is used to obtain frost thickness values at varying time and RH values.

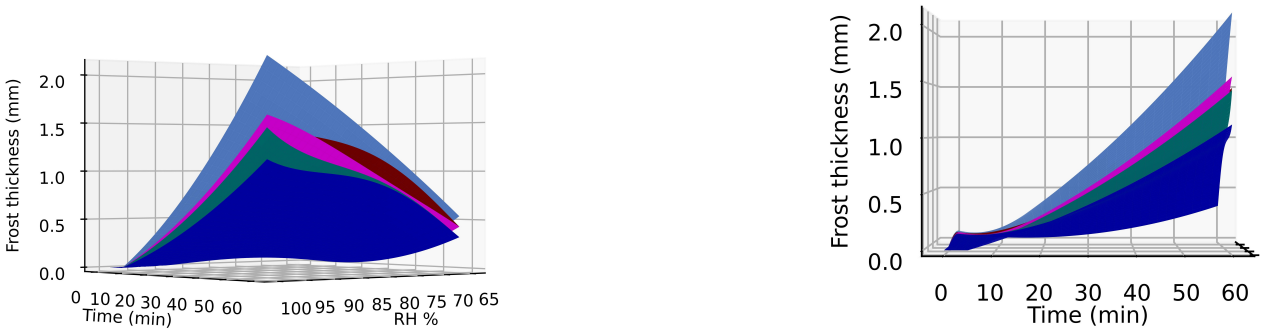


Figure 6.7: Frost thickness over time at varying relative humidity and ambient air temperatures

6.2.1.1. Defrosting strategy and control

As discussed in Section 3.1.4, many forms of defrosting exist. For simplicity, in this model, defrosting is achieved with an electric heater. During a defrosting cycle, the HP is turned off, but circulation of glycol through the dry cooler and evaporator continue. A 2500 W electric heater is turned on, heating the glycol in the circuit. When the glycol begins losing heating to its surroundings instead of gaining it (i.e. when the sum of thermal energy gained by the HEX, $\sum q_i$ (J) becomes negative), the frost begins melting. The thickness of melted frost is characterised by

$$t_m = \frac{\sum q_i}{\Delta h_{fus} \rho_{ice} A_{ice}}, \tag{6.18}$$

where t_m is the thickness of frost that has melted (m), Δh_{fus} is the latent heat of fusion of water (J/kg) and A_{ice} is approximately equal to the heat transfer area of the dry cooler (m^2). The heat released due to fusion is not taken into account in the defrosting model.

As frost grows, it reduces the overall heat transfer coefficient through an additional conduction term and a decrease in the convective term due to reduced air flow. Once frost thickness reaches half of the fin spacing, air flow stops. Therefore, in the model, the condition for a defrosting cycle to turn on is when frost thickness reaches half of fin spacing. Once this condition is met, the HP turns off and the electric heater is turned on. The defrosting cycle ends when frost thickness is 0.5 mm. In reality, it is not possible to easily measure frost thickness. Thus, defrosting cycles begin when the evaporating temperature in the HP drops drastically due to little air flow. However, this method of control was not chosen to avoid additional complexity in the model and to reduce the model's dependence on sub-optimal control. However, in future work, the control of the defrosting cycle should be studied further.

6.3. Frost growth experimental validation

A PhD defence by K.P. Sankaranarayanan [51] investigated frost growth on HEXs used in ASHPs. The experimental data from this study is used to validate the frost growth model of the dry cooler.

Experimental data in the study is given for 3 relative humidities; 72%, 82%, 92% at an ambient air temperature of 1.7 °C. These values were used as inputs to the model and the frost growth was computed. The comparison in frost thickness between experimental and numerical values is displayed in Figure 6.8. From observation, frost thickness is well-predicted for a relative humidity of 92%. For relative humidity values of 82% and 72%, experimental data shows less frost growth than is predicted by the model.

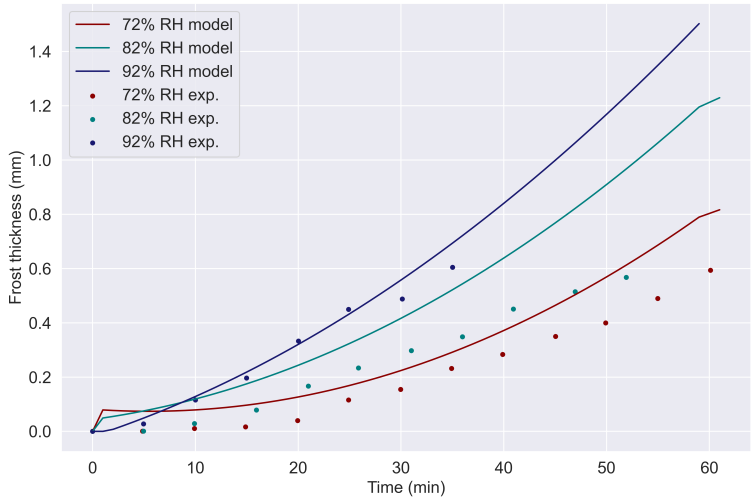


Figure 6.8: Comparison between experimental and numerical frost thickness

The percentage error between experimental and numerical results is displayed in Figure 6.8. For relative humidity values of 72% and 82%, the error generally lies around 50-75%. For 92%, the error is approximately 10%, a significant portion. However a better measure of error is the difference between calculated overall heat transfer coefficients using the experimental and numerical frost thickness values, as this is the parameter that uses the frost thickness in the dry cooler model. Using equation 6.6, this comparison is seen in Figure 6.10. It is seen that the error remains below 1%. This error is acceptable and therefore the frost growth in the model is considered validated.

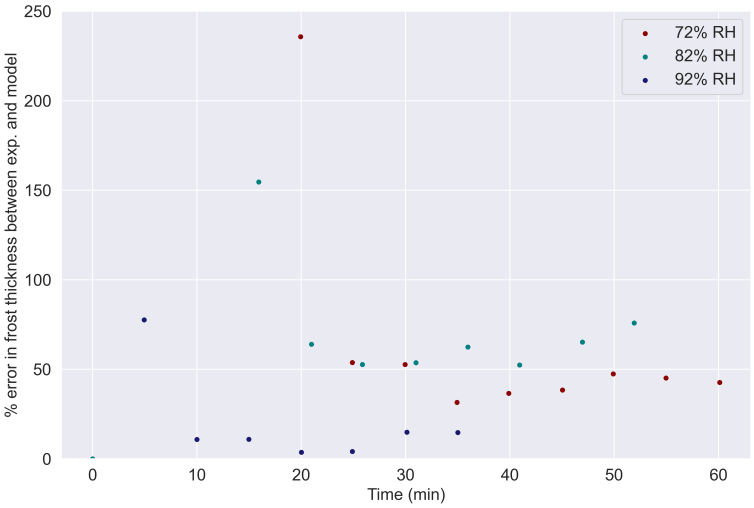


Figure 6.9: Error between experimental and numerical frost thickness

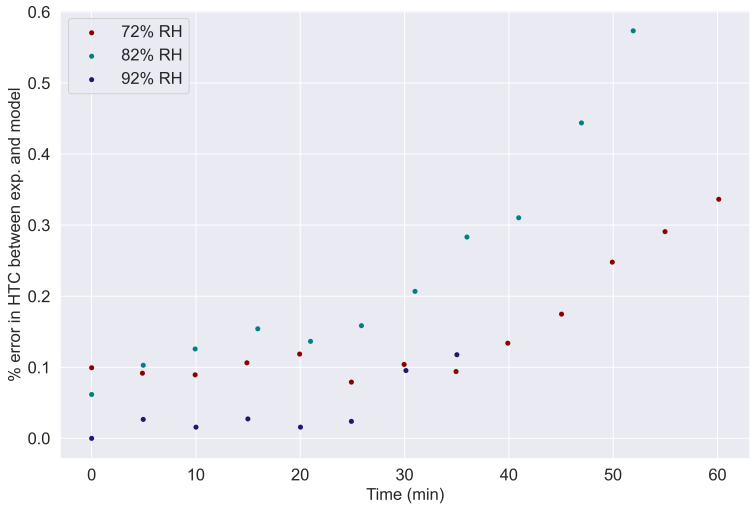


Figure 6.10: Error between overall heat transfer coefficient using experimental and numerical frost thickness

6.4. Air source experimental validation

6.4.1. Experimental setup

An experiment was set up to test the dry cooler model's ability to predict outflow temperature. The experimental setup is displayed in Figure 6.11. The setup consists of a dry cooler, a boiler and a pump. An expansion vessel is used to regulate the pressure in the system and a pressure gauge is used to monitor it. The tank is filled with tap water. Table 6.2 displays the components and sensors used in the experiment.

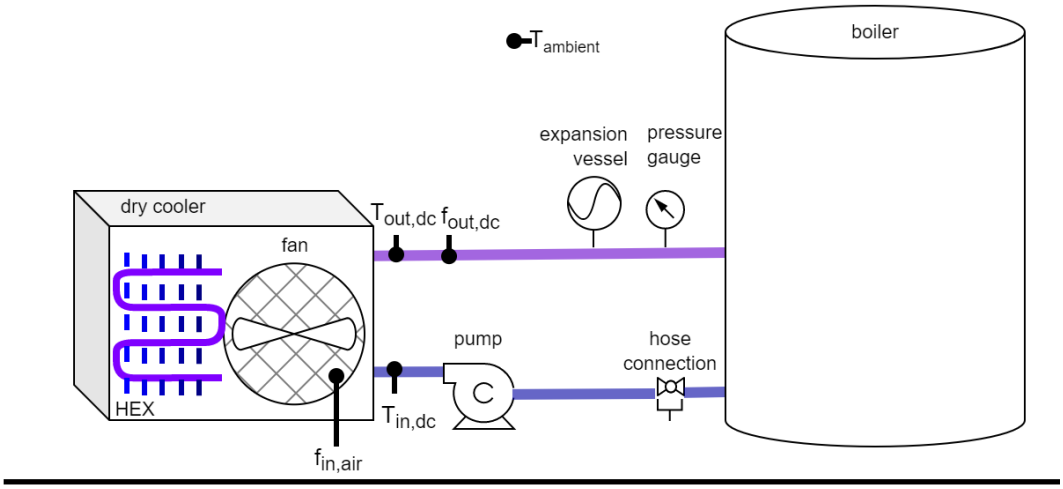


Figure 6.11: Setup for dry cooler experimental validation

Table 6.2: Air source experiment - components and sensors used

Component Type	Model	Error
Boiler	DeJong M150	-
Pump	GreenPro EAB circulating pump	-
Dry Cooler	Intersam ISAK 14	-
Temperature sensor	DS18B20	±0.5 °C
Flow sensor (water)	Huba Control 210	±1%
Flow sensor (air)	BT-866A Anemometer	±2%

6.4.2. Methodology

Tap water in the boiler was cooled to 9 °C using DeWarmte's ASHP. Once the water reached the setpoint temperature, the pump and dry cooler fan were turned on. The mass flow rate of air was not measured, as air flow over the dry cooler is dependent on the measurement location. Thus, the fan speed was set to its maximum power. According to the datasheet, this is equivalent to an air flow rate of 0.37 (kg/s). The anemometer was used to test whether this air speed is reached. The water was allowed to circulate in the system and heat up until it reached a steady state temperature close to that of the ambient air the room. The experiment was repeated 3 times.

Ambient air temperature, inflow and outflow temperatures, mass flow of water and air are monitored. The sensors and their corresponding errors are presented in Section 6.4.3.2.

Inflow temperature $T_{in,dc}$, ambient air temperature $T_{ambient}$, flow of water $f_{out,dc}$ and flow of air $f_{in,air}$ are used as inputs in the dry cooler model. The calculated and experimentally measured $T_{out,dc}$ are compared.

6.4.3. Results

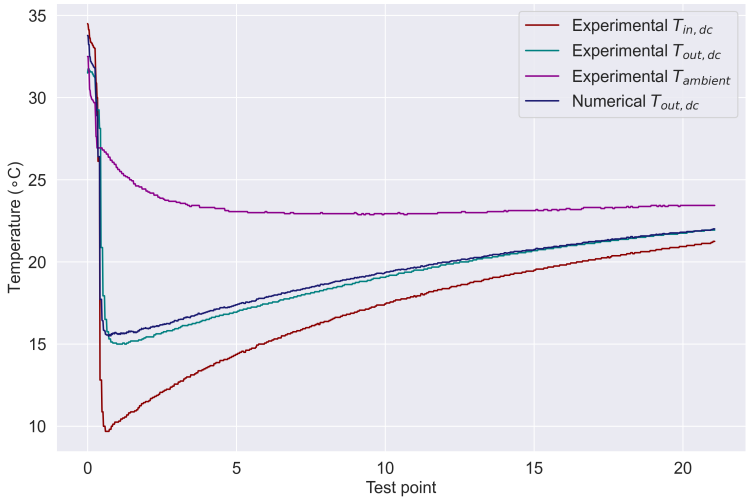
Figure 6.12 displays the experimentally measured temperatures $T_{ambient}$, $T_{in,dc}$, $T_{out,dc}$ and calculated $T_{out,dc}$ for the 3 performed experiments.

The initial steep drop in temperature for $T_{in,dc}$ and $T_{out,dc}$ is due to existing warm water in the pipes connecting the boiler and dry cooler passing through the sensors before it is replaced with cold water from the boiler. The laboratory's ambient temperature is initially at a high of approximately 33°C due to heat being removed from the boiler by DeWarmte's ASHP and released to the surrounding air. The ambient temperature stabilizes to 23 °C. $T_{in,dc}$ is initially measured at approximately 10 °C and increases gradually to 21 °C. $T_{out,dc}$ is initially measured at approximately 16 °C and increases gradually to 22 °C.

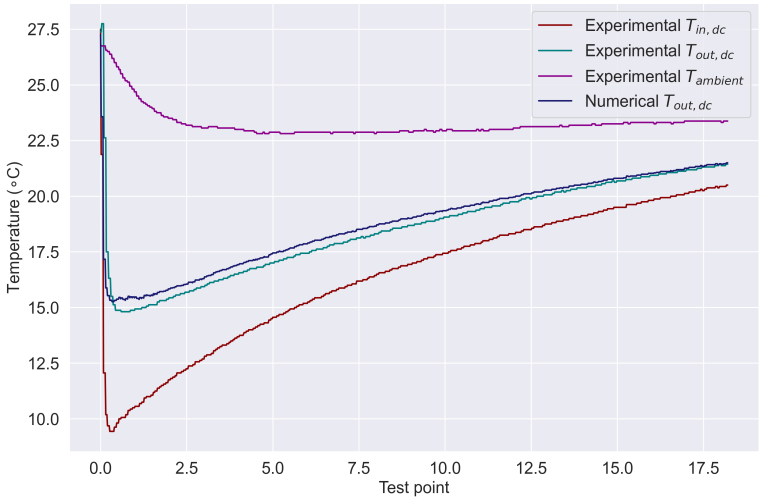
6.4.3.1. Error analysis

Figure 6.13 displays the numerical and experimental $T_{out,dc}$ and corresponding error for the 3 performed experiments. Experimental $T_{out,dc}$ is plotted with its sensor error of 0.5 ±°C.

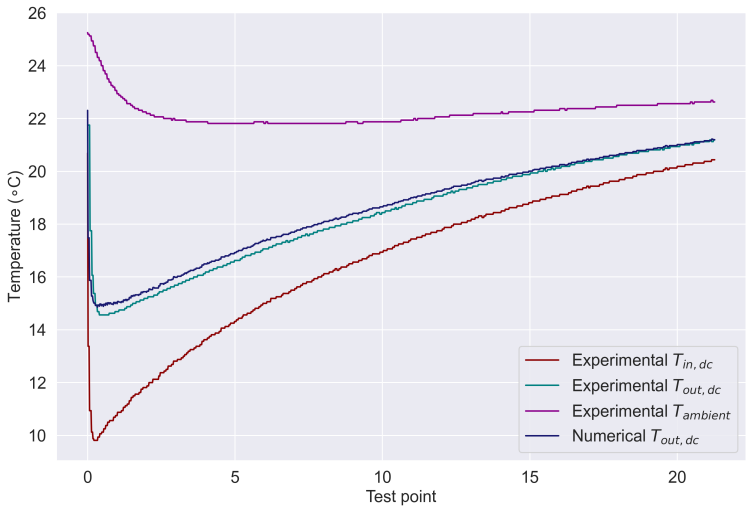
The model predicts the outflow temperature well during each experiment. The error reaches a maximum of 3.5 % due to the steep drop in temperature when cool water from the boiler is registered by the inflow temperature sensors. Generally, the error lies between 0 and 0.25 %. The error gradually decreases as the outflow temperature reaches its steady state value. The reason for this unclear; it may be due to the model performing worse in predicting heat transfer at higher temperature differences between inflow and outflow. This effect requires further investigation. Nevertheless, the model predicts the outflow temperature with a small error and therefore the air source is considered validated.



(a) Experiment 1

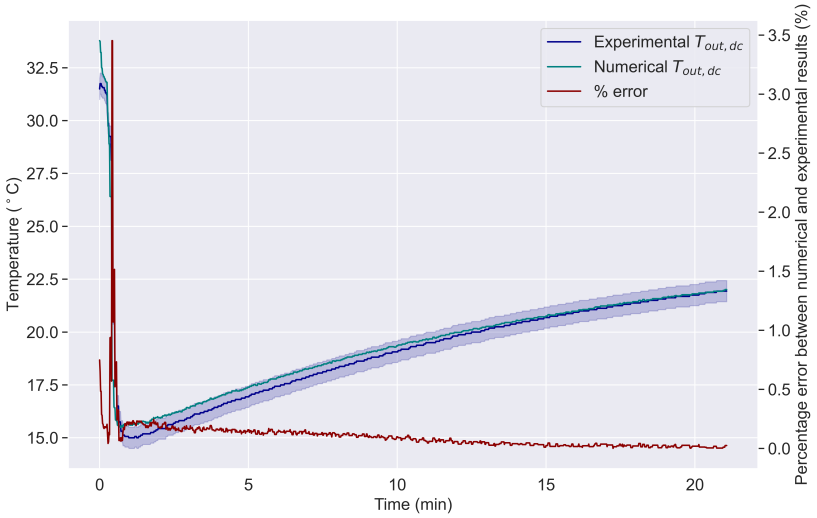


(b) Experiment 2

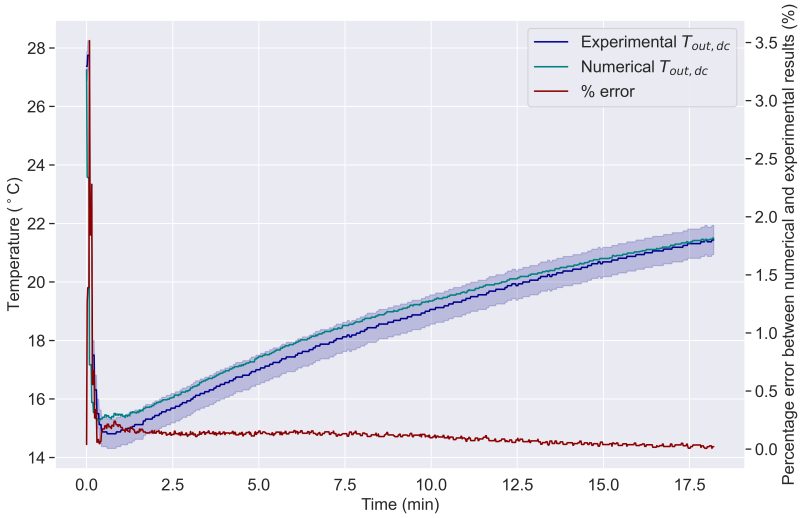


(c) Experiment 3

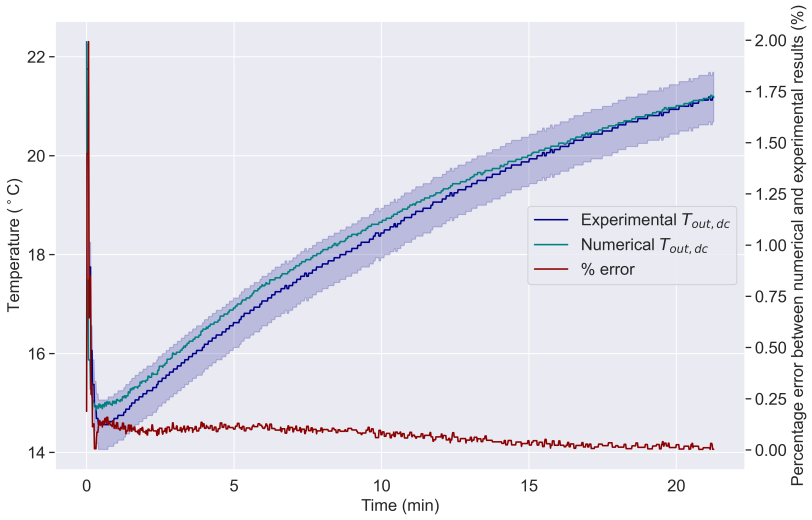
Figure 6.12: Measured experimental temperatures and calculated $T_{out,dc}$



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 6.13: Comparison between experimental and numerical $T_{out,dc}$

6.4.3.2. Sobol's variance-based sensitivity analysis

A Sobol sensitivity analysis was conducted on the air source model. For theoretical background on the method, refer to Section 5.3.2.2. The tested output was the outlet dry cooler temperature T_{out}. The inputs were dry cooler input temperature, T_{in}, ambient air temperature $T_{ambient}$, dry cooler input mass flow, $\dot{m}_{in}$ and air mass flow, $\dot{m}_{air}$. The sensor errors as described in Table 6.2 were used as upper and lower bounds for the inputs. The mass flow rate of air was not measured. Instead, the fan was set to its maximum power. It was assumed that this value could vary by ±30%, depending on the location on the dry cooler. 128 random samples were used to obtain indices at each test point.

The sensitivity analysis was performed for all three experiments, the results of which are displayed in Figure 6.14. Note that both mass flow rates have a Sobol index of approximately zero, thus only one line is seen.

The results of the analysis show that the variance in T_{in} has the largest weight on the output's variance, with $T_{ambient}$ following. This results may be explained as follows; T_{out} may be decomposed into two parts: T_{in} and an additional gained ΔT. Variance in T_{in} will have the largest effect on T_{out}, as it has a direct relation with it. As the source temperature, $T_{ambient}$ determines ΔT, but it has an indirect relation with T_{out}. However, the sensitivity analysis shows that variance in mass flow rates has no effect on the variance of the output. A potential explanation could be that the model is not sensitive to the tested variations in the mass flow rates. Due to the low specific heat of air, it is possible that the ±30% variation in its mass flow may not have a very significant impact on the output temperature. This variation is significantly lower for the measured mass flow of water (±2%), making up for its higher specific heat value. Furthermore, due to low temperature gradients, heat added to the fluid at each test point is low, meaning the effect of variation in mass flow has less impact on the output temperature. This results in no variation in the Sobol indices as the experiment runs and ΔT decreases.

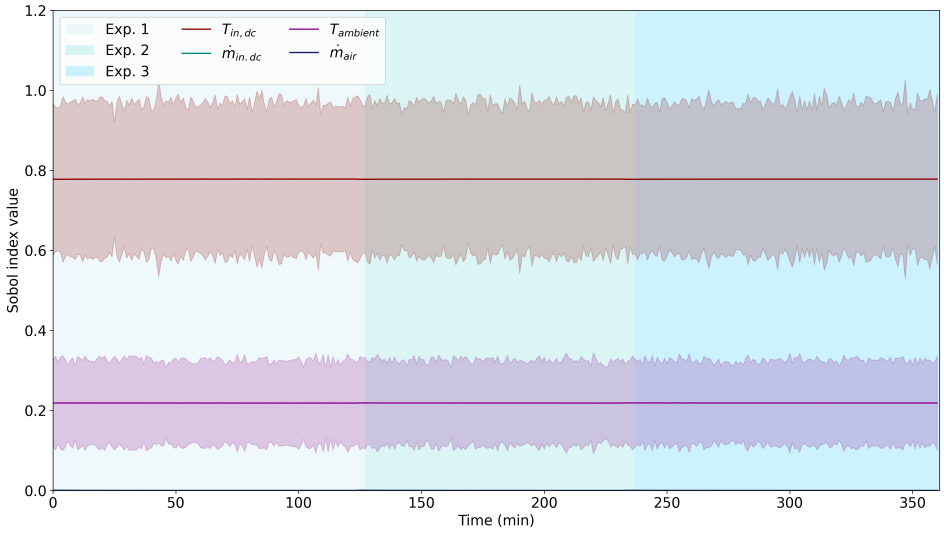


Figure 6.14: Sobol analysis of dry cooler model

7

Wastewater bag model

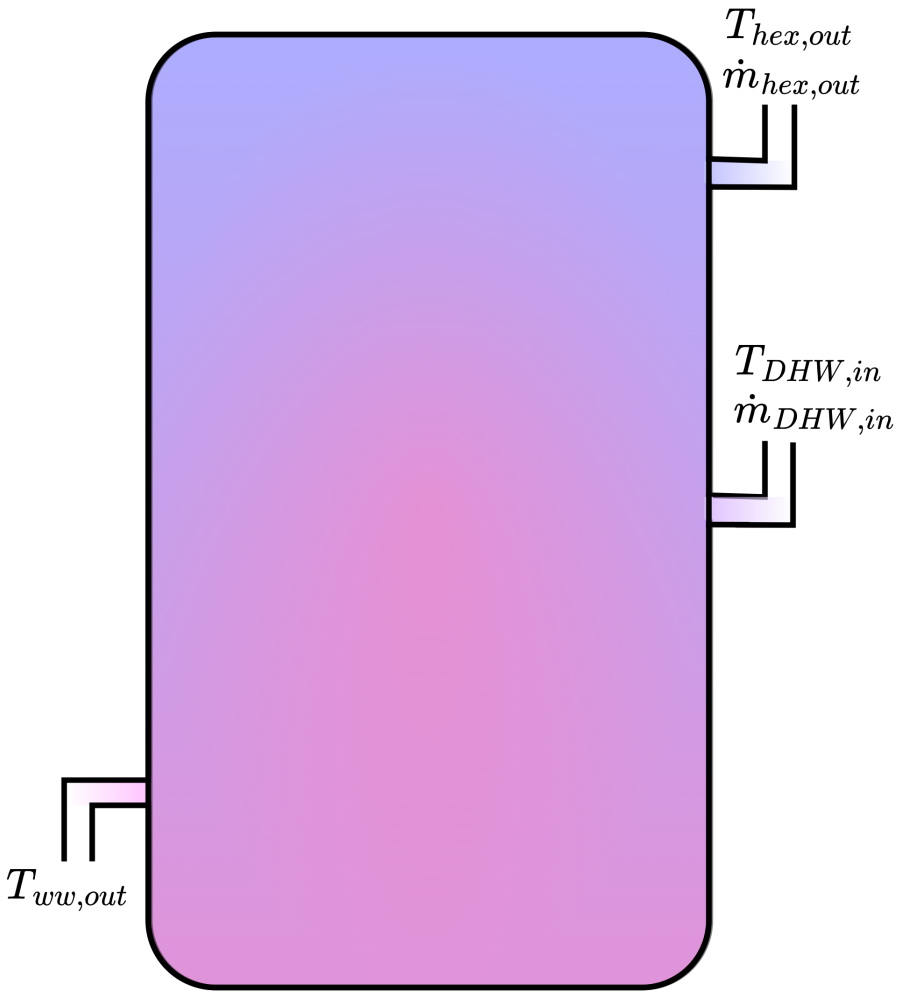


Figure 7.1: Wastewater bag class

Table 7.1: Inputs and outputs of wastewater bag class

Variable	Unit	Type	Description	Source
$T_{hex,out}$	K	Input	Temperature of incoming wastewater from HEX	Wastewater-to-glycol HEX class
$\dot{m}_{hex,out}$	kg/s	Input	Mass flow rate of incoming wastewater from HEX	Wastewater-to-glycol HEX class
$T_{DHW,in}$	K	Input	Temperature of incoming wastewater from sewage	DHW class
$\dot{m}_{DHW,in}$	kg/s	Input	Mass flow of incoming wastewater from sewage	Filter class
$T_{ww,out}$	K	Output	Temperature of outgoing wastewater	-

7.1. Numerical modelling approach

The flexible bag used to store wastewater has multiple sources of heat loss and gain. When the HP is turned on, water is pumped from the wastewater bag and its heat is transferred to the HP's refrigerant circuit. Heat is gained when new wastewater is flushed into the bag. Depending on the temperature of the bag and its environment, poor insulation of the bag can result in heat loss or gain. These effects are modelled with a one-dimensional nodal model of the bag and the crawlspace surrounding it, seen in Figure 7.2. Each node represents a calculated temperature. Note dashed lines between nodes 2-3 and 9-10 represent multiple nodes used to calculate conduction through respective mediums.

- **1** - Ambient room temperature T_∞
- **2** - Room floor temperature T_f
- **3** - Crawlspace ceiling temperature T_c
- **4** - Crawlspace air temperature T_a
- **5** - Top external bag temperature T_{bet}
- **6** - Top internal bag temperature T_{bit}
- **7** - Wastewater temperature T_w
- **8** - Bottom internal bag temperature T_{bib}
- **9** - Top soil temperature T_s
- **10** - Soil at specified depth temperature $T_{s,in}$

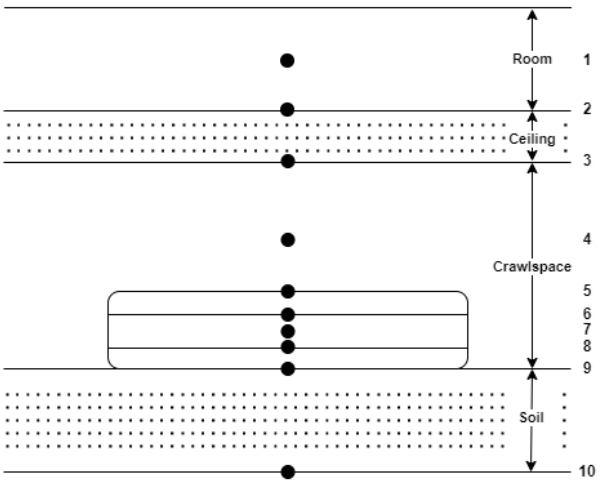


Figure 7.2: Nodal model of wastewater bag

7.1.1. Energy balance equations in nodal model

An energy balance expression is required for each node. These expressions are then solved simultaneously using a matrix configuration and Euler backward implicit time stepping to obtain temperatures at each node.

Crawlspace ceiling

The crawlspace ceiling temperature distribution was found by calculating temperatures at multiple, equally spaced nodes throughout the ceiling. Two ghost nodes, one above the living room floor and the other below the ceiling surface, were used to

assign the required boundary conditions. The conduction equation describes the energy balance in the inner ceiling domain, given by

$$\rho_c C_{p,c} \frac{\partial T_c}{\partial t} = k_c \frac{\partial^2 T}{\partial y^2}, \tag{7.1}$$

where ρ_c $C_{p,c}$ and k_c are the density (kg/m^3), specific heat (J/kgK) and thermal conductivity (W/mK) of the ceiling material respectively.

In discretized form, this equation becomes

$$\rho_c C_{p,c} \frac{T_{c,i}^{n+1} - T_{c,i}^{n}}{\Delta t} = k_c \frac{T_{c,i-1}^{n+1} - 2T_{c,i}^{n+1} + T_{c,i+1}^{n+1}}{\Delta y_c^2}, \tag{7.2}$$

where Δt is the timestep (s), Δy_c is the distance between each ceiling node (m) and subscripts i and n indicate the spatial and temporal step respectively.

At the living room floor, a convective boundary condition is imposed, given by

$$h_{c,t}(T_\infty - T_f^{n+1}) = -k_c \frac{(T_{c,1}^{n+1} - T_f^{n+1})}{\Delta y_c/2}, \tag{7.3}$$

where $h_{c,t}$ is the convective heat transfer coefficient above the crawlspace ceiling (W/m^{2}K). Furthermore, it is assumed that the floor temperature is the mean of the upper ghost node and the first inner ceiling node

$$T_f = \frac{T_{c,0} + T_{c,1}}{2}. \tag{7.4}$$

At the crawlspace ceiling surface, a convective and radiative boundary condition is imposed, given by

$$h_{c,b}(T_a^{n+1} - T_c^{n+1}) + h_{r,f}(T_s^{n+1} - T_c^{n+1}) + h_{r,b}(T_{bet}^{n+1} - T_c^{n+1}) = -k_c \frac{(T_{i_{max,c}-1}^{n+1} - T_c^{n+1})}{\Delta y_c/2}, \tag{7.5}$$

where $h_{c,b}$, $h_{r,f}$ and $h_{r,b}$ are the convective crawlspace ceiling bottom, radiative crawlspace floor and radiative wastewater bag heat transfer coefficients (W/m^{2}K).

It is again assumed that ceiling surface temperature is the mean of the last inner ceiling node and lower ghost node

$$T_c = \frac{T_{c,i_{max,c}-1} + T_{c,i_{max,c}}}{2}. \tag{7.6}$$

A system of equations describing the temperature distribution throughout the ceiling may be constructed in matrix form:

$$\begin{bmatrix} b & c & 0 & 0 \\ a & b & c & 0 \\ 0 & a & b & c \\ \dots & \dots & \dots & \dots \\ 0 & a & b & c \\ 0 & 0 & a & b \end{bmatrix} \cdot \begin{Bmatrix} T_{c,1}^{n+1} \\ T_{c,2}^{n+1} \\ T_{c,3}^{n+1} \\ \dots \\ T_{c,i_{max,c}-2}^{n+1} \\ T_{c,i_{max,c}-1}^{n+1} \end{Bmatrix} = \begin{Bmatrix} T_{c,1}^{n} \\ T_{c,2}^{n} \\ T_{c,3}^{n} \\ \dots \\ T_{c,i_{max,c}-2}^{n} \\ T_{c,i_{max,c}-1}^{n} \end{Bmatrix} \tag{7.7}$$

For nodes $i = 2$ to $i = i_{max,c} - 2$, coefficients a, b, c in matrix M_c are found by rearranging Equation 7.2, giving

$$a = -\frac{\alpha_c \Delta t}{\Delta y_c^2},$$

$$b = 1 + 2\frac{\alpha_c \Delta t}{\Delta y_c^2},$$

$$c = -\frac{\alpha_c \Delta t}{\Delta y_c^2}.$$

However, for nodes $i = 1$ and $i = i_{max,c}-1$, coefficient b in matrix M_c is found by imposing corresponding boundary conditions. Coefficients a, c remain identical to those found above. For the convective boundary condition at the living room floor, Equation 7.4 is rearranged for $T_{c,0}$. From Equation 7.3, an expression is obtained for T_f and is substituted into newly rearranged Equation 7.4. This expression is substituted into Equation 7.2 to obtain coefficient b and the corresponding right-hand side expression. This is found to be

$$b_1 = 1 + 2\frac{\alpha_c \Delta t}{\Delta y_c^2} + \frac{\alpha_c \Delta t}{\Delta y_c^2}\left(\frac{2}{\frac{-\Delta y_c h_{c,t}}{2k_c} - 1} + 1\right),$$

$$RHS_1 = T_{c,1}^n + \frac{\alpha_c \Delta t}{\Delta y_c^2}\left(\frac{2}{\frac{-\Delta y_c h_{c,t}}{2k_c} - 1}\right) T_\infty.$$

The same process is repeated to find the corresponding coefficient b for the convective and radiative boundary condition at the ceiling surface. Equation 7.6 is rearranged for $T_{c,i_{max,c}}$ into which an expression for T_c from Equation 7.5 is substituted. This is then substituted into Equation 7.2, giving

$$b_{c,i_{max,c}-1} = 1 + 2\frac{\alpha_c \Delta t}{\Delta y_c^2} - \frac{\alpha_c \Delta t}{\Delta y_c^2}\left(\frac{2}{1 - \frac{\Delta y_c}{2k_c}(h_{c,b} + h_{r,f} + h_{r,b})} - 1\right).$$

It should be noted that $RHS_{i_{max,c}-1}$ remains $T_{c,i_{max,c}-1}^n$ to ensure a fully implicit timestepping method. The implementation of the dependency of $T_{c,i_{max,c}-1}^n$ on $T_a^{n+1}, T_{f,b}^{n+1}, T_{bet}^{n+1}$ is further discussed in Section 7.1.1.1.

Soil

To model the temperature distribution throughout the soil, the same approach as that of the ceiling temperature distribution was taken. The conduction equation, given by

$$\rho_s C_{p,s} \frac{T_{s,i}^{n+1} - T_{s,i}^n}{\Delta t} = k_s \frac{T_{s,i-1}^{n+1} - 2T_{s,i}^{n+1} + T_{s,i+1}^{n+1}}{\Delta y_s^2}, \tag{7.8}$$

where ρ_s $C_{p,s}$ and k_s are the density (kg/m^3), specific heat (J/kgK) and thermal conductivity (W/mK) of the soil and Δy_s is the distance between soil nodes (m).

Coefficients a, b, c in matrix M_s are found to be

$$a = -\frac{\alpha_s \Delta t}{\Delta y_s^2},$$

$$b = 1 + 2\frac{\alpha_s \Delta t}{\Delta y_s^2},$$

$$c = -\frac{\alpha_s \Delta t}{\Delta y_s^2}.$$

Again, for soil nodes $i = 1$ and $i = i_{max,s} - 2$, coefficients a, c remain the same as those shown above. To find $b_{imax,s-1}$ and $RHS_{imax,s-1}$, a constant temperature boundary condition is imposed at the wall between soil nodes $i = i_{max,s-1}$ and $i = i_{max,s}$. It is again assumed that

$$T_{s,in} = \frac{T_{s,i_{max,s}-1} + T_{s,i_{max,s}}}{2}. \tag{7.9}$$

Rearranging for $T_{s,i_{max,s}}$ and substituting the expression into Equation 7.8, we find

$$b_{i_{max,s}-1} = 1 + 3\frac{\alpha_s \Delta t}{\Delta y_s^2},$$

$$RHS_{i_{max,s}-1} = T_{s,i_{max,s}-1}^n + 2\frac{\alpha_s \Delta t}{\Delta y_s^2} T_{s,in}$$

The boundary condition at the top internal soil node, $i = 1$ is given by

$$-k_s \frac{(T_s^{n+1} - T_{s,1}^{n+1})}{\Delta y_s/2} = -k_{pvc} \frac{(T_{bib}^{n+1} - T_s^{n+1})}{t_b}, \tag{7.10}$$

where k_{pvc} is the thermal conductivity of the bag (made of PVC) (W/m^{2}K) and t_b is its thickness. Furthermore, it is assumed that

$$T_s = \frac{T_{s,0} + T_{s,1}}{2}. \tag{7.11}$$

Rearranging Equation 7.11 for $T_{s,0}$, Equation 7.10 for T_s and substituting this expression into Equation 7.8, we find that

$$b_1 = 1 + 2\frac{\alpha_s \Delta t}{\Delta y_s^2} + \left(\frac{\frac{k_s t_b}{k_{pvc}\Delta y_s} - 0.5}{\frac{k_s t_b}{k_{pvc}\Delta y_s}}\right)$$

It should again be noted that RHS_1 remains $T_{s,1}^n$. The dependence of $T_{s,1}^{n+1}$ on T_{bib}^{n+1} will be discussed in Section 7.1.1.1.

Air

The discretized energy balance at the air node is given by

$$\rho_a C_{p,a} V_{cs} \frac{T_a^{n+1} - T_a^n}{\Delta t} = h_{c,b} A_b (T_{bet}^{n+1} - T_a^{n+1}) + h_{c,b} A_f (T_s^{n+1} - T_a^{n+1}) + h_{c,b} A_c (T_c^{n+1} - T_a^{n+1}), \tag{7.12}$$

where $C_{p,a}$ is the specific heat of air (J/kgK) and V_{cs} is the volume of the crawlspace (m^3).
Therefore, we obtain the following equation to describe the energy balance at the air node:

$$\rho_a C_{p,a} \Delta x_{cs} \Delta y_{cs} \Delta z_{cs} \frac{T_a^{n+1} - T_a^n}{\Delta t} = h_{c,b} \Delta x_b \Delta z_b (T_{bet}^{n+1} - T_a^{n+1}) + h_{c,b} (\Delta x_{cs} \Delta z_{cs} - \Delta x_b \Delta z_b)(T_s^{n+1} - T_a^{n+1}) + h_{c,b} \Delta x_{cs} \Delta z_{cs} (T_c^{n+1} - T_a^{n+1}), \tag{7.13}$$

where Δx and Δz are length and width (m) respectively, with subscripts b and cs representing the bag and crawlspace respectively.

