# Oil Temperature Control Unit

by

Ediz Ferit Kula, Serdal en


Department of Mechanical Engineering
Bogazici University
Turkey

January 2023

**Abstract**

This report aims to introduce a dynamic model with Matlab/Simulink that simulates the oil cycle part of an ORC setup and proposes a solution for the oil's oscillating temperature. The heater of this cycle regularly overshoots and undershoots the set temperature of the oil, and this results in an oscillation of the temperature of the primary working fluid, which is not desired. A dynamic model is created by applying thermodynamic and heat transfer principles to the components of the oil cycle. The unknown variables are either calculated by the experimental data or found by using the experimental data as training data to the model. Afterward, the model is, compared with the experimental data to demonstrate that the model is valid; as it turned out, the model data and experimental data are coherent. Next, a proposed solution is created including a bypass pipeline between the inlet and exit and producing a reverse flow. The simulation of the proposed solution is done with the validated model. The simulation demonstrated that the oscillations could be reduced with the bypass pipeline based on how much of the flow goes through the bypass pipeline. Furthermore, a concept experimental setup is created by using a pump, a check valve, a 3-way valve, and an actuator. The real experiment, however, only consisted of a 3-way valve and an actuator, demonstrating that the 3-way valve can be controlled with the actuator to change the fluid flow direction.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

There is an organic rankine cycle (ORC) experiment setup located in the BURET (Bogazici University Renewable Energy Technologies) in Kilyos, Istanbul (See Figure 1.1). The ORC setup's primary objective is to extract energy from low-temperature heat sources such as solar, waste, or biomass by using fluids with low boiling temperatures. The setup consists of two separate fluid cycles: the main working fluid cycle and the oil cycle. The oil cycle extracts energy from a heater and transfers the energy to the main working fluid. Meanwhile, hot and pressurized main working fluid spends its energy rotating a shaft that generates electricity.

Figure 1.1: ORC Setup in BURET Lab

The main working fluid cycle had already been investigated by Altun et al. Altun developed a dynamic simulation model that predicts the temperatures and pressures across the cycle, including the pressure and heat losses, by using Modelica and validated his model with the experiment results of the ORC setup. In this project, we focus on the other side of the ORC, the oil cycle, and we propose a way to improve the cycle.

The oil cycle (See Figure 1.2) consists of 4 main components: the heater, the pump, the evaporator, and the oil tank. Basically, the oil is heated at the heater, which has a resistor and an on-off temperature control unit. The hot oil is pumped toward the evaporator by a gear pump, and the oil transfers its energy to the cold main working fluid inside the evaporator, which is a plate-type heat exchanger. The heater can be controlled by setting a temperature. When the thermocouple at the exit of the heater determines a higher temperature than the set temperature, the heater is closed. When the temperature at the exit of the heater is determined to be less than the set temperature, the heater is turned on.

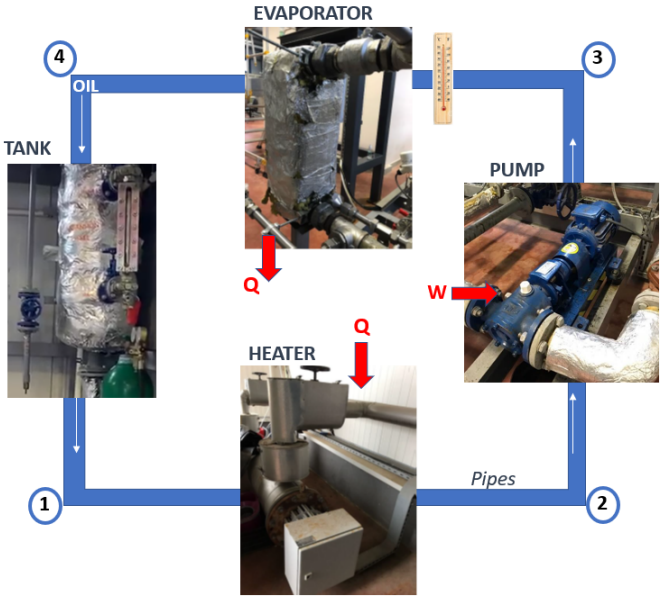



Figure 1.2: The Oil Cycle

The problem that is aimed to be fixed in this project occurs because of the oil heater's on-off type temperature controller. The thermocouple used to determine the exit temperature at the exit has a hysteresis of $\pm 3°C$. That is why the oil is either heated or left to be cooled much more than the desired set temperature. Additionally, the resistor inside the heater, which has a heating capacity of 100 kW, remains hot even after the heater is closed, resulting in an additional temperature rise after the set temperature is reached. Both of these reasons result in the oscillation of the oil temperature at the heater exit (See Figure 1.3). The oscillation temperature of the oil causes the temperature of the main working fluid to oscillate.