External bag top

The energy balance at the surface of the bag is given by

$$h_{c,b}(T_a^{n+1} - T_{bet}^{n+1}) + h_{r,b} F_{cb} (T_c^{n+1} - T_{bet}^{n+1}) = -k_{pvc} \frac{(T_{bit}^{n+1} - T_{bet}^{n+1})}{t_b}, \tag{7.14}$$

where F_{cb} is the radiative shape factor between bag and ceiling.

Internal bag top

The energy balance at the internal top node of the bag is given by

$$-k_{pvc} \frac{(T_{bit}^{n+1} - T_{bet}^{n+1})}{t_b} = h_w (T_{bit}^{n+1} - T_w^{n+1}), \tag{7.15}$$

where h_w is the convective heat transfer coefficient in the wastewater bag (W/m^{2}K).

Wastewater

The energy balance at the water node is given by

$$\rho_w C_{p,w} V_b \frac{T_w^{n+1} - T_w^n}{\Delta t} = h_w A_b (T_{bit}^{n+1} - T_w^{n+1}) + h_w A_b (T_{bib}^{n+1} - T_w^{n+1}), \tag{7.16}$$

where $C_{p,w}$ is the specific heat of water (J/kgK) and V_b is the volume of the bag (m^3). Note it is assumed that wastewater has comparable fluid properties to water.

The final equation is therefore

$$\rho_w C_{p,w} \Delta x_b \Delta y_b \Delta z_b \frac{T_w^{n+1} - T_w^n}{\Delta t} = h_w \Delta x_b \Delta z_b (T_{bit}^{n+1} - T_w^{n+1}) + h_w \Delta x_b \Delta z_b (T_{bib}^{n+1} - T_w^{n+1}), \tag{7.17}$$

where y_b is the height of the bag (m).

Internal bag bottom

The energy balance at the internal node of the bag bottom is given by

$$-k_{pvc}\frac{(T_{bib}^{n+1}-T_{s}^{n+1})}{t_b}=h_w(T_{bib}^{n+1}-T_{w}^{n+1}). \tag{7.18}$$

7.1.1.1. Matrix implementation

The temperature nodes are treated as a system of equations and are solved for at each timestep for time level n+1. After a quick node number independence study, it is found that the solution for temperature distribution throughout the soil and crawlspace ceiling converges when the distance between nodes is 1.5 cm. Thus, the equivalent number of temperature nodes are used. This is implemented in the following form:

$$\left[\mathbf{M}_{ij}\right]\cdot\begin{Bmatrix} T_{c,1}^{n+1}\\ T_{c,2}^{n+1}\\ \dots\\ T_{c,i_{max,c}-1}^{n+1}\\ T_{c}^{n+1}\\ T_{a}^{n+1}\\ T_{bet}^{n+1}\\ T_{bit}^{n+1}\\ T_{w}^{n+1}\\ T_{bib}^{n+1}\\ T_{s}^{n+1}\\ T_{s,1}^{n+1}\\ T_{s,2}^{n+1}\\ \dots\\ T_{s,i_{max,s}-1}^{n+1}\end{Bmatrix}=\begin{Bmatrix} T_{c,1}^{n}\\ T_{c,2}^{n}\\ \dots\\ T_{c,i_{max,c}-1}^{n}\\ T_{c}^{n}\\ T_{a}^{n}\\ T_{bet}^{n}\\ T_{bit}^{n}\\ T_{w}^{n}\\ T_{bib}^{n}\\ T_{s}^{n}\\ T_{s,1}^{n}\\ T_{s,2}^{n}\\ \dots\\ T_{s,i_{max,s}-1}^{n+1}\end{Bmatrix}.$$

Matrices M_c and M_s are placed in the top left and bottom right corners respectively of matrix M. The correct RHS expressions are implemented for $T_{c,1}^{n+1}$ and $T_{s,i_{max,s}-1}^{n+1}$. As previously stated, the RHS stays the same of $T_{s,1}^{n+1}$ and $T_{c,i_{max,c}-1}^{n+1}$. The dependencies on temperatures from varying nodes at time level n+1 are implemented in matrix M. Furthermore, the expressions from Equation 7.13 to 7.18 are rearranged and implemented in the matrix. Therefore, the final system of equations is of the following

form, with $\times$ indicating a non-zero coefficient that may be deduced from Equation 7.13 to 7.18:

$$\begin{bmatrix}
b_{c,1} & c_{c,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_c & b_c & c_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & a_{c,i_{max,c}-1} & b_{c,i_{max,c}-1} & 0 & \times & \times & 0 & 0 & 0 & \times & 0 & 0 & 0 \\
0 & 0 & \times & 1 & \times & \times & 0 & 0 & 0 & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & 1 & \times & 0 & 0 & 0 & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & \times & 1 & \times & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \times & 1 & \times & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \times & 1 & \times & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \times & 1 & \times & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \times & 1 & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \times & 0 & b_{s,1} & c_{s,1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_s & b_s & c_s \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{s,i_{max,s}-1} & b_{s,i_{max,s}-1}
\end{bmatrix}
\cdot
\begin{Bmatrix}
T_{c,1}^{n+1} \\ T_{c,2}^{n+1} \\ \dots \\ T_{c,i_{max,c}-1}^{n+1} \\ T_c^{n+1} \\ T_a^{n+1} \\ T_{bet}^{n+1} \\ T_{bit}^{n+1} \\ T_w^{n+1} \\ T_{bib}^{n+1} \\ T_s^{n+1} \\ T_{s,1}^{n+1} \\ T_{s,2}^{n+1} \\ \dots \\ T_{s,i_{max,s}-1}^{n+1}
\end{Bmatrix}
=
\begin{Bmatrix}
T_{c,1}^{n} \\ T_{c,2}^{n} \\ \dots \\ T_{c,i_{max,c}-1}^{n} \\ 0 \\ T_a^{n} \\ 0 \\ 0 \\ T_w^{n} \\ 0 \\ 0 \\ T_{s,1}^{n} \\ T_{s,2}^{n} \\ \dots \\ T_{s,i_{max,s}-1}^{n+1}
\end{Bmatrix}.$$

7.2. Heat transfer coefficients

The heat transfer coefficients in the energy balances must be defined. However, they vary with temperature. To avoid additional computational cost, the coefficients will be considered temperature independent. This assumption is considered valid as no stark temperature differences over short periods of time exist.

7.2.1. Convection within wastewater bag

The wastewater heat transfer coefficient is dependent solely on natural convection within the bag. However, depending on air and soil temperatures, it may be heated from the top or bottom bag surface. Therefore two correlations for calculating the heat transfer coefficient of a fluid between two plates were investigated, one for a lower heated plate and the other for an upper heated plate.

Lower heated surface

To compute the water heat transfer coefficient for the condition of $T_{bib} > T_w > T_{bit}$, an empirical correlation for water between two heated plates was used [33]. The paper suggests the following empirical correlation:

$$Nu = 1 + 1.44\left(1 - \frac{1708}{Ra}\right) + \left(\left(\frac{Ra}{5830}\right)^{1/3} - 1\right) + 2\left(\frac{Ra^{1/3}}{140}\right)^{1 - ln\left(\frac{Ra^{1/3}}{140}\right)}, \tag{7.19}$$

where Nu is the Nusselt number and Ra is the Rayleigh number, given by

$$Ra = \frac{\beta g \Delta T l_c^3}{v\alpha}. \tag{7.20}$$

β is the thermal expansion coefficient (1/K), g is gravitational acceleration (m/s^2), ΔT is the temperature difference between surface and fluid (K), l_c is characteristic length, in this case the height of the bag (m), v is kinematic viscosity of the fluid (m^2/s) and α is its thermal diffusivity (m^2/s).

The heat transfer coefficient is calculated by rearranging

$$Nu = \frac{hl_c}{k}, \tag{7.21}$$

where h is the heat transfer coefficient (W/m^{2}K) and k is the thermal conductivity of the fluid (W/mK).

Upper heated surface

To compute the water heat transfer coefficient for the condition $T_{bit} > T_w > T_{bib}$, an empirical correlation was used [28]. The following expression is suggested for the characteristic length of two parallel plates for an upper heated surface:

$$l_c = \frac{\Delta x_b \Delta z_b}{2(\Delta x_b + \Delta z_b)}. \tag{7.22}$$

The Nusselt number is given by

$$Nu = 0.15(Raf_2)^{1/3}, \tag{7.23}$$

with

$$f_2 = (1 + (\frac{0.322}{Pr})^{11/20})^{-20/11}, \tag{7.24}$$

where Pr is the Prandtl number. The heat transfer coefficient may be computed using the Nusselt number.

The Rayleigh number is temperature dependent. Typical ΔT values (5°C and 10 °C for lower and upper heated surfaces respectively) were taken to compute the value of the two heat transfer coefficients. Fluid properties were taken at a temperature of 15 °C. It was found that the heat transfer coefficients are a similar order of magnitude, but the correlation for the upper heated plate predicts lower heat transfer. Thus, this correlation is used for the heat transfer coefficient of water to avoid overprediction of heat transfer.

7.2.2. Convection in air

The mode of heat transfer in the air is natural convection. As the ceiling temperature remains above the soil temperature for the whole duration of the year, the same correlation as described in Section 7.2.1 [28] was used to obtain the heat transfer coefficient. An expected temperature difference value of ΔT = 10 °C was used to calculate the Rayleigh number. Fluid properties were taken at a temperature of 15 °C.

7.2.3. Radiation

To calculate the radiative heat transfer occurring between the ceiling surface, bag and floor, an approximation for determining a radiative heat transfer coefficient from the was used [47]:

$$h_r = 4\sigma\epsilon(\frac{T_1 + T_2}{2})^3, \tag{7.25}$$

where h_r is the heat transfer coefficient (W/m^{2}K), σ is the Stefan-Boltzmann constant, ε is the emissivity of the surface and T_1 and T_2 are the surface temperatures between which radiation is occurring (K).

This approximation is only valid under the temperature condition $\frac{3}{4} < \frac{T_1}{T_2} < \frac{4}{3}$. It is expected that T_{bet}, T_c and T_f will always be between 10-15 °C. With this assumption, the use of this approximation is valid for most use cases.

When calculating the heat transfer due to radiation, a shape factor must be taken into consideration. As the model is one dimensional and side wall effects were not taken into account, the geometry to calculate the shape factors was simplified. Figure 7.3 displays this geometry, where

- **1** - Δx_{cs}
- **2** - Δx_b
- **3** - $\Delta x_f = \Delta x_{cs} - \Delta x_b$.

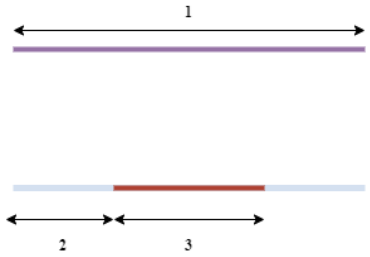


Figure 7.3: Crawlspace geometry

The shape factor matrix is in the following form:

$$F_{ij} = \begin{Bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33}. \end{Bmatrix}$$

From Figure 7.3 it is seen that $F_{11}, F_{22}, F_{33}, F_{23}, F_{32} = 0$. Furthermore, $F_{21}\ and\ F_{31} = 1$. To find the remaining components of the shape factor matrix, the rule

$$A_1 F_{12} = A_2 F_{21} \quad and \quad A_1 F_{13} = A_3 F_{31}$$

is used. The following shape factor matrix is obtained:

$$F_{ij} = \begin{Bmatrix} 0 & \frac{\Delta x_{cs}\Delta z_{cs} - \Delta x_b \Delta z_b}{\Delta x_{cs}\Delta z_{cs}} & \frac{\Delta x_b \Delta z_b}{\Delta x_{cs}\Delta z_{cs}} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{Bmatrix}.$$

7.2.4. Inflow and outflow of energy

Thus far, the numerical model of the wastewater bag is able to capture the heat gains and losses to the surrounding area of the crawlspace. The inflow of energy from additional wastewater being added to the system and outflow from the HP extracting from the system must be modelled.

7.2.4.1. Inflow - domestic wastewater entering system

The mass, temperature and time profile at which domestic wastewater is produced in a home is described in Section 10.1. The wastewater passes through DeWarmte's filtering system, losing energy and mass, as described in Section 9.4. It then enters the wastewater bag at a certain mass flow and temperature.

It is assumed that the bag is well-mixed and therefore the addition of this energy contributes to the temperature change of the whole bag. Thus, the inflow of energy to the bag is given by

$$q_{in} = m_{DHW,in} C_p (T_{DHW,in} - T_w), \tag{7.26}$$

where $m_{DHW,in}$ is the mass of the incoming wastewater (kg), C_p is it's specific heat, assumed to be that of water (J/kgK), $T_{DHW,in}$ is the temperature of incoming wastewater (K) and T_w is the wastewater temperature in the bag (K), calculated as described in Section 7.1.1.

7.2.4.2. Outflow - heat extraction by heat pump

When the HP is turned on, water is pumped from the bag to the HEX that transfers heat from wastewater to glycol. The glycol is then pumped to the HP evaporator. The numerical modelling of this HEX is described in Section 9.1.

Thus, the outflow of energy from the wastewater is described by

$$q_{out} = m_{hex,out} C_p (T_w - T_{hex,out}), \tag{7.27}$$

where $m_{hex,out}$ is the mass of wastewater leaving the bag (kg), $T_{hex,out}$ is the temperature of wastewater leaving the HEX (K).

For each timestep *n* of the simulation, q_{in} and q_{out} are calculated. When no wastewater enters the system or the HP is turned off, q_{in} and q_{out} are set to zero respectively. Furthermore, the new mass of the bag is also calculated, using the previous timestep's value and current timestep's inflow and outflow mass. Therefore, a new T_w is calculated using

$$T_w^n = T_w^n + \frac{q_{in} - q_{out}}{m_{bag} C_p}. \tag{7.28}$$

7.2.5. Control strategy

The current HeatCycle control strategy is implemented into the numerical model;

- To prevent freezing of the wastewater, the HP will turn off once the water in the bag is 2 °C. It is assumed that all possible heat is extracted from the wastewater bag, therefore its contents will be dumped to sewage.
- To prevent pump malfunction, a minimum amount of mass is required to remain in the bag. Therefore, dumping will stop once this minimum is reached. This mass is set to 20 kg.
- Once the maximum volumetric capacity of the storage bag is reached, excess wastewater will not enter the system and instead will be dumped to sewage. This capacity is set to 140 liters.

7.3. Experimental validation

The experimental validation of the numerical model provides insight into whether the model predicts the heat transfer from the bag to its surroundings correctly. Furthermore, when modelling the water temperature as a single, well-mixed node, two assumptions are made about the processes within the bag: it is assumed that the water in the bag does not become thermally stratified and that there are no hot or cold spots in the bag when the HP is on or when wastewater is flushed into the bag. These assumptions, alongside the heat transfer from the bag's surroundings, are validated in a single experiment. Temperatures throughout the bag are measured in a test during which a HP extracts heat from the bag. This experiment replicates the working conditions of the wastewater bag.

7.3.1. Experimental setup

To test the temperatures throughout the bag, an array of sensors must be fit throughout the bag. Figure 7.4 displays the positions and labels of the sensors.

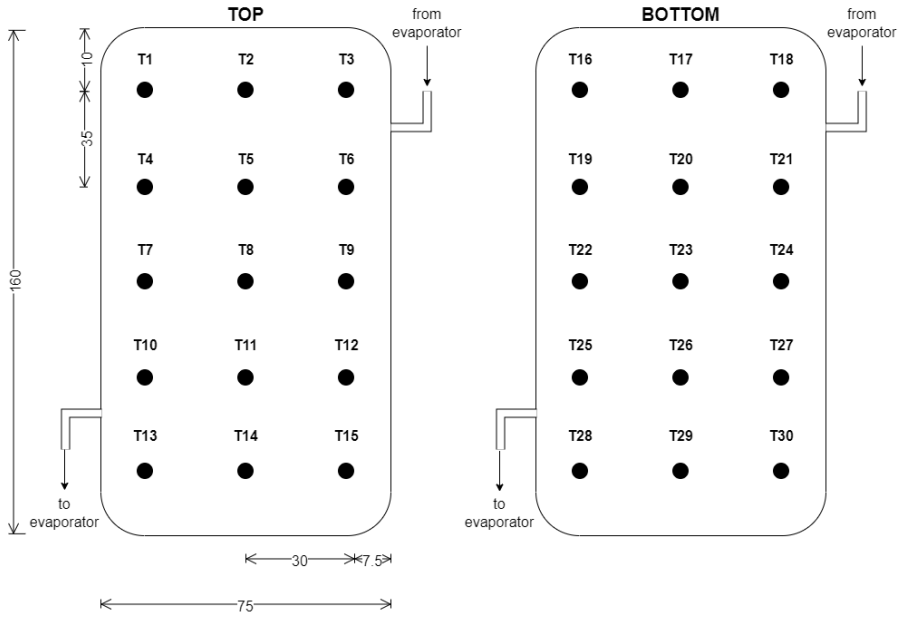


Figure 7.4: Location of temperature sensors on wastewater bag (measurements given in cm)

The used temperature sensors were DS18B20 sensors with an M8 thread size. A custom-made PVC bag used in DeWarmte's commercial installations was used for the experiment. Holes with an M6 drill bit were made for the temperature sensors. A sensor was placed in each hole. The following components, displayed in Figure 7.5, were used to ensure that the system was water-tight:

- **M8 rubber ring**, placed onto the thread of the temperature sensor, between its base and the top of the bag. The sensor and rubber ring are pushed into the hole in the bag
- **M8 washer**, placed onto the thread of the temperature sensor, between the inside of the bag and the nut.
- **M8 nut**, placed onto the thread of the temperature sensor, under the washer. The nut is tightened to ensure the bag is watertight.

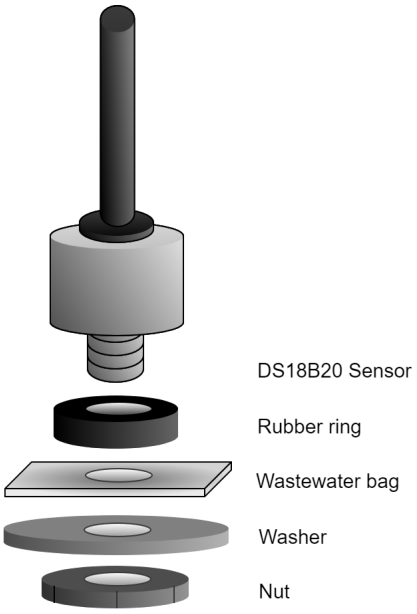


Figure 7.5: Components used to ensure bag is watertight

Temperature sensors were connected to DeWarmte's custom sensor HAT and used with a Raspberry Pi. Ceiling and floor sensors were covered in thermal insulation tape. The bag was connected in a standard HeatCycle testing system, consisting of the wastewater bag, HP and boiler. The system is displayed in Figure 7.6. The specifications of each component is displayed in Table 7.2.

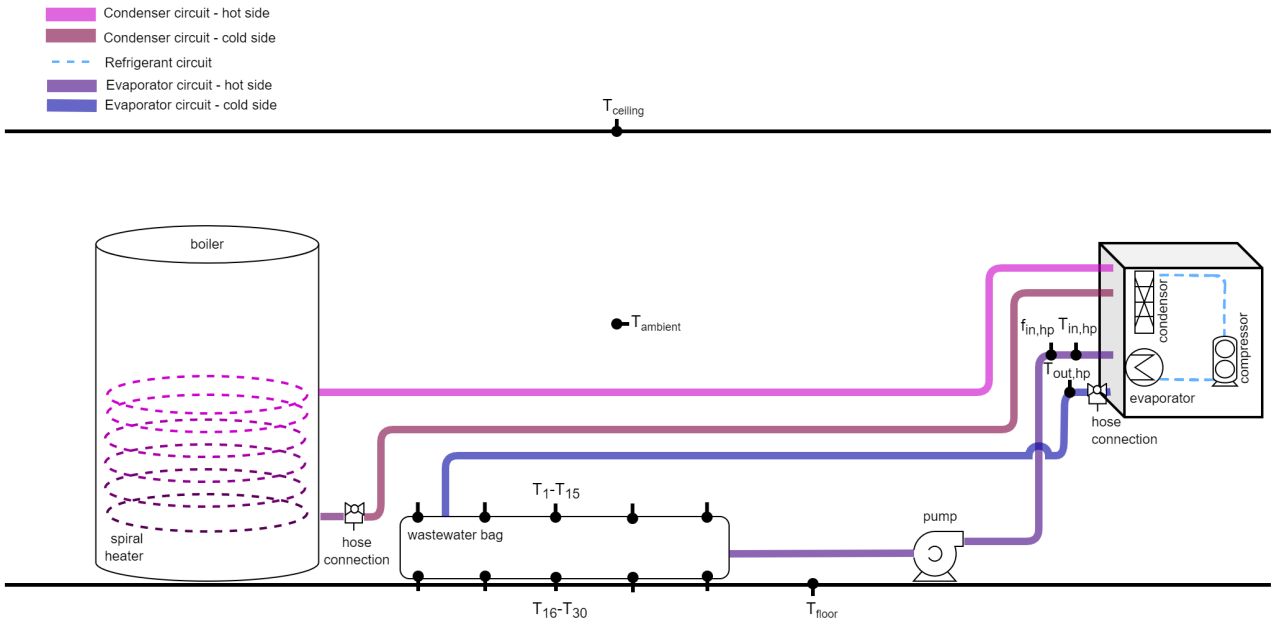


Figure 7.6: Setup of wastewater bag validation experiment

Table 7.2: Water bag experiment - Components and sensors used

Component Type	Model	Error
Boiler	DeJong M150	-
Pump	GreenPro EAB circulating pump	-
Heat Pump	DeWarmte WP0	-
Wastewater Bag	DeWarmte custom PVC bag	-
Temperature sensor	DS18B20	±0.5 °C
Flow sensor	Huba Control 210	±1%

7.3.1.1. Methodology

Initially, the water bag was fully filled with tap water and allowed to reach room temperature of 25 °C. The HP was turned on, cooling the bag and heating the boiler. The HP was allowed to run until it heated the boiler to its set-point temperature of 55°C. Temperatures and flow rates were monitored and logged using 2 sensor HATs, 2 Raspberry Pis and DeWarmte's in-house code. The experiment was repeated 3 times.

Once the experimental data was obtained, temperature and flow readings were used as inputs in the model to test the calculated temperatures against measured temperatures. Missing experimental values for a timestep were filled-in using linear interpolation between the previous and next timestep value. Temperatures $T_{ceiling}$, $T_{ambient}$, T_{floor}, $T_{in,hp}$, $T_{out,hp}$ and flow $f_{in,hp}$ are inputs in the numerical model. T_{floor} is used as a replacement to the constant soil temperature boundary condition, $T_{s,in}$. It is assumed that the measured floor temperature is equal to the floor temperature 20 cm below the floor surface. This assumption is considered valid due to the large size of the laboratory and therefore its floor being largely unaffected by the change in temperature in the bag.

The model was run and experimentally obtained temperatures T_1 - T_{30} are compared amongst themselves and with numerically calculated temperature T_w.

7.3.2. Results

Figure 7.7 displays the temperatures of the bag over time for all conducted experiments. Attention must be brought to the high temperatures recorded by sensors T_5, T_8 and T_{11}. Their locations are displayed in Figure 7.4. These sensors correspond to the highest point in the bag once it is filled with water. Therefore, in experiments 1 and 2, sensors T_5 and T_8 were in contact with air, instead of water. Due to movement of water within the bag caused by the circulation pump, sensor T_{11} became in contact with air throughout the experiment.

Neglecting the anomalous readings, no significant variation in temperature due to sensor position or thermal stratification can be seen in the heat-maps. A more in-depth analysis must be conducted.

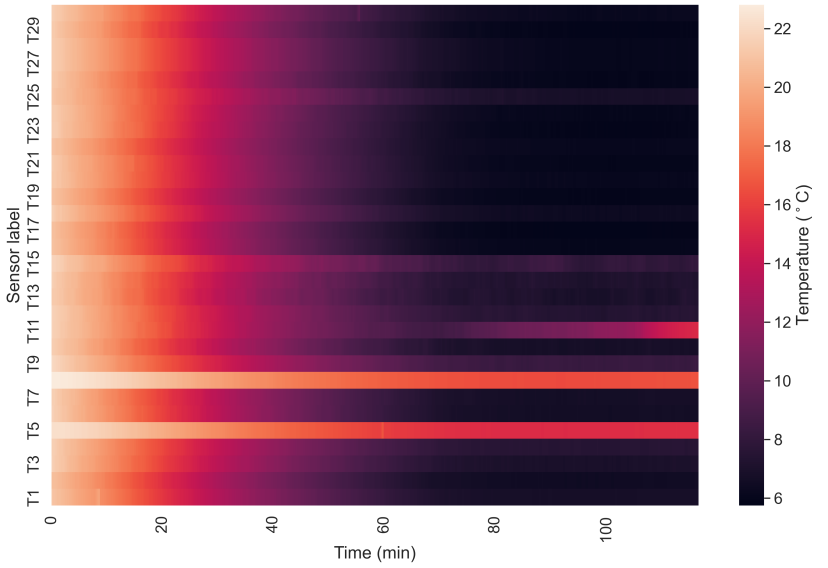
Figure 7.8 displays temperatures T_1 - T_{30} over time. Their mean, alongside the mean of the top and bottom temperature sensors is plotted. Note the anomalous readings of T_5, T_8 and T_{11} are neglected from the calculation of the mean. Figure 7.9 displays the deviation of the sensor readings from the mean, alongside the deviation of the top and bottom mean values. It is clear that the bag becomes thermally stratified as the experiment continues. However, the stratification is not considered significant, reaching a maximum of approximately 2.5 degrees at the end of the experiment.

7.3.2.1. Error analysis

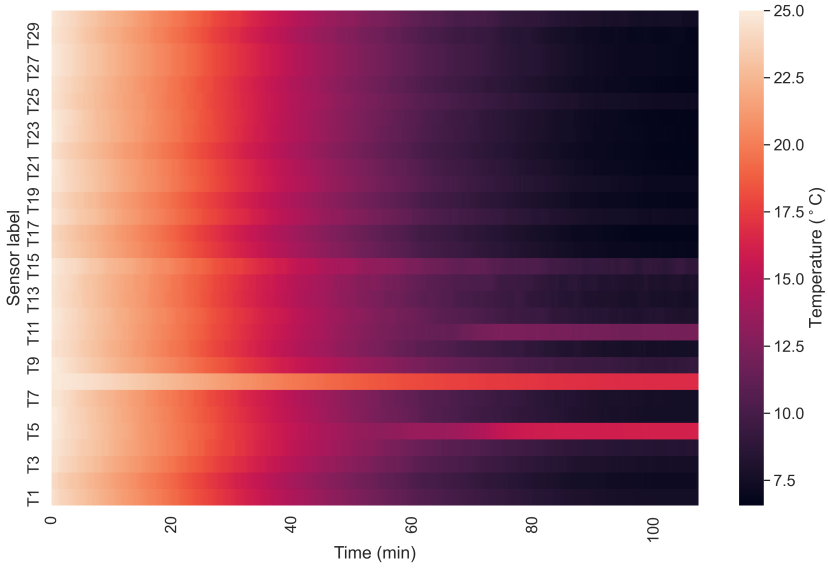
Figure 7.10 displays the comparison between the mean experimental temperature throughout the experiment and the numerically modelled value of T_w. The experimental mean temperature is plotted with its sensor error of ±0.5 °C. Percentage errors between experimental and numerical values are displayed.

Calculated T_w follows closely to the experimental mean. During experiment 2 and 3, T_w stays within the bounds of sensor error for the duration of the experiments. Experiment 1 performs the worst, with the tested temperature staying 0.5 °C above the sensor error bounds for half of the duration of the experiment. Nevertheless, temperatures show close resemblance.

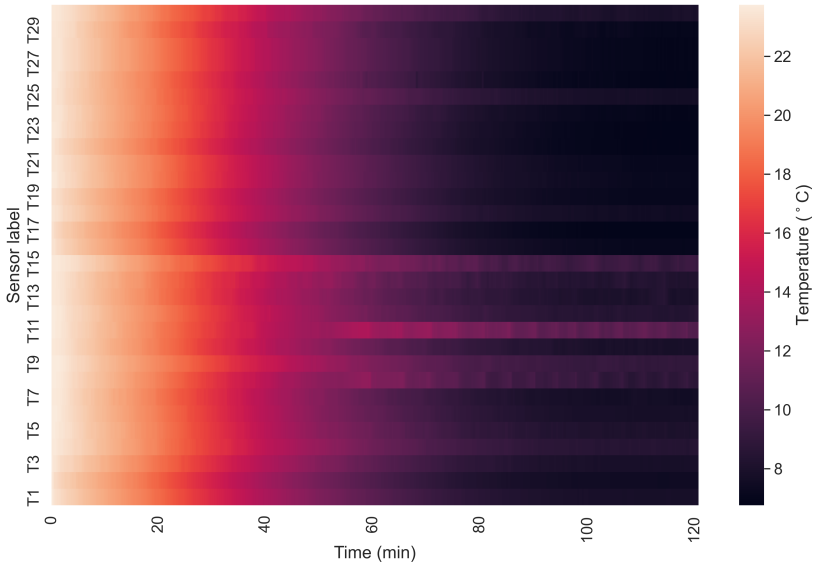
Thus, the numerical model of the wastewater bag is considered well-validated due to its close representation of the mean temperature, and therefore energy transfer, within the bag during a HP cycle. Furthermore, it is found that there is no significant variation in temperature throughout the bag. Some stratification is present, with a maximum of 2.5 °C difference. Maximum temperature deviations from the mean due to sensor positions at the top and bottom of the bag are 2 °C and 1 °C respectively. These variations cause hot or cold zones in the bag and thus may fail to accurately predict the amount of heat being extracted by the HP. However, due to the relatively small temperature variations and the computational power required to model varying temperature zones within the bag, this error is considered acceptable for the numerical model.



(a) Experiment 1

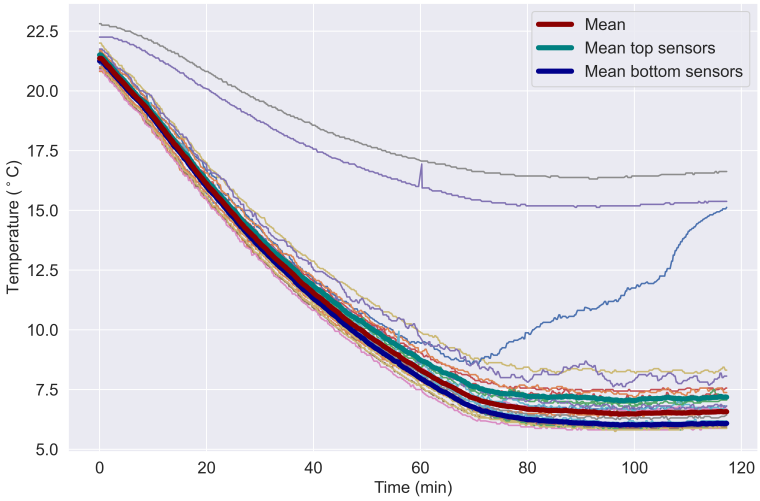


(b) Experiment 2

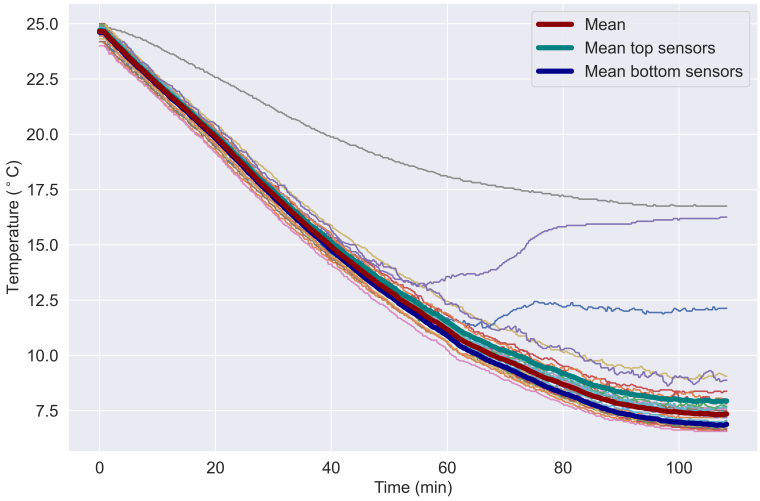


(c) Experiment 3

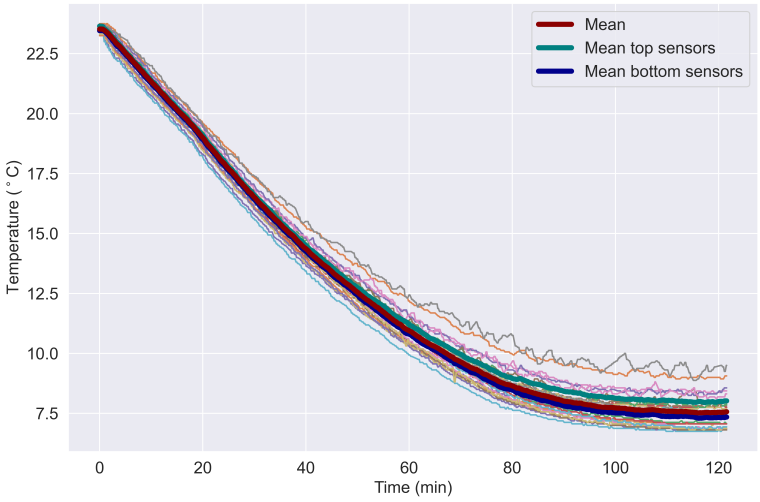
Figure 7.7: Heat-map of bag temperatures over time



(a) Experiment 1

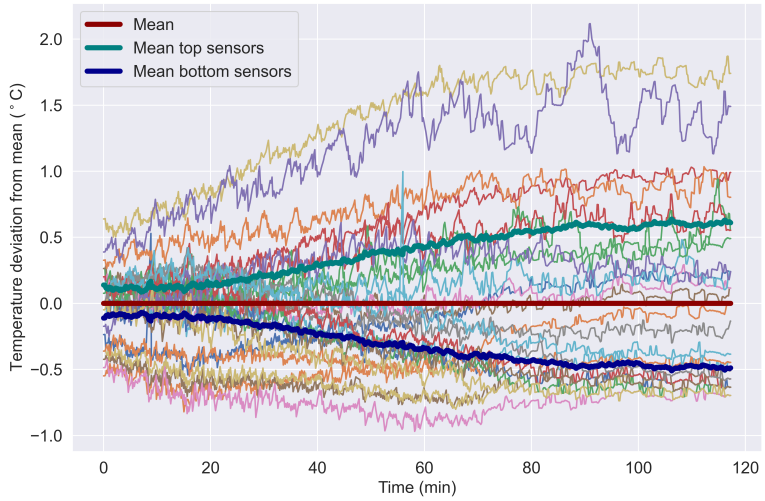


(b) Experiment 2

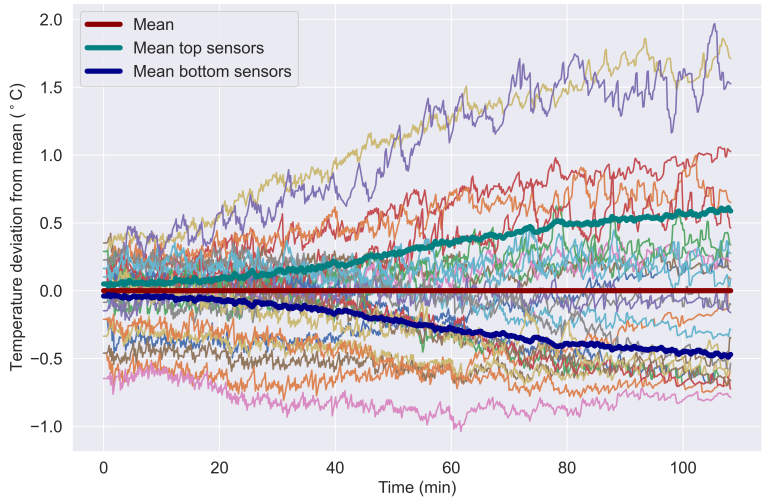


(c) Experiment 3

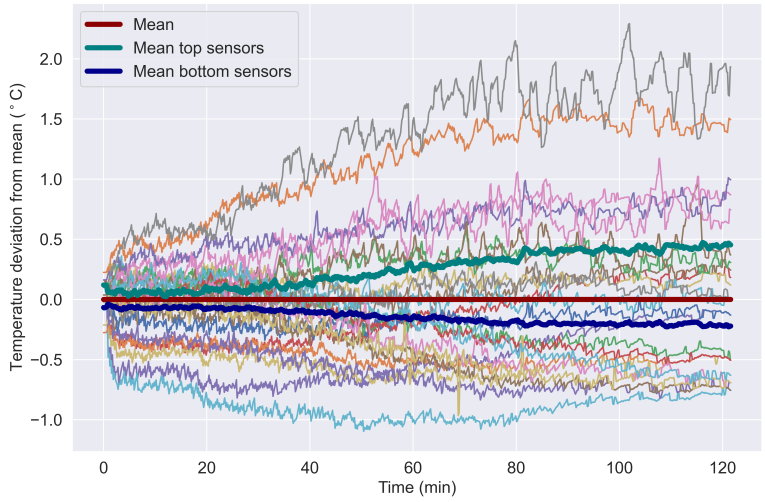
Figure 7.8: Bag temperatures over time



(a) Experiment 1

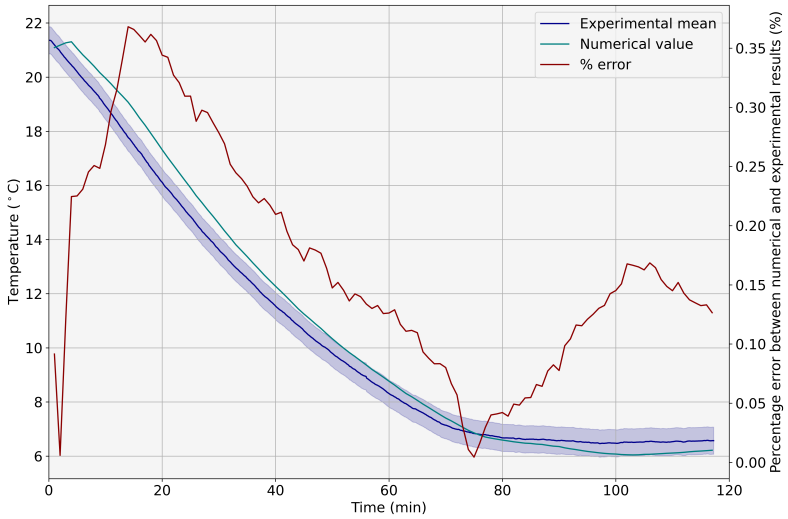


(b) Experiment 2

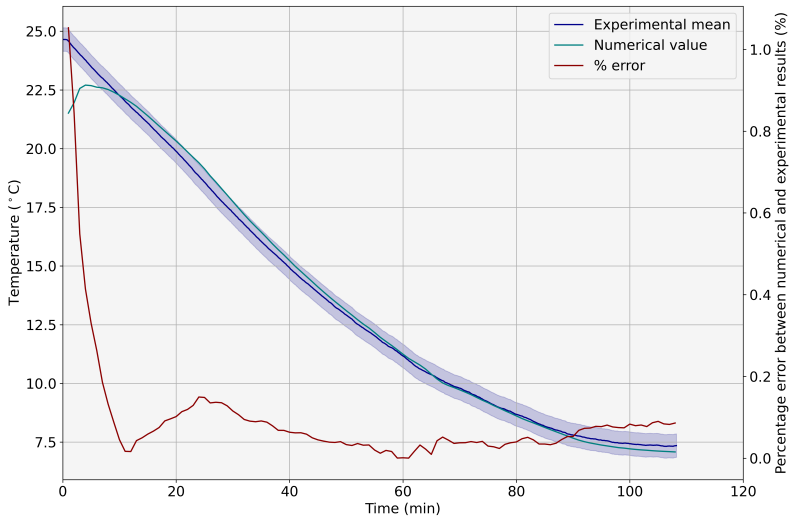


(c) Experiment 3

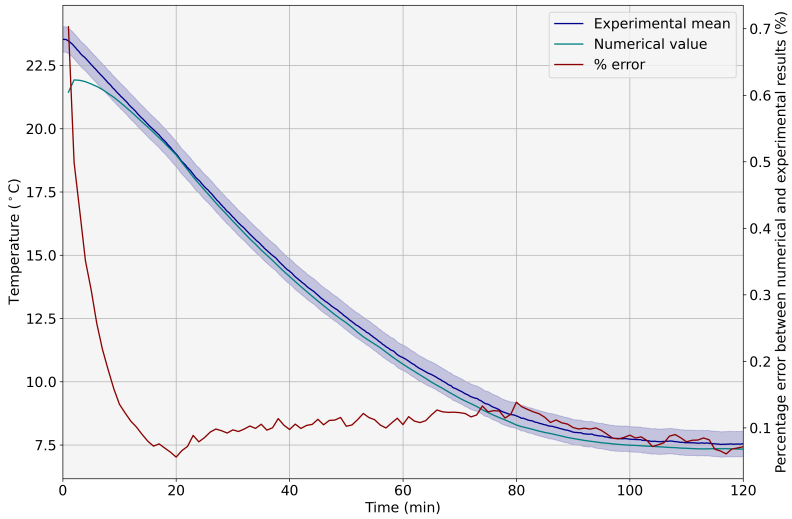
Figure 7.9: Bag temperature deviations from the mean over time



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 7.10: Comparison between mean experimental bag temperature and numerical T_w

7.3.2.2. Sensitivity analysis

A Sobol sensitivity analysis was conducted on the wastewater bag model. For theoretical background on the method, refer to Section 5.3.2.2. The tested output was the water temperature T_w in the bag. The tested inputs were outlet temperature of heatpump $T_{out,hp}$, HP mass flow $\dot{m}_{out,hp}$, ambient air temperature $T_{ambient}$, ceiling temperature $T_{ceiling}$ and floor temperature T_{floor}. The sensor errors as described in Table 7.2 were used as upper and lower bounds for the inputs. The sensitivity analysis was performed for all three experiments, the results of which are displayed in Figure 7.11.

The analysis shows that variations in all inputs have an effect on the output. Variations in temperatures in the surroundings of the bag, T_{floor}, $T_{ceiling}$, $T_{ambient}$ have the largest effect, meaning the variations in heat gain from the ambient are more significant than variations in the cooling effect of colder inflow of water from the HP. Variation in mass flow of cooled water has more impact on the output than its temperature, the reason for which is unknown. Considering the heat removed by the HP: $\dot{m}\ C_p\ \Delta T$, a flow sensor error of 1% causes an equivalent 1% fluctuation in its value. Thus, ΔT would need equal 50 °C for the temperature sensor error of ±0.5 °C to cause an equivalent 1% difference in its value. ΔT is significantly lower in the system, thus the temperature error should have a higher effect on output's variation. However, as the model has very close correlation to experimental results, this result is not looked into further in this study. However, deeper analysis should be conducted.

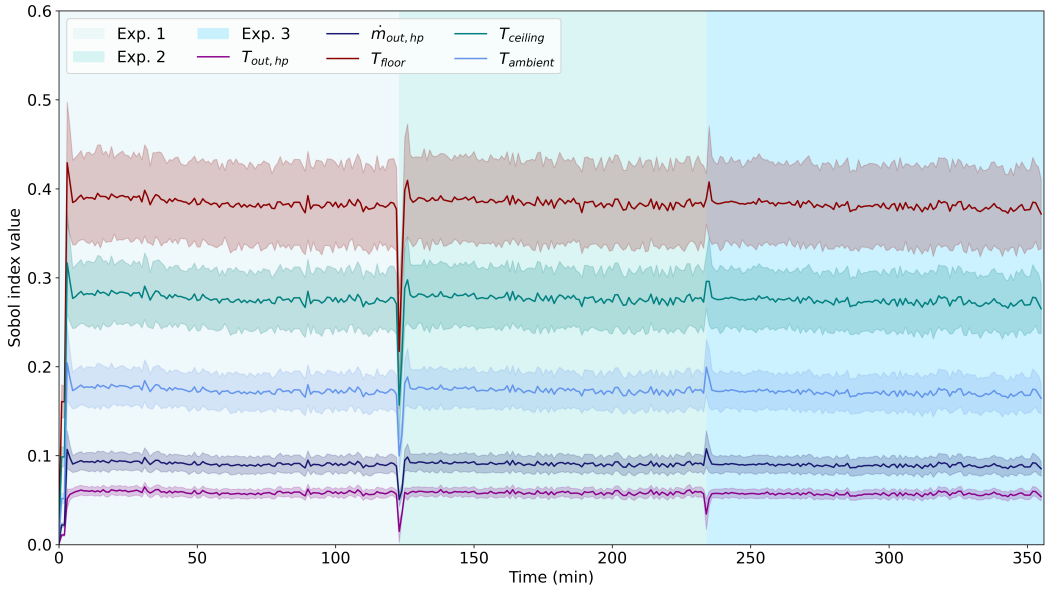


Figure 7.11: Sobol analysis of waterbag model

8

Boiler model

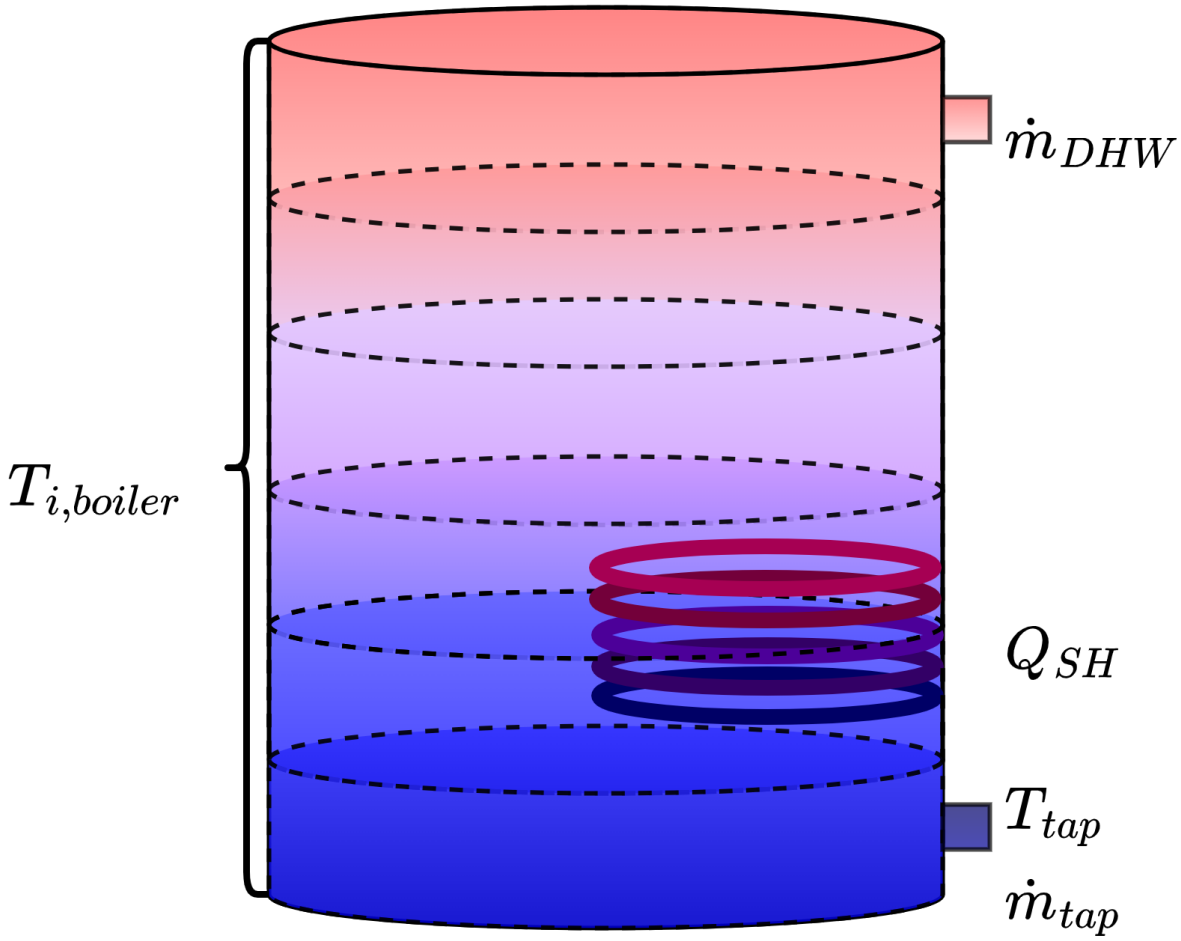


Figure 8.1: Boiler class

Table 8.1: Inputs and outputs of boiler class

Variable	Unit	Type	Description	Source
$\mathbf{Q_{SH}}$	**W**	Input	Heat output from spiral heater	Spiral heater class
$\mathbf{T_{tap}}$	**K**	Input	Temperature of incoming tap water	User input
$\mathbf{\dot{m}_{tap}}$	**kg/s**	Input	Mass flow rate of incoming tap water	User input
$\mathbf{\dot{m}_{DHW}}$	**kg/s**	Input	Mass flow rate of outgoing DHW	DHW class
$\mathbf{T_{i,boiler}}$	**K**	Output	Temperature of each node in boiler	-

8.1. Numerical modelling approach

A commonly used heat transfer modelling approach for boilers or storage vessels is that of 1-dimensional heat transfer. In this method, the boiler model is split into equal volumes. Within these volumes, it is assumed that the temperature remains constant. Each volume's temperature is represented by a node at which the temperature is calculated. TRNSYS, a commonly used software for building and energy simulations, uses this method with varying spatial and time-stepping schemes, depending on computational time requirements [61]. Furthermore, research papers using the method have seen good correlation with experimental results [39], [19]. It should be noted that these papers built upon the approach, by adding a new function to encompass thermal stratification [39] and an empirical mixing function for cold tap water inflow [19]. Aguilar et al. [19] performed a spatial and temporal independence study and found that 100 nodes and a 1 minute timestep is sufficient to model the temperature in the boiler accurately. Thus, the boiler in this model consists of 100 nodes. The performance model is already calculated at 1 minute timesteps. Inflow of tap water occurs at the bottom node (i = 0) at 10 °C and outflow of hot water occurs at the top node (i = i_{max}).

8.2. Energy balance equations

Middle nodes

Figure 8.2 displays the energy changes in the middle nodes (i = 1 to i = i_{max} - 1). $\mathbf{Q_{loss}}$ is the energy lost to ambient air. $\mathbf{Q_{HP}}$ is the heat gained from the spiral heater connected to the HP. $\mathbf{Q_{in}}$ and $\mathbf{Q_{out}}$ represent the energy transfer due to conduction between nodes and mass transfer when tap water enters and hot water leaves the boiler.

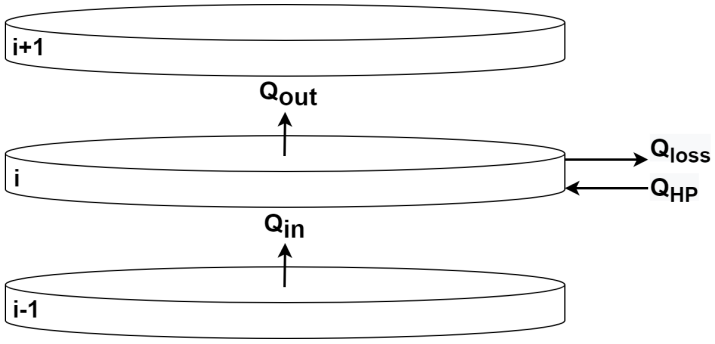


Figure 8.2: Energy balance in middle nodes of boiler

In equation form, the energy balance for the middle nodes is given by

$$\underbrace{\rho C_p \frac{\partial T_i}{\partial t}}_{\text{Temporal change in CV's energy}} = \underbrace{k \frac{\partial^2 T_i}{\partial z^2}}_{\text{Energy exchange due to conduction}} + \underbrace{\frac{\dot{m}_{tap} C_p}{A_i \Delta z} (T_{i-1} - T_i)}_{\text{Energy exchange due to mass flow}} + \underbrace{\frac{R_{side}}{A_i \Delta z} (T_i - T_{ambient})}_{\text{Insulation losses}} + \underbrace{\frac{Q_{HP,i}}{A_i \Delta z_i}}_{\text{Heat added by spiral heater}}, \tag{8.1}$$

where $\dot{m}_{tap}$ is the mass flow of cold water entering the boiler (kg/s), A_i is the CV's cross sectional area (m^2), Δz is the height of the CV (m), Δx, R_{side} is the estimated overall heat transfer coefficient of the side wall (W/K), given by:

$$R_{side} = \frac{A_{side} h_{conv,side} k}{\Delta x h_{conv,side} + k}, \tag{8.2}$$

where Δx is the insulation layer thickness, A_{side} is the CV's side-wall area and $h_{conv,side}$ is the convective heat transfer at the outside of the boiler (W/m^2K). Awbi [20] conducted a numerical and experimental study on the heat transfer coefficient in a room and its walls for thermal modelling of buildings. It was found that the heat transfer remains between 3 to 4 W/m^2K for a large range of temperature differences between room and ambient air. Thus, 3.5 W/m^2K is taken as $h_{conv,side}$.

The equation is discretized using Euler implicit timestepping. This method is chosen to avoid computational complexity, but maintain stability of the solution. The discretized equation reads

$$\rho C_p \frac{T_i^{n+1} - T_i^n}{\Delta t} = k \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta z^2} + \frac{\dot{m}_{tap} C_p}{A_i \Delta z} \left(T_{i-1}^{n+1} - T_i^{n+1}\right) + \frac{R_{side}}{A_i \Delta z} \left(T_i^{n+1} - T_{ambient}\right) + \frac{Q_{HP,i}^{n+1}}{A_i \Delta z_i} \tag{8.3}$$

Top node

The energy balance of the top node changes with an additional loss term at the top lid of the boiler, displayed in Figure 8.3.

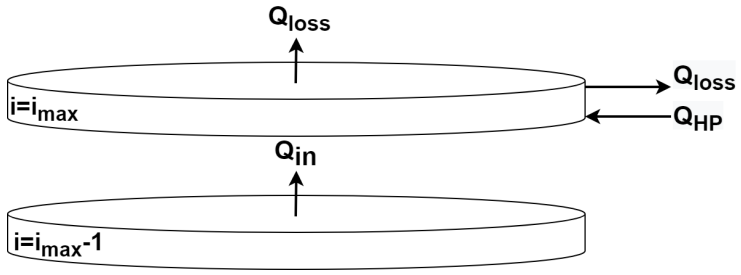


Figure 8.3: Energy balance in top node of boiler

The discretized equation reads

$$\rho C_p \frac{T_{i_{max}}^{n+1} - T_{i_{max}}^n}{\Delta t} = k \frac{T_{i_{max}-1}^{n+1} - T_{i_{max}}^{n+1}}{\Delta z^2} + \frac{R_{top}\left(T_{i_{max}} - T_{ambient}\right)}{\Delta z} + \frac{\dot{m}_{tap} C_p}{A_i \Delta z} \left(T_{i_{max}-1}^{n+1} - T_{i_{max}}^{n+1}\right) + \frac{R_{side}}{A_i \Delta z} \left(T_{i\,max}^{n+1} - T_{ambient}\right) + \frac{Q_{HP,i}^{n+1}}{A_i \Delta z_i}, \tag{8.4}$$

where R_{top} is the lumped heat transfer coefficient (W/K)at the boiler top, consisting of a conductive and convective component. It is given by

$$R_{top} = \frac{A_{top} h_{conv,top} k}{\Delta x h_{conv,top} + k}, \tag{8.5}$$

where A_{top} is the boiler's top area, $h_{conv,top}$ is the convective heat transfer coefficient at the top of the boiler (W/m^2K). [20] predicts a floor heat transfer coefficient between 3.5-5 (W/m^2K) and a ceiling heat transfer coefficient of approximately 0.5 (W/m^2K), depending on the temperature difference between room and the ambient air temperature. As the boiler lid is of the same orientation as these surfaces and is approximately midway between them, an average h_{conv} of 2.5 (W/m^2K) is used.

Bottom node

It is assumed that the bottom of the boiler is perfectly insulated to the ground. The energy balance of this node is displayed in Figure 8.4. The assumption is made to prevent additional complexity in the model's user inputs, as boilers may be placed in different locations in a home, requiring varying boundary conditions. Furthermore, this assumption will not affect the results drastically, as, due to stratification, the bottom of the boiler remains at lower temperatures, thus decreasing the insulation losses.

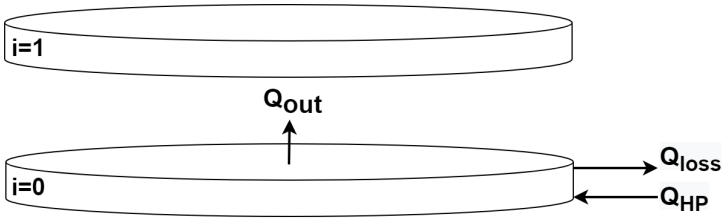


Figure 8.4: Energy balance in bottom node of boiler

Thus, in discretized form, the energy balance is given by

$$\rho C_p \frac{T_0^{n+1} - T_0^n}{\Delta t} = k \frac{T_1^{n+1} - T_0^{n+1}}{\Delta z^2} + \frac{\dot{m}_{tap} C_p}{A_i \Delta z} \left(T_{tap}^{n+1} - T_0^{n+1}\right) + \frac{R_{side}}{A_i \Delta z} \left(T_0^{n+1} - T_{ambient}\right) + \frac{Q_{HP,i}^{n+1}}{A_i \Delta z_i}, \tag{8.6}$$

where T_{tap} is the cold tap water temperature, entering at the bottom of the boiler (K).

8.2.1. Matrix implementation

The following equation is set up to solve the system of equations implicitly

$$[M_{ij}] \cdot \{T_i^{n+1}\} = \{T_i^n\} + \{d\} \tag{8.7}$$

Where $\{T_i^{n+1}\}$ is a vector containing all temperature nodes at timestep n+1, $\{T_i^n\}$ is a vector containing all temperature nodes at timestep n, {d} is a vector containing all constants, as derived from Equations 8.3 - 8.6 and $[M_{ij}]$ is a matrix containing coefficients that must be multiplied by $\{T_i^{n+1}\}$. Thus, the final equation resembles

$$\begin{bmatrix} b & c & 0 & 0 \\ a & b & c & 0 \\ 0 & a & b & c \\ \dots & \dots & \dots & \dots \\ 0 & a & b & c \\ 0 & 0 & a & b \end{bmatrix} \cdot \begin{Bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ \dots \\ T_{i_{max}-2}^{n+1} \\ T_{i_{max}-1}^{n+1} \end{Bmatrix} = \begin{Bmatrix} T_0^n \\ T_1^n \\ T_2^n \\ \dots \\ T_{i_{max}-2}^n \\ T_{i_{max}-1}^n \end{Bmatrix} + \begin{Bmatrix} d_0 \\ d_1 \\ d_2 \\ \dots \\ d_{i_{max}-2} \\ d_{i_{max}-1} \end{Bmatrix}, \tag{8.8}$$

where coefficients a, b, c are found by rearranging Equations 8.3 - 8.6. The matrix is inversed to solve for $\{T_i^{n+1}\}$.

8.3. Mixing function

Thus far, the presented model does not simulate buoyancy effects causing thermal stratification in the boiler. Buoyancy is an effect of mass transfer, therefore modelling solely energy transfer is not sufficient to account for it and computational fluid dynamics are required. To prevent the additional complexity and computational power requirements, a mixing function is implemented into the model. The function assumes that if a CV is at a temperature higher than that of the CV volume above it, the two CVs will mix to their average temperature. The mixing function continues to loop through CVs until the boiler is thermally stratified. The following logic is used to implement the function:

$$\begin{aligned} while \quad min\left((diff\left(\{T_i^{n+1}\}\right)\right) \leq -\epsilon: \\ for \quad i \quad in \quad range(1, i_{max}): \\ if\, T_{i-1} \geq T_i: \\ T_{i-1} = T_i = \frac{T_{i-1} + T_i}{2}, \end{aligned}$$

where min is a function that finds the minimum (negative) value of an array, $diff$ is a function that subtracts the next element from the previous one in an array, returning an array of the differences and ε is a user-defined acceptable temperature error between T_i and T_{i+1}.

8.4. Boiler control

The final output of the boiler's numerical model is an array of temperatures throughout its height. This output is used to dictate whether the HP turns on and/or is set to the DHW setting. The control implemented is that of the current HeatCycle system. At a user-specified height, a temperature sensor is placed in the boiler. If this sensor logs a temperature that is below a lower threshold value, the HP is turned on and heats the boiler until the sensor logs an upper threshold value. Throughout this time, the HP is considered to be in DHW mode. The HP's prioritization of DHW or SH mode is described in Section 5.2. The model is run using a lower and upper threshold of 45 to 50 °C respectively. The user-specified sensor height is transformed into the corresponding boiler node number and is used to monitor the temperature.

A second control is required in the boiler. DHW may be required and drawn from the boiler even if it is not at a high temperature. Thus, another user-variable is introduced; the temperature at which DHW must enter the system. This temperature is reached by means of an auxiliary heater. This may be a gas-fired boiler or an electric heater. Further analysis of the cost-effectiveness and efficiency of these solutions will be discussed in Section 11. The energy required to reach the setpoint temperature is found with

$$Q = \dot{m}_{DHW} C_p (T_{setpoint} - T_{imax}). \tag{8.9}$$

The current set point in the model is set to 55 °C.
Finally, incoming tap water is set to a temperature of 10 °C.

8.5. Experimental validation

The boiler model is experimentally validated using data from an experiment conducted by Aguilar et al [19]. The study developed a numerical model of a boiler being drained over the course of a day, in accordance to tapping cycle's set out by EU standard EN 16147:2017. This standard concerns performance ratings of HPs for domestic use. The model was compared to experimental results following the same tapping cycle.

8.5.1. Experimental data

The tapping cycle used is displayed in Table 8.2. The boiler is initially heated to 55 °C. Tapping occurs at the specified flow rate for 5 minutes and 30 seconds at the specified times.

Table 8.2: Tapping cycle in accordance to EN 16147:2017

Tapping cycle	Flow rate	7:00	8:00	9:00	13:00	19:15	20:15
TC1—6×22 L	4 L/min	22 L	22 L	22 L	22 L	22 L	22 L

Tables 8.3 and 8.4 displays the relevant experimental components, conditions and errors to use as inputs in the model.

Table 8.3: Boiler specifications

Component	Capacity	Dimensions	Insulation thickness	Insulation conductivity	Steel wall thickness	Steel wall conductivity
Boiler	190 L	ϕ0.47 m x 1.1 m	0.045 m	0.04 W/m2K	0.003 m	50 W/m2K

Table 8.4: Temperature sensor specifications

Sensor	Measured value	Set value	Error
Incoming tap water temperature	-	15 °C	±0.05 °C
Ambient air temperature	19-23 °C	-	±0.3 °C
Boiler temperature	-	-	±1.5 °C

8.5.2. Error analysis

Results between experimental and numerical values were compared. Figure 8.5 displays the comparison between temperatures at top (94 cm from bottom of boiler) , bottom (16 cm from bottom of boiler) and middle temperature (39 cm from bottom of boiler) of the boiler. Experimental results are plotted with corresponding sensor error. The results are closely aligned. The numerical results fall within the error of the measured temperatures for most of the test.

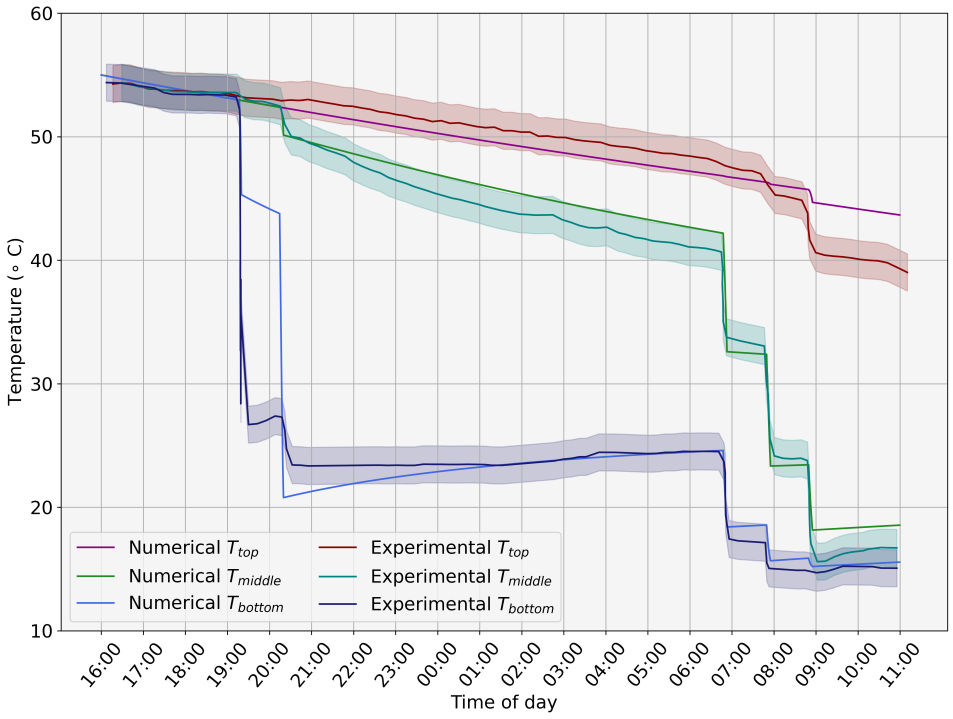


Figure 8.5: Comparison between experimental and numerical temperatures in boiler during tapping cycle

Figure 8.6 displays the percentage error between experimental and numerical results. The largest peak in error is that of the first tapping for the bottom node. Experimentally, the temperature drops to approximately 27 °C, while the numerically-calculated temperature remains at 45°C. The referenced study proposes the implementation of a mixing function for the nodes at the bottom of the boiler, stating that the inflow of cold tap water causes circulation, and thus mixing in the lower part of the boiler. During tapping from a high temperature, this causes a significant temperature change. This may explain the error between numerical and experimental results, as the boiler model implemented in this study does not use the mixing function. The temperatures at the top node vary too. The numerical error displays slightly higher insulation losses and less sensitivity

to the final tappings. The former may be due to the high variability in room temperature in the experiment, while in the model it is set to a constant 23 °C. Furthermore, it could be due to variation in modelling insulation losses in the boiler. The study by Aguilar et al. [19] calculated temperature distribution throughout additional nodes in the boiler wall, thus more accurately encompassing losses. Overall, the numerical results of the boiler show more stark thermal stratification than is present in reality. However, as the error between numerically-calculated and experimental temperatures are within less than 10%, the boiler model is considered validated.

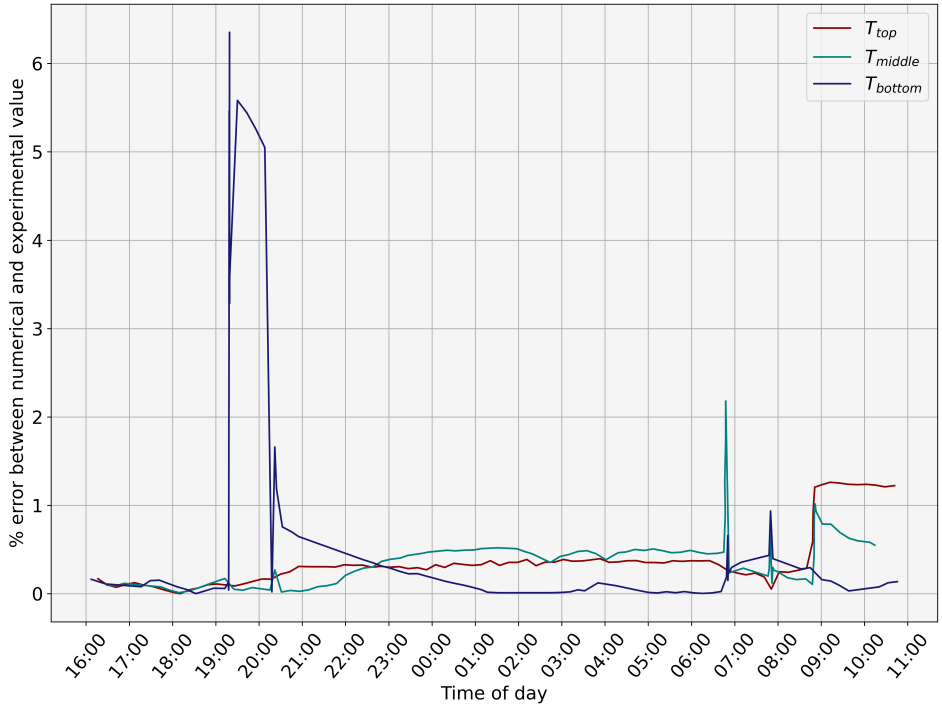


Figure 8.6: Error between experimental and numerical temperatures in boiler during tapping cycle

9

Additional components

9.1. Wastewater-to-glycol heat exchanger model

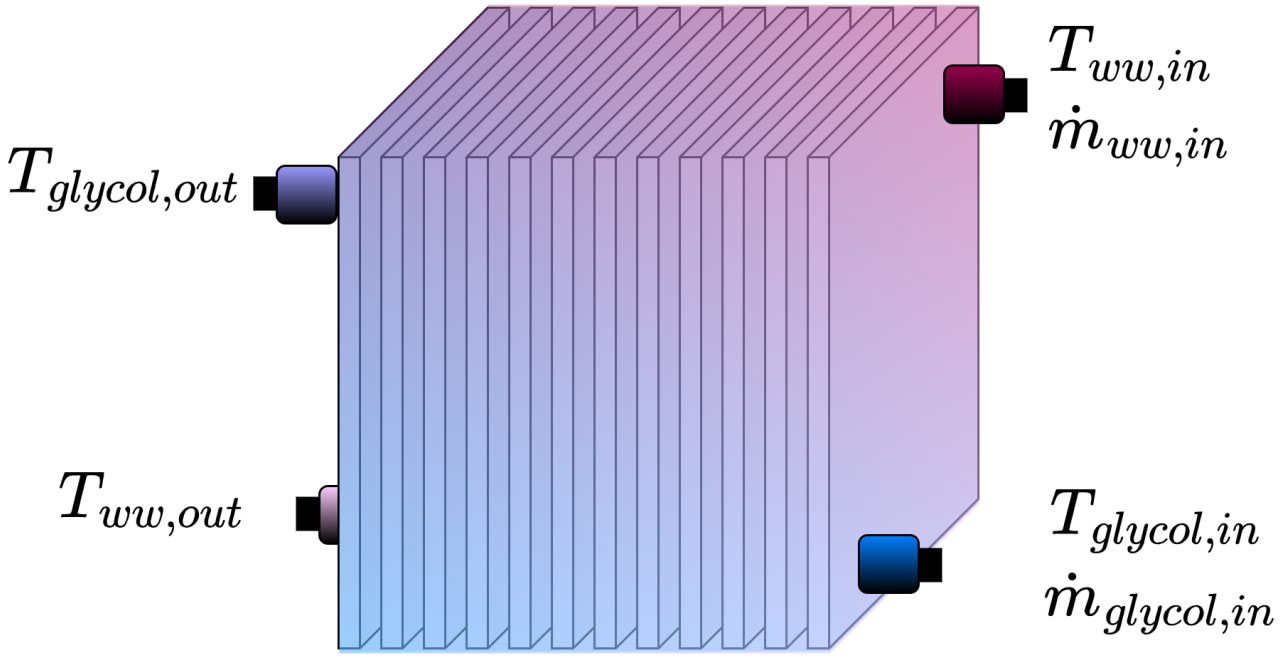


Figure 9.1: Heat exchanger class

Table 9.1: Inputs and outputs of heat exchanger class

Variable	Unit	Type	Description	Source
$T_{ww,in}$	K	Input	Temperature of incoming wastewater	Wastewater bag class
$\dot{m}_{ww,in}$	kg/s	Input	Mass flow rate of incoming wastewater	User input
$T_{glycol,in}$	K	Input	Temperature of incoming glycol	HP class
$\dot{m}_{glycol,in}$	kg/s	Input	Mass flow of incoming glycol	User input
$T_{ww,out}$	K	Output	Temperature of outgoing wastewater	-
$T_{glycol,out}$	K	Output	Temperature of outgoing glycol	-

9.1.1. Heat exchanger model

A plate HEX (PLV6-C10-22D) from AliExpress was modelled to exchange heat between wastewater and glycol. This HEX is used in a pilot by DeWarmte. The HEX was chosen due to its higher performance in comparison to a shell and tube HEX, and large plate spacing, which is needed for wastewater flow.

The NTU method was used in the numerical model. The method is described in Section 5.1.4.1.

For the counter-current flow in a plate HEX, heat exchange between the two fluids is given by

$$Q = \epsilon\ min(\dot{m}C_p)(T_{hot,in} - T_{cold,in}), \tag{9.1}$$

where Q is the transferred heat (W), $T_{hot,in}$ is the incoming hot fluid temperature (K), in this case the wastewater, $T_{cold,in}$ is the incoming cold fluid temperature (K), in this case the glycol and ε is the effectiveness, for a counter-current flow defined as

$$\epsilon = \frac{1 - e^{-NTU\left(1-\frac{C_{p,min}}{C_{p,max}}\right)}}{1 - \frac{C_{p,min}}{C_{p,max}} e^{-NTU\left(1-\frac{C_{p,min}}{C_{p,max}}\right)}} \tag{9.2}$$

Using the calculated Q, the outlet temperatures of wastewater and glycol may be calculated using

$$Q = \dot{m}C_p\Delta T\,. \tag{9.3}$$

9.1.2. Heat transfer coefficients

The overall heat transfer coefficient, U is given by

$$\frac{1}{U} = \frac{1}{h_{ww}} + \frac{1}{h_{glycol}} + \frac{t_p}{k_{copper}}, \tag{9.4}$$

where h_{ww} is the convective heat transfer coefficient (W/m^2K) on the wastewater side, h_{glycol} is the convective heat transfer coefficient (W/m^2K) on the glycol side, t_p is the plate thickness (m) and k_{copper} is the thermal conductivity of copper (W/mK).

Heat transfer coefficients are calculated using Equation 5.22, as described in Section 5.1.4.1.

9.1.3. Experimental validation

The plate HEX between glycol and wastewater has not been experimentally validated. However, it uses the same calculation method and empirical heat transfer coefficient relations as the plate HEX in the condenser of the heat pump, which has been validated. Thus it is assumed that the heat exchange prediction of this model will be sufficiently accurate. However, the component must be validated in further work.

9.2. Spiral heater model

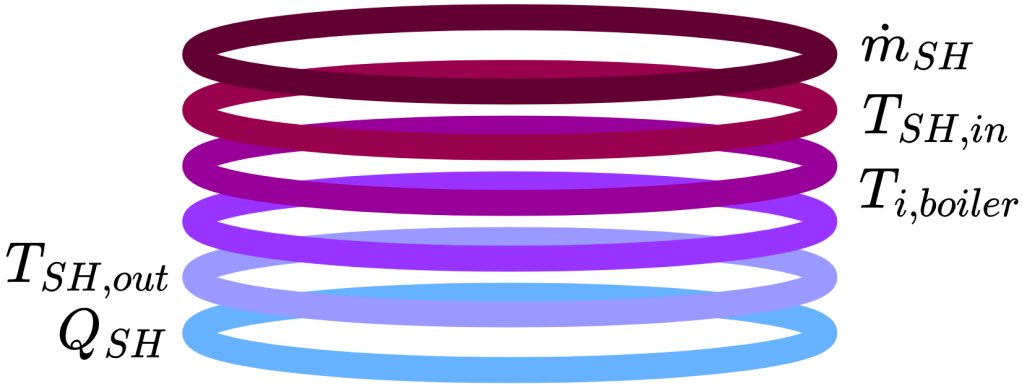


Figure 9.2: Spiral heater class

Table 9.2: Inputs and outputs of spiral heater class

Variable	Unit	Type	Description	Source
$T_{SH,in}$	K	Input	Temperature of incoming water	HP class
$\dot{m}_{SH}$	kg/s	Input	Mass flow rate of incoming water	HP class
$T_{i,boiler}$	K	Input	Temperatures in boiler	Boiler class
$T_{SH,out}$	K	Output	Temperature of outgoing water	-
Q_{SH}	W	Output	Heat released to boiler	-

The performance model developed in this study uses a spiral heater numerical model developed by DeWarmte. The internal document relating to this study [46] may be obtained upon request. In this chapter, the calculation method and its experimental validation will be presented.

9.2.1. Modelling approach of spiral heater by DeWarmte

The spiral heater is based on a model by Abdelsalam et al. [18]. It is modelled as a network of control volumes (CVs). The following energy balance applies to each CV i at each timestep n:

$$\rho C_p V \frac{\partial T}{\partial t} = \dot{m}_{SH} C_p (T_{i-1} - T_i) - Q_{SH,i}, \tag{9.5}$$

where ρ is the density of water (kg/m^3), C_p is its specific heat (J/kgK), V is the volume of the CV (m^3), T is its temperature (K), $\dot{m}_{SH}$ is the mass flow of fluid in the spiral heater (kg/s) and $Q_{SH,i}$ is the heat lost to the surrounding water in the boiler from CV i (W). Discretized using Euler backward timestepping scheme, the equation becomes

$$\rho C_p V \frac{T_i^{n+1} - T_i^n}{\Delta t} = \dot{m} C_p (T_{i-1}^{n+1} - T_i^{n+1}) - Q_{SH,i}^{n+1}, \tag{9.6}$$

where Δt is the model's timestep. The system of equations is solved with a matrix of the form:

$$[M_{ij}] \cdot \{T_i^{n+1}\} = \{T_i^n\} + \{d\} \tag{9.7}$$

Where $\{T_i^{n+1}\}$ is a vector containing all temperature nodes at each CV at timestep n+1, $\{T_i^n\}$ is a vector containing all temperature nodes at timestep n, {d} is a vector containing the term that is a function of $Q_{SH,i}^{n+1}$, and $[M_{ij}]$ is a matrix containing coefficients that must be multiplied by $\{T_i^{n+1}\}$. Thus, the final equation resembles

$$\begin{bmatrix} b & c & 0 & 0 \\ a & b & c & 0 \\ 0 & a & b & c \\ \dots & \dots & \dots & \dots \\ 0 & a & b & c \\ 0 & 0 & a & b \end{bmatrix} \cdot \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ \dots \\ T_{i_{max}-2}^{n+1} \\ T_{i_{max}-1}^{n+1} \end{bmatrix} - \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \dots \\ d_{i_{max}-2} \\ d_{i_{max}-1} \end{bmatrix} = \begin{bmatrix} T_0^{n} \\ T_1^{n} \\ T_2^{n} \\ \dots \\ T_{i_{max}-2}^{n} \\ T_{i_{max}-1}^{n} \end{bmatrix}, \tag{9.8}$$

with coefficients

$$a = 1 + \frac{\Delta t \dot{m}}{\rho V} \quad b = -\frac{\Delta t \dot{m}}{\rho V} \quad d_i = \frac{\Delta t}{\rho C_p V} Q_{SH,i}^{n+1} \quad d_0 = \frac{\Delta t}{\rho C_p V} \quad Q_{SH,0}^{n+1} - \frac{\Delta t \dot{m}}{\rho V} T_{in},$$

where T_{in} is the outlet temperature of the water-side of the condenser.
$Q_{SH,i}$ is found using the following relation:

$$Q_{SH,i}^{n+1} = U A_i (T_i^{n+1} - T_{boiler,i}^{n}), \tag{9.9}$$

where U is the overall heat transfer coefficient (W/m²K), A_i is the area of the CV (m²) (inner and outer tube area is approximated to be equal) and $T_{boiler,i}$ is the temperature of the water surrounding the spiral heater at timestep *n*. The reason for this assumption will be presented shortly.

Overall heat transfer coefficient U is given by

$$\frac{1}{U} = \frac{1}{h_i} + \frac{t}{k} + \frac{1}{h_o}, \tag{9.10}$$

where t is the wall thickness of the spiral heater (m), k is its thermal conductivity (W/mK) and h_i and h_o are the inner and outer tube convective heat transfer coefficients respectively (W/m²K). These are calculated by using the correlations presented by Abdelsalam et al. [18] in Appendix A. Notably, the coefficients depend on the temperatures of the CV and the water surrounding it. Thus, a convergence must be reached to obtain a value of $Q_{SH,i}$, the logic of which is the following:

- Initial value for $Q_{SH,i}$ is estimated.
- T_i^{n+1} is calculated using Eq. 9.8.
- $Q_{SH,i}$ is calculated using Eqs. 9.9 and 9.10.
- The estimated and newly calculated $Q_{SH,i}$ are compared. If the difference is not below a threshold value, the calculated value is set to its estimate and the process is repeated.

Thus, the outlet temperature of and the amount of heat released by the spiral heater can be calculated. It must be noted that this model contains a very significant assumption. Temperatures calculated in the boiler are an input to the spiral heater and therefore remain constant while calculating the value for Q_{SH}. A correct approach would be using a convergence loop between the spiral heater and boiler temperatures. However, this process would be computationally expensive. Thus, an experimental validation of the spiral heater was performed by DeWarmte to test whether this assumption can be used.

9.2.2. Experimental validation

Using experimental data from Abdalsalam et al. [18], DeWarmte tested the spiral heater with the boiler model described in Section 8.1. The following conditions were set in the experiment:

- Initial boiler temperature of 20 °C
- Constant spiral heater inflow temperature of 55 °C
- Constant mass flow of 0.05 kg/s
- Ambient air conditions of 25 °C
- Experiment test time of 5.5 hrs

The boiler in the experiment was a rectangular tank. Thus, the boiler model, specifically terms being multiplied by its cross sectional area, were slightly modified. The following boiler and spiral dimensions were set:

- Boiler dimensions of 0.527 x 0.533 x 0.72 m
- Spiral inner diameter of 19.5 mm
- Spiral wall thickness of 2.5 mm
- Spiral length of 7.5 m

Figure 9.3 displays the results of the comparison. It is seen that the mean boiler temperature is very well predicted throughout the experiment. The exit temperature of the spiral heater is not predicted well at the beginning of the experiment, with a maximum temperature difference of approximately 4 °C. However, as the experiment progresses, the numerical and experimental results converge towards the same temperature. For the use of this performance model, this level of accuracy is considered sufficient, as it will be used in the temperature ranges that show close correlation between experimental and numerical results. However, in further work, the reason for this error must be further analyzed. If the issue lays in the calculation of Q_{SH}, the boiler and spiral heater model must be changed and the interaction between updated boiler temperatures and Q_{SH} must be encompassed.

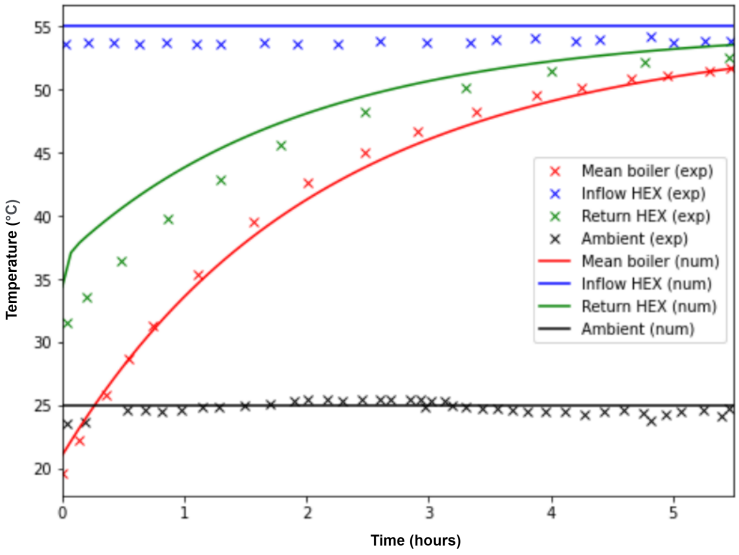


Figure 9.3: Experimental validation of spiral heat exchanger model

9.3. Household radiator model

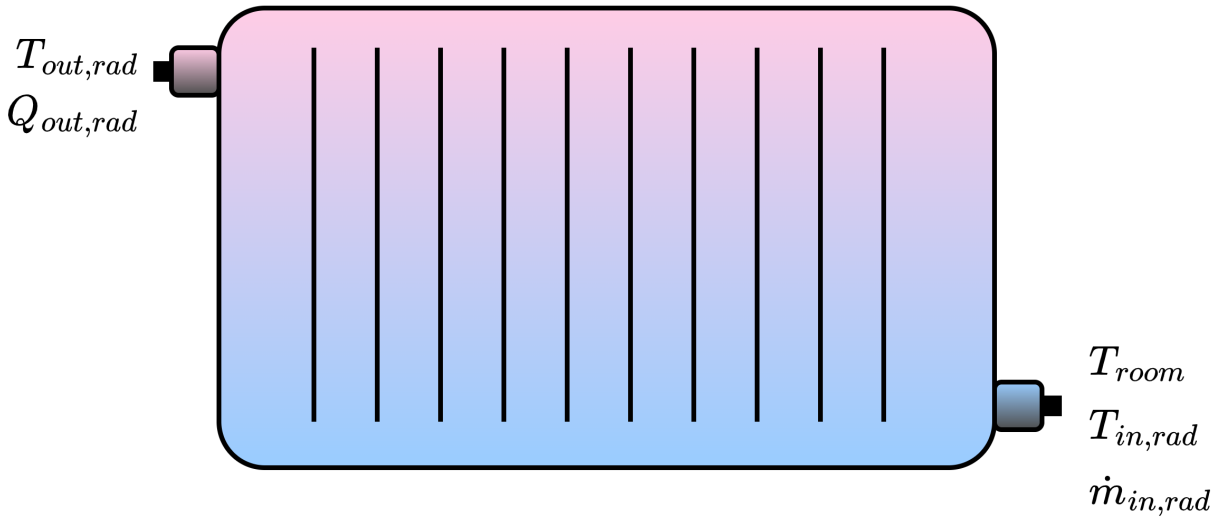


Figure 9.4: Radiator class

Table 9.3: Inputs and outputs of radiator class

Variable	Unit	Type	Description	Source
$\dot{m}_{in,rad}$	kg/s	Input	Mass flow of water entering radiator	User input
$T_{in,rad}$	K	Input	Temperature of water entering radiator	Heat pump class
T_{room}	K	Input	Ambient air temperature of room heated by radiator	User input
$T_{out,rad}$	K	Output	Temperature of water exiting radiator	-
$Q_{out,rad}$	W	Output	Heat emitted by radiator	-

EU Norm EN442 2014 1-2 [3] [4] outlines the specifications that must be fulfilled by radiators sold within the EU. It defines the requirement-for and the method-of calculating thermal output of radiators [5]. This method is used in the radiator class of the numerical model.

9.3.1. Methodology

The standard defines that the heat output of a radiator may be approximated with the relation

$$Q_{out,rad} = K \Delta T_{lm}^{n}, \tag{9.11}$$

where $Q_{out,rad}$ is the radiator's heat output (W), K and n are constants, and ΔT_{lm} is the logarithmic mean temperature difference, defined as

$$\Delta T_{lm} = \frac{T_{in,rad} - T_{out,rad}}{ln \frac{T_{in,rad} - T_{room}}{T_{out,rad} - T_{room}}}, \tag{9.12}$$

where $T_{in,rad}$ is the incoming temperature into the radiator, $T_{out,rad}$ is the output temperature of the radiator and T_{room} is the room temperature in which the radiator is placed, all measured in (K).

Within the datasheet of a radiator sold in the EU, $Q_{out,rad}$ is defined for 2 combinations of $T_{in,rad}$, $T_{out,rad}$ and T_{room}. Thus, constants K and n are known for all radiators sold in the EU. With known constants K and n, the heat output for the radiator

is known for any input, output and room temperature combination.

Within the model, radiator type is a user input. Thus, constants K and n are defined. It is assumed that when the SH circuit is turned on, water from the circuit at a temperature of 20 °C enters the HP condenser. Thus, initially at t_{SH} = 0, $T_{out,rad}$ is set as 20 °C. As described in Section 5.1.4.1, the HP class outputs a condenser outlet temperature. This is the temperature entering the SH circuit, and is therefore set as $T_{in,rad}$ for time t_{SH} = 0. Throughout the model, it is assumed that the room temperature remains constant at a user-defined set-point.

Thus, $Q_{out,rad}$ may be calculated. $Q_{out,rad}$ is subtracted from the SH demand for timestep t_{SH} = 0, as described in Section 10.3.

A new condenser inlet temperature, $T_{out,rad}$ is calculated using the following relation

$$Q_{out,rad} = \dot{m}_{in,rad} C_p (T_{in,rad} - T_{out,rad}) \tag{9.13}$$

and timestep's t_{SH} = 0 values of $Q_{out,rad}$ and $T_{in,rad}$.

When SH demand is not being fulfilled quickly enough (the definition of which is presented in Section 10.3), an electric heater of an assumed 2500 W is turned on. Then, $T_{out,rad}$ is found by substituting $Q_{out,rad} = Q_{out,rad} + 2500$ in Equation 9.13.

Additional functionality is implemented in the model. In real-life SH systems, when the DHW demand is being fulfilled, the SH circuit may continue circulating without heat being added to it. This allows for the efficient usage of all energy stored in the thermal inertia of the system. Thus, in the model, when the HP is providing heat for DHW, but there still exists a SH demand, the circuit continues circulating. However, without the known temperatures entering and exiting the HP's condenser, a convergence loop is required to calculate temperature $T_{in,rad}$ and heat output $Q_{out,rad}$.

$T_{out,rad}$ is assumed to be equal to the previous timestep's $T_{in,rad}$, while $T_{in,rad}$ is assumed to be equal to $T_{out,rad}$ - 1. With this assumption, $Q_{out,rad}$ is calculated using Equation 9.11. Using newly calculated $Q_{out,rad}$, a new value of $T_{in,rad}$ is calculated and compared with its previously calculated counterpart. If the difference between the two values is larger than a defined error, the old $T_{in,rad}$ is set to the new $T_{in,rad}$. The process is repeated until the results converge.

9.3.2. Space heating circuit

The number of radiators in the home is a user input. It is assumed that the radiators are all connected in parallel. This is a common configuration in homes due the highest temperature difference being reached between water in the radiator and its surroundings. Furthermore, parallel configurations allow for easier maintenance when one radiator malfunctions. Thus, $T_{in,rad}$ and $T_{out,rad}$ are the same for each radiator and the total SH demand provided at a timestep is the product of the number of radiators and calculated $Q_{out,rad}$.

9.4. Wastewater filter model

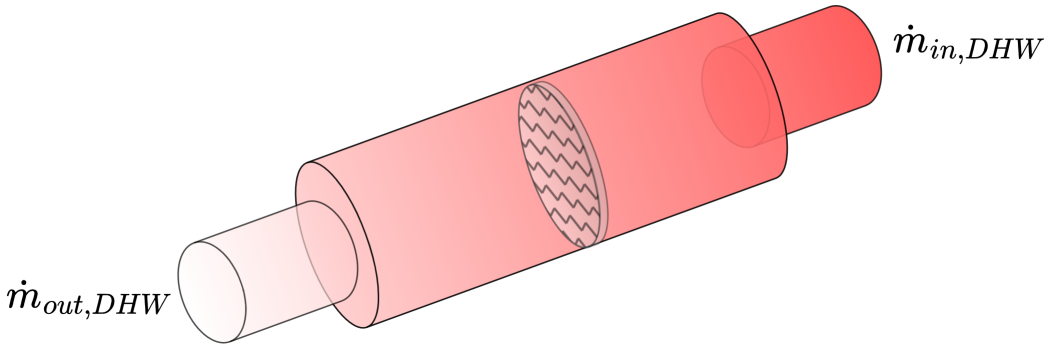


Figure 9.5: Filter class

Table 9.4: Inputs and outputs of filter class

Variable	Unit	Type	Description	Source
$\dot{m}_{in,DHW}$	kg/s	Input	Mass flow of domestic wastewater entering filter	DHW class
$\dot{m}_{out,DHW}$	kg/s	Output	Mass flow of domestic wastewater exiting filter	-

9.4.1. Loss of mass

Wastewater entering the HeatCycle system from the sewage initially is passed through a filter developed by DeWarmte. In this filter, mass is lost. Thus, the mass flow as defined by the DHW class, described in Chapter 10.1, is reduced.

DeWarmte investigated the ratio of outflowing to inflowing mass in their first pilot HeatCycle installation in the Green Village, Delft. During the 5 month experiment, monthly filter ratios were computed. The ratios were calculated using

$$FR = \frac{V_{dumped,month}}{V_{DHW,month}}, \tag{9.14}$$

where V_{dumped} is the amount of wastewater that was dumped in a month (m^3), found by monitoring flow sensor data at the outlet of the HP when it was set to dump wastewater to the sewage. $V_{DHW,month}$ is the monthly water usage of the inhabitants of the home (m^3). This value was given by the inhabitants.

The average filter ratio was found to be approximately 0.8. Thus, in the filter class, mass flow of incoming wastewater from the DHW class is multiplied by 0.8. The newly calculated mass flow is passed to the wastewater bag.

Part III: Model Application and Performance Evaluation

10

Domestic hot water and space heating demand: An input

In this chapter, the DHW and SH demands will be described. They are inputs to the model. Furthermore, from the DHW demand, a wastewater production profile will be created.

The performance and feasibility of the system is highly influenced by the DHW and SH demands. For this project, the implemented demands correspond to a typical Dutch home and hot water usage profile. Thus, the hybrid HP system will be tested as an alternative to gas heating in a typical Dutch home. However, more demand types must be added to predict the performance of the system in various homes with different user usage patters. Recommendations on this will be discussed in Section 12.

10.1. Domestic hot water demand

The European Union's Commission Delegated Regulation No. 812/2013 outlines the requirements for energy labelling of water heaters with a rated heat output $\leq$ 70 kW, under which domestic HPs fall. Within the norm, energy demand profiles are established for various consumption amounts, ranging from 3XS - XXL. These profiles are given for times throughout the day, deemed to be a representation of real-life user DHW consumption.

The DHW demand in this numerical model uses a medium (M) sized profile. The profile is displayed in Table 10.1.

The norm defines $\mathbf{Q_{tap}}$ as the energy requirement that arises at a corresponding time. The minimum flow rate and temperature at which the demand must be provided are **f** and $\mathbf{T_P}$ respectively. Absence of data for $\mathbf{T_P}$ means there is no minimum temperature requirement. $\mathbf{T_m}$ is the minimum temperature requirement from which the heat demand begins to be fulfilled. The boiler temperature is not allowed to drop below 45 °C by the HP, therefore this column is insignificant, as the boiler will always be providing heat at a temperature of at least 45 °C (the control of the boiler is described in Section 8.4).

The DHW demand is assumed to remain the same for every day in a year. To illustrate the model's performance, an example will be used; At the timestep corresponding to 07:00, a DHW demand of 0.105 kWh is logged. Water is extracted from the boiler at the temperature of its utmost top node and new water enters at a user-defined tap water temperature. In the model it is defined as 10 °C. The flow rate at which the exchange of water occurs in the boiler is **f**. Thus, the heat provided at the timestep corresponding to 07:00 is

$$Q_{provided,n=0} = \rho_w f C_p (T_{boiler,topnode} - T_{tap}).$$

If $Q_{provided,n=0} < Q_{tap}$, the difference between the two will be carried onto timestep n=1 and so forth until all demand is fulfilled. At the timestep corresponding to 07:05, a new demand will be added and the process will be repeated.

Table 10.1: Medium-sized DHW daily profile as defined by EU Norm. No. 812/2013

Time	Q_{tap} (kWh)	f (l/min)	T_m (°C)	T_p (°C)	Time	Q_{tap} (kWh)	f (l/min)	T_m (°C)	T_p (°C)
07:00	0.105	3	25		**11:45**	0.105	3	25	
07:05	1.4	6	40		**12:45**	0.315	4	10	55
07:30	0.105	3	25		**14:30**	0.105	3	25	
08:01	0.105	3	25		**15:30**	0.105	3	25	
08:15	0.105	3	25		**16:30**	0.105	3	25	
08:30	0.105	3	25		**18:00**	0.105	3	25	
08:45	0.105	3	25		**18:15**	0.105	3	40	
09:00	0.105	3	25		**18:30**	0.105	3	40	
09:30	0.105	3	25		**19:00**	0.105	3	25	
10:30	0.105	3	10	40	**20:30**	0.735	4	10	55
11:30	0.105	3	25		**21:15**	0.105	3	25	
					21:30	1.4	6	40	
Q_{total}	**5.845**								

10.2. Wastewater production

Wastewater as a source for the HP must be matched to the DHW demand. However, EU Norm. No, 812/2013 outlines the DHW demand as an energy requirement. These values must be transformed into a temperature and mass of incoming wastewater.

Kantar, a Dutch data collection and analysis company, conducts studies every 3 years in collaboration to Vewin, the association of water companies in the Netherlands to investigate daily domestic water usage patterns. Table 10.2 displays results from their study in 2016 [64].

V_{use} is the used volume of water per day per person in a household, T_{use} is the temperature the water is used at. Therefore, this data provides the source mass and temperature of wastewater for the HP per person. This data must be matched with the energy demand in Table 10.1 to obtain a usage pattern for a house.

Table 10.2: Daily domestic hot water usage per person

Source	V_{use} (l/day)	% total	T_{use}
Bath	1.9	2	38
Shower	49.2	42	38
Bathroom sink	5.2	4	30
Toilet flush	34.6	29	18
Washing clothing (by hand)	1.3	1	30
Washing machine	14.1	12	40
Washing dishes (by hand)	3.5	3	30
Dishwasher	2.5	2	70
Food preparation	1.2	1	30
Hot drink preparation	0.8	1	-
Drinking water	0.5	0	-
Other kitchen sink	4.5	4	13
Total	**118**	100	

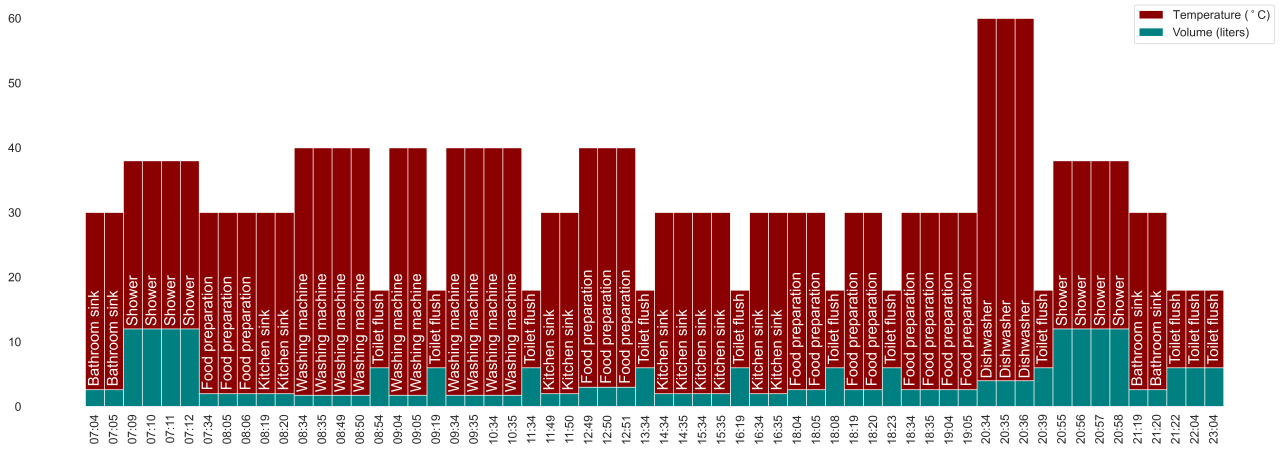


Figure 10.1: Wastewater production profile based on Table 10.1 and 10.2

Figure 10.1 displays a made-up wastewater production profile for a home. It is assumed that a family of three (two parents with a young child) live in the home, with one parent staying at home during the day. The profile is made by assuming household activities that would require hot water based on the demand and time taken to fulfill the demand described in Table 10.1. The temperatures of the tasks are set to be equal to those described in Table 10.2. Tap water is mixed with incoming hot water to provide the required temperature. The total volume of wastewater produced in a day is 257 liters. Assuming a young child produces a third of the amount of wastewater that an adult produces, the total volume produced in this profile is similar to the total amount estimated by Kantar (2(118) + 0.3(118) = 271 liters).

It is assumed that 4 minutes pass from the time of the demand requirement to wastewater entering into the wastewater bag. In this way, the demand will not be fulfilled by the consumed hot water.

10.3. Space heating demand

The numerical model uses a yearly SH demand from a previous study by DeWarmte. The internal document relating to this study [67] may be obtained upon request.

During the preliminary study of the feasibility of combining an additional source to the HeatCycle (described in Section 2.1, DeWarmte developed a method of calculating the SH demand in a typical home. This method is briefly reviewed in this section.

SH demand may be seen as the loss of heat from a home below a desired temperature. Thus, SH is used to make up for the loss. However, a home gains heat from sources such as solar radiation, heat dissipation from electrical appliances etc. Thus, SH demand is the balance between desired thermal energy within a home, and its loss and gain. Heat losses and gains are heavily dependent on home construction, location, orientation and inhabitant patterns. Thus, DeWarmte assumes typical home and usage patterns to model this thermal exchange.

During DeWarmte's research, it was found that the most common house in the Netherlands is one built between 1975 and 1991. Thus, typical properties of corresponding construction materials and values for building area and orientation are used. Furthermore, it was found that a desired temperature of 18 °C is most common during heating season. This is set as the temperature from which additional loss in energy within the home must be supplemented by SH. It is assumed that the home is a single volume with no internal walls. Therefore, a single temperature node may be used to described the complete home temperature.

The modelled sources of heat gain and loss are as follows:

- Ventilation and infiltration due to air change
- Forced convection outside the building due to wind

- Natural convection in the crawlspace of the home
- Conduction through building walls
- Radiation through windows and to walls/roof, taking into account the orientation of the surface with respect to the sun
- Average heat gain from lighting, electrical appliances and home inhabitants

Thus, temperatures are calculated at the following locations:

- Outside of roof
- Inside of roof
- Outside wall of building
- Inside wall of building
- Inside ground floor
- Outside ground floor
- Air temperature
- Crawlspace air temperature

Boundary conditions are taken from weather data transformed to an average mean year, as described in Appendix B.11. The relevant parameters are air temperature, wind speed, direct solar irradiance, diffuse solar irradiance and ground temperature.

The energy balance equations calculating temperatures at each node and deeper justification of choices and assumptions may be found in DeWarmte's internal report [67].

10.4. Use of space heating demand in numerical model

The demand as calculated by DeWarmte is given at hourly timesteps. Thus, in this numerical model, it is assumed that a value for SH demand stays constant for an hour, equivalent to 60 timesteps. If not fulfilled, in the next hour (after 60 timesteps), an additional SH demand will be added onto the existing one for another 60 timesteps.

The SH demand class is closely linked to the radiator class (described in Section 9.3). An output of the radiator class is $Q_{out,rad}$, the amount of heat that has been released by the radiator to its surroundings. This is given at each timestep the SH circuit is on. Thus, $Q_{out,rad}$ is subtracted from the existing SH demand at each timestep. In such a way, the model is able to capture under and over production of heat. If there is an over production of heat, it will be subtracted from the following hour's SH demand.

An electric heater is turned on to supplement the HP when SH demand is not being fulfilled. The bounds of what is considered to be an unfulfilled demand is a control issue that should be optimized. However, in this model, it is assumed that the electric heater will turn on when there is a buildup of 2 kWh of demand, and will turn off when it has been reduced to 1 kWh. In the home that is modelled (2 story home of 150 m^2 and a total height of approximately 4 m), the equivalent temperature deviations from the set point for a demand buildup of 2 and 1 kWhs are 3.5 and 1.7 °C respectively. It should be noted that these bounds may be too broad, as a drop of temperature in 3.5 °C in a home is noticeable and may cause significant discomfort. However, when using the model to estimate a households savings, these bounds may be a user input depending on the customer's wishes relating to comfort.

11

Results

11.1. Predicted HeatCycle performance

The performance of the HeatCycle as a standalone system was initially run to compare the results to real-life HeatCycle performance as an additional model validation. For comparison purposes, the system was run in January (coldest month of the year), August (warmest month of the year) and for a full year.

Figure 11.1 displays the HeatCycle's performance over the month of January. The heat output of the HP is plotted alongside its work input and COP. The performance of each day remains relatively constant. This should be the case, as the DHW demand of each day is constant in the performance model. However, some fluctuations in the performance exist, due to variations in soil temperature over the month or SH demand being fulfilled at times when it is present and when the boiler has been sufficiently heated. Furthermore, initially, the boiler is set to a temperature 20 °C. Thus, for the first 2 days it is heating up. A higher COP is observed in these days, as the supply temperature of the DHW is lower due to the lower boiler temperature. After, the COP of the system generally lies between 2 and 4. When SH is being provided, the COP is between 4.5-8. This fluctuation may be explained by variations in wastewater and DHW and SH circuit temperatures while the heat pump is turned on. Peak COP values of 11-14 are observed when the wastewater is the warmest and the SH circuit has cooled to room temperature.

Figure 11.2 displays the HeatCycle's performance over the month of August. The power of the HP throughout the month does not differ significantly to that of January. This is expected, as only a small portion of SH demand is being delivered by the HeatCycle in January. However, in January, two ranges of COPs are seen due to SH and DHW being provided at different supply temperatures. In August, only the lower range COP is seen. This is expected, as only DHW demand is being fulfilled in August.

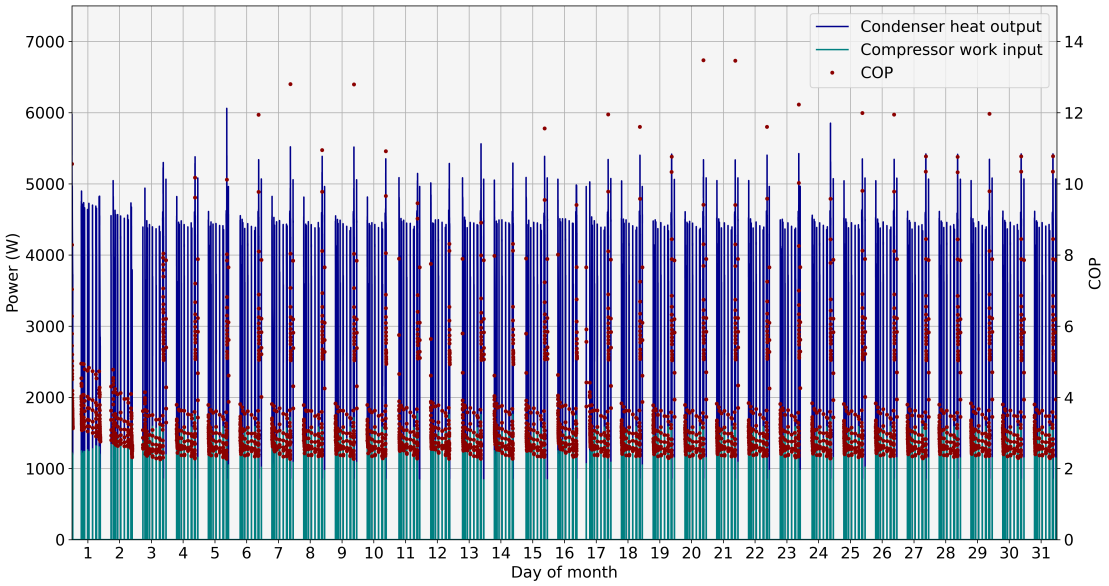


Figure 11.1: HeatCycle's heat output, work input and COP during January

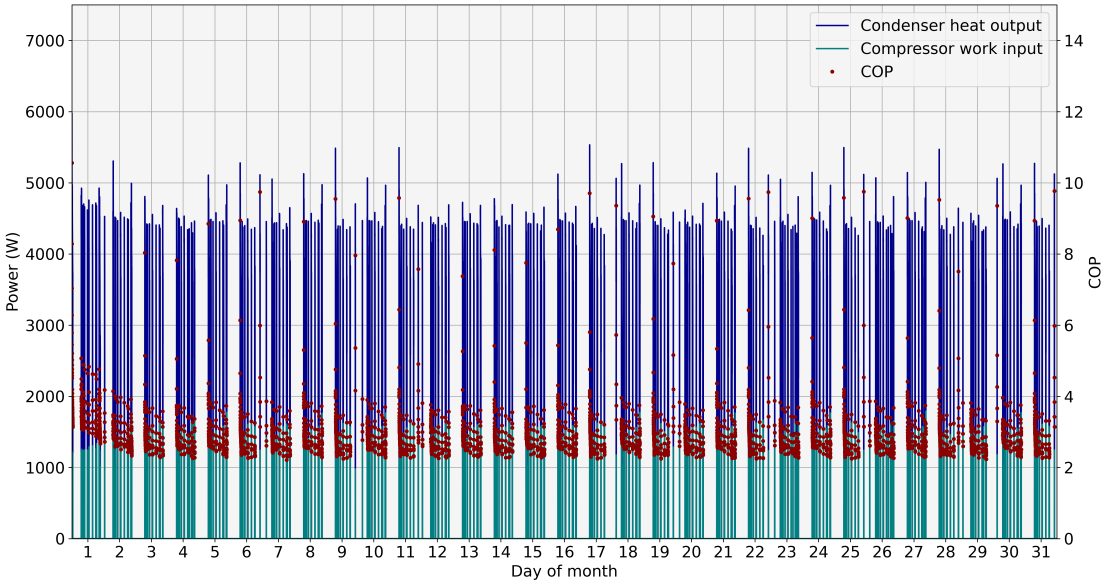


Figure 11.2: HeatCycle's heat output, work input and COP during August

Table 11.1 displays the key performance indicators (KPIs) of the system in a year and for the months of January and August. It is assumed that the supplementary heater is a gas boiler. From the data, it is seen that the HeatCycle is able to provide the full DHW demand and 7% of the SH. Additional heat is required to reach the high DHW supply temperature. However, this is set in the control. Instead, a user could opt for a lower DHW supply temperature or a higher boiler setpoint temperature. There are variations in the amount of DHW demand fulfilled in January and August. However, with the additional heat supplied by the gas boiler, the total energy required for fulfilling DHW demand is 231 kWh and 238 kWh for January and August respectively.

The difference between the two is slight, and occurs due to dynamic effects of model, such as switches between SH and DHW circuits and different wastewater temperatures due to variations in soil temperature. The HP operates at a yearly COP of 3.1. In January, the COP is higher than in August, as SH is being provided. This results in times of a lower supply temperature and therefore higher average COP.

Table 11.1: Performance of HeatCycle

	January	August	Yearly
Delivered heat (kWh)	260	213	2948
Delivered heat ratio of DHW:SH	202:58	213:0	2425:523
Additional heat required from gas boiler for DHW (kWh)	29	25	292
Additional heat required from gas boiler for SH (kWh)	1552	0	7704
Compressor work input required (kWh)	81	73	944
Av. HP COP	3.2	2.9	3.1

11.1.1. Comparison of HeatCycle performance with current data

CE Delft, an independent research and consultancy firm specialising in sustainability consulting has conducted research on the performance of gas boiler in homes. Taking into account the losses due to short DHW tappings, their research concluded that the efficiency of gas boiler is 0.72. For SH, taking into account the heat gain due to condensation, this efficiency lies at 1.04. Assuming the average between the lower and higher heating value of natural gas, it releases 38.6 MJ/m^3 [38] when burned. These values will be used for the cost calculation of a fully gas system.

DeWarmte has developed a simple excel model to calculate the gas savings of a household with the HeatCycle. These values have been checked and tailored using customer data. DeWarmte estimates that using the HeatCycle amounts to gas savings of 322 m^3 and 483 m^3 for a 2 and 3 person household respectively. Using corresponding efficiencies for DHW and SH supply, the model developed in this study predicts a saving of 361 m^3 for the household modelled: i.e. a 3 person household, one of which is a young child. This aligns well to the estimate of DeWarmte. The average performance of 50 of DeWarmte's HeatCycle systems over the month of August was calculated (not enough data is available for January). The average heat delivered is 200 kWh. This aligns closely to the delivered heat in August. The average COP is 4.0. This value is significantly higher than the predicted COP by the model. The difference between the source and supply temperatures was checked. Customer data showed an average temperature difference of 35.7 °C, while the model predicted one of 42.4 °C. This explains the disparity of COP values, but alignment of total heat delivered; the amount of heat available and delivered is predicted well by the model, but the supply temperature is much higher, thus resulting in a lower COP. The reason for variations in temperature difference between source and supply is due to the size of the boiler and set points in boiler control. These are user-specific values.

11.2. Predicted HeatCycle and air source combination performance

A simulation was run for the combination system of a HeatCycle and air source. This was done for the month of January (coldest month of the year), August (warmest month of the year) and for a full year.

Figure 11.3 displays the performance of the HP over the month of January. It is seen that for the first 2 weeks of January, the most heat is required. From day 11-14, 17-12 and 22-14, significantly less heat is released. During these days, frosting conditions are not met, therefore the HP does not need to shut off during defrosting, allowing for a larger supply of heat. Figure 11.4 shows that the defrost heating element is not turned on during these days. The COP of the system stays between 2-3 and 4-6. The former is the COP for providing DHW at a higher supply temperature, while the latter is for providing SH at a lower supply temperature. A slight increase in COP can be seen during the days when defrosting is not needed, due to the higher ambient air temperature.

The energy provided by the HP and required auxiliary energy supply for SH is displayed in Figure 11.4. Note that the defrosting line is pink in the legend, but appears purple on the plot, as it is on top of, and therefore required at the same time as the supplementary SH line. The most defrosting and auxiliary SH power is required at the beginning and end of January,

when weather conditions are the coldest. Auxiliary heating for SH demand is required often and required only slightly less when frosting conditions are not met. Thus it can be concluded that during winter months, the system is not able to fulfill the full SH demand. A larger capacity HP could potentially solve this problem.

Figure 11.5 displays the HP's performance in the month of August. A notable difference between the performance in January and August is the frequency at which the system is operating. During August, only DHW demand is present. It is easily and quickly fulfilled by the system. Furthermore, the average COP of the HP drops in August, as heat is being solely supplied to a high temperature for DHW.

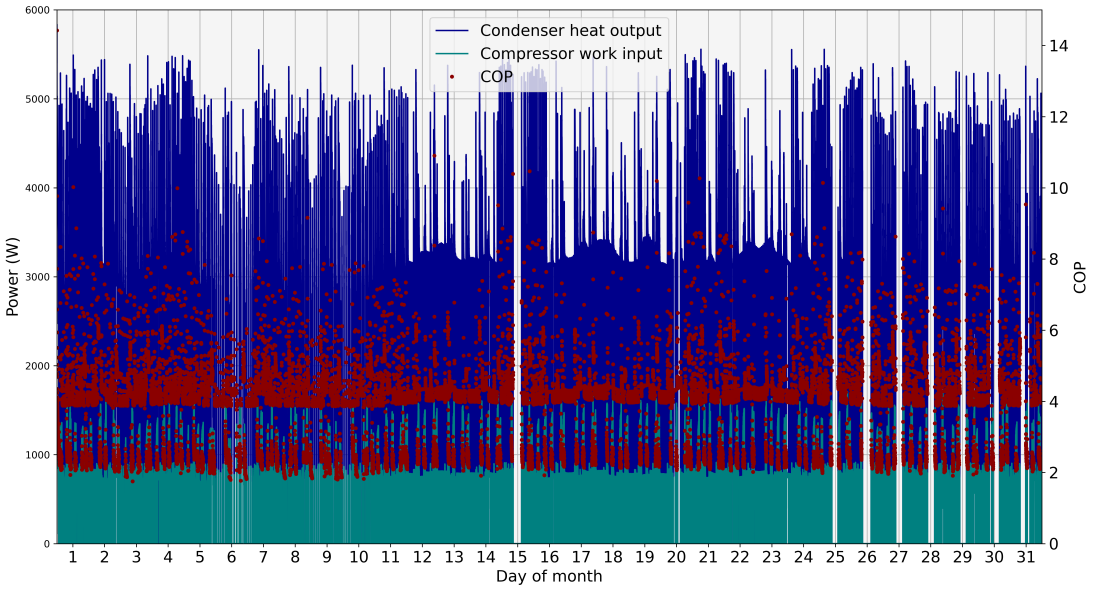


Figure 11.3: Combination system's heat output, work input and COP during January

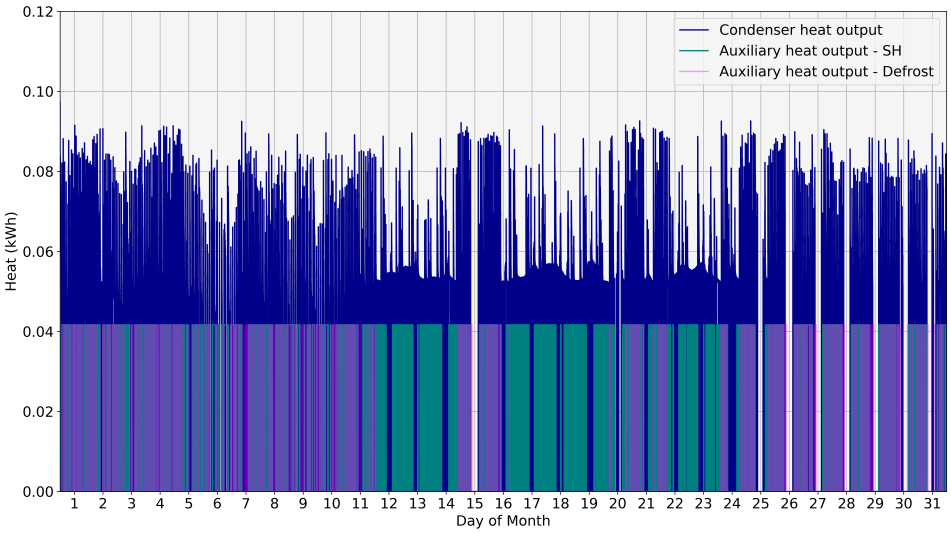


Figure 11.4: Combination system's heat output and required auxiliary power input for SH during January

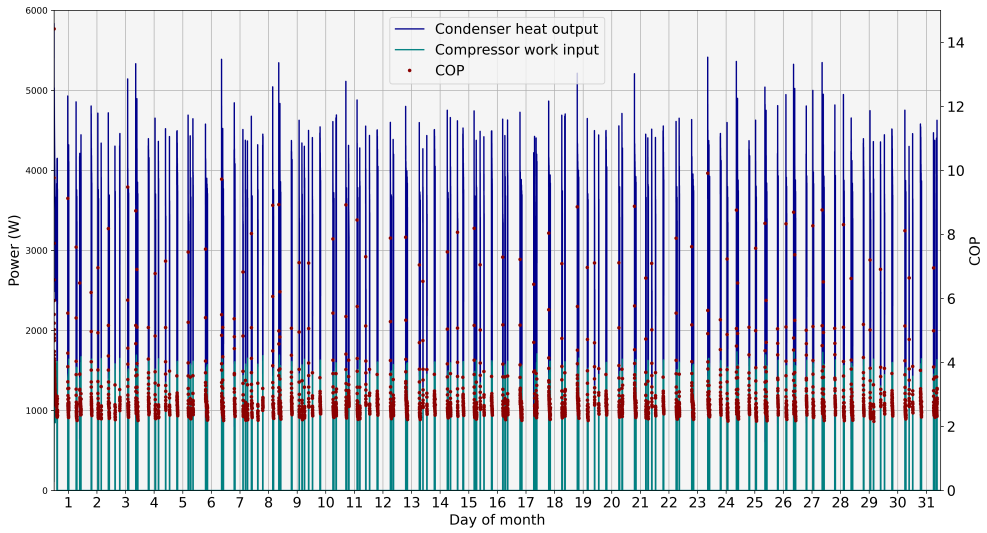


Figure 11.5: Combination system's heat output, work input and COP during August

Table 11.2 displays the yearly performance of the system, alongside its performance in January and August. In January, the system delivers 1559 kWh of heat, of which 82% is delivered to SH. The rest is used for DHW. In August, 238 kWh are supplied to DHW. The discrepancy between January and August DHW energy requirement could be explained by the HP control. In January, when the air source is being used, its source temperature is very low. This results in the compressor operating at a high frequency, thus resulting in higher condenser capacity and temperatures in the system. This effect may be seen when comparing Figures 11.3 and 11.5, where the capacity of the condenser in August is usually less than around 4.5 kW, while in January it is usually around 5 kW. With higher temperature and quicker heat supply, more volume of the boiler will be heated. However, with lower heat supply, as occurs in August due to lower compressor speeds with a significantly higher source temperature, the bottom volume of the boiler may not be heated. This does not affect the supply of DHW, as not all of the boiler is drained when DHW is needed. Therefore, with thermal stratification of the boiler, a lower amount of energy in the boiler can still supply

the same demand.

To check the validity of this theory, a quick calculation was performed. The difference between energy required to supply DHW in January and August is 47 kWh, equivalent to a difference of 1.5 kWh per day. In the 300 liter boiler modelled, a difference of 1.5 kWh in energy is equivalent to a difference of 4.5 °C in temperature. The HP is on DHW mode approximately 5 times a day. Thus, each time the HP turns on the temperature difference throughout the boiler must be less than 1 °C to satisfy the 1.5 kWh per day discrepancy between January and August. This value is reasonable and could occur due to varying condenser capacities.

The COPs of the system are calculated. Note that av. HP COP refers to SPF_{H1} (as defined by Equation 3.9) and av. system COP refers to SPF_{H3} (as defined by Equation 3.11), neglecting energy usage of circulation pumps in the HP. The HP COP is high for January and lower for August, as SH is being supplied in January at a low supply temperature. This also explains why the yearly HP COP lies between that of January and August. The opposite trend is seen for the average system COPs, as additional electricity is used in winter months to fulfill SH demand.

Table 11.2: Performance of HeatCycle and air source combination

	January	**August**	**Yearly**
Delivered heat (kWh)	1559	238	7640
Delivered heat ratio of DHW:SH	274:1285	238:0	2995:4685
Additional heat required from electric heater for DHW (kWh)	18	8	197
Additional heat required from electric heater for SH (kWh)	550	0	3019
Additional heat required from electric heater for defrosting (kWh)	119	0	299
Compressor work input required (kWh)	342	87	2149
Av. HP COP	3.7	2.7	3.5
Av. system COP	1.9	2.6	2

11.3. Economic analysis

To test the feasibility of the modeled HeatCycle + air source system, an economic analysis must be performed. Here, a comparison will be drawn between an all-electric combination system versus an all gas system. The savings of the system and its payback time will be calculated.

The Dutch government has recently announced the cap on energy prices in the Netherlands. These are limited to 1.45 €/m^3 up to a yearly usage of 1200 m^3 of gas and 0.4 €/kWh up to a yearly usage of 2900 kWh of electricity. Energy prices beyond these points depend on the energy market, geopolitics and a user's energy contract. Thus, they are difficult to estimate. Currently, typical values are approximately 2.1 €/m^3 of gas and 0.5 €/kWh of electricity. These values will be used for the calculation, however they must be user-specific variables.

Using the values in Table 11.2, cost calculations are performed. In an all gas system, it is assumed that the same amount of heat must be delivered as was delivered by the combination system. Thus, the amount of required energy for DHW is set to the amount of energy that was provided by the HP in DHW mode and the additional energy required by the DHW electric heater. This will slightly overestimate the energy requirement, as gas fired boilers directly heat the boiler, while a HP heats it indirectly through the spiral heater. The energy requirement for SH is set to the energy provided by the HP in SH mode and additionally required by the SH electric heater. As both the HP and gas-fired boiler heat the SH circuit directly, this is an accurate representation of the energy requirement. Defrosting energy is neglected, as it is obviously not required by a gas-fired boiler. Table 11.3 displays the yearly cost comparison between an all-electric HeatCycle + air source combination and a fully gas system.

Table 11.3: Comparison of all-electric and all-gas systems

HeatCycle + AirSource			All gas			
Consumer	**Electricity use (kWh)**	**Cost (€)**	**Consumer**	**Energy use (kWh)**	**Gas use (m^3)**	**Cost (€)**
DHW heater	197	99	**DHW**	3192	413	470
SH heater	3019	1219	**SH**	7704	691	1157
Defrost heater	299	100				
Compressor	2149	1075				
Total	5664	2450		10896	1104	1601

The all-electric system is more expensive than an all-gas system. Thus, the modelled hybrid HP cannot currently be seen as an alternative to an all gas system for a typical Dutch home. However, a few factors must be considered. Firstly, the effect of varying energy prices will be checked. The subsidised pricing for gas and electricity are a temporary measure. Therefore, assuming current typical values of gas and electricity, a price comparison for the two systems must be conducted. Secondly, the system could be implemented as a hybrid electric-gas system. Gas could be used to supplement the additional SH, DHW and defrosting requirements. Table 11.4 summarizes these comparisons. Payback time is calculated using a total system cost of 5900 € (after governmental subsidy on the HP).

Table 11.4: Summary of economic analysis

System type	Operation cost (€/yr)	Savings (€/yr)	Payback time (yrs)
All-electric, subsidised energy prices	2450	None	None
All-electric, non-subsidised energy prices	2832	None	None
Hybrid electric-gas, subsidised energy prices	1375	226	26.1
Hybrid electric-gas, non-subsidised energy prices	1822	496	11.9
All-gas, subsidised energy prices	1601	-	-
All-gas, non-subsidised energy prices	2318	-	-

Thus, for the modelled demands, the system can only potentially be worthwhile when it is in combination with gas. However, the payback times are high. The system can be optimised. The most obvious method is using reverse cycle defrosting instead of an electric element. In this method, the condenser would function as an evaporator, using heated water from the boiler as its source. The evaporator would become the condenser. Assuming a COP of 6 for the process (due to the high source temperature of DHW), the electricity required to defrost would drop from 300 kWh to 50 kWh. Further optimisations could include a decrease in desired incoming DHW temperature, decrease in required room temperature for SH, especially during the night or times when no one is home.

However, the primary reason for the system's failure as an all-electric solution is the high SH demand. The HP capacity is too low to produce the required heat. As a guideline, an electric heater should supplement only 5% of a home's demand, the rest should be supplied by the HP. In the modelled system, the electric heater is supplementing almost 40% of the demand, making the system very cost-inefficient. Interestingly, the average HP COP is quite high for the whole year, indicating that the hybrid HP is working efficiently. Thus, to potentially make the system worthwhile, the demand must be reduced (with a smaller, better insulated home) or the HP capacity must be increased (by using a larger-sized HP).

A quick calculation will be performed to estimate the energy savings when using an appropriately sized HP, with the same SH and DHW demands; Assuming the HP capacity has increased to one that can supply 95% of the total SH demand, the delivered heat to SH increases from 4685 kWh to 7319 kWh, and the required supplemental heat by the electric heater is reduced from 3019 kWh to 366 kWh. If the yearly average COP of the system remains the same, this additional heat being generated by the HP will increase the compressor work input by 753 kWh. Thus, the total energy consumption of the system will be reduced from 5664 kWh to 3764 kWh, a decrease of 33.5%. Note these values are only an estimate to show the potential system enhancement. However, many factors must be taken into account when calculating savings. These include the additional cost of the higher capacity HP, variations in HP efficiency due to differently sized components, variations in average HP COP due to more SH being provided and additional defrosting power requirements due to more frequent usage of the air source.

A cost comparison between an all-electric, all gas and hybrid electric-gas system for a well-sized HP system is seen in Table 11.5. In comparison to an all-gas system, the HeatCycle + air source has the same operational costs with subsidised energy prices. However, with non-subsidised energy prices, the savings of the HeatCycle + air source system are more significant. The hybrid electric-gas system has the most significant savings. Payback time is calculated assuming the higher capacity HP will cost an additional 1000 €.

Table 11.5: Summary of economic analysis using a higher capacity heat pump

System type	Operation cost (€/yr)	Savings (€/yr)	Payback time (yrs)
All-electric, subsidised energy prices	1592	None	None
All-electric, non-subsidised energy prices	1882	436	15.8
Hybrid electric-gas, subsidised energy prices	1280	321	21.5
Hybrid electric-gas, non-subsidised energy prices	1624	692	10.0
All-gas, subsidised energy prices	1601	-	-
All-gas, non-subsidised energy prices	2318	-	-

Finally, a comparison will be drawn between an all-electric ASHP system and a well-sized, all-electric HeatCycle + air source system. Using the previously calculated energy consumption values for the HeatCycle + air source system, a yearly average COP of 3.0 is reached. CE Delft estimates that all-electric ASHP systems run at an average system COP of 2-2.6 for DHW and 3.5-4.5 for SH being provided at 35 °C [25]. Note that this is on the lower side of the temperature the HeatCycle + air source is providing SH at, therefore the comparison will be slightly skewed in favor of the ASHP. Assuming the same distribution of DHW to SH demand as in the modelled system, i.e. 29% to 71% (taking into account additional electric heater consumption), an all-electric ASHP operates at a yearly system COP of 3.0-3.9. Thus, the HeatCycle + air source system operates at a similar, albeit lower than average, efficiency as an ASHP.

12

Conclusions and further work

Experimental validation showed that the model sufficiently captures the behaviour of the individually modelled components. The condenser of the HP, air source and wastewater bag align very closely to experimental results. The evaporator of the HP, boiler and spiral heater perform worse. Nevertheless, it may be concluded that the real performance of the hybrid HP can be well-predicted by the numerical model. For further work on improvements in the model and therefore its ability to simulate reality, refer to Section 12.1.

Yearly performance of the system shows that the hybrid HP is not capable of supplying the required SH demand, resulting in high electricity usage. Therefore, a higher capacity HP is required for the modelled household. However, although the system is not able to supply the demand of the modelled household, the HP operates efficiently. DHW is provided at approximately 45-50 °C, with a HP COP between 2 and 3. SH is provided at approximately 35-45 °C, with a HP COP of 4-6. Therefore, it may be concluded that even with the additional losses due to the heat exchange between dry cooler and HP evaporator, the system design remains relatively efficient. However, when compared to an ASHP, the system operates comparably, but on the lower side of average ASHP performance. This shows that the benefit of using wastewater as a high temperature source is counteracted by the additional losses caused by the indirect heat exchange between air and the HP.

Thus, the overarching conclusion of the performance model is that the hybrid HP's current design is not able to fulfill the required heat demand of a typical Dutch home. Therefore, it is not a worthwhile all-electric alternative to a gas-heated home. However, improvements in the system design could make it worthwhile. Refer to Section 12.2 for an outline of possible system improvements.

12.1. Further work in model improvement

Further work must be done to improve the components of the numerical model. This includes refinement in calculation methods to improve their alignment to experiments, additional experiments for validation and improvements in the model's modularity for its function as a tool for further research in hybrid HP systems.

Heat pump:
The numerical model of the HP requires improvement in its prediction of heat transfer at high compressor frequencies, especially for the evaporator. Testing different heat transfer coefficient correlations for the HEXs or the addition of superheat in the thermodynamic model could improve the validation. Furthermore, a higher capacity HP must be added. To do so, a new compressor datasheet for such a HP must be used.

Air source:
For the numerical model of the air source, most improvements relate to the frost growth model. Firstly, the control of turning on defrosting mode must be changed. It is currently controlled by the frost thickness, a variable that is not possible to

measure in real HP systems. It should be controlled by stark changes in evaporation temperature in the HP once air flow stops/starts again. The numerical model of this component showed close correlation with experimental results. However, it was only tested in ambient room conditions. Tests in a low temperature and high humidity environment should be performed.

Wastewater bag:
The wastewater bag model showed very close correlation to mean experimental results. Therefore, no additional work is recommended for this component. However, the Sobol analysis showed a surprising lack of sensitivity to incoming mass flow and fluid temperature. This result should be examined and analysed.

Filter:
Currently, it is assumed that the wastewater produced by the home enters the wastewater bag with no heat losses. Research should be conducted on the amount of heat lost to pipes in homes. Thus, the filter could have both a mass and heat loss factor.

Wastewater-to-glycol heat exchanger:
This component must be experimentally validated.

Boiler:
The experimental validation of the boiler showed that the model underestimates some insulation losses and fails to encompass the mixing at the bottom of the boiler due to inflow of cold water. The calculation of insulation losses could be better predicted by adding temperature nodes in the structure of the boiler, instead of calculating the losses using an overall heat transfer coefficient. Mixing could be introduced by using an empirically-derived mixing function at various mass flow rates. Lastly, the change in tap water temperature over the year should be implemented in the model. This would result in less energy requirement, and therefore better performance of the system in hotter months.

Spiral heater:
The spiral heater model's convergence loop calculation for heat release must be coupled with the boiler's temperature calculations. This is expected to improve the model's correlation to experimental results.

Radiator:
To improve the model's modularity, a function containing a calculation method for the heat loss coefficients should be developed. With an input of the radiator input/output and ambient air temperatures defined by EU Norm EN442 2014 1-2, the model would be able to calculate the heat released by any radiator sold in the EU.

Domestic hot water demand and wastewater production:
To improve the model's modularity, all sizes of demand profiles as defined by standard No. 812/2013 should be implemented in the model and corresponding wastewater production profiles should be created. This would allow the model to predict performance of the system for all types of users.

Space heating demand:
The SH demand is currently calculated assuming typical characteristics of a Dutch home. Therefore, the calculation method for demand must be changed to encompass any modelled home. Standard ISO 52016-1:2017 [17] defines calculation methods for SH demand, with inputs of home characteristics and weather data. The standard is complex, taking into account factors such as home orientation, shared walls with other buildings, thermal zones within the home and thermal buffering by its walls. The method defined in this standard must be implemented in the model, allowing for the calculation of heat demand for all home types in any climate.

General:
A few general recommendations for model improvement must be made. Firstly, pipes between components and their losses should be modelled. The length of the pipes must be a user input. Look-up tables could be used to prevent additional computational time, i.e. based on the length and inlet temperature of a pipe, the outlet temperature is found in a look-up table. Furthermore, system electricity usage must be calculated. This included the usage of pumps, valves and the controller. Finally, the computational time of the model must be improved. A yearly simulation currently takes approximately 9 hours to run on a laptop using an Intel Core i7-9750H processor. This time can be reduced by creating more look-up tables where applicable, improvement in memory storage and reduction in the use of excel files in the code.

12.2. Recommendations for system improvements

The following recommendations may be implemented or investigated to improve the performance and therefore savings of the HeatCycle + air source system:

Replacement of dry cooler with an ASHP:
The use of a dry cooler in the system leads to additional losses between source and supply. Therefore, a potential solution for this problem would be to replace the dry cooler with an ASHP. The advantage of using such a system would be lower losses due to direct heat exchange with refrigerant. Furthermore, by cascading an air source HP and the HeatCycle HP, the stages between low temperature heat extraction and high temperature supply would be split. This can theoretically maximise the COP of the two HPs. However, the overall efficiency of the system must be investigated, as the additional compressor power may not be worthwhile. A significant benefit would be that the ASHP could be sized appropriately to the home's demand. The cost of the system would increase. However, the current Dutch governmental subsidy on HPs is effective with a purchase of two HPs, thus minimising its cost.

Better control on compressor frequency:
The current control of the HP chooses compressor frequency based on source temperature, as described in Section 5.2. Figure 3.2 displays the relationship between COP, compressor frequency and evaporation and condensation temperatures. Using this relationship, the frequency of the HP could be controlled by maximising COP based off of measured evaporation and condensation temperatures, thus maximising its efficiency.

Defrosting strategy:
As seen in Table 11.2, the electric heater for defrosting uses a significant amount of electricity. Thus, reverse cycle defrosting must be implemented to reduce electricity consumption.

Direct heating of boiler:
DeWarmte is currently experimenting with direct heating of the boiler. In this type of heating, water passes through the HP condenser and returns to the boiler. Keeping in mind safety regulations, this method could be implemented into this system too, enhancing its performance.

Bibliography

[1] Climate policy | climate change | government.nl. URL https://www.government.nl/topics/climate-change/climate-policy.

[2] Market data - ehpa. URL https://www.ehpa.org/market-data/.

[3] Nen connect-en 442-1, . URL https://connect.nen.nl/Family/Detail/32602?compId=10037&collectionId=0.

[4] Nen connect-en 442-2, . URL https://connect.nen.nl/Family/Detail/29215?compId=10037&collectionId=0.

[5] En 442-12:2014 radiators and convectors – policies - iea, . URL https://www.iea.org/policies/7036-en-442-122014-radiators-and-convectors.

[6] Drain water heat recovery systems. URL https://m5shops.tk/ProductDetail.aspx?iid=589328732&pr=67.88.

[7] Innovative wastewater energy tech to be installed at toronto western hospital, . URL https://canada.constructconnect.com/dcn/news/technology/2021/08/innovative-wastewater-energy-tech-to-be-installed-at-toronto-western-hospital.

[8] Tank version of huber heat exchanger rowin, . URL https://www.huber.de/huber-report/ablage-berichte/energy-from-wastewater/tank-version-of-huber-heat-exchanger-rowin.html.

[9] Daggegevens van het weer in nederland, . URL https://www.knmi.nl/nederland-nu/klimatologie/daggegevens.

[10] Bodemtemperaturen, . URL https://www.knmi.nl/nederland-nu/klimatologie/bodemtemperaturen.

[11] Feasibility of small scale heat recovery from sewers. URL www.kwrwater.nl.

[12] The netherlands 2020 – analysis - iea. URL https://www.iea.org/reports/the-netherlands-2020.

[13] Rabtherm ag - member of the world alliance. URL https://solarimpulse.com/companies/rabtherm-ag.

[14] Cop of air refrigerator working on reversed carnot cycle with pv and ts diagram | mecholic. URL https://www.mecholic.com/2018/03/cop-reversed-carnot-cycle.html.

[15] Heat4cool sewage heat recovery for district heating and cooling, pilot installation in budapest, hungary. 2017. URL https://www.heat4cool.eu/wp-content/uploads/2019/04/2.-thermowatt-.pdf.

[16] Climate agreement, 2019. URL https://www.klimaatakkoord.nl/documenten/publicaties/2019/06/28/national-climate-agreement-the-netherlands.

[17] Iso 52016-1:2017, July 2022. URL https://www.iso.org/standard/65696.html.

[18] M.Y. Abdelsalam, M.F. Lightstone, and J.S. Cotton. A novel approach for modelling thermal energy storage with phase change materials and immersed coil heat exchangers. *International Journal of Heat and Mass Transfer*, 136:20–33, 2019. ISSN 0017-9310. doi: https://doi.org/10.1016/j.ijheatmasstransfer.2019.02.047. URL https://www.sciencedirect.com/science/article/pii/S0017931018349743.

[19] F. Aguilar, D. Crespí-Llorens, S. Aledo, and P. V. Quiles. One-dimensional model of a compact dhw heat pump with experimental validation. *Energies*, 14, 6 2021. ISSN 19961073. doi: 10.3390/en14112991.

[20] Hazim B. Awbi. Calculation of convective heat transfer coefficients of room surfaces for natural convection. *Energy and Buildings*, 28:219–227, 1998. ISSN 03787788. doi: 10.1016/s0378-7788(98)00022-x.

[21] Ugne Bunikyte. Numerical model of additional heat source to the heatcycle, 2021. Available upon request.

[22] Jingyong Cai, Zhouhang Li, Jie Ji, and Fan Zhou. Performance analysis of a novel air source hybrid solar assisted heat pump. *Renewable Energy*, 139:1133–1145, 8 2019. ISSN 0960-1481. doi: 10.1016/J.RENENE.2019.02.134.

[23] Jingyong Cai, Haihua Zhou, Zhengrong Shi, Haifei Chen, and Tao Zhang. Analysis and optimization on the performance of a heat pump water heater with solar-air dual series source. *Case Studies in Thermal Engineering*, 28:101577, 12 2021. ISSN 2214-157X. doi: 10.1016/J.CSITE.2021.101577.

[24] Jingyong Cai, Haihua Zhou, Lijie Xu, Zhengrong Shi, Tao Zhang, and Jie Ji. Energy and exergy analysis of a novel solar-air composite source multi-functional heat pump. *Renewable Energy*, 185:32–46, 2 2022. ISSN 0960-1481. doi: 10.1016/J.RENENE.2021.12.033.

[25] CE Delft. Luchtwarmtepomp. URL https://cedelft.eu/.

[26] Miguel Duarte, Luis Pires, Pedro Silva, and Pedro Gaspar. Experimental comparison between r409a and r437a performance in a heat pump unit. *Open Engineering*, 7, 04 2017. doi: 10.1515/eng-2017-0011.

[27] J.R. García-Cascales, F. Vera-García, J.M. Corberán-Salvador, and J. Gonzálvez-Maciá. Assessment of boiling and condensation heat transfer correlations in the modelling of plate heat exchangers. *International Journal of Refrigeration*, 30(6):1029–1041, 2007. ISSN 0140-7007. doi: https://doi.org/10.1016/j.ijrefrig.2007.01.004. URL https://www.sciencedirect.com/science/article/pii/S0140700707000138.

[28] VDI Gesellschaft. *VDI Heat Atlas*. VDI-Buch. Springer Berlin Heidelberg, 2010. ISBN 9783540778769.

[29] Farzin Golzar and Semida Silveira. Impact of wastewater heat recovery in buildings on the performance of centralized energy recovery – a case study of stockholm. *Applied Energy*, 297:117141, 9 2021. ISSN 0306-2619. doi: 10.1016/J.APENERGY.2021.117141.

[30] Bruno Hadengue, Prabhat Joshi, Alejandro Figueroa, Tove A. Larsen, and Frank Blumensaat. In-building heat recovery mitigates adverse temperature effects on biological wastewater treatment: A network-scale analysis of thermal-hydraulics in sewers. *Water Research*, 204:117552, 10 2021. ISSN 0043-1354. doi: 10.1016/J.WATRES.2021.117552.

[31] Linda P Haugerud and Ingvild Lien. Analyse av feltmålinger av varmepumper i boliger, 2015. URL https://www.enova.no/download?objectPath=upload_images/A39CCBB0659E45289D1FFC54EBF71CFA.pdf.

[32] Arif Hepbasli, Emrah Biyik, Orhan Ekren, Huseyin Gunerhan, and Mustafa Araz. A key review of wastewater source heat pump (wwshp) systems, 2014. ISSN 01968904.

[33] K.G.T. Hollands, G.D. Raithby, and L. Konicek. Correlation equations for free convection heat transfer in horizontal layers of air and water. *International Journal of Heat and Mass Transfer*, 18(7):879–884, 1975. ISSN 0017-9310.

[34] Hossein Javadi, Seyed Soheil Mousavi Ajarostaghi, Marc A. Rosen, and Mohsen Pourfallah. Performance of ground heat exchangers: A comprehensive review of recent advances. *Energy*, 178:207–233, 7 2019. ISSN 0360-5442. doi: 10.1016/J.ENERGY.2019.04.094.

[35] Sung Jhee, Kwan-Soo Lee, and Woo-Seung Kim. Effect of surface treatments on the frosting/defrosting behavior of a fin-tube heat exchanger. URL www.elsevier.com/locate/ijrefrig.

[36] Alco Kieft, Robert Harmsen, and Marko P. Hekkert. Heat pumps in the existing dutch housing stock: An assessment of its technological innovation system. *Sustainable Energy Technologies and Assessments*, 44:101064, 4 2021. ISSN 2213-1388. doi: 10.1016/J.SETA.2021.101064.

[37] Tomas Kropas, Giedrė Streckienė, and Juozas Bielskus. Experimental investigation of frost formation influence on an air source heat pump evaporator. *Energies*, 14, 9 2021. ISSN 19961073. doi: 10.3390/en14185737.

[38] Ridge National Laboratory. Biomass energy data book: Edition 4. 2011. URL www.e85fuel.com.

[39] Jesus Lago, Fjo De Ridder, Wiet Mazairac, and Bart De Schutter. A 1-dimensional continuous and smooth model for thermally stratified storage tanks including mixing and buoyancy. *Applied Energy*, 248:640–655, 8 2019. ISSN 03062619. doi: 10.1016/j.apenergy.2019.04.139.

[40] Mooyeon Lee, Yonghan Kim, Hosung Lee, and Yongchan Kim. Air-side heat transfer characteristics of flat plate finned-tube heat exchangers with large fin pitches under frosting conditions. *International Journal of Heat and Mass Transfer*, 53:2655–2661, 6 2010. ISSN 0017-9310. doi: 10.1016/J.IJHEATMASSTRANSFER.2010.02.047.

[41] Hong Li and Hongxing Yang. Study on performance of solar assisted air source heat pump systems for hot water production in hong kong. *Applied Energy*, 87:2818–2825, 9 2010. ISSN 0306-2619. doi: 10.1016/J.APENERGY.2009.06.023.

[42] Zuqiang Li and Xinghua Huang. Simulation analysis on operation performance of a hybrid heat pump system integrating photovoltaic/thermal and air source. *Applied Thermal Engineering*, 200:117693, 1 2022. ISSN 1359-4311. doi: 10.1016/J.APPLTHERMALENG.2021.117693.

[43] Cai Hua Liang, Xiao Song Zhang, Xiu Wei Li, and Xia Zhu. Study on the performance of a solar assisted air source heat pump system for building heating. *Energy and Buildings*, 43:2188–2196, 9 2011. ISSN 0378-7788. doi: 10.1016/J.ENBUILD.2011.04.028.

[44] Fengzhen Liu, Liang Wang, Qiang Wang, and Hongfei Wang. Experiment study on heating performance of solar-air source heat pump unit. *Procedia Engineering*, 205:3873–3878, 2017. ISSN 18777058. doi: 10.1016/j.proeng.2017.10.054.

[45] Allison J. Mahvi, Kalyan Boyina, Andy Musser, Stefan Elbel, and Nenad Miljkovic. Superhydrophobic heat exchangers delay frost formation and enhance efficency of electric vehicle heat pumps. *International Journal of Heat and Mass Transfer*, 172:121162, 6 2021. ISSN 0017-9310. doi: 10.1016/J.IJHEATMASSTRANSFER.2021.121162.

[46] Daniel Mensink. Modelling new sources - internship report. July 2022.

[47] A.F. Mills. *Basic Heat and Mass Transfer*. Prentice Hall, 1999. ISBN 9780130962478.

[48] Kashif Nawaz, Ahmed Elatar, and Brian Fricke. A critical literature review of defrost technologies for heat pumps and refrigeration systems. 2018. URL http://www.osti.gov/scitech/.

[49] Behrouz Nourozi, Qian Wang, and Adnan Ploskic. Energy and defrosting contributions of preheating cold supply air in buildings with balanced ventilation. *Applied Thermal Engineering*, 146:180–189, 1 2019. ISSN 1359-4311. doi: 10.1016/J.APPLTHERMALENG.2018.09.118.

[50] Nordic Council of Ministers. Nordsyn study on air-to-water heat pumps in humid nordic climate. 2019. doi: 10.6027/TN2019-502. URL http://dx.doi.org/10.6027/TN2019-502.

[51] K Sankaranarayanan P. Study of frost growth on heat exchangers used as outdoor coils in air source heat pump systems. 2011. URL https://hvac.okstate.edu/sites/default/files/pubs/theses/PhD/14-Sankar%20Thesis.pdf.

[52] Sustainable Technologies Evaluation Program. Evaluation of residential drain water heat recovery unit: Technical brief. URL https://sustainabletechnologies.ca/app/uploads/2015/06/DWHR_TechBrief_June2015.pdf.

[53] Siyuan Ran, Xianting Li, Wei Xu, and Baolong Wang. A solar-air hybrid source heat pump for space heating and domestic hot water. *Solar Energy*, 199:347–359, 3 2020. ISSN 0038-092X. doi: 10.1016/J.SOLENER.2020.02.038.

[54] Mohamed Salama and Mostafa H. Sharqawy. Experimental investigation of the performance of a falling-film drain water heat recovery system. *Applied Thermal Engineering*, 179:115712, 10 2020. ISSN 1359-4311. doi: 10.1016/J.APPLTHERMALENG.2020.115712.

[55] Frank Schmid. Sewage water: interesting heat source for heat pumps and chillers. URL https://heatpumpingtechnologies.org/publications/sewage-water-interesting-heat-source-forheat.pumps-and-chillers/.

[56] R.K. Sinnott and G. Towler. *Chemical Engineering Design*. Butterworth-Heinemann/IChemE series. Butterworth-Heinemann, 2009. ISBN 9780750685511. URL https://books.google.nl/books?id=ISQ9lAEACAAJ.

[57] Daniel Slys and Sabina Kordana. Financial analysis of the implementation of a drain water heat recovery unit in residential housing. *Energy and Buildings*, 71:1–11, 3 2014. ISSN 0378-7788. doi: 10.1016/J.ENBUILD.2013.11.088.

[58] I.M. Sobol. Sensitivity estimates for non-linear mathematical models. *Mathematical modelling and computational experiments*, 1:407–414, 1993.

[59] Mengjie Song, Jiankai Dong, Chili Wu, Yiqiang Jiang, and Minglu Qu. Improving the frosting and defrosting performance of air source heat pump units: review and outlook. *HKIE Transactions Hong Kong Institution of Engineers*, 24:88–98, 4 2017. ISSN 1023697X. doi: 10.1080/1023697X.2017.1313134.

[60] Wei Su, Weihao Li, Junming Zhou, and Xiaosong Zhang. Experimental investigation on a novel frost-free air source heat pump system combined with liquid desiccant dehumidification and closed-circuit regeneration. *Energy Conversion and Management*, 178:13–25, 12 2018. ISSN 0196-8904. doi: 10.1016/J.ENCONMAN.2018.09.085.

[61] Cody Unrau. Numerical investigation of one-dimensional storage tank models and the development of analytical modelling techniques. 2017. URL https://macsphere.mcmaster.ca/bitstream/11375/21274/2/Unrau_Cody_M_2017March_MASc.pdf.

[62] Ministerie van Binnenlandse Zaken en Koninkrijksrelaties. Hybride warmtepomp de nieuwe standaard vanaf 2026, May 2022. URL https://www.rijksoverheid.nl/actueel/nieuws/2022/05/17/hybride-warmtepomp-de-nieuwe-standaard-vanaf-2026.

[63] Ministerie van Economische Zaken en Klimaat. Klimaatplan. 2019.

[64] Lisanne van Thiel. Watergebruik thuis rapport 2016. URL www.tns-nipo.comhttps://www.vewin.nl/SiteCollectionDocuments/Publicaties/Cijfers/Watergebruik-Thuis-2016.pdf.

[65] Elumalai Vengadesan and Ramalingam Senthil. A review on recent developments in thermal performance enhancement methods of flat plate solar air collector. *Renewable and Sustainable Energy Reviews*, 134:110315, 12 2020. ISSN 1364-0321. doi: 10.1016/J.RSER.2020.110315.

[66] Pamela Vocale, Gian Luca Morini, and Marco Spiga. Influence of outdoor air conditions on the air source heat pumps performance. volume 45, pages 653–662. Elsevier Ltd, 2014. doi: 10.1016/j.egypro.2014.01.070.

[67] Sander Wapperom, Auke de Vries, Simon Terluij, Sytse Rinia van Nauta, Bo Verduijn, and Thomas Meyer Ranneft. Heatcycle combined with sustainable heating systems. June 2021.

[68] Zhanli Xi, Runming Yao, Jinbo Li, Chenqiu Du, Zixian Yu, and Baizhan Li. Experimental studies on hot gas bypass defrosting control strategies for air source heat pumps. *Journal of Building Engineering*, 43:103165, 11 2021. ISSN 23527102. doi: 10.1016/j.jobe.2021.103165.

[69] Dong Keun Yang, Kwan Soo Lee, and Simon Song. Fin spacing optimization of a fin-tube heat exchanger under frosting conditions. *International Journal of Heat and Mass Transfer*, 49:2619–2625, 7 2006. ISSN 0017-9310. doi: 10.1016/J.IJHEATMASSTRANSFER.2006.01.016.

[70] Zuliang Ye, Yikai Wang, Xiang Yin, Yulong Song, and Feng Cao. Comparison between reverse cycle and hot gas bypass defrosting methods in a transcritical co2 heat pump water heater. *Applied Thermal Engineering*, 196:117356, 9 2021. ISSN 1359-4311. doi: 10.1016/J.APPLTHERMALENG.2021.117356.

[71] Penglei Zhang, Xingyue Rong, Xiaorui Yang, and Dalin Zhang. Design and performance simulation of a novel hybrid pv/t-air dual source heat pump system based on a three-fluid heat exchanger. *Solar Energy*, 191:505–517, 10 2019. ISSN 0038-092X. doi: 10.1016/J.SOLENER.2019.09.024.

[72] Rijing Zhao, Dong Huang, Xueyuan Peng, and Lin Qiao. Comprehensive measures to enhance electric heater defrosting (ehd) performance for household frost-free refrigerators. *International Journal of Refrigeration*, 111:1–8, 3 2020. ISSN 0140-7007. doi: 10.1016/J.IJREFRIG.2019.12.002.

[73] Wen Zhong Zhou, Jian Xing Li, orderlithtml Zhou, Wen Zhong, Jian Xing, and Wenzhong Zhou. Sewage heat source pump system's application examples and prospect analysis in china. 2004. URL http://docs.lib.purdue.edu/iracc/734.

A

HeatCycle description

CONFIDENTIAL

B

Model inputs

B.1. Fluid properties

B.1.1. Air

Variable	Symbol	Value	Unit
Thermal conductivity	k	0.024	W/mK
Density	rho	1.2	kg/m^3
Thermal diffusivity	alpha	1.85E-05	m^2/s
Thermal expansion coefficient	beta	3.40E-03	1/K
kinematic viscosity	nu	1.60E-05	m^2/s
Prandtl number	Pr	0.71	
Dynamic viscosity	mu	1.81E-05	Pa s
Ambient temperature	T_inf	293	K
Specific heat	cp	1005	J/kgK
Fouling factor	F	1000	m^{2}K/W

B.1.2. Water

Variable	Symbol	Value	Unit
Density	rho	1000	kg/m^3
Specific heat capacity	cp	4186	J/kg/K
Thermal conductivity	k	0.6	W/mK
Dynamic viscosity	mu	1.00E-03	Pa s
Kinematic viscosity	nu	1.00E-06	Js/kg
Volumetric thermal expansion coefficient	beta	2.10E-04	1/K
Prandtl number	Pr	3.00E+00	
Thermal diffusivity	alpha	1.43E-07	m^2/s
Cold tap water temperature	T_cold_tap	283	K

B.1.3. Glycol

Variable	Symbol	Value	Unit
Density	rho	1110	kg/m^3
Specific heat capacity	cp	2433	J/kg/K
Thermal conductivity	k	0.25	W/mK
Thermal diffusivity	alpha	9.26E-08	m^2/s
Viscosity	mu	1.61E-02	Pa s
Kinematic viscosity	nu	0.0000052	m^2/s
Prandtl number	Pr	9.5	
Fouling factor	F	500	m^2K/W

B.2. Heat pump

Variable	**Symbol**	**Value**	**Unit**
Fluid	R410a	"R410a"	
Heat transfer area (evaporator)	A_evap	1.57	m^2
Heat transfer area (condenser)	A	0.8	m^2
Water volumetric flow rate through condenser	V_w	0.00022	m^3/s
Condenser plate thickness	t	0.0025	m
Condenser plate spacing	s	0.0018	m
Condenser equivalent diameter	d_e	0.0036	m
Condenser number of channels	N_ch	12	
Condenser plate width	w	0.124	m
Thermal conductivity of copper	k_cop	398	W/mK
Thermal conductivity of aluminium	k_al	247	W/mK
Evaporator equivalent diameter (shell)	d_e_w	0.048	m
Evaporator equivalent diameter (tube)	d_e_ref	0.01588	m
Evaporator tube thickness	t_evap	0.0015	m

B.2.1. Compressor datasheet

	Displacement	Frequency	Power supply	Capacity	Input power	Flow rate	Current
	cc	rps	V	W	W	kg/h	A
KSN98D22UFZ	9. 71	30	220	1920. 0	304. 0	25. 2	2. 26
*	Evaporating Temp.(℃)	Condensing Temp.(℃)	Suction Temp.(℃)	Liquid Temp.(℃)	Ambient Temp.(℃)		
	8	34. 2	18. 1	30. 1	35		

2、Data under different condition

Capacity(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60. 0	574. 5	766. 7	1012. 9	1303. 8	1629. 8	1981. 5
	55. 0	620. 8	816. 7	1061. 6	1346. 1	1660. 7	1996. 1
	50. 0	669. 5	874. 0	1122. 5	1405. 6	1713. 7	2037. 6
	45. 0	728. 1	946. 2	1203. 2	1489. 8	1796. 4	2113. 8
	40. 0	804. 2	1040. 8	1311. 3	1606. 3	1916. 4	2232. 2
	35. 0	905. 5	1165. 5	1454. 4	1762. 8	2081. 3	2400. 4
	30. 0	1039. 6	1327. 9	1640. 2	1966. 9	2298. 6	2626. 0

Input Power(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60. 0	530. 1	574. 3	607. 2	627. 5	634. 0	625. 6
	55. 0	473. 3	510. 8	537. 3	551. 5	552. 3	538. 3
	50. 0	425. 7	456. 6	476. 8	485. 0	480. 1	460. 7
	45. 0	385. 3	409. 7	423. 7	426. 1	415. 5	390. 9
	40. 0	350. 3	368. 3	376. 3	372. 9	356. 9	327. 0
	35. 0	318. 7	330. 6	332. 6	323. 5	302. 1	267. 2
	30. 0	288. 9	294. 5	290. 7	276. 1	249. 5	209. 7

Flow Rate(kg/h)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60. 0	9. 2	12. 2	16. 0	20. 4	25. 4	30. 8
	55. 0	9. 5	12. 4	16. 0	20. 1	24. 7	29. 6
	50. 0	9. 8	12. 7	16. 2	20. 1	24. 5	29. 0
	45. 0	10. 2	13. 2	16. 7	20. 5	24. 6	28. 9
	40. 0	10. 9	14. 0	17. 5	21. 3	25. 3	29. 4
	35. 0	11. 8	15. 1	18. 7	22. 6	26. 6	30. 5
	30. 0	13. 1	16. 6	20. 4	24. 4	28. 4	32. 3

Current(A)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60. 0	3. 94	4. 27	4. 51	4. 66	4. 71	4. 65
	55. 0	3. 52	3. 80	3. 99	4. 10	4. 11	4. 00
	50. 0	3. 16	3. 39	3. 54	3. 61	3. 57	3. 42
	45. 0	2. 86	3. 05	3. 15	3. 17	3. 09	2. 91
	40. 0	2. 60	2. 74	2. 80	2. 77	2. 65	2. 43
	35. 0	2. 37	2. 46	2. 47	2. 41	2. 25	1. 99
	30. 0	2. 15	2. 19	2. 16	2. 05	1. 85	1. 56

.	Displacement	Frequency	Power supply	Capacity	Input power	Flow rate	Current
	cc	rps	V	W	W	kg/h	A
KSN98D22UFZ	9.71	60	220	3110.0	791.0	42.3	5.37
*	Evaporating Temp.(℃)	Condensing Temp.(℃)	Suction Temp.(℃)	Liquid Temp.(℃)	Ambient Temp.(℃)		
	2.7	42.3	12.8	34.3	35		

2、Data under different condition

Capacity(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	1465.8	1816.3	2228.0	2709.4	3268.7	3914.2
	55.0	1590.4	1964.8	2402.6	2912.0	3501.4	4179.2
	50.0	1713.0	2109.7	2571.8	3107.7	3725.7	4434.1
	45.0	1833.7	2251.0	2735.9	3296.7	3941.6	4679.0
	40.0	1952.3	2388.8	2894.8	3478.8	4149.0	4913.8
	35.0	2069.1	2522.9	3048.5	3654.1	4348.1	5138.6
	30.0	2183.8	2653.6	3197.1	3822.7	4538.7	5353.4

Input Power(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	1017.9	1086.2	1138.5	1172.9	1187.6	1180.5
	55.0	933.7	991.4	1033.1	1057.0	1061.1	1043.6
	50.0	854.9	902.2	933.6	947.1	940.9	913.0
	45.0	780.9	818.1	839.3	842.7	826.3	788.3
	40.0	711.4	738.5	749.8	743.2	716.9	668.9
	35.0	645.6	663.0	664.5	648.1	612.0	554.3
	30.0	583.1	590.9	582.8	556.9	511.3	444.0

Flow Rate(kg/h)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	23.4	28.8	35.1	42.5	51.0	60.8
	55.0	24.3	29.8	36.1	43.6	52.2	62.0
	50.0	25.0	30.6	37.1	44.5	53.2	63.1
	45.0	25.7	31.3	37.9	45.4	54.1	64.0
	40.0	26.3	32.0	38.6	46.2	54.8	64.7
	35.0	26.9	32.7	39.2	46.8	55.5	65.4
	30.0	27.5	33.2	39.8	47.4	56.0	65.9

Current(A)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	6.91	7.37	7.73	7.96	8.06	8.01
	55.0	6.34	6.73	7.01	7.18	7.20	7.08
	50.0	5.80	6.12	6.34	6.43	6.39	6.20
	45.0	5.30	5.55	5.70	5.72	5.61	5.35
	40.0	4.83	5.01	5.09	5.05	4.87	4.54
	35.0	4.38	4.50	4.51	4.40	4.16	3.76
	30.0	3.96	4.01	3.96	3.78	3.47	3.01

	Displacement	Frequency	Power supply	Capacity	Input power	Flow rate	Current
	cc	rps	V	W	W	kg/h	A
KSN98D22UFZ	9.71	90	220	3508.0	1389.0	51.9	8.95
*	Evaporating Temp.(℃)	Condensing Temp.(℃)	Suction Temp.(℃)	Liquid Temp.(℃)	Ambient Temp.(℃)		
	-3.4	49.4	2.3	41.4	35		

2、Data under different condition

Capacity(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	2355.7	2846.9	3425.8	4130.4	4998.5	6068.1
	55.0	2536.3	3056.6	3669.9	4414.0	5326.9	6446.4
	50.0	2712.3	3259.0	3903.8	4684.6	5639.4	6806.0
	45.0	2883.1	3453.3	4126.9	4941.6	5935.5	7146.5
	40.0	3048.3	3639.2	4338.6	5184.5	6214.7	7467.2
	35.0	3207.1	3816.0	4538.5	5412.7	6476.4	7767.6
	30.0	3359.2	3983.1	4726.0	5625.7	6720.1	8047.1

Input Power(W)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	1533.7	1661.6	1755.5	1813.1	1832.3	1810.7
	55.0	1401.3	1512.7	1591.2	1634.4	1640.1	1606.1
	50.0	1280.6	1375.9	1439.1	1468.2	1460.8	1414.7
	45.0	1170.6	1249.9	1298.3	1313.5	1293.3	1235.3
	40.0	1070.1	1133.8	1167.6	1169.3	1136.5	1066.9
	35.0	978.1	1026.5	1046.0	1034.4	989.3	908.5
	30.0	893.5	926.9	932.5	907.8	850.8	759.0

Flow Rate(kg/h)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	37.7	45.2	54.0	64.7	78.0	94.3
	55.0	38.7	46.3	55.2	66.1	79.3	95.7
	50.0	39.6	47.3	56.3	67.2	80.5	96.8
	45.0	40.4	48.1	57.1	68.1	81.4	97.7
	40.0	41.1	48.8	57.9	68.8	82.1	98.3
	35.0	41.8	49.4	58.4	69.3	82.6	98.8
	30.0	42.3	49.8	58.8	69.7	83.0	99.0

Current(A)		Evaporating Temp.(℃)					
		-10.0	-5.0	0.0	5.0	10.0	15
Condensing Temp.(℃)	60.0	9.88	10.71	11.31	11.68	11.81	11.67
	55.0	9.03	9.75	10.25	10.53	10.57	10.35
	50.0	8.25	8.87	9.27	9.46	9.41	9.12
	45.0	7.54	8.05	8.37	8.46	8.33	7.96
	40.0	6.89	7.31	7.52	7.53	7.32	6.87
	35.0	6.30	6.61	6.74	6.67	6.37	5.85
	30.0	5.76	5.97	6.01	5.85	5.48	4.89

B.2.2. Fitted surfaces

To find Q_{evap}, Q_{cond}, COP, W_{in}, T_e or $\dot{m}_{ref}$ at a frequency of 30, 60 or 90 Hz, substitute corresponding coefficients in the equation.

Equation: $a*T_{cond}^3 + b*T_{evap}^3 + c*T_{cond}^2 + d*T_{evap}^2 + e*T_{cond}\,T_{evap} + f*T_{cond} + g*T_{evap} + h$

	Q_{evap}			Q_{cond}			COP		
Coefficient	**30 Hz**	**60 Hz**	**90 Hz**	**30 Hz**	**60 Hz**	**90 Hz**	**30 Hz**	**60 Hz**	**90 Hz**
a	0	0	0	0	0	0	-0.00016	-7.71E-05	-4.30E-05
b	0	0	0	0	0	0	0.000124	0.00014	0.000149
c	0.597595	-0.11997	-0.25465	0.701165	-0.12806	-0.21826	0.027284	0.013744	0.008113
d	0.496398	1.60013	3.2586	0.309607	1.37076	2.98116	0.003447	0.003853	0.004187
e	66.0614	148.105	208.635	58.4102	137.868	195.644	0.497095	0.478302	0.381909
f	-0.2392	-0.96158	-1.25712	-0.00029	-0.65968	-0.85208	-0.00794	-0.00779	-0.00614
g	-72.8452	-22.0584	-21.6836	-78.5039	-13.1876	-12.1	-1.58978	-0.88996	-0.58045
h	3269.93	3969.77	5603.69	3517.65	4043.36	5808.29	34.1796	22.9017	17.2351

Equation: $a*T_{cond}^3 + b*T_{evap}^3 + c*T_{cond}^2 + d*T_{evap}^2 + e*T_{cond}\,T_{evap} + f*T_{cond} + g*T_{evap} + h$									
	W_{in}			T_e			$\dot{m}_{ref}$		
Coefficient	**30 Hz**	**60 Hz**	**90 Hz**	**30 Hz**	**60 Hz**	**90 Hz**	**30 Hz**	**60 Hz**	**90 Hz**
a	0.002493	0.000711	0.001419	0.000587	8.07E-05	0.000117	0	0	0
b	-0.00165	-0.00254	-0.00303	0.000558	0.000287	0.00065	0	0	0
c	-0.22333	0.000778	0.014286	-0.04894	0.004219	0.00346	2.15E-06	-5.49E-07	-9.44E-07
d	-0.2331	-0.35684	-0.66388	-0.01907	-0.00439	-0.01934	1.86E-06	5.92E-06	1.20E-05
e	-8.75903	-15.502	-18.1361	-0.38138	-0.40198	-0.2198	0.00017	0.00041	0.000571
f	0.232935	0.402294	0.548665	-0.01484	-0.01511	-0.01678	8.40E-07	-3.84E-07	-8.16E-08
g	14.866	13.9698	16.9415	3.07728	1.20312	1.01509	-0.00023	7.14E-06	4.10E-05
h	-20.2408	144.101	377.461	-6.6951	15.1218	24.872	0.010489	0.011302	0.015885

B.3. Air source

Variable	Symbol	Value	Unit
Frost density	rho_f	951	kg/m^3
Latent heat of fusion	h_fus	5566	W/kgmin
Frost thermal conductivity	k_f	2.22	W/mK
Diameter of fan	d_fan	0.315	m
Air mass flow	m_a	0.43	kg/s
Glycol mass flow	m_w	0.25	kg/s
HEX area	A	13	m^2
Number of rows	N_rows	10	
Number of tubes	N_tubes	48	
Number of passes	N_pass	2	
Outer diameter	d_o	0.013	m
Inner diameter	d_i	0.01	m
Fin thickness	t_fin	0.0002	m
Fin spacing	s	0.003	m
Horizontal pitch	s_hor	0.028	m
Vertical pitch	s_vert	0.022	m
Heat transfer factor	j_h	0.004	

B.3.1. Fitted surfaces

To find frost thickness (mm) at various ambient temperatures for a set of RH % (x) and time (min) (y), substitute coefficients in the equation.

Equation: $(a*x^2 + b*y^2 + c^{**x} + d*y + e*x + f)*(h*x^3 + i*x^2 + j*x + k)$					
Coefficient	**0 °C**	**3 °C**	**5 °C**	**-5 °C**	**-8 °C**
a	-0.00022	-0.00022	-0.00022	-0.00022	-0.00022
b	0.000265	0.000265	0.000265	0.000265	0.000265
c	0.000843	0.000843	0.000843	0.000843	0.000843
d	0.031037	0.031037	0.031037	0.031037	0.031037
e	-0.06306	-0.06306	-0.06306	-0.06306	-0.06306
f	-1.00365	-1.00365	-1.00365	-1.00365	-1.00365
g	0	2.25E-05	-8.67E-06	2.96E-05	2.80E-05
h	0	-0.00595	0.002225	-0.00749	-0.00723
i	0	0.511332	-0.18952	0.62519	0.61037
j	0	-13.4926	6.09756	-16.5122	-16.2278

B.4. Wastewater bag

Variable	Symbol	Value	Unit
Length	dx	1.4	m
Width	dy	0.6	m
Height	dz	0.17	m
Crawlspace temperature	T_cs	283.15	K
Surface area of the bag	A	0.84	m^2
Dumped mass when bag is full	m_dump	15	kg
Minimum mass in bag	m_min	30	kg
Maximum mass in bag	m_max	195	kg
Mass flow of wastewater	m_ww	0.25	kg/s

```
# -*- coding: utf-8 -*-

g = 9.81
#%%
#Setting material choices for model
material_b = 'bag' #material of which bag is made of
material_f = 'water' #fluid in bag
material_cs = 'air' #fluid in crawlspace, always air
material_c = 'ceiling' #material of which ceiling is made of
material_s = 'soil_med' #type of soil
#%%
#Bag variables
t_b = 2*10**(-3) #thickness of bag
#%%
#Crawlspace variables
dx_cs = 4 #length of crawlspace
dy_cs = 0.7 #height of crawlspace
dz_cs = 4 #width of crawlspace
A_win = 0.1 #area of window in crawlspace, if non-existent set ACPH = 0
ACPH = 0/3600 #air change rate in crawlspace in 1/s
#%%
#Ceiling variables
y_c = 0.45 #thickness of ceiling
dy_c = 0.015625 #distance between ceiling nodes in m
```

```
imax_c = round(y_c/dy_c + 2) #calculation of number of nodes in ceiling
#%%s
#Soil variables
y_s = 0.4 #depth of soil modelled
dy_s = 0.015625 #distance between soil nodes in m
imax_s = round(y_s/dy_s + 2) #calculation of number of nodes in soil
#%%
#Timestepping variables, currently held constant but code should be improved in such a way
    ↪ that they do not need to remain constant
dt = 60 #timestep size in s, keep constant
nstep = 8760 #number of timesteps in a year, keep constant
#%%
#Temperatures used to calculate heat transfer coefficients (K)
T_mbet = 288
T_mbib = 278
T_mbit = 288
T_mc = 293
T_ms = 278
#%%
#Initial guesses for temps at timestep 0 in degrees Celsius
T_bib = 22
T_bit = 22
T_bet = 22
T_a = 22
T_w = 22
T_s = 22
T_c = 22
T_amb_top = 22
#%%
#Heat transfer coefficient estimate in the room above crawlspace
h_conv_top = 2
```

B.5. Boiler

Variable	Symbol	Value	Unit
Diameter	D	0.47	m
Height	H	1.75	m
Height of the vessel above the exit of the mass flow	H_above_exit	1.75	m
Height of the vessel below the exit of the mass flow	H_under_entry	0	m
Thermal conductivity of the wall	k	0.02	W/mK
Initial temperature	T_hot_0	298	K
Thickness of insulation	L_ins	0.102	m
No. of nodes	N_layers	100	-
Upper control T	T_up	323	K
Lower control T	T_low	318	K
Location of the DHW demand control temperature sensor	H_control	0.5	m
Height of Spiral	H_s	0.45	m
Top heat transfer coefficient	h_top	2.5	W/m^2K
Side heat transfer coefficient	h_side	3.5	W/m^2K

B.6. Wastewater-to-glycol heat exchanger

Variable	Symbol	Value	Unit
Plate spacing	s	0.0126	m
Equivalent diameter	d_e	0.0252	m
Width of plate	w	0.17	m
Cross sectional area of flow channel	A_f	0.002142	m^2
Initial temp.	T_init	293	K
Heat transfer area	A_ht	0.85	m^2
Mass flow	m_ww_g	0.25	kg/s

B.7. Radiator

Variable	Symbol	Value	Unit
Room set-point temperature	T_room	291.15	K
Coefficient K	K	13.06	
Coefficient n	n	1.315	
Supply mass flow	m_supply	0.25	kg/s
No. of radiators	n_rad	15	
Power of aux. heater	P_aux	2500	W
Upper SH buildup limit	up_lim	72000000	J
Lower SH buildup limit	low_lim	36000000	J

B.8. Other

Variable	Symbol	Value	Unit
Power of electric heater	P_el	2500	W
Set point temperature	T_sp	328.15	K

B.9. Domestic hot water demand

Time	Minute	Q (kWh)	f (l/min)	f (m^3/s)	T_m (°C)	T_p (°C)	V (m^3)
07:00	420	0.105	3	0.00005	25		0.002013
07:05	425	1.4	6	0.0001	40		0.026835
07:30	450	0.105	3	0.00005	25		0.002013
08:01	481	0.105	3	0.00005	25		0.002013
08:15	495	0.105	3	0.00005	25		0.002013
08:30	510	0.105	3	0.00005	25		0.002013
08:45	525	0.105	3	0.00005	25		0.002013
09:00	540	0.105	3	0.00005	25		0.002013
09:30	570	0.105	3	0.00005	25		0.002013
10:30	630	0.105	3	0.00005	10	40	0.002013
11:30	690	0.105	3	0.00005	25		0.002013
11:45	705	0.105	3	0.00005	25		0.002013
12:45	765	0.315	4	6.67E-05	10	55	0.006038
14:30	870	0.105	3	0.00005	25		0.002013
15:30	930	0.105	3	0.00005	25		0.002013
16:30	990	0.105	3	0.00005	25		0.002013
18:00	1080	0.105	3	0.00005	25		0.002013
18:15	1095	0.105	3	0.00005	40		0.002013
18:30	1110	0.105	3	0.00005	40		0.002013
19:00	1140	0.105	3	0.00005	25		0.002013
20:30	1230	0.735	4	6.67E-05	10	55	0.014088
21:15	1275	0.105	3	0.00005	25		0.002013
21:30	1290	1.4	6	0.0001	40		0.026835

B.10. Wastewater production

Time	Flow rate (l/min)	Temperature (C)	Action
07:04	2.6	30	Bathroom sink
07:05	2.6	30	
07:09	12	38	Shower
07:10	12	38	
07:11	12	38	
07:12	12	38	
07:34	2	30	Food preparation
08:05	2	30	
08:06	2	30	
08:19	2	30	Kitchen sink
08:20	2	30	
08:34	1.75	40	Washing machine
08:35	1.75	40	
08:49	1.75	40	
08:50	1.75	40	
08:54	6	18	Toilet flush

09:04	1.75	40	Washing machine
09:05	1.75	40	
09:19	6	18	Toilet flush
09:34	1.75	40	Washing machine
09:35	1.75	40	
11:34	6	18	Toilet flush
11:49	2	30	Kitchen sink
11:50	2	30	
13:34	6	18	Toilet flush
14:34	2	30	Kitchen sink
16:19	6	18	Toilet flush
16:34	2	30	Kitchen sink
16:35	2	30	
18:04	2.6	30	Food preparation
18:05	2.6	30	
18:08	6	18	Toilet flush
18:19	2.6	30	Food preparation
18:20	2.6	30	
18:23	6	18	Toilet flush
18:34	2.6	30	Food preparation
18:35	2.6	30	
19:04	2.6	30	
19:05	2.6	30	
20:34	4	60	Dishwasher
20:35	4	60	
20:36	4	60	
20:39	6	18	Toilet flush
20:55	12	38	Shower
20:56	12	38	
20:57	12	38	
20:58	12	38	
21:19	2.6	30	Bathroom sink
21:20	2.6	30	
21:22	6	18	Toilet flush
22:04	6	18	Toilet flush
23:04	6	18	Toilet flush

B.11. Climate data

DeWarmte has constructed a 10-year mean climate data-set. The hourly climate data was taken from KNMI, measured in De Bilt, from year 2007-2016 [9]. For each year, mean monthly weather conditions were calculated. The 10-year mean was also calculated for each month. For each month, the year whose monthly mean was closest to the 10-year mean of that month was found. The climate data measured in that month was used for the 10-year mean. This method prevents the averaging out of stark changes in weather conditions, thus not overestimating the system's performance. The years used for each month are:

- January - 2009
- February - 2013
- March - 2008
- April - 2015
- May - 2015
- June - 2015
- July - 2014
- August - 2015
- September - 2010
- October - 2010
- November - 2008
- December - 2011

The climate data is given in hourly timesteps. The data is transformed into minute timesteps by linearly interpolating between two hourly values.

B.11.1. Soil temperature

KNMI has also measured soil temperature at 1 meter depth in De Bilt [10]. The measurements fluctuate between a minimum of 8 °C in January and 17.5 °C in August. However, these values are measured in a field. The thermal charging in the soil due to insulation losses in suburban areas raises the soil temperature significantly. A previous study in DeWarmte [21] (may be obtained upon request), found that throughout a 4 month period in January to May, the soil temperature at a 40 cm depth stayed at approximately 11 °C. Thus, the constant temperature boundary condition in the wastewater bag model is set to a linear profile between 11 to 17.5 °C from January 1st to July 31st, and decreases back to 11 °C from August 1st to December 31st.

C

Model code

C.1. Heat pump

```
import numpy as np
import pandas as pd
import math
import copy
from CoolProp.CoolProp import PropsSI

class HeatPump:
    '''
    Class that models the heat pump
    '''
    def __init__(self, constants):
        '''
        Object initialization

        Parameters
        ----------
        constants : object
            Constants object created in read_constants.py.

        Returns
        -------
        None.

        '''
        # Heat pump constants
        self.fluid = constants.HP['R410a'] # Refrigerant type
        self.U = constants.HP['U'] # Overall heat transfer coefficient
        self.A = constants.HP['A'] # Heat transfer area of HEX in HP
        self.t = constants.HP['t'] # Condenser plate thickness
        self.s = constants.HP['s'] # Condenser plate spacing
        self.d_e = constants.HP['t'] # Condenser equivalent diameter
        self.N_ch = constants.HP['s'] # Condenser number of channels
        self.w = constants.HP['t'] # Condenser plate width
        self.k_cop = constants.HP['s'] # Thermal conductivity of copper
        self.k_al = constants.HP['t'] # Thermal conductivity of aluminium
        self.d_e_w = constants.HP['s'] # Evaporator equivalent diameter (shell)
        self.d_e_ref = constants.HP['t']# Evaporator equivalent diameter (tube)
```

```
        self.t_evap = constants.HP['s'] # Evaporator tube thickness
        # Coefficients from compressor datasheet lookup table
        self.coeff_Q = pd.read_excel('databases/HP_fit.xlsx', sheet_name='Q')
        self.coeff_COP = pd.read_excel('databases/HP_fit.xlsx', sheet_name='COP')
        self.coeff_W_in = pd.read_excel('databases/HP_fit.xlsx', sheet_name='W_in')
        self.coeff_T_e = pd.read_excel('databases/HP_fit.xlsx', sheet_name='T_e')
        self.coeff_m_ref = pd.read_excel('databases/HP_fit.xlsx', sheet_name='m_ref')
        # Water properties
        self.rho_water = constants.water['rho'] # Water density [kg/m3]
        self.cp_water = constants.water['cp'] # Water specific heat [J/kgK]
        self.k_water = constants.water['k'] # Water thermal conductivity
        self.nu_water = constants.water['nu'] # Water kinematic viscosity
        self.Pr_water = constants.water['Pr'] # Water Prandtl number
        # Glycol properties
        self.rho_glycol = constants.glycol['rho'] # Glycol density [kg/m3]
        self.cp_glycol = constants.glycol['cp'] # Glycol specific heat [J/kgK]
        self.fluid = 'R32'
        # Refrigerant properties
        self.rho_ref = constants.refrigerant['rho_ref'] # Refrigerant density
        self.nu_ref = constants.refrigerant['nu_ref'] # Refrigerant kinematic viscosity
        self.Pr_ref = constants.refrigerant['Pr_ref'] # Refrigerant Prandtl number
        self.k_ref = constants.refrigerant['k_ref'] # Refrigerant thermal conductivity
        self.P_cr_ref = constants.refrigerant['P_cr_ref'] #Refrierant critical pressure
        # Initialisations
        self.T_cond = 293 # Condensor refrigerant temperature [K]
        self.T_evap = 290 # Evaporator refrigerant temprature [K]
        self.T_e = 293 # Exhaust temperature [K]
        self.T_source_out = self.T_evap + 1 # Source outflow temperature [K]
        self.T_supply_out = 300 # Supply outflow temperature [K]
        self.T_supply_out_DHW= 300 # Supply outflow temperature for DHW [K]
        self.T_supply_out_SH = 300 # Supply outflow temperature for SH [K]
        self.W_compressor_in = 0 # Compressor work input [W]
        self.W_el = 0 # Electric heater for defrosting work input [W]
        self.Q_evaporator = 0 # Heat input [W]
        self.Q_condensor = 0 # Heat output [W]

    def heat_flow(self, f, T_cond, T_evap, HEX_type):
        '''
        Function used in the evaporator and the condenser to find the heat flow through the
            ↪ evaporator
        and condenser based on the compressor look up table.

        Parameters
        ----------
        f : float
            Compressor frequency [Hz].
        T_cond : float
            Refrigerant temperature in the condenser [K].
        T_evap : float
            Refrigerant temperature in the evaporator [K].
        HEX_type : string
            Type of heat exchanger, supports 'cond' and 'evap'.

        Returns
        -------
        Q : float
```

```
        Heat flow through the heat exchanger [W].

    '''

    if HEX_type !='cond' and HEX_type != 'evap':
        print('Unknown HEX_type')
    if HEX_type == 'cond':
        #Second order surface fit for condenser and evaporator capacities based off of
            ↪ compressor datasheet
        Q_cond_30 = self.coeff_Q.loc[4,'30c']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 30c']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'30c']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'30c']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'30c']*(T_evap-273.15) + self.coeff_Q.loc[0,'30c']
        Q_cond_60 = self.coeff_Q.loc[4,'60c']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 60c']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'60c']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'60c']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'60c']*(T_evap-273.15) + self.coeff_Q.loc[0,'60c']
        Q_cond_90 = self.coeff_Q.loc[4,'90c']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 90c']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'90c']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'90c']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'90c']*(T_evap-273.15) + self.coeff_Q.loc[0,'90c']
        if f < 25:
            # If the compressor is off, there is no heat transfer.
            Q = 0
        elif 25 <= f < 60:
            # Linear interpolation between values found in compressor datasheet
            Q = Q_cond_30 + (Q_cond_60-Q_cond_30)*(f - 30)/(60-30)
        elif 60 <= f <= 90:
            # Linear interpolation between values found in compressor datasheet
            Q = Q_cond_60 + (Q_cond_90-Q_cond_60)*(f - 60)/(90-60)
    if HEX_type == 'evap':
        #Second order surface fit for condenser and evaporator capacities based off of
            ↪ compressor datasheet
        Q_evap_30 = self.coeff_Q.loc[4,'30e']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 30e']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'30e']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'30e']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'30e']*(T_evap-273.15) + self.coeff_Q.loc[0,'30e']
        Q_evap_60 = self.coeff_Q.loc[4,'60e']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 60e']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'60e']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'60e']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'60e']*(T_evap-273.15) + self.coeff_Q.loc[0,'60e']
        Q_evap_90 = self.coeff_Q.loc[4,'90e']*(T_cond-273.15)**2. + self.coeff_Q.loc[5,'
            ↪ 90e']*(T_evap-273.15)**2. + self.coeff_Q.loc[3,'90e']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_Q.loc[1,'90e']*(T_cond-273.15) + self.coeff_Q
            ↪ .loc[2,'90e']*(T_evap-273.15) + self.coeff_Q.loc[0,'90e']
        if f < 25:
            # If the compressor is off, there is no heat transfer.
             Q = 0
        elif 25 <= f < 60:
            # Linear interpolation between values found in compressor datasheet
            Q = Q_evap_30 + (Q_evap_60-Q_evap_30)*(f - 30)/(60-30)
        elif 60 <= f <= 90:
            # Linear interpolation between values found in compressor datasheet
            Q = Q_evap_60 + (Q_evap_90-Q_evap_60)*(f - 60)/(90-60)
    return Q
```

```
def massflow_ref(self, f, T_cond, T_evap):
    '''
    Function used to calculate the mass flow of refrigerant based on compressor look-up
        ↪  table.

    Parameters
    ----------
    f : float
        Compressor frequency [Hz].
    T_cond : float
        Refrigerant temperature in the condenser [K].
    T_evap : float
        Refrigerant temperature in the evaporator [K]..

    Returns
    -------
    m_ref : float
        Mass flow of refrigerant [kg/s].

    '''
    #Surface fit for mass flow at each frequency
    m_30 = self.coeff_m_ref.loc[4,'f30']*(T_cond-273.15)**2. + self.coeff_m_ref.loc[5,'
        ↪ f30']*(T_evap-273.15)**2. + self.coeff_m_ref.loc[3,'f30']*(T_cond-273.15)*(
        ↪ T_evap-273.15) + self.coeff_m_ref.loc[1,'f30']*(T_cond-273.15) + self.
        ↪ coeff_m_ref.loc[2,'f30']*(T_evap-273.15) + self.coeff_m_ref.loc[0,'f30']
    m_60 = self.coeff_m_ref.loc[4,'f60']*(T_cond-273.15)**2. + self.coeff_m_ref.loc[5,'
        ↪ f60']*(T_evap-273.15)**2. + self.coeff_m_ref.loc[3,'f60']*(T_cond-273.15)*(
        ↪ T_evap-273.15) + self.coeff_m_ref.loc[1,'f60']*(T_cond-273.15) + self.
        ↪ coeff_m_ref.loc[2,'f60']*(T_evap-273.15) + self.coeff_m_ref.loc[0,'f60']
    m_90 = self.coeff_m_ref.loc[4,'f90']*(T_cond-273.15)**2. + self.coeff_m_ref.loc[5,'
        ↪ f90']*(T_evap-273.15)**2. + self.coeff_m_ref.loc[3,'f90']*(T_cond-273.15)*(
        ↪ T_evap-273.15) + self.coeff_m_ref.loc[1,'f90']*(T_cond-273.15) + self.
        ↪ coeff_m_ref.loc[2,'f90']*(T_evap-273.15) + self.coeff_m_ref.loc[0,'f90']
    # Linear interpolation between values found in compressor datasheet
    if f < 25:
         m_ref = 0
    elif 25 <= f < 60:
        m_ref = m_30 + (m_60-m_30)*(f - 30)/(60-30)
    elif 60 <= f <= 90:
        m_ref = m_60 + (m_90-m_60)*(f - 60)/(90-60)
    return m_ref

def T_exhaust(self, f, T_cond, T_evap):
    '''
    Function used to calculate the exhaust temperature of compressor based on
        ↪ compressor look-up table.

    Parameters
    ----------
    f : float
        Compressor frequency [Hz].
    T_cond : float
        Refrigerant temperature in the condenser [K].
    T_evap : float
        Refrigerant temperature in the evaporator [K]..
```

```
        Returns
        -------
        T_e : float
            Exhaust temperature [K].
        '''
        # Surface fits
        T_e_30 = self.coeff_T_e.loc[4,'f30']*(T_cond-273.15)**2. + self.coeff_T_e.loc[5,'
            ↪ f30']*(T_evap-273.15)**2. + self.coeff_T_e.loc[3,'f30']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_T_e.loc[1,'f30']*(T_cond-273.15) + self.
            ↪ coeff_T_e.loc[2,'f30']*(T_evap-273.15) +self.coeff_T_e.loc[0,'f30']
        T_e_60 = self.coeff_T_e.loc[4,'f60']*(T_cond-273.15)**2. + self.coeff_T_e.loc[5,'
            ↪ f60']*(T_evap-273.15)**2. + self.coeff_T_e.loc[3,'f60']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_T_e.loc[1,'f60']*(T_cond-273.15) + self.
            ↪ coeff_T_e.loc[2,'f60']*(T_evap-273.15) +self.coeff_T_e.loc[0,'f60']
        T_e_90 = self.coeff_T_e.loc[4,'f90']*(T_cond-273.15)**2. + self.coeff_T_e.loc[5,'
            ↪ f90']*(T_evap-273.15)**2. + self.coeff_T_e.loc[3,'f90']*(T_cond-273.15)*(
            ↪ T_evap-273.15) + self.coeff_T_e.loc[1,'f90']*(T_cond-273.15) + self.
            ↪ coeff_T_e.loc[2,'f90']*(T_evap-273.15) +self.coeff_T_e.loc[0,'f90']
        # Linear interpolation between values found in compressor datasheet
        if f < 25:
            if self.T_e > 293.15:
                self.T_e = self.T_e - 5
            else:
                self.T_e = 293.15
        elif 25 <= f < 60:
            self.T_e = T_e_30 + (T_e_60-T_e_30)*(f - 25)/(60-25) + 273.15
        elif 60 <= f <= 90:
            self.T_e = T_e_60 + (T_e_90-T_e_60)*(f - 60)/(90-60) + 273.15
        return self.T_e


    def HTC(self, hex_type, m_cond, P_cond, m_w_cond, m_w_evap, P_evap, m_evap):
        '''
        Function calculating the overall heat transfer coefficient in the condenser and
            ↪ evaporator

        Parameters
        ----------
        hex_type : string
            Type of heat exchanger, supports 'cond' and 'evap'.
        m_cond : float
            mass flow of refrigerant in condenser [kg/s].
        P_cond : float
            Pressure in condenser [Pa].
        m_w_cond : float
            mass flow of water in condenser [kg/s].
        m_w_evap : float
            mass flow of water in evaporator[kg/s].
        P_evap : float
            Pressure in evaporator [Pa].
        m_evap : float
            mass flow of refrigerant in evaporator [kg/s].

        Returns
        -------
```

```
        U_cond or U_evap : float
            Overall heat transfer coefficient of condenser or evaporator [W/m2K].

        '''
        #Approximate properties of water
        Pr_w = self.Pr_water
        k_w = self.k_water
        rho_w = self.rho_water
        if hex_type == 'cond':
           #HEX dimensions
           t = self.t
           s = self.s
           d_e = self.d_e
           N_ch = self.N_ch
           w = self.w
           k_cop = self.k_cop
           #Approximate R32 properties in condenser
           nu = self.nu_ref
           k = self.k_ref
           Pr = self.Pr_ref
           P_cr = self.P_cr_ref
           P_red = P_cond/P_cr
           Pr_w = self.Pr_water
           k_w = self.k_water
           rho_w = self.rho_water
           if m_cond != 0:
               G = m_cond/(N_ch*s*w)
               Re_l = G*d_e/nu
               x = np.array([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
               h_ref = np.mean(0.023*Re_l**(0.8)*Pr**(0.4)*k/d_e*((1-x)**(0.8)+ 3.8*x
                   ↪ **(0.76)*(1-x))/P_red**(0.38))
           else:
               h_ref = 0
           if m_w_cond != 0:
               U_w = m_w_cond/(rho_w*s*w*N_ch)
               Re_w = U_w*4*2*s/self.nu_water
               h_w = k_w/(4*2*s)*0.26*Re_w**(0.65)*Pr_w**(0.4)
           else:
               h_w - 0
           if m_w_cond != 0 and m_cond != 0:
               U_cond = (1/h_w +1/h_ref * t/k_cop)**(-1)
           elif m_w_cond == 0 and m_cond != 0:
               U_cond = (1/h_ref * t/k_cop)**(-1)
           elif m_w_cond != 0 and m_cond == 0:
               U_cond = (1/h_w * t/k_cop)**(-1)
           else:
               U_cond = k_cop/t
           return U_cond
        if hex_type == 'evap':
        #Shell side HTC
          k_al = self.k_al
          d_e_w = self.d_e_w
          U_w = 4*m_w_evap/(rho_w*d_e_w**2*np.pi)
          Re_w = U_w*d_e_w/self.nu_water
          if m_w_evap != 0:
              h_w = 1*10**(-2)*k_w/d_e_w * Re_w*Pr_w**0.33
```

```
        else:
            h_w = 0
        #Tube side HTC
        P_cr = self.P_cr_ref
        h_b = (0.104*(P_cr*10**(-5))**(0.69)*1**(1.7*(P_evap/P_cr)**(0.17)+4*(P_evap/P_cr)
            ↪ **(1.2)+10*(P_evap/P_cr)**(10)))**(1/0.3)
        v_ref = self.nu_ref
        k_ref = self.k_ref
        Pr_ref = self.Pr_ref
        rho_ref = self.rho_ref
        d_e_ref = self.d_e_ref
        U_ref = 4*m_evap/(rho_ref*d_e_ref**2*np.pi)
        Re_ref = U_ref*d_e_ref/v_ref
        h_fc = 3*10**(-3)*k_ref/d_e_ref * Re_ref*Pr_ref**0.33
        h_ref = h_b + h_fc
        t_evap = 1.5*10**(-3)
        if m_w_evap != 0 and m_evap != 0:
            U_evap = (1/h_w +1/h_ref * t_evap/k_al)**(-1)
        elif m_w_evap == 0 and m_evap != 0:
            U_evap = (1/h_ref * t_evap/k_al)**(-1)
        elif m_w_evap != 0 and m_evap == 0:
            U_evap = (1/h_w * t_evap/k_al)**(-1)
        else:
            U_evap = k_al/t_evap
        return U_evap

    def condenser(self, f, m_flow_supply, T_supply_in, T_supply_in_DHW, mode):
        '''
        Condenser function.

        Parameters
        ----------
        f : float
            Compressor frequency [Hz].
        m_flow_supply : float
            Mass flow through the supply side of the condensor [kg/s].
        T_supply_in : float
            Inflow temperature of the supply side [K].
        T_supply_in_DHW : float
            Inflow temperature of spiral heater [K]. Used as placeholder variable.
        mode: string
            'DHW' or 'SH'

        Returns
        -------
        None.

        '''


        if m_flow_supply != 0:
            # Initial guess for the supply outflow temperature

            self.T_supply_out = T_supply_in + 2
            # Calculate number of transfer units
            NTU = self.U*self.A/(m_flow_supply*self.cp_water)
```

```
        while 1:
            # Calculate the condenser refrigerant temperature based on the NTU method
            self.T_cond = T_supply_in + (self.T_supply_out - T_supply_in)/(1 - np.exp(-
                ↪ NTU))

            # Find heat transfer in the condensor for a given temperature and compressor
                ↪  frequency
            self.Q_condensor = self.heat_flow(f,self.T_cond, self.T_evap,'cond')

            # Calculate the temperature of the outflow based on the transferred heat.
            T_supply_out_calculated = T_supply_in + self.Q_condensor/(m_flow_supply*self
                ↪ .cp_water)

            # If the initial guess is close enough to the calculated temperature, the
                ↪ value is accepted.
            if abs(T_supply_out_calculated - self.T_supply_out ) < 0.001:
                break
            else:
                # The calculated value is not close enough, the loop is performed once
                    ↪ again with the mean of the guess and the calculated value
                self.T_supply_out = self.T_supply_out + (T_supply_out_calculated - self.
                    ↪ T_supply_out)/2
                self.m_ref = self.massflow_ref(f, self.T_cond, self.T_evap)
                self.P_cond = PropsSI('P','T',self.T_cond,'Q',0,self.fluid)
                # Calculation of new heat transfer coefficient
                self.U_cond = self.HTC('cond', self.m_ref, self.P_cond, m_flow_supply, 0,
                    ↪  0, 0)
                NTU = self.U_cond*self.A/(m_flow_supply*self.cp_water)
        #Placeholders when switches between circuits occur
        if mode == 'DHW':
            self.T_supply_out_DHW = self.T_supply_out
        if mode == 'SH':
            self.T_supply_out_SH = self.T_supply_out
            self.T_supply_out_DHW = T_supply_in_DHW




def evaporator(self, f, m_flow_source, T_source_in):
    '''
    Evaporator function.

    Parameters
    ----------
    f : float
        Compressor frequency [Hz].
    m_flow_source : float
        Mass flow through the source side of the evaporator [kg/s].
    T_source_in : float
        Inflow temperature of the source side [K].

    Returns
    -------
    None.
```

```
    '''

    if m_flow_source != 0:
        # Calculate number of transfer units
        NTU = self.U*self.A/(m_flow_source*self.cp_glycol)
        # Initial guess for the source outflow temperature
        self.T_source_out = T_source_in - 2
        # There must be a mass flow to transfer heat to the evaporator.
        while 1:
            # Calculate the evaporator refrigerant temperature based on the NTU method
            self.T_evap = T_source_in + (self.T_source_out - T_source_in)/(1 - np.exp(-
                ↪ NTU))
            # Find heat transfer in the evaporator for a given temperature and
                ↪ compressor frequency
            self.Q_evaporator = self.heat_flow(f, self.T_cond, self.T_evap, 'evap')
            # Calculate the temperature of the outflow based on the transferred heat.
            T_source_out_calculated = T_source_in - self.Q_evaporator/(m_flow_source*
                ↪ self.cp_water)
            # If the initial guess is close enough to the calculated temperature, the
                ↪ value is accepted.
            if abs(T_source_out_calculated - self.T_source_out ) < 0.001:
                break
            else:
                # The calculated value is not close enough, the loop is performed once
                    ↪ again with the mean of the guess and the calculated value.
                self.T_source_out = self.T_source_out + (T_source_out_calculated - self.
                    ↪ T_source_out)/2
                self.m_ref = self.massflow_ref(f, self.T_cond, self.T_evap)
                self.P_evap = PropsSI('P','T',self.T_evap,'Q',0,self.fluid)
                self.U_evap = self.HTC('evap', self.m_ref, self.P_cond,0, m_flow_source,
                    ↪ self.P_evap, self.m_ref)
                NTU = self.U_evap*self.A/(m_flow_source*self.cp_water)


def evaporator_defrost(self, f, m_flow_source, T_source_in, P_el):
    '''
    Function that models the behaviour of the evaporator during defrosting. Additional
        ↪ heat is added from electric heater.
    Parameters
    ----------
    f : float
        Compressor frequency [Hz].
    m_flow_source : float
        Mass flow through the source side of the evaporator [kg/s].
    T_supply_in : float
        Inflow temperature of the source side [K].
    P_el : float
        Power of defrosting heater [W]

    Returns
    -------
    None.

    '''
    if m_flow_source != 0:
```

```
        # Calculate number of transfer units
        NTU = self.U_evap*self.A/(m_flow_source*self.cp_glycol)
        # Initial guess for the source outflow temperature
        self.T_source_out = T_source_in + 2
        self.Q_evaporator = P_el
        # Calculate the temperature of the outflow based on the transferred heat by
            ↪ electric heater.
        self.T_source_out = T_source_in + self.Q_evaporator/(m_flow_source*self.
            ↪ cp_glycol)
        self.T_evap = T_source_in + abs(self.T_source_out - T_source_in)/(1 - np.exp(-
            ↪ NTU))


    def compressor(self, f, T_cond, T_evap):
        '''
        Compressor function.

        Parameters
        ----------
        f : float
            Compressor frequency [Hz]
        T_cond : float
            Refrigerant condensing temperature [K].
        T_evap : float
            Refrigerant evaporating temperature [K].

        Returns
        -------
        W_in : float
            Workout input of compressor [W].

        '''

        # Find the work input of the compressor based on the look-up table
        W_in_30 = self.coeff_W_in.loc[6,'f30']*(T_cond-273.15)**3. + self.coeff_W_in.loc[7,
            ↪ 'f30']*(T_evap-273.15)**3. + self.coeff_W_in.loc[4,'f30']*(T_cond-273.15)
            ↪ **2. + self.coeff_W_in.loc[5,'f30']*(T_evap-273.15)**2. + self.coeff_W_in.
            ↪ loc[3,'f30']*(T_cond-273.15)*(T_evap-273.15) + self.coeff_W_in.loc[1,'f30'
            ↪ ]*(T_cond-273.15) + self.coeff_W_in.loc[2,'f30']*(T_evap-273.15) + self.
            ↪ coeff_W_in.loc[0,'f30']
        W_in_60 = self.coeff_W_in.loc[6,'f60']*(T_cond-273.15)**3. + self.coeff_W_in.loc[7,
            ↪ 'f60']*(T_evap-273.15)**3. + self.coeff_W_in.loc[4,'f60']*(T_cond-273.15)
            ↪ **2. + self.coeff_W_in.loc[5,'f60']*(T_evap-273.15)**2. + self.coeff_W_in.
            ↪ loc[3,'f60']*(T_cond-273.15)*(T_evap-273.15) + self.coeff_W_in.loc[1,'f60'
            ↪ ]*(T_cond-273.15) + self.coeff_W_in.loc[2,'f60']*(T_evap-273.15) + self.
            ↪ coeff_W_in.loc[0,'f60']
        W_in_90 = self.coeff_W_in.loc[6,'f90']*(T_cond-273.15)**3. + self.coeff_W_in.loc[7,
            ↪ 'f90']*(T_evap-273.15)**3. + self.coeff_W_in.loc[4,'f90']*(T_cond-273.15)
            ↪ **2. + self.coeff_W_in.loc[5,'f90']*(T_evap-273.15)**2. + self.coeff_W_in.
            ↪ loc[3,'f90']*(T_cond-273.15)*(T_evap-273.15) + self.coeff_W_in.loc[1,'f90'
            ↪ ]*(T_cond-273.15) + self.coeff_W_in.loc[2,'f90']*(T_evap-273.15) + self.
            ↪ coeff_W_in.loc[0,'f90']
        #Linear interpolation
        if f < 25.:
            self.W_compressor_in = 0
```

```
        elif 25 <= f < 60:
            self.W_compressor_in = W_in_30 + (W_in_60-W_in_30)*(f - 30)/(60-30)
        elif 60 <= f <= 90:
            self.W_compressor_in = W_in_60 + (W_in_90-W_in_60)*(f - 60)/(90-60)


    def heat(self, mode, f, m_flow_supply, T_supply_in_DHW, T_supply_in_SH, m_flow_source,
        ↪ T_source_in, P_el):
        '''
        Function describing the heat pump components operating together
        Parameters
        ----------
        All inputs described in functions above

        Returns
        -------
        None.

        '''
        #Exhaust temperature calculating
        self.T_e = self.T_exhaust(f, self.T_cond, self.T_evap)

        if mode == 'DHW':
            self.W_el = 0
            i = 0
            while 1:
                # Convergence loop ensuring condenser and evaporator functions coincide
                T_cond_assumed = self.T_cond
                T_evap_assumed = self.T_evap
                # Compute the temperatures in the condenser
                self.condenser(f, m_flow_supply, T_supply_in_DHW, T_supply_in_DHW, mode)
                # Compute the temperatures in the evaporator
                self.evaporator(f, m_flow_source, T_source_in)
                i = i + 1
                # If convergence is reached, the loop is broken.
                if (abs(T_evap_assumed - self.T_evap) < 0.1) & (abs(T_cond_assumed - self.
                    ↪ T_cond) < 0.1) or i > 100:
                    # Compute the work input for the compressor:
                    self.compressor(f, self.T_cond, self.T_evap)
                    break

        if mode == 'SH':
            self.W_el = 0
            while 1:
                #Same convergence loop
                T_cond_assumed = self.T_cond
                T_evap_assumed = self.T_evap
                # Compute the temperatures in the condenser
                self.condenser(f, m_flow_supply, T_supply_in_SH, T_supply_in_DHW, mode)
                # Compute the temperatures in the evaporator
                self.evaporator(f, m_flow_source, T_source_in)
                # If convergence is reached, the loop is broken.
                if (abs(T_evap_assumed - self.T_evap) < 0.1) & (abs(T_cond_assumed - self.
                    ↪ T_cond) < 0.1) :
                    # Compute the work input for the compressor:
                    self.compressor(f, self.T_cond, self.T_evap)
```

```
                break


    if mode == 'off':
        # Shut off compressor
        self.compressor(0, self.T_cond, self.T_evap)
        self.T_supply_out = 293
        self.W_el = 0


    if mode == 'Defrosting':
        while 1:
          #Same convergence loop
            T_cond_assumed = self.T_cond
            T_evap_assumed = self.T_evap
            # Compute the temperatures in the condenser
            self.condenser(0, m_flow_supply, T_supply_in_SH, T_supply_in_DHW, mode)
            # Compute the temperatures in the evaporator
            self.evaporator_defrost(0, m_flow_source, T_source_in, P_el)
            # If convergence is reached, the loop is broken.
            if (abs(T_evap_assumed - self.T_evap) < 0.1) & (abs(T_cond_assumed - self.
                ↪ T_cond) < 0.1) :
                # Compute the work input for the compressor, equal to 0 as HP is turned
                    ↪ off for defrosting
                self.compressor(0, self.T_cond, self.T_evap)
                break
        self.W_compressor_in = 0
        self.W_el = P_el
```

C.2. Air source

```
import numpy as np
import pandas as pd

class AirSourceHeatPump:
    '''
    Object calculating heat exchange between air and outside unit of HP.
    Accounts for frosting and defrosting.
    Primary output is T_out of circulating fluid in outside HEX --> used as input for HP
        ↪ evap.

    '''
    def __init__(self, constants):
        #Air properties
        self.cp_a = constants.air['cp'] #Specific heat
        self.k_a = constants.air['k'] #Thermal conductivity
        self.Pr_a = constants.air['Pr'] #Prandtl number
        self.v_kin_a = constants.air['nu']#Kinematic viscosity
        self.F_a = constants.air['F']#Fouling factor
        #Glycol properties
        self.cp_w = constants.glycol['cp']#Specific heat
        self.k_w = constants.glycol['k']#Thermal conductivity
        self.Pr_w = constants.glycol['Pr']#Prandtl number
        self.v_kin_w = constants.glycol['nu']#Kinematic viscosity
        self.F_w = constants.glycol['F']#Fouling factor
        #Frost properties
        self.k_f = constants.airsource['k_f']#Thermal conductivity
        self.h_fus = constants.airsource['h_fus']#Latent heat of fusion
        self.rho_f = constants.airsource['rho_f']#Density
        #Drycooler geometry
        self.d_fan = constants.airsource['d_fan']#Fan diameter
        self.A = constants.airsource['A']#Heat transfer area
        self.k_al = constants.HP['k_al']#Thermal conductivity (aluminium)
        self.k_cu = constants.HP['k_cop']#Thermal conducitivity (copper)
        self.d_o = constants.airsource['d_o']#Outer diameter
        self.d_i = constants.airsource['d_i']#Inner diameter
        self.r_in = self.d_i/2
        self.t_fin = constants.airsource['t_fin']#Fin thickness
        self.s = constants.airsource['s']#Fin spacing
        self.s_vert = constants.airsource['s_vert']#Vertical pitch
        self.s_hor = constants.airsource['s_hor']#Horizontal pitch
        self.h_fin = self.s_vert - self.d_o
        self.N_tubes = constants.airsource['N_tubes']#No. of tubes
        self.N_pass = constants.airsource['N_pass']#No. of passes
        self.N_rows = constants.airsource['N_rows']#No. of rows
        self.A_fin_A = (self.A - np.pi*self.d_o*(self.s - self.t_fin))/self.A
        self.A_A_to = 1 + 2*(self.h_fin*(self.h_fin+self.d_o + self.t_fin))/(self.s * self.
            ↪ d_o)
        self.phi = (self.h_fin/self.d_o - 1)*(1+0.35*np.log(self.h_fin/self.d_o))
        self.X = self.phi*(self.d_o/2)*(2/(self.k_al*self.t_fin))**(0.5)
        self.j_h = constants.airsource['j_h']#Heat transfer factor
        #Initializations
        self.m_w = constants.airsource['m_w'] #Glycol mass flow
        self.m_a = constants.airsource['m_a']#Air mass flow
        v_a = self.m_a/(np.pi*self.d_fan**2/4)
```

```
        v_w = self.m_w/(1000*np.pi*self.d_i**2/4*self.N_tubes/self.N_pass)
        self.N_nodes = 100 #No. of nodes
        self.base_case = np.zeros(120)
        self.T_w_out_array = np.full(self.N_nodes,273.15)
        self.Q_array = np.zeros(self.N_nodes)
        self.defrosty_real = False
        self.dT_lm = 0
        self.U = self.heat_transfer_coeff(v_a, v_w)
        self.U_0 = self.U
        self.defrosty_loop = []
        self.defrosty_loop.append(False)
        self.t_f = 0
        self.t_m = 0
        x_time = np.arange(120)
        for i in range (0,120):
            self.base_case[i] = self.frosty(273.15, 100, x_time[i] , False)

    def heat_transfer_coeff(self, v_a, v_w):
        '''
        Parameters
        ----------
        v_a : float
            Velocity of air [m/s].
        v_w : float
            Velocity of water [m/s].

        Returns
        -------
        U : float
           Overall heat transfer coefficient.

        '''
        #Air heat transfer coefficient
        v_sa_den = self.s*((self.s_vert**2+self.s_hor**2/4)**(0.5) - self.d_o) + self.t_fin
            ↪ *((self.s_vert**2+self.h_fin**2/4)**(0.5))
        v_sa = v_a*self.s_hor*self.s/(2*v_sa_den)
        self.h_a = (self.k_a/self.d_o)*self.A_A_to**(-0.15)*self.Pr_a**(1/3)*(v_sa*self.d_o
            ↪ /self.v_kin_a)
        self.n_fin = np.tanh(self.X*(self.h_a)**0.5)/(self.X*(self.h_a)**0.5)
        self.h_vir = self.h_a*(1-self.n_fin)*self.A_fin_A
        #Water heat transfer coefficient
        v_sw = v_w
        Re_w = self.d_i*v_sw/self.v_kin_w
        self.h_w = self.k_w/self.d_i*self.j_h*Re_w*self.Pr_w**(0.33)*1
        A_A_i = self.A/(48*0.589*np.pi*self.d_i)
        self.U = (A_A_i*(1/self.F_w + 1/self.h_w)+1/self.F_a+1/self.h_vir+self.d_o*np.log(
            ↪ self.d_o/self.d_i)/(2*self.k_cu))**(-1)
        self.U_cond = (1/2+self.d_o*np.log(self.d_o/self.d_i)/(2*self.k_cu))**(-1)
        return self.U

    def Q_in(self, T_in, T_a, RH, time, m_w, T_evap):
        '''
        Method defining the heat flux (in W) from each CV in HEX.

        Parameters
        ----------
```

```
        T_in : float
            Input temperature to HEX in K.
        T_a : float
            Air temperature (from weatherdata) in K .
        RH : float
            % Relative humidity (from weatherdata).
        time : int
            The time at which the rate of frosting is defined in min.
        m_w : float
            Mass flow rate of glycol through drycooler [kg/s]
        T_evap: float
            Heat pump evaporating temperature [K]

        Returns
        -------
        None.
        '''
        #Frosting conditions
        if RH >= 65 and T_a <= 273.15 + 2.5 and m_w != 0:
            t_f = self.frosty(T_a, RH, time, self.defrosty_real)
        else:
            t_f = 0
        #Limit on frost thickness based on fin spacing
        if t_f > 3*10**(-3):
           t_f = 3*10**(-3)
        # Modify U based on frost thickness
        if t_f > 0:
            self.U = (1/self.U_0 + (2*(self.r_in+t_f)*np.log((self.r_in+t_f)/(self.r_in))
                ↪ /(2*self.k_f)))**(-1)
        #Initialization of variables for heat exchange loop
        if self.defrosty == False:
            self.T_w_out_est = T_evap + (4/self.N_nodes)
        if self.defrosty == False and self.defrosty_loop[-2] == True:
            self.T_in = T_evap
            self.T_w_out_est = T_in + (4/self.N_nodes)
        else:
            self.T_w_out_est = T_in - (4/self.N_nodes)
        self.Q_array = np.zeros(self.N_nodes)
        self.T_w_out_array = np.ones(self.N_nodes)*self.T_w_out_est
        self.T_a_out_array = np.zeros(self.N_nodes)
        T_w_out = self.T_w_out_est - 1
        i = 0
        #Heat exchange loop
        for j in range (0, self.N_nodes):
            if i > 0 and self.defrosty == False:
                T_in = T_w_out
                self.T_w_out_est = T_in + (4/self.N_nodes)
                i = 0
            if i > 0 and self.defrosty == True:
               T_in = T_w_out
               self.T_w_out_est = T_in - (4/self.N_nodes)
               i = 0
            if m_w == 0:
                self.Q = 0
                T_w_out = self.T_w_out_est
                self.T_a_out = T_a
```

```
            while abs(self.T_w_out_est - T_w_out) > 0.001:
                if t_f == 0:
                    self.U = self.U_0
                else:
                    self.U = (1/self.U_0 + (2*(self.r_in+t_f)*np.log((self.r_in+t_f)/(self.
                        ↪ r_in))/(2*self.k_f)))**(-1)
                if i > 0:
                    self.T_w_out_est = T_w_out
                i = i + 1
                self.m_a = 0.43
                self.Q = m_w*self.cp_w*(self.T_w_out_est - T_in)
                self.T_a_out = T_a - self.Q/(self.m_a*self.cp_a)
                self.dT_lm = ((T_a - self.T_w_out_est) - (self.T_a_out - T_in))/np.log((T_a
                    ↪ - self.T_w_out_est) /(self.T_a_out - T_in))
                if np.isnan(self.dT_lm) == True:
                   break
               #Use NTU method is dT_lm method fails
                else:
                    self.Q = self.U*(self.A/self.N_nodes)*self.dT_lm
                    T_w_out = self.Q/(m_w*self.cp_w) + T_in
            if np.isnan(self.dT_lm) == True:
                C_r = (self.cp_a*self.m_a)/(self.cp_w*m_w)
                NTU = self.U*self.A/(self.cp_a*self.m_a)
                eff = (1-np.exp(-NTU*(1+C_r)))/(1+C_r)
                self.Q = -eff*self.cp_a*self.m_a*(T_in - T_a)
                T_w_out = self.Q/(m_w*self.cp_w) + T_in
            self.Q_array[j] = self.Q
            self.T_w_out_array[j] = T_w_out
            self.T_a_out_array[j] = self.T_a_out
            self.check_i = i
        self.t_f = t_f

    def frosty(self, T_a, RH, time, defrosty):
        '''
        Method determining the frost thickness (in m) depending on ambient temp, RH, time
            ↪ passed since frosting started
        Parameters
        ----------
        T_a : float
            Air temperature (from weatherdata) in K .
        RH : float
            % Relative humidity (from weatherdata).
        time : int
            The time at which the rate of frosting is defined in min.
        defrosty : Bool
            if TRUE: defrosting on, if FALSE: defrosting off

        Returns :
            t_f
        -------
        float
            Frost thickness in m.
        '''
        #Surface fits
        fit = pd.read_excel('databases/frost_growth.xlsx', sheet_name = 'Sheet5')
        fit_T0 = fit['Ta_0']
```

```
        fit_T3= fit['Ta_3']
        fit_T5 = fit['Ta_5']
        fit_Tn5 = fit['Ta_-5']
        fit_Tn8 = fit['Ta_-8']
        #x is %RH,y is time (min)
        def surface_T0(x,y):
            return fit_T0[0]*x**2 + fit_T0[1]*y**2 + fit_T0[2]*x*y + fit_T0[3]*x + fit_T0
                ↪ [4]*y+fit_T0[5]
        def surface_T3(x,y):
            return (fit_T3[0]*x**3 + fit_T3[1]*x**2 + fit_T3[2]*x + fit_T3[3])*(fit_T0[0]*x
                ↪ **2 + fit_T0[1]*y**2 + fit_T0[2]*x*y + fit_T0[3]*x + fit_T0[4]*y+fit_T0
                ↪ [5])
        def surface_T5(x,y):
            return (fit_T5[0]*x**3 + fit_T5[1]*x**2 + fit_T5[2]*x + fit_T5[3])*(fit_T0[0]*x
                ↪ **2 + fit_T0[1]*y**2 + fit_T0[2]*x*y + fit_T0[3]*x + fit_T0[4]*y+fit_T0
                ↪ [5])
        def surface_Tn5(x,y):
            return (fit_Tn5[0]*x**3 + fit_Tn5[1]*x**2 + fit_Tn5[2]*x + fit_Tn5[3])*(fit_T0
                ↪ [0]*x**2 + fit_T0[1]*y**2 + fit_T0[2]*x*y + fit_T0[3]*x + fit_T0[4]*y+
                ↪ fit_T0[5])
        def surface_Tn8(x,y):
            return (fit_Tn8[0]*x**3 + fit_Tn8[1]*x**2 + fit_Tn8[2]*x + fit_Tn8[3])*(fit_T0
                ↪ [0]*x**2 + fit_T0[1]*y**2 + fit_T0[2]*x*y + fit_T0[3]*x + fit_T0[4]*y+
                ↪ fit_T0[5])
        #Linear interpolation between surfaces of varying ambient temperatures
        if 273.15+3 < T_a <= 273.15+5:
            t_f1 = surface_T5(RH, time)
            t_f2 = surface_T3(RH, time)
            t_f = (t_f1-t_f2)/2*(T_a-3-273.15) + t_f2
        if 273.15 < T_a <= 273.15+3:
            t_f1 = surface_T3(RH, time)
            t_f2 = surface_T0(RH, time)
            t_f = (t_f1-t_f2)/3*(T_a-273.15) + t_f2
        if 268.15 < T_a <= 273.15:
            t_f1 = surface_T0(RH, time)
            t_f2 = surface_Tn5(RH, time)
            t_f = (t_f1-t_f2)/(5)*(T_a - 268.15) + t_f2
        if 265.15 <= T_a <= 268.15:
            t_f1 = surface_Tn5(RH, time)
            t_f2 = surface_Tn8(RH, time)
            t_f = (t_f1-t_f2)/(3)*(T_a - 265.15) + t_f2
        if T_a < 265.15:
            t_f = surface_Tn8(RH, time)
        if t_f < 0:
            t_f = 0
        self.defrosty_real = False
        if sum(self.Q_array) < 0:
           #Thickness of frost that has melted
           self.defrosty_real = True
           self.t_m = sum(abs(self.Q_array))/(self.h_fus*self.rho_f*self.A)*10**(3)
           t_f = t_f - self.t_m
        #change from mm to m
        t_f = t_f * 10**(-3)
        return t_f

    def defrost(self, T_a, T_evap, defrosty_loop):
```

```
        '''
        Method defining whether defrosting is occuring or not
        Parameters
        ----------
        T_a : float
            Air temperature (from weatherdata) in K .
        T_evap : float
            Evaporating temperature in K.
        defrosty_loop : float array
            Array containing frost thickness from start of simulation
        Returns :
        -------
            None
        '''
        self.defrosty_loop = defrosty_loop
        if self.t_f > 1.7*10**(-3):
            self.defrosty = True
        elif defrosty_loop[-1] == True and self.t_f > 0.5*10**(-3):
              self.defrosty = True
        else:
            self.defrosty = False

    def time_pass(self, sim_time, defrosting_loop, t_f_prev):
        '''
        Method to aid in defining the initial time when defrosting stops and frosting
            ↪ begins again

        Parameters
        ----------
        sim_time : int
            time since simulation started.
        defrosting_loop : list of bool
            contains all previous bool conditions if frosting was occuring.
        t_f_prev : float
            previous timestep frost thickness (m).

        Returns : None

        ----------
        '''
        self.time = sim_time
        #if defrosting is occuring, at each timestep,
        #self.time corresponds to the time at which the frost growth rate should be defined
        if defrosting_loop[-1] == True:
           for i in range(len(self.base_case)-1, 0, -1):
               if self.base_case[i] > t_f_prev:
                   self.time = i
        self.counter = 0
        #at timestep at which defrosting stops,
        #self.time defines which rate to begin frosting again
        if sim_time > 0 and defrosting_loop[-1] == False and defrosting_loop[-2] == True:
            for i in range(len(self.base_case)-1, 0, -1):
                if self.base_case[i] > t_f_prev:
                    self.counter = i
                pass
            self.time = self.counter
```

C.3. Wastewater bag

```
import pandas as pd
import numpy as np
from source.water_bag_ugne import user_inputs as u_in
import scipy.linalg as sc
from source.water_bag_ugne.Functions import water_htc, air_htc, rad_htc, temp_soil_in

class WaterBag:
    '''Class of wastewater bag'''

    def __init__(self, constants, T_a, dt):
        '''Function for initialisation of properties and values'''
        # Bag properties
        self.m_max = constants.waterbag['m_max'] # Maximum amount of water in the bag [kg]
        self.m_min = constants.waterbag['m_min'] # Minimum amount of water in the bag [kg]
        self.m_dump = constants.waterbag['m_dump'] # Mass that is dumped when the
            ↪ temperature gets too cold [kg]
        self.dx = constants.waterbag['dx'] # Length of the bag [m2]
        self.dy = constants.waterbag['dy'] # Width of the bag [m2]
        self.dz = constants.waterbag['dz'] # Height of the bag [m2]
        self.A = constants.waterbag['A'] # Surface area of the bag [m2]
        self.T_cs = constants.waterbag['T_cs'] # Crawlspace temperature [K]
        # Air properties
        self.alpha = constants.air['alpha'] # Thermal diffusivity [m2/s]
        self.nu = constants.air['nu'] # Kinematic viscosity [m2/s]
        self.beta = constants.air['beta'] # Thermal expansion coefficient [1/K]
        self.Pr = constants.air['Pr'] # Prandtl number [-]
        self.k = constants.air['k'] # Thermal conductivity [W/m/K]
        # Water properties
        self.cp = constants.water['cp'] # Specific heat capacity [J/kgK]
        self.rho = constants.water['rho'] # Density [kg/m3]
        # Other properties
        self.g = constants.other['g'] # Acceleration due to gravity [m/s2]
        # Initialize bag water properties
        self.m_bag = 100 # Mass in the bag [kg]
        #Initialization from user input file
        self.b = bag()
        self.b.dx_b = self.dx
        self.b.dx_y = self.dz
        self.b.dx_z = self.dy
        self.t_b = u_in.t_b
        self.cs = crawlspace()
        self.s = soil()
        self.c = ceiling()
        #%%
        #Creating an instance of the temperature initializing class
        self.T_init = temp_initial(self.s,self.c, T_a)
       #Initialize temps:
        self.T_mbet = u_in.T_mbet
        self.T_mbib = u_in.T_mbib
        self.T_mbit = u_in.T_mbit
        self.T_mc = u_in.T_mc
        self.T_ms = u_in.T_ms
        self.T_amb_top = u_in.T_amb_top + 273.15
```

```
        #Initializing temperatures throughout soil and ceiling as arrays of zeros
        self.T_new_s = np.zeros(self.s.imax_s-2)
        self.T_old_s = np.zeros(self.s.imax_s-2)
        self.T_new_c = np.zeros(self.c.imax_c-2)
        self.T_old_c = np.zeros(self.c.imax_c-2)
        #Setting outside and soil at 1m depth boundary conditions
        self.T_in_s = temp_soil_in(u_in.nstep)
        self.T_outside = T_a
        #Initializations
        self.T_bib = 20 + 273.15
        self.T_bit = 20 + 273.15
        self.T_bet =20 + 273.15
        self.T_a = 20 + 273.15
        self.T_w = 20 + 273.15
        self.T_s = 20 + 273.15
        self.T_c = 20 + 273.15
        #Setting initial guesses for soil and ceiling temperature distributions (linear)
        self.T_old_s = self.T_init.initial_cond(self.T_bib,self.T_in_s[0], self.s.imax_s)
        self.T_old_c = self.T_init.initial_cond(self.T_amb_top-2, self.T_a+2, self.c.imax_c
            ↪ )
        #Initializing array of all temperatures used to timestep
        self.T_all_new = np.zeros((self.c.imax_c - 2 + 7 + self.s.imax_s - 2))
        self.T_all_old = np.zeros((self.c.imax_c - 2 + 7 + self.s.imax_s - 2))
        self.Q_tot = 0
        #Creating an instance of the matrix that remains constant during timestepping
        self.M = main_matrix(self.b, self.cs, self.s, self.c, self.T_init, dt)
        self.M_inv = self.M.M
#%%
    def TempCalc(self, iTime, dt, T_return, m_return, T_in, m_in):
        '''
        Function to calculate temperature inside the bag
        Parameters
        ----------
        iTime : int
            nth timestep of simulation.
        dt : int
            timestep size [s].
        T_return : float
            Temperature of fluid returning from HEX to bag [K].
        m_return : float
            Mass flow of fluid returning from HEX to bag [kg/s].
        T_in : float
            Temperature of new wastewater entering bag [K].
        m_in : float
            Mass flow of new wastewater entering bag [kg/s].
        Returns
        -------
        None.
        '''
        #Defining heat transfer coefficients
        T_mbet = self.T_init.T_mbet
        T_mbib = self.T_init.T_mbib
        T_mbit = self.T_init.T_mbit
        T_mc = self.T_init.T_mc
        T_ms = self.T_init.T_ms
        h_rad_cb = rad_htc(self.c.material_c.epsilon, T_mc, T_mbet)
```

```
        h_conv_top = u_in.h_conv_top
        h_conv_bot = air_htc(self.cs.dx_cs, self.cs.dy_cs, self.cs.dz_cs, u_in.A_win, u_in.
            ↪ ACPH, self.cs.material_cs.nu, self.cs.material_cs.alpha, self.cs.material_cs
            ↪ .k, self.cs.material_cs.Pr, self.cs.material_cs.beta, u_in.g, T_ms, T_mc)
        h_w = water_htc(u_in.g, T_mbib, T_mbit, self.b.dy_b, self.b.material_f.nu, self.b.
            ↪ material_f.alpha, self.b.material_f.k, self.b.material_f.beta)
        #Defining coefficients used to simplify making RHS vector
        F_cb = self.b.dx_b*self.b.dz_b/self.cs.dx_cs*self.cs.dz_cs
        Fo_s = self.s.material_s.alpha*dt/self.s.dy_s**2
        A_top_c = -self.c.dy_c*h_conv_top/(2*self.c.material_c.k)
        C_top_c = 2*A_top_c/(1-A_top_c)
        Fo_c = self.c.material_c.alpha*dt/self.c.dy_c**2
        D_a = 1/(1+(h_conv_bot*dt*2/(self.cs.material_cs.rho*self.cs.material_cs.Cp*self.cs
            ↪ .dy_cs))+dt*u_in.ACPH)
        D_w = 1/(1+(2*h_w*dt)/(self.b.material_f.rho*self.b.material_f.Cp*self.b.dy_b))
        #Initializing RHS vector and RHS soil and ceiling vectors
        R = np.zeros((self.c.imax_c - 2 + 7 + self.s.imax_s - 2))
        R_c = np.zeros(self.c.imax_c-2)
        R_s = np.zeros(self.s.imax_s - 2)
        for i in range(1,self.c.imax_c-3):
           R_c[i] = self.T_old_c[i]
        R_c[0] = ( self.T_old_c[0]-C_top_c*Fo_c* self.T_amb_top)
        R_c[self.c.imax_c-3] = self.T_old_c[self.c.imax_c-3]
        #Calculating soil RHS vector
        for i in range(1,self.s.imax_s-3):
           R_s[i] = self.T_old_s[i]
        R_s[0] = self.T_old_s[0]
        R_s[self.s.imax_s-3] = (self.T_old_s[self.s.imax_s-3] + 2*Fo_s* self.T_in_s[iTime
            ↪ -1])
        #RHS value of air temperature node
        R[self.c.imax_c-1] = D_a*((dt*u_in.ACPH* self.T_outside[iTime-1])+ self.T_a)
        #RHS value of water temperature node
        R[self.c.imax_c + 2] = D_w* self.T_w
        #Creating full RHS vector with ceiling and soil vectors
        R[0:self.c.imax_c-2] = R_c
        R[self.c.imax_c + 5: self.c.imax_c + 5 + self.s.imax_s-2] = R_s
        #Solving system of equations
        self.T_all_new[:] = np.dot(self.M_inv,R)
        #Setting newly calculated temperature values as old values at timestep n
        self.T_all_old[:] = self.T_all_new
        self.T_old_c = self.T_all_new[0:self.c.imax_c - 2]
        self.T_old_s = self.T_all_new[self.c.imax_c+5: self.c.imax_c + self.s.imax_s + 3]
        #Separating to arrays to create time sequence for temperatures of interest
        self.T_c = self.T_all_new[self.c.imax_c-2]
        self.T_a = self.T_all_new[self.c.imax_c-1]
        self.T_bet = self.T_all_new[self.c.imax_c]
        self.T_bit = self.T_all_new[self.c.imax_c+1]
        self.T_w = self.T_all_new[self.c.imax_c+2]
        #Imposing control at temperature threshold for water
        if self.T_w > 273.15+2:
            self.Q_return = m_return*dt*self.cp*(T_return - self.T_w)
        else:
            self.m_bag = self.m_bag - m_return*dt
            self.Q_return = 0
        if self.m_bag < self.m_min:
            self.m_bag = self.m_min
```

```
        if self.m_bag < self.m_max:
            self.Q_in = m_in*dt*self.cp*(T_in - self.T_w)
            self.m_bag = self.m_bag + m_in*dt
        if self.m_bag > self.m_max:
            self.Q_in = 0
            self.m_bag = self.m_max
        #Calculating heat extracted when threshold temperature is reached
        self.T_w = self.T_w + (self.Q_return + self.Q_in)/(self.m_bag*self.cp)
        #Continuing to separate arrays to create time sequence for temperatures of interest
        self.T_bib = self.T_all_new[self.c.imax_c+3]
        self.T_s = self.T_all_new[self.c.imax_c+4]
#%%
"""
Class of materials read in from Materials.csv
"""
class material:
    materials = pd.read_csv("C:/Users/ugneb/OneDrive/Documents/GitHub/Performance_model/
        ↪ LEEF␣Performance␣model/source/water_bag_ugne/Materials.csv", index_col=('Props')
        ↪ )
#%%
class bag:
    """
    Bag class
    Consists of x,y,z dimensions of bag, its thickness, its material and fluid in bag
    All values are input from user_inputs.py
    """
    def __init__(self):
        self.dx_b = u_in.dx_b
        self.dy_b = u_in.dy_b
        self.dz_b = u_in.dz_b
        self.t_b = u_in.t_b
        self.material_b = material.materials.loc[:,u_in.material_b]
        self.material_f = material.materials.loc[:,u_in.material_f]
class crawlspace:
    """
    Crawlspace class
    Consists of x,y,z dimensions of the crawlspace and fluid in crawlspace (air)
    All values are input from user_inputs.py

    """
    def __init__(self):
        self.dx_cs = u_in.dx_cs
        self.dy_cs = u_in.dy_cs
        self.dz_cs = u_in.dz_cs
        self.material_cs = material.materials.loc[:,u_in.material_cs]
#%%
class ceiling:
     """
     Ceiling class
     Consists of the ceiling's material, number of nodes in the ceiling and the spacing
         ↪ between them
     All values are input from user_inputs.py

     """
     def __init__(self):
        self.dy_c = u_in.dy_c
```

```
        self.imax_c = u_in.imax_c
        self.material_c = material.materials.loc[:,u_in.material_c]
#%%
class soil:
    """
    Soil class
    Consists of the soil's material, number of nodes in the soil and the spacing between
        ↪ them
    All values are input from user_inputs.py
    """
    def __init__(self):

        self.dy_s = u_in.dy_s
        self.imax_s = u_in.imax_s
        self.material_s = material.materials.loc[:,u_in.material_s]

class temp_initial:
    """
    Class used to initialize all temperature values, temperature arrays and set limits
        ↪ depending on which month simulation is chosen to run in
    Requires inputs of soil and ceiling instances
    Initial conditions for soil and ceiling temperature arrays are linear lines between two
        ↪  temperature guesses which may be varied in user_inputs.py
    T_mXXXs represent the mean yearly temperature of T_XXX, used to calculate heat transfer
        ↪  coefficients and may be varied in user_inputs.py
    Initial guesses for all T_XXXs may be varied in user_inputs.py
    T_outside is based on a 10-year mean year, taken from DeWarmte's Weather_import.KNMI
    T_soil_in is based on a 10-year mean year, taken from KNMIs website at DeBilt location
        ↪ at 1m depth
    """

    def initial_cond(self, T_up, T_bot, imax):
        #function creates linear profile used to initialize temperatures in soil and
            ↪ ceiling nodes
        self.I = np.zeros(imax-2)
        for i in range(imax-2):
            self.I[i] = T_up - (i)*(T_up-T_bot)/(imax-2)
        return self.I
    def __init__(self, soil, ceiling, T_a):
        #Mean temperatures throughout simulation, used to calculate heat transfer
            ↪ coefficients
        self.T_mbet = u_in.T_mbet
        self.T_mbib = u_in.T_mbib
        self.T_mbit = u_in.T_mbit
        self.T_mc = u_in.T_mc
        self.T_ms = u_in.T_ms
        self.T_amb_top = u_in.T_amb_top + 273.15
        #Nodal temperatures initialized as array of zeros
        self.T_s = np.zeros(u_in.nstep)
        self.T_c = np.zeros(u_in.nstep)
        self.T_bib = np.zeros(u_in.nstep)
        self.T_bit = np.zeros(u_in.nstep)
        self.T_a = np.zeros(u_in.nstep)
        self.T_w = np.zeros(u_in.nstep)
        self.T_bet = np.zeros(u_in.nstep)
        #Initializing temperatures throughout soil and ceiling as arrays of zeros
```

```
        self.T_new_s = np.zeros(soil.imax_s-2)
        self.T_old_s = np.zeros(soil.imax_s-2)
        self.T_new_c = np.zeros(ceiling.imax_c-2)
        self.T_old_c = np.zeros(ceiling.imax_c-2)
        #Setting outside and soil at 1m depth boundary conditions
        self.T_in_s = temp_soil_in(u_in.nstep)
        #Setting initial guesses in nodal temperatures
        self.T_bib[0] = u_in.T_bib + 273.15
        self.T_bit[0] = u_in.T_bit + 273.15
        self.T_bet[0] = u_in.T_bet + 273.15
        self.T_a[0] = u_in.T_a + 273.15
        self.T_w[0] = u_in.T_w + 273.15
        self.T_s[0] = u_in.T_s + 273.15
        self.T_c[0] = u_in.T_c + 273.15
        #Setting initial guesses for soil and ceiling temperature distributions (linear)
        self.T_old_s = self.initial_cond(self.T_bib[0],self.T_in_s[0], soil.imax_s)
        self.T_old_c = self.initial_cond(self.T_amb_top-2, self.T_a[0]+2, ceiling.imax_c)
        #Initializing array of all temperatures used to timestep
        self.T_all_new = np.zeros((ceiling.imax_c - 2 + 7 + soil.imax_s - 2))
        self.T_all_old = np.zeros((ceiling.imax_c - 2 + 7 + soil.imax_s - 2))

class main_matrix():
    """
    Creates the matrix that does not change with time and is inversed when timestepping
    Requires instances of bag, crawlspace, soil, ceiling and initialized temperatures
        ↪ classes as inputs
    Calculates a ceiling and soil matrix that are put into the top-left and bottom-right of
        ↪  the final matrix respectively
    Temperature node coefficients inbetween are defined
    """
    def __init__(self, bag, crawlspace, soil, ceiling, temp_initial, dt):
        #Creating soil matrix
        M_s = np.zeros((soil.imax_s - 2, soil.imax_s - 2))
        A_s = (soil.material_s.k*bag.t_b)/(bag.material_b.k*soil.dy_s)+0.5
        B_s = (1-A_s)/A_s
        C_s = 1/A_s
        Fo_s = soil.material_s.alpha*dt/soil.dy_s**2
        for i in range(1,soil.imax_s-3):
            M_s[i,i-1] = -Fo_s
            M_s[i,i] = 1+2*Fo_s
            M_s[i,i+1] = -Fo_s
        M_s[0,0] = 1+2*Fo_s+B_s*Fo_s
        M_s[0,1] = -Fo_s
        M_s[soil.imax_s-3,soil.imax_s-4] = -Fo_s
        M_s[soil.imax_s-3,soil.imax_s-3] = 1+3*Fo_s
        #Defining heat transfer coefficients
        T_mbet = temp_initial.T_mbet
        T_mbib = temp_initial.T_mbib
        T_mbit = temp_initial.T_mbit
        T_mc = temp_initial.T_mc
        T_ms = temp_initial.T_ms
        h_rb = rad_htc(bag.material_b.epsilon, T_mc, T_mbet)
        h_rf = rad_htc(soil.material_s.epsilon, T_mc, T_ms)
        F_cb = bag.dx_b*bag.dz_b/crawlspace.dx_cs*crawlspace.dz_cs
        h_rad_cb = rad_htc(ceiling.material_c.epsilon, T_mc, T_mbet)
        h_conv_top = u_in.h_conv_top
```

```
h_conv_bot = air_htc(crawlspace.dx_cs, crawlspace.dy_cs, crawlspace.dz_cs, u_in.
    ↪ A_win, u_in.ACPH, crawlspace.material_cs.nu, crawlspace.material_cs.alpha,
    ↪ crawlspace.material_cs.k, crawlspace.material_cs.Pr, crawlspace.material_cs.
    ↪ beta, u_in.g, T_ms, T_mc)
h_w = water_htc(u_in.g, T_mbib, T_mbit, bag.dy_b, bag.material_f.nu, bag.material_f
    ↪ .alpha, bag.material_f.k, bag.material_f.beta)
#Creating ceiling matrix
M_c = np.zeros((ceiling.imax_c-2,ceiling.imax_c-2))
A_top_c = -ceiling.dy_c*h_conv_top/(2*ceiling.material_c.k)
A_bot_c = -ceiling.dy_c/(2*ceiling.material_c.k)
B_top_c = 2/(1-A_top_c) - 1
B_bot_c = 2/(1-A_bot_c*(h_rf+h_rb+h_conv_bot)) - 1
C_bot_c = 2*A_bot_c/(1-A_bot_c*(h_rf+h_rb+h_conv_bot))
Fo_c = ceiling.material_c.alpha*dt/ceiling.dy_c**2
for i in range(1,ceiling.imax_c-3):
    M_c[i,i-1] = -Fo_c
    M_c[i,i] = 1+2*Fo_c
    M_c[i,i+1] = -Fo_c
M_c[0,0] = 1+2*Fo_c-B_top_c*Fo_c
M_c[0,1] = -Fo_c
M_c[ceiling.imax_c-3,ceiling.imax_c-4] = -Fo_c
M_c[ceiling.imax_c-3,ceiling.imax_c-3] = 1+2*Fo_c-B_bot_c*Fo_c
#Initializing full matrix
M = np.zeros((ceiling.imax_c -2 + 7 + soil.imax_s - 2, ceiling.imax_c -2 + 7 + soil
    ↪ .imax_s-2))
#Setting ceiling and soil matrices in their respective places in the full matrix
M[0:ceiling.imax_c -2, 0:ceiling.imax_c - 2] = M_c
M[ceiling.imax_c -2 + 7:ceiling.imax_c -2 + 7 + soil.imax_s ,ceiling.imax_c -2 + 7:
    ↪ ceiling.imax_c -2 + 7+ soil.imax_s] = M_s
#Creating coefficients to simplify calculations of coefficients in the full matrix
D_c = 1/(1-A_bot_c*(h_rf+h_rb+ h_conv_bot))
D_a = 1/(1+(h_conv_bot*dt*2/(crawlspace.material_cs.rho*crawlspace.material_cs.Cp*
    ↪ crawlspace.dy_cs))+dt*u_in.ACPH)
D_bet = 1/(1+(bag.t_b/bag.material_b.k)*(h_conv_bot+h_rad_cb*F_cb))
D_bit = 1/(1+bag.material_b.k/(bag.t_b*h_w))
D_w = 1/(1+(2*h_w*dt)/(bag.material_f.rho*bag.material_f.Cp*bag.dy_b))
D_bib = 1/(1+bag.material_b.k/(bag.t_b*h_w))
D_s = 1/(1+(bag.material_b.k*soil.dy_s)/(2*soil.material_s.k*bag.t_b))
#Last T_c node matrix line
#air temperature node
M[ceiling.imax_c -3, ceiling.imax_c-1]= Fo_c*C_bot_c*h_conv_bot
#bag top external temperature node
M[ceiling.imax_c - 3, ceiling.imax_c] = Fo_c*C_bot_c*h_rb
#soil temperature node
M[ceiling.imax_c - 3, ceiling.imax_c+4] = Fo_c*C_bot_c*h_rf
#T_c matrix line
#last temperature node in ceiling
M[ceiling.imax_c - 2, ceiling.imax_c -3] = -D_c
#ceiling temperature node
M[ceiling.imax_c - 2, ceiling.imax_c -2] = 1
#air temperature node
M[ceiling.imax_c - 2, ceiling.imax_c -1] = D_c*(A_bot_c*h_conv_bot)
#bag top temperature node
M[ceiling.imax_c - 2, ceiling.imax_c] = D_c*(A_bot_c*h_rb)
#soil temperature node
M[ceiling.imax_c - 2, ceiling.imax_c+4] = D_c*(A_bot_c*h_rf)
```

```
        #T_a matrix line
        #ceiling temperature node
        M[ceiling.imax_c - 1, ceiling.imax_c -2] = -D_a*(h_conv_bot*dt/(crawlspace.
            ↪ material_cs.rho*crawlspace.material_cs.Cp*crawlspace.dx_cs*crawlspace.dy_cs*
            ↪ crawlspace.dz_cs))*crawlspace.dx_cs*crawlspace.dz_cs
        #air temperature node
        M[ceiling.imax_c - 1, ceiling.imax_c -1] = 1
        #bag top temperature node
        M[ceiling.imax_c - 1, ceiling.imax_c] = -D_a*(h_conv_bot*dt/(crawlspace.material_cs
            ↪ .rho*crawlspace.material_cs.Cp*crawlspace.dx_cs*crawlspace.dy_cs*crawlspace.
            ↪ dz_cs))*bag.dx_b*bag.dz_b
        #soil temperature node
        M[ceiling.imax_c - 1, ceiling.imax_c+4] = -D_a*(h_conv_bot*dt/(crawlspace.
            ↪ material_cs.rho*crawlspace.material_cs.Cp*crawlspace.dx_cs*crawlspace.dy_cs*
            ↪ crawlspace.dz_cs))*(crawlspace.dx_cs*crawlspace.dz_cs-bag.dx_b*bag.dz_b)
        #T_bet matrix line
        #ceiling temperature node
        M[ceiling.imax_c, ceiling.imax_c -2] = -D_bet*(bag.t_b/bag.material_b.k)*h_rad_cb*
            ↪ F_cb
        #air temperature node
        M[ceiling.imax_c, ceiling.imax_c - 1] = -D_bet*(bag.t_b/bag.material_b.k)*
            ↪ h_conv_bot
        #bag top temperature node
        M[ceiling.imax_c, ceiling.imax_c] = 1
        #bag internal top temperature node
        M[ceiling.imax_c, ceiling.imax_c+1] = -D_bet
        #T_bit matrix line
        #bag top temperature node
        M[ceiling.imax_c+1, ceiling.imax_c] = -D_bit*(bag.material_b.k/(bag.t_b*h_w))
        #bag top internal temperature node
        M[ceiling.imax_c+1, ceiling.imax_c+1] = 1
        #water temperature node
        M[ceiling.imax_c+1, ceiling.imax_c+2] = -D_bit
        #T_w matrix line
        #bag top internal temperature node
        M[ceiling.imax_c+2, ceiling.imax_c+1] = -h_w*dt/(bag.material_f.rho*bag.material_f.
            ↪ Cp*bag.dy_b)*(D_w)
        #water temperature node
        M[ceiling.imax_c+2, ceiling.imax_c+2] = 1
        #bag bottom internal temperature node
        M[ceiling.imax_c+2, ceiling.imax_c+3] = -h_w*dt/(bag.material_f.rho*bag.material_f.
            ↪ Cp*bag.dy_b)*(D_w)
        #T_bib matrix line
        #water temperature node
        M[ceiling.imax_c+3, ceiling.imax_c + 2] = -D_bib
        #bag bottom internal temperature node
        M[ceiling.imax_c + 3, ceiling.imax_c+3] = 1
        #soil temperature node
        M[ceiling.imax_c+3, ceiling.imax_c+4] = -D_bib*(bag.material_b.k/(bag.t_b*h_w))
        #T_soil matrix line
        #bag bottom internal temperature node
        M[ceiling.imax_c + 4, ceiling.imax_c+3] = -D_s*(bag.material_b.k*soil.dy_s/(2*soil.
            ↪ material_s.k*bag.t_b))
        #soil temperature node
        M[ceiling.imax_c+4, ceiling.imax_c+4] = 1
        #first temperature node in soil
```

```
        M[ceiling.imax_c + 4, ceiling.imax_c+5] = -D_s
        #first T_soil node matrix line
        M[ceiling.imax_c + 5, ceiling.imax_c + 3] = -C_s*Fo_s
        self.M = sc.inv(M)
```

C.3.1. Additional functions

```
import numpy as np

def temp_soil_in(nstep):
    '''
    Function that calculates constant temperature boundary condition in soil
    Parameters
    ----------
    nstep : number of timesteps simulated.

    Returns
    -------
    T_in_s : Array of float [K]
        Constant temperature boundary condition of soil throughout the year.
    '''
    T_in_s = np.zeros(nstep*60)
    #Linear interpolation of temperatures, as explained in Appendix
    for i in range(0,8760*60):
        if i in range(0,5871*60):
           T_in_s[i] = 10.1 + (16.9-10.1)/(5871*60)*(i) + 273.15
        else:
            T_in_s[i] = 16.9 - (16.9 - 10.1)/(5870*60)*(i-(5870*60)) +273.15
    return T_in_s

def water_htc(g, T_2, T_1, l_c, nu, alpha, k, beta):
    '''
    Function that calculates water HTC

    Parameters
    ----------
    g : float
        Acceleration due to gravity [m2/s].
    T_2 : float
        Upper temperature bound [K].
    T_1 : float
        Lower temperature bound [K].
    l_c : float
        Characteristic length [m].
    nu : float
        Kinematic viscosity [Js/kg]
    alpha : float
        Thermal diffusivity [m2/s]
    k : float
        Thermal conductivity [W/mK].
    beta : float
        Thermal expansion [1/K].

    Returns
    -------
```

```
    h_w : float
        Water HTC [W/m2K].

    '''
    Ra = beta*g*(T_2 - T_1)*l_c**3/(nu*alpha)
    if Ra > 5830:
        Nu = 1 + 1.44*(1-1708/Ra)+((Ra/5830)**(1/3)-1)+2*(Ra**(1/3)/140)**(1-np.log(Ra
            ↪ **(1/3)/140))
        h_w = Nu*k/l_c
    else:
        Nu = 1
        h_w = Nu*k/l_c
    return h_w

def air_htc(dx, dy, dz, A_win, ACPH, nu, alpha, k, Pr, beta, g, T_2, T_1):
    '''
    Function that calculates air HTC

    Parameters
    ----------
    dx : float
        Crawlspace length [m].
    dy : float
        Crawlspace height [m].
    dz : float
        Crawlspace width [m].
    nu : float
        Kinematic viscosity [Js/kg]
    alpha : float
        Thermal diffusivity [m2/s]
    k : float
        Thermal conductivity [W/mK].
    beta : float
        Thermal expansion [1/K].
    Pr : float
        Prandtl number.
    g : float
        Acceleration due to gravity [m2/s].
    T_2 : float
        Upper temperature bound [K].
    T_1 : float
        Lower temperature bound [K].

    Returns
    -------
    h_comb : float
        Air HTC [W/m2K].

    '''
    #Forced convection htc
    U = ACPH*dx*dy*dz/A_win
    Re = U*dx/nu
    Nu_forced = 0.664*Re**(0.5)*Pr**(1/3)
    #Free convection htc
    l_c = dx*dz/(2*(dx+dz))
    f2 = (1+(0.322/Pr)**(11/20))**(-20/11)
```

```
    Ra = beta*g*(T_2 - T_1)*l_c**3/(nu*alpha)
    Nu_free = 0.15*(Ra*f2)**(1/3)
    Nu_comb = (Nu_free**3 + Nu_forced**3)**(1/3)
    h_comb = Nu_comb*k/l_c
    return h_comb

def rad_htc(epsilon, T_1, T_2):
    '''
    Parameters
    ----------
    epsilon : float
        Material emissivity.
    T_2 : float
        Upper temperature bound [K].
    T_1 : float
        Lower temperature bound [K].

    Returns
    -------
    h_rad : float
        Heat transfer coefficient [W/m2K].
    '''
    sigma = 5.67 * 10**(-8)
    h_rad = 4*sigma*epsilon*((T_1 + T_2)/2)**3
    return h_rad
```

C.4. Boiler

```
import numpy as np

class Boiler():
    ''' Class that calculates the boiler temperature based on the heat inflow from the
        ↪ spiral heater and the mass flow required for DHW use.'''
    def __init__(self, constants):
        # Boiler specifications
        self.D = constants.boiler['D'] # Diameter [m]
        self.N_layers = int(constants.boiler['N_layers'] ) # Amounts of layers [-]
        self.H = constants.boiler['H'] # Height [m]
        self.H_above_exit = constants.boiler['H_above_exit'] # Height of the vessel above
            ↪ the exit of the mass flow [m]
        self.H_under_entry = constants.boiler['H_under_entry'] # Height of the vessel under
            ↪  the entry of the mass flow [m]
        self.H_control = constants.boiler['H_control'] # Height of demand control sensor [m
            ↪ ]
        self.control_layer = int( self.N_layers * (self.H_control / self.H)) #Layer at
            ↪ which demand is measured
        self.k_ins = constants.boiler['k'] # Thermal conductivity of sides [W/m/K]
        self.L_ins = constants.boiler['L_ins'] # Thickness of the insulation layer [m]
        self.H_s = constants.boiler['H_s'] #Height of top of spiral heater [m]
        # Entry and exit node for spiral heater
        self.entry_layer = round((self.H_under_entry / self.H ) * self.N_layers) - 1
        self.exit_layer = round((1 - self.H_above_exit / self.H ) * self.N_layers) - 1
        # Initializations
        self.T = np.ones((self.N_layers,1))*constants.boiler['T_hot_0']
```

```
        self.T_outflow = constants.boiler['T_hot_0'] # Outflow boiler temperature [K]
        self.T_cold = constants.water['T_cold_tap'] # Cold tap water temperature [K]
        self.T_control = constants.boiler['T_hot_0'] # Demand control temperature [K]
        self.T_inf = constants.air['T_inf'] # Ambient temperature [K]
        self.g = constants.other['g'] # Gravitation accelaration [m/s2]
        self.T_up = constants.boiler['T_up'] # Upper control temperature
        self.T_low = constants.boiler['T_low'] # Lower control temperature
        # Water properties
        self.rho = constants.water['rho'] # Density [kg/m3]
        self.cp = constants.water['cp'] # Specific heat capacity [J/kg/K]
        self.alpha = constants.water['alpha'] # Thermal diffusivity [m2/s]
        self.k = constants.water['k'] # Thermal conductivity[W/mK]
        self.mu = constants.water['mu'] # Dynamic viscosity [Pa s]
        # Air properties
        self.beta_air = constants.air['beta'] # Thermal expansion coefficient [1/K]
        self.nu_air = constants.air['nu'] # Kinematic viscosity [m2/s]
        self.Pr_air = constants.air['Pr'] # Prandtl number [-]
        self.k_air = constants.air['k'] # Thermal conductiviy [W/m/K]
        # Variables for calculation
        self.dz_layer = self.H/self.N_layers # Layer height [m]
        self.lambda_i = 1/(self.rho*self.cp*np.pi/4*self.D**2)
        self.phi_i = 1/(self.rho*np.pi/4*self.D**2)
        self.h_top = constants.boiler['h_top']
        self.h_side = constants.boiler['h_side']
        self.R_top = np.pi/4*self.D**2 * (self.h_top*self.k_ins )/ (self.h_top*self.L_ins +
            ↪  self.k_ins)
        self.R_side = np.pi*self.D*(self.h_side*self.k_ins)/(self.k_ins + self.h_side*self.
            ↪ L_ins)
        self.beta_i = self.R_side/(self.L_ins*self.rho*self.cp)

    def MixTemperature(self,T):
        '''
        Function that accounts for bouyancy effects in the boiler.

        Parameters
        ----------
        T : Array of float
            Array of temperatures throughout boiler [K].

        Returns
        -------
        T : Array of float
            Array of mixed temperatures throughout boiler [K].
        '''
        while min(np.diff(T,axis=0)) < -0.001:
            for i in range(1,self.N_layers):
                if T[i-1,0] > T[i,0]:
                    dT= T[i-1,0] - T[i,0]
                    T[i-1,0] = T[i-1,0] - dT/2
                    T[i,0] = T[i,0] + dT/2
        return T

    def UpdateTemperature(self, dt, Q_sh, m_flow):
        '''
        Function that calculates temperatures throughout boiler based on energy balances
        Parameters
```

```
        ----------
        dt : int
            Timestep size [s].
        Q_sh : Array of float
            Array of heat from SH [W].
        m_flow : float
            Mass inflow into boiler [kg/s].
        Returns
        -------
        None.
        '''
        #Define matrices
        A = np.diag([1 + 2*self.alpha*dt/self.dz_layer**2 +self.beta_i*dt + self.phi_i*dt/
            ↪ self.dz_layer *m_flow]*self.N_layers)
        A[0,0] = 1 + self.alpha*dt/self.dz_layer**2 +self.beta_i*dt + self.phi_i*dt/self.
            ↪ dz_layer *m_flow
        A[self.N_layers-1,self.N_layers-1] = 1 + (1-self.dz_layer*self.R_top/self.k)*self.
            ↪ alpha*dt/self.dz_layer**2 +self.beta_i*dt + self.phi_i*dt/self.dz_layer *
            ↪ m_flow
        A = A + np.diag([-self.alpha*dt/self.dz_layer**2]*(self.N_layers-1),1) + np.diag([-
            ↪ self.alpha*dt/self.dz_layer**2 - self.phi_i*dt/self.dz_layer*m_flow]*(self.
            ↪ N_layers-1),-1)
        B = np.zeros((self.N_layers,1))+Q_sh*(-self.lambda_i*dt/self.dz_layer) - self.
            ↪ beta_i*dt*self.T_inf
        B[0,0] = B[0,0] - self.phi_i*dt/self.dz_layer*m_flow*self.T_cold
        B[self.N_layers-1] = B[self.N_layers-1] + self.alpha*dt*self.R_top*self.T_inf/(self
            ↪ .dz_layer*self.k)
        for i in range(self.entry_layer):
            if i != self.entry_layer - 1:
                A[i,i] = A[i,i] - self.phi_i*dt/self.dz_layer *m_flow
            if i != 0:
                A[i,i-1] = A[i,i-1] + self.phi_i*dt/self.dz_layer*m_flow
            if i == 0:
                B[i,0] = B[i,0] + self.phi_i*dt/self.dz_layer*m_flow*self.T_cold
            if i == self.entry_layer - 1:
                 B[i,0] = B[i,0] - self.phi_i*dt/self.dz_layer*m_flow*self.T_cold
        for i in range(self.exit_layer,self.N_layers):
            A[i,i] = A[i,i] - self.phi_i*dt/self.dz_layer *m_flow
            A[i,i-1] = A[i,i-1] + self.phi_i*dt/self.dz_layer*m_flow
        T_new = np.matmul(np.linalg.inv(A), (self.T - B))
        #Calculate new temperature distribution
        self.T = self.MixTemperature(T_new)
        self.T_outflow = self.T[self.exit_layer][0]
        self.T_control = self.T[self.control_layer][0]
```

C.5. Wastewater-to-glycol heat exchanger

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 9 15:19:43 2022

"""

import numpy as np
```

```
class PlateHEX():
    '''
    This object describes the behaviour of a tube in shell heat exchanger. The
    tube in shell heat exchanger can be thought of as a small vessel containing
    a spiral heat exchanger. Therefore, the code for the spiral heat exchanger
    and the vessel used in multiple heat storage objects are reused in this
    element.
    '''
    def __init__(self, constants):

        self.s = constants.tsHEX['s'] #Plate spacing
        self.d_e = constants.tsHEX['d_e'] #Equivalent diameter
        self.w = constants.tsHEX['w'] #Plate width
        self.A_f = constants.tsHEX['A_f'] #Flow area
        self.A_ht = constants.tsHEX['A_ht'] #Heat transfer area
        #Water properties
        self.mu_w = constants.water['mu'] #Viscosity
        self.k_w = constants.water['k'] #Thermal conductivity
        self.nu_w = constants.water['nu'] #Kinematic viscosity
        self.alpha_w = constants.water['alpha'] #Thermal diffusivity
        self.Pr_w = self.nu_w/self.alpha_w
        self.cp_w = constants.water['cp'] #Specific heat
        #Glycol properties
        self.mu_g = constants.glycol['mu'] #Viscosity
        self.k_g = constants.glycol['k'] #Thermal conductivity
        self.nu_g = constants.glycol['nu'] #Kinematic viscosity
        self.alpha_g = constants.glycol['alpha'] #Thermal diffusivity
        self.Pr_g = self.nu_g/self.alpha_g
        self.cp_g = constants.glycol['cp'] #Specific heat
        self.k_g = constants.glycol['k'] #Thermal conductivity
        #Initializations
        self.T_out_water = 293.15
        self.T_out_glycol = 293.15
        self.Q = 0

    def exchange_heat(self, dt, T_in_water, m_flow_water, T_in_glycol, m_flow_glycol):
        '''
        Function to calculates heat transfer between the glycol and wastewater.

        Parameters
        ----------
        dt : float
            time-step.
        T_in_water : float
            Inflow temperature of the water [K].
        m_flow_water : float
            Mass flow of water flowing through the tube [kg/s].
        T_in_glycol : float
            Inflow temperature of the glycol [K].
        m_flow_glycol : float
            Mass flow of the glycol flowing through the shell [kg/s].

        Returns
        -------
            None
        '''
```

```
        if m_flow_glycol != 0 and m_flow_water !=0:
            c_g = m_flow_glycol*self.cp_g
            c_w = m_flow_water*self.cp_w
            c_min = c_g
            c_max = c_w
            self.c_r = c_min/c_max
            if c_min == c_g:
                min_fluid = 'glycol'
                Re = m_flow_glycol*self.d_e/(self.A_f*self.mu_g)
                self.h = self.k_g/self.d_e*0.26*Re**0.65*self.Pr_g**0.4
                self.NTU = self.h*self.A_ht/c_min
            else:
                min_fluid = 'water'
                Re = m_flow_water*self.d_e/(self.A_f*self.mu_w)
                self.h = self.k_w/self.d_e*0.26*Re**0.65*self.Pr_w**0.4
                self.NTU = self.h*self.A_ht/c_min
            self.eff = (1-np.exp(-self.NTU*(1-self.c_r)))/(1-self.c_r*np.exp(-self.NTU*(1-
                ↪ self.c_r)))
            self.Q = self.eff*c_min*(T_in_water-T_in_glycol)
            # Calculate the temperature of outflow
            self.T_out_water = T_in_water - self.Q/(m_flow_water*self.cp_w)
            self.T_out_glycol = T_in_glycol + self.Q/(m_flow_glycol*self.cp_g)
        else:
            self.T_out_water = T_in_water
            self.T_out_glycol = T_in_water
```

C.6. Radiator

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 15 12:01:27 2022

"""

import numpy as np

class Radiator:

    def __init__(self, constants):
        self.T_room = constants.radiator['T_room'] #Desired room temperature
        self.K = constants.radiator['K'] #Coefficient (for explanation, look at thesis)
        self.n = constants.radiator['n'] #Coefficient (for explanation, look at thesis)
        self.m_supply = constants.radiator['m_supply'] #SH supply mass flow rate
        self.n_rad = constants.radiator['n_rad'] #Number of radiators in home
        self.P_aux = constants.radiator['P_aux'] #Power of auxiliary heater
        self.up_lim = constants.radiator['up_lim'] #Upper limit of SH demand buildup
        self.low_lim = constants.radiator['low_lim'] # Lower limit of SH demand buildup
        self.cp_water = constants.water['cp'] # Specific heat of water
        #Initialisations
        self.T_supply_in = 273.15 + 21
        self.P_rad = 0
        self.Q_left = 0
        self.aux = False
        self.W_aux = 0

    def SH_supply(self, T_supply_out, T_supply_in, Q_req, mode):
        '''
        Function calculating the supplied heat and temperature output of radiators
        Calculation method can be found in Section: Numerical model of radiators in thesis

        Parameters
        ----------

        T_supply_out : float
            Water temperature leaving condenser [K].
        T_supply_in : float
            Water temperature entering condenser [K].
        Q_req : float
            Required SH demand.
        mode : string
            Heat pump mode: 'DHW' or 'SH' or 'off'.

        Returns
        -------
        None.

        '''
        #If HP is delivering SH and the backup heater is required
        if mode == 'SH' and Q_req > self.up_lim:
            self.aux = True
            T_supply_out = T_supply_out+self.P_aux/(self.m_supply*self.cp_water)
```

```
            self.P_rad = self.n_rad*(self.K*((T_supply_out - T_supply_in)/(np.log((
                ↪ T_supply_out - self.T_room)/(T_supply_in - self.T_room))))**(self.n))
            self.E_rad = self.P_rad*60
            self.Q_left = Q_req - self.E_rad
            self.T_supply_in = T_supply_out - self.P_rad/(self.m_supply*self.cp_water)
            self.T_supply_out = T_supply_out
            self.W_aux = self.P_aux
        #If HP is delivering SH and the backup heater was required and the lower threshold
            ↪ has not yet been reached
        elif mode == 'SH' and Q_req > self.low_lim and self.aux == True:
            self.aux = True
            T_supply_out = T_supply_out+self.P_aux/(self.m_supply*self.cp_water)
            self.P_rad = self.n_rad*(self.K*((T_supply_out - T_supply_in)/(np.log((
                ↪ T_supply_out - self.T_room)/(T_supply_in - self.T_room))))**(self.n))
            self.E_rad = self.P_rad*60
            self.Q_left = Q_req - self.E_rad
            self.T_supply_in = T_supply_out - self.P_rad/(self.m_supply*self.cp_water)
            self.T_supply_out = T_supply_out
            self.W_aux = self.P_aux
        #If HP is delivering SH and the backup heater is not required
        elif mode == 'SH':
            self.aux = False
            self.P_rad = self.n_rad*(self.K*((T_supply_out - T_supply_in)/(np.log((
                ↪ T_supply_out - self.T_room)/(T_supply_in - self.T_room))))**(self.n))
            self.E_rad = self.P_rad*60
            self.Q_left = Q_req - self.E_rad
            self.T_supply_in = T_supply_out - self.P_rad/(self.m_supply*self.cp_water)
            self.T_supply_out = T_supply_out
            self.W_aux = 0
        #If HP is not delivering SH, but there is a remaining SH demand, turn on circulation
            ↪  without heating until the circuit is just above room temperature
        elif (mode == 'DHW' or 'off') and Q_req > 0 and T_supply_out > self.T_room+1 and
            ↪ T_supply_in > self.T_room+1:
            if Q_req > self.up_lim:
                self.aux = True
                T_supply_out = T_supply_out + self.P_aux/(self.m_supply*self.cp_water)
                self.W_aux = self.P_aux
            elif Q_req > self.low_lim and self.aux == True:
                self.aux = True
                T_supply_out = T_supply_out + self.P_aux/(self.m_supply*self.cp_water)
                self.W_aux = self.P_aux
            else:
                self.aux = False
                self.W_aux = 0
                T_supply_out = T_supply_in
            T_supply_in_ass = T_supply_in-1
            self.T_supply_out = T_supply_out
            i = 0
            while 1:
                self.P_rad = (self.n_rad*(self.K*((T_supply_out - T_supply_in_ass)/(np.log((
                    ↪ T_supply_out - self.T_room)/(T_supply_in_ass - self.T_room))))**(self
                    ↪ .n)))
                self.T_supply_in = T_supply_out - self.P_rad/(self.m_supply*self.cp_water)

                if abs(T_supply_in_ass - self.T_supply_in > 0.001) or i > 100:
                    break
```

```
                else:
                    T_supply_in_ass = self.T_supply_in
                    i = i+1
            self.E_rad = self.P_rad*60
            self.Q_left = Q_req - self.E_rad
        #If none of the above, no SH being fulfilled
        else:
            self.P_rad = 0
            self.E_rad = 0
            self.Q_left = Q_req
            self.T_supply_out = T_supply_in
```

C.7. Other

C.7.1. Filter

```
class Filter:
    ''' Class defining DeWarmte's filter
    '''
    def __init__(self):

        self.filter_ratio = 0.8

    def filter_water(self, m_flow_in):
        '''
        Mass lost to the filter
        Parameters
        ----------
        m_flow_in : Incoming wastewater flow.

        Returns
        -------
        None.

        '''
        self.m_flow_out = self.filter_ratio * m_flow_in
```

C.7.2. Electric heater

```
# -*- coding: utf-8 -*-

class ElectricHeater:
    '''Class that defines electric heaters used in system'''
    def __init__(self, constants):
        #Initializations
        self.P = constants.heaters['P_el']
        self.COP = 1
        self.Q_out = self.P*self.COP
```

C.7.3. Gas boiler

```
class GasBoiler():
    '''Class that defines additional heat required by a gas boiler '''
    def __init__(self, constants):
        #Setpoint temperature
        self.T_sp = constants.heaters['T_sp']
        # Water property
        self.cp = constants.water['cp'] # Water specific heat [J/kgK]
        #Initialization
        self.Q_req = 0

    def heat_req(self, m_flow, T_b):
        '''
        Parameters
        ----------
        m_flow : float
```

```
        Flow into gas boiler [kg/s].
    T_b : float
        Temperature from outflow of (heat pump heated) boiler [K].

    Returns
    -------
    None.

    '''
    #No flow, no requirement
    if m_flow == 0:
        self.Q_req = 0
    #Heat requirement
    if m_flow != 0:
        self.Q_req = m_flow*self.cp*(self.T_sp - T_b)
    #If temperature is higher than required, no requirement
    if self.Q_req < 0:
        self.Q_req = 0
```

C.8. Heat pump control

```
class HeatpumpController:

    def __init__(self, constants):

        #Initializations
        self.On = True
        self.f = 0 # Hz
        self.safety = False
        self.mode = 'DHW'
        self.i = 0
        #Boiler control temperatures
        self.T_up = constants.boiler['T_up']
        self.T_low = constants.boiler['T_low']
        #Mass flows
        self.m_supply_DHW = constants.spiralHEX['m_supply']
        self.m_supply_SH = constants.radiator['m_supply']
    def ChooseFrequency(self,T_source_input, T_e):
        '''
        Parameters
        ----------
        T_source_input : float
            Source input temperature [K]
        T_e : float
           Exhaust temperature of compressor [K].

        Returns
        -------
        None.

        '''
        #Compressor frequency control based on the source temperature (HP control in thesis
            ↪ )
        if T_e >= 105 + 273.15:
            self.f = 0
            self.safety = True
            self.i = 0
        elif self.safety == True and self.i < 3:
            self.safety = True
            self.f = 0
            self.i = self.i + 1
        elif T_e >= 90 + 273.15 and self.safety == True:
            self.safety = False
            self.f = 0
        elif 97 + 273.15 <= T_e < 105 + 273.15:
            self.f = self.f - 15
            self.safety = False
        elif 92 + 273.15 <= T_e < 97 + 273.15:
            self.f = self.f - 7.5
            self.control_1 = False
            self.safety = False
        elif T_e >= 83 + 273.15:
            self.control_1 = True
            self.f = self.f
```

```
            self.safety = False
        elif T_source_input >= 273.15+30 :
            self.f = 50
            self.control_1 = False
            self.safety = False
        elif T_source_input >= 273.15+20 :
            self.f = 54
            self.control_1 = False
            self.safety = False
        elif T_source_input >= 273.15+12 :
            self.f = 65
            self.control_1 = False
            self.safety = False
        elif T_source_input >= 273.15+4:
            self.f = 76
            self.control_1 = False
            self.safety = False
        elif T_source_input < 273.15+4:
            self.f = 80
            self.control_1 = False
            self.safety = False

    def control(self, T_e, T_storage_sensor, T_source_input, SH_demand, mode, defrosty,
        ↪ source):
        '''
        Parameters
        ----------
        T_e : float
            Compressor exhaust temperature [K].
        T_storage_sensor : float
            Temperature used to control boiler demand.
        T_source_input : float
            Source temperature [K].
        SH_demand : float
            Current SH demand [J].
        mode : string
            Previous timestep's HP mode (off, defrosting, DHW, SH).
        defrosty : bool
            Says whether defrosting is occuring (if yes, TRUE).
        source : string
            HP source, (AirSource or HeatCycle).

        Returns
        -------
        None.

        '''
        #Set compressor frequency
        self.ChooseFrequency(T_source_input, T_e)
        #If no source chosen, HP is off
        if source == 'None':
            self.f = 0
            self.On = False
            self.mode = 'off'
            self.m_flow_supply = 0
        #If air source is defrosting, HP is off
```

```
        if defrosty == True and source == 'AirSource':
            self.mode = 'Defrosting'
            self.m_flow_supply = 0
            self.On = False
        #If boiler as a demand
        elif T_source_input > 250 and T_storage_sensor < self.T_low:
            self.mode = 'DHW'
            self.m_flow_supply = self.m_supply_DHW
            self.On = True
        #Hystereses for boiler demand being fulfilled
        elif mode == 'DHW' and T_storage_sensor < self.T_up:
            self.mode = 'DHW'
            self.m_flow_supply = self.m_supply_DHW
            self.On = True
        #If there is no DHW demand, but there is SH demand
        elif T_source_input > 250 and SH_demand > 0:
            self.mode = 'SH'
            self.m_flow_supply = self.m_supply_SH
            self.On = True
        #No demands -> HP is off
        else:
            self.f = 0
            self.On = False
            self.mode = 'off'
            self.m_flow_supply = 0
```

C.9. Source control

```
class SourceController:
    def __init__(self, constants):
        #Initializations
        self.T_source = 280 # [K]
        self.T_return_wastewater_circuit = 280 # [K]
        self.m_flow_waterbag = 0
        self.m_flow_glycol_wb = 0
        self.m_flow_glycol_extra = 0
        self.m_flow_extra = 0
        self.selected_source = 'HeatCycle'
        self.on_HC = True
        self.hyst = False

        self.m_airsource = constants.airsource['m_w']
        self.m_ww = constants.waterbag['m_ww']
        self.m_ww_g = constants.tsHEX['m_ww_g']

    def control(self, T_extra_circuit, T_wastewater_circuit, T_out_heatpump, m_waterbag,
        ↪ T_waterbag):
        '''
        Parameters
        ----------
        T_extra_circuit : float
            AirSource temperature [K].
        T_wastewater_circuit : float
           Wastewater temperature [K].
        T_out_heatpump : float
            Condenser output temperature [K].
        m_waterbag : float
            Mass in wastewater bag [kg].
        T_waterbag : float
            Temperature of waterbag [K].

        Returns
        -------
        None.

        '''
        #Source is HeatCycle if it has a higher temperature and has a mass higher than the
            ↪ minimum allowable
        if T_waterbag > T_extra_circuit and m_waterbag > 30 and self.hyst == False:
            self.m_flow_waterbag = self.m_ww
            self.m_flow_glycol_wb = self.m_ww_g
            self.m_flow_glycol_extra = 0
            self.m_flow_extra = 0
            self.m_flow_source = self.m_ww_g
            self.T_source = T_wastewater_circuit
            self.T_return_extra_circuit = T_extra_circuit
            self.T_return_wastewater_circuit = T_out_heatpump
            self.selected_source = 'HeatCycle'
            self.on_HC == True
        #Use HeatCycle until dumped to avoid toggling between sources
        elif self.selected_source == 'HeatCycle' and m_waterbag > 30:
```

```
            self.on_HC = True
            self.m_flow_waterbag = self.m_ww
            self.m_flow_glycol_wb = self.m_ww_g
            self.m_flow_glycol_extra = 0
            self.m_flow_extra = 0
            self.m_flow_source = self.m_ww_g
            self.T_source = T_wastewater_circuit
            self.T_return_extra_circuit = T_out_heatpump
            self.T_return_wastewater_circuit = T_out_heatpump
            self.selected_source = 'HeatCycle'
        #Use AirSource if it is not below 250 K (to prevent extrapolation from compressor
            ↪ datasheet)
        elif T_extra_circuit > 250:
            self.m_flow_waterbag = 0
            self.m_flow_glycol_wb = 0
            self.m_flow_glycol_extra = self.m_airsource
            self.m_flow_source = self.m_airsource
            self.m_flow_extra = self.m_airsource
            self.T_source = T_extra_circuit
            self.T_return_extra_circuit = T_out_heatpump
            self.T_return_wastewater_circuit = T_wastewater_circuit
            self.selected_source = 'AirSource'
            self.on_HC = False
        #If none of the conditions are met, no source is chosen
        else:
            self.m_flow_waterbag = 0
            self.m_flow_glycol_wb = 0
            self.m_flow_glycol_extra = 0
            self.m_flow_source = 0
            self.selected_source = 'None'
            self.T_source = T_extra_circuit
            self.T_return_extra_circuit = T_extra_circuit
            self.T_return_wastewater_circuit = T_wastewater_circuit
            self.on_HC = False
```

C.10. Domestic hot water demand

```
import pandas as pd

class ReadDHW:
    '''DHW demand and wastewater production'''
    def __init__(self,constants):
        # Load DHW data
        self.df_DHW = pd.read_excel('databases/DHW/DHW_EU811.xlsx', sheet_name='Medium')
        self.df_DHW = self.df_DHW.fillna(0)
        self.df_DHW = self.df_DHW.append([self.df_DHW]*364, ignore_index = True)
        self.Q_demand_array = [Q * 3.6e6 for Q in self.df_DHW.loc[:,'Q␣(kWh)'] ]
        self.V_DHW_array = self.df_DHW.loc[:,'f␣(m3/s)']
        # Water properties
        self.cp = constants.water['cp'] # Specific heat capacity [J/kgK]
        self.rho = constants.water['rho'] # Density [kg/m3]
        self.T_cold = constants.water['T_cold_tap'] # Tap cold water temperature [K]
        # Initialization
        self.m_flow = 0
        self.Q_demand = 0

    def demand(self, demand, iTime, dt, T_outflow_boiler, Q_aux):
        '''

        Parameters
        ----------
        demand : bool
            True if HP is on DHW demand.
        iTime : int
            nth timestep.
        dt : int
            timestep [s].
        T_outflow_boiler : float
            Temperature of water leaving boiler.
        Q_aux : float
            Additional heat added by gas boiler (J).

        Returns
        -------
        None.

        '''
        if demand == True:
            # Find required mass flow and heat demand for timestep i.
            m_flow_i = self.V_DHW_array[iTime]*self.rho
            Q_demand_i = self.Q_demand_array[iTime]

            self.Q_demand = self.Q_demand - self.m_flow*dt*self.cp*(T_outflow_boiler-self.
                ↪ T_cold) + Q_demand_i - Q_aux

            # If there is a heat demand, mass must flow out of the vessel
            if self.Q_demand > 0:
                self.m_flow = max(self.m_flow,m_flow_i)
            else:
                self.m_flow = 0
```

```
        else:
            self.Q_demand = self.Q_demand_array[iTime]/dt
```

C.11. Wastewater production

```
import pandas as pd

class WasteWater():
    '''Provides amount of waste water added to waterbag.'''
    def __init__(self,constants):
        # Daily wastewater profile
        self.df = pd.read_csv('databases/totalusage.csv')
        self.df = self.df.fillna(0)
        self.df = self.df.append([self.df]*364, ignore_index = True)
        self.V_array = [float(V) / (60*1000) for V in self.df.loc[:, 'Flow␣rate␣(l/min)'] ]
            ↪  # Flow rate [m3/s]
        self.T_array = [float(T) + 273.15 for T in self.df.loc[:, 'Temperature␣(C)'] ] #
            ↪ Temperature [K]
        # Water properties
        self.cp = constants.water['cp'] # Specific heat capacity [J/kgK]
        self.rho = constants.water['rho'] # Density [kg/m3]
        #Initialisations
        self.m_flow = 0
        self.T = 273.15

    def supply(self,iTime):
        '''
        Function defining the mass flow and temperature of incoming wastewater at a minute
            ↪ of the simulation
        Parameters
        ----------
        iTime : nth minute of simulation.

        Returns
        -------
        None.

        '''
        self.m_flow = self.V_array[iTime] * self.rho
        self.T = self.T_array[iTime]
```

C.12. Main file

```
#Read in classes
import numpy as np
from demand.DHW import ReadDHW
from databases.read_constants import ReadConstants
constants = ReadConstants()
from source.air_source import AirSourceHeatPump
from storage.boiler import Boiler
from HEX.heat_pump import HeatPump
from HEX.spiral_heater import SpiralHeater
from databases.read_weather import ReadWeather
from control.heatpump_controller import HeatpumpController
from control.source_controller import SourceController
from demand.SH import SpaceHeatingDemand
from HEX.radiator import Radiator
from source.wastewater import WasteWater
from source.water_bag_ugne.water_bag_ugne import WaterBag
from HEX.PlateHEX import PlateHEX
from other.filter import Filter
from source.electric_heater import ElectricHeater
from source.gas_boiler import GasBoiler
#Simulation time definitions
dt= 60
t_sim = 100
start = 0
#Define objects
hp = HeatPump(constants)
s = SpiralHeater(constants, constants.spiralHEX)
r = Radiator(constants)
pHEX = PlateHEX(constants)
waste_filter = Filter()
DHW = ReadDHW(constants)
controllerHP = HeatpumpController(constants)
controllerSource = SourceController(constants)
SH_demand = SpaceHeatingDemand(constants)
DHW_demand = ReadDHW(constants)
e = ElectricHeater(constants)
gb = GasBoiler(constants)
b = Boiler(constants)
ashp = AirSourceHeatPump(constants)
weather = ReadWeather()
ww_bag = WaterBag(constants, weather.T_a, dt)
ww = WasteWater(constants)
#Required variables for main loop
check_source = []
source_change = np.zeros(t_sim)
defrost_loop = []
defrost_loop.append(False)
defrost_loop[-1] = False
t_f_loop = []
t_f_loop.append(0)
t_f_loop[-1] = 0
time_loop = []
x = np.zeros(t_sim+1)
counter_loop = np.zeros(t_sim)
```

```
mode = []
mode.append('DHW')
#Time loop
for i in range (start,start+t_sim):
    #Initialise wastewater supply
    ww.supply(i)
    #Wastewater passes through filter
    waste_filter.filter_water(ww.m_flow)
    #Choose heat source
    controllerSource.control(ashp.T_w_out_array[ashp.N_nodes-1], pHEX.T_out_glycol, hp.
        ↪ T_source_out, ww_bag.m_bag, ww_bag.T_w)
    #Wastewater bag temperature calculation
    ww_bag.TempCalc(i, dt, pHEX.T_out_water, controllerSource.m_flow_waterbag, ww.T,
        ↪ waste_filter.m_flow_out)
    #Plate heat exchanger --> heat exchange between glycol and wastewater temperature
        ↪ calculation
    pHEX.exchange_heat(dt, ww_bag.T_w, controllerSource.m_flow_waterbag, controllerSource.
        ↪ T_return_wastewater_circuit, controllerSource.m_flow_glycol_wb)
    #Air source defrosting calculation
    if (i-start) > 3 and ((check_source[-1] == 'None' and check_source[-2] == 'HeatCycle')
        ↪ or (check_source[-1] == 'HeatCycle' and check_source[-2] == 'AirSource')) :
        source_change[i-start] = i - start
    check_source.append(controllerSource.selected_source)
    ashp.defrost(weather.T_a[i], hp.T_evap, defrost_loop)
    #Getting array of whether defrosting is True or False
    defrost_loop.append(ashp.defrosty_real)
    #Timestep at which defrosting stops -> used to recalculate new start time
    ashp.time_pass(i-start, defrost_loop, t_f_loop[i-start])
    time_loop.append(ashp.time)
    t_f_loop.append(ashp.t_f)
    #Loop that defines two counters at timestep when defrosting ends and frosting begins:
    #x - counter that defines timestep from start of simulation
    #counter_loop - counter that defines the time that corresponds to the current frost
        ↪ thickness based off of base case (Ta = 0 C and RH 85%), used to find the correct
        ↪  timestep to start frosting rate once defrosting ends
    if (i-start) > 0:
        #if defrosting has turned off
        if defrost_loop[i-start+1] == False and defrost_loop[i-start] == True:
            x[i-start:t_sim + 1] = time_loop.index(ashp.counter, len(time_loop)-2, len(
                ↪ time_loop))
            counter_loop[i-start:t_sim+1]= ashp.counter
        time = i - start - x[i-start] + counter_loop[i-start] - source_change[i-start]
        if time < 0:
            time = 0
    else:
        time = 0
    if defrost_loop[i-start+1] == True:
        time = ashp.time
    #Air source temperature calculation
    ashp.Q_in(controllerSource.T_return_extra_circuit, weather.T_a[i], weather.RH[i], time,
        ↪  controllerSource.m_flow_extra, hp.T_evap)
    #Calculation of supplied space heating
    SH_demand.demand('Boiler', i, None, dt, r.T_supply_in, r.Q_left)
    #Heat pump control
    controllerHP.control( hp.T_e, b.T_control, controllerSource.T_source, SH_demand.Q_req,
        ↪ mode[-1], ashp.defrosty, controllerSource.selected_source)
```

```
    mode.append(controllerHP.mode)
    #Calculation of heat pump parameters
    hp.heat(controllerHP.mode, controllerHP.f, controllerHP.m_flow_supply, s.T_out,
        ↪ SH_demand.T_supply_in, controllerSource.m_flow_source, controllerSource.T_source
        ↪ , e.P)
    #Radiator heat release
    r.SH_supply(hp.T_supply_out_SH, SH_demand.T_supply_in, SH_demand.Q_req, controllerHP.
        ↪ mode)
    # Spiral heater temperature calculation
    Q_to_boiler = s.heat(controllerHP.m_flow_supply, hp.T_supply_out_DHW , b.T, dt)
    #Boiler temperature calculation
    b.UpdateTemperature(dt, Q_to_boiler, DHW.m_flow)
    # Domestic hot water usage
    DHW.demand(True, i, dt, b.T_outflow, gb.Q_req)
    #Gas boiler additional heat requirement
    gb.heat_req(DHW.m_flow, b.T[b.N_layers-1])
```