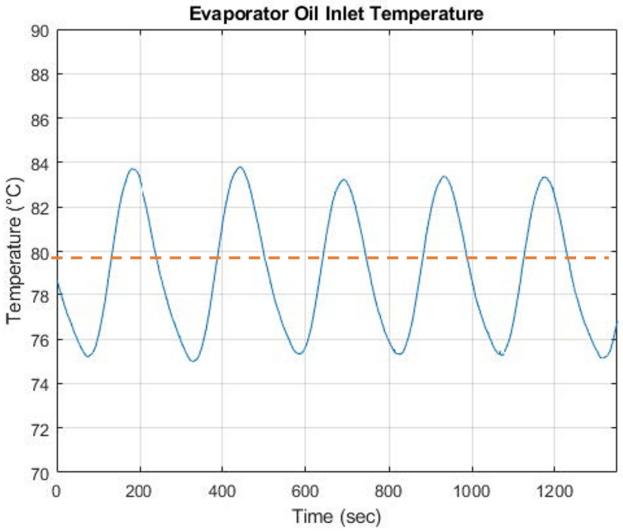



Figure 1.3: The Oscillation of Oil's temperature when set Temperature is $80°C$

This project aims to develop a simulation model that predicts temperature across the oil cycle, propose a mechanical solution to decrease the oil oscillation, and modeling of the proposed solution. According to the literature, there is a couple of software that can be used to model a thermodynamic cycle, like

ORC setups such as GtSuite, Modelica, and Matlab/Simulink.

An example study on the dynamic modeling of ORC and optimization of some components, such as turbines and heat exchangers, was conducted by Marchionni et al. [1]. A 1D computer-aided modeling software called GTSuite was employed in his model. Rather than MW scale ORC systems, this research concentrates on kW scale systems. The goal of this study is to model ORC systems with power output ranging from 34.5 kW to 55.5 kW. To observe the cycle's transient reaction, various mass flow rates and intake temperatures of cooling water and hot oil are investigated.

Another software that can be employed to make a dynamic model for ORC is Modelica which is used by Wei et al. [2]. The cycle's heat source is a power plant's exhaust gas, and the air is used for cooling. They examined the accuracy, complexity, and simulation durations of moving boundary (MB) and finite volume (FV) methods. Results of the transient analysis are presented for both scenarios and contrasted with experimental data. It is discovered that the MB strategy is quicker and simpler than the FV approach as a result. The FV method, however, is more precise than the MB one. As a result, MB is more appropriate than the FV technique if the simulation time is the primary concern, whereas the FV approach would make more sense if it is not. Additional to Wei et al.'s study, Modelica/Dymola was used by Zhang et al. [3]. to develop a dynamic model for kW scale ORC. Their steps were to model the components of ORC first and then compare the model's simulation results with the experiment results. As it turned out, the model's result was compatible with the results of the experiments. By validating the model, they also validated the thermal efficiency that they calculated by this model, which was 6.94%. Other users of Modelica/Dynamo for ORC modeling were Quolin et al. [4]. They also compared the model results with the experiment results, and they discovered that the model was accurate. Ertugrul's research can be considered the predecessor of this project [5]. In his thesis, which is about dynamic modeling of the ORC setup (except the oil cycle section), he made a transient simulation of ORC and found out the temperature of the R134a across the ORC cycle by using Modelica/Dymola. He also considered simulating pressure losses across ORC and validated the results with the readings from the pressure sensor in ORC. Unlike his study, we neglected the pressure effects on the system since their effects are small, and it would be costly to implement pressure sensors in the oil cycle for validation.

Matlab/Simulink is another applicable software for the dynamic modeling of ORC. Carraro et al. [6]. made a study targeting to model and design of a small ORC setup with the help of Matlab. They investigated the ORC system's response to the oscillating temperature of the oil between 410 K and 435 K. When they compared their dynamic model's results with the experiment's results; they calculated the maximum relative error of 6.6%.

A similar problem is solved by Akmal et al. [7]. They used Matlab/Simulink to dynamically model an underfloor heating system that has an on-off type resistor. Their aim was to create a valid model that could optimize the heating of a room with underfloor heating. Similar to the heater in the oil cycle, the heater of the underfloor heating system overshoots or undershoots the set temperature because of the delayed response of the resistor.

As can be seen, there are many ways to create a dynamic model of a part of the ORC setup. Modelica/Dymola, GT-Suite, and Matlab/Simulink are generally used for this purpose, and they give compatible results with the results of the experiment. In this project, Matlab/Simulink is chosen to model the oil cycle. This is mainly because of time scarcity since Matlab/Simulink is a more familiar software for us, and it would take time to learn new software. Additionally, Simulink's easy-to-use and practical interface is beneficial for us.

# Chapter 2

# Theory

In order to provide a transient simulation model of this oil cycle, some governing equations must be established for each component in the cycle. Additionally, in order to appropriately simulate the cooling of the system when the heater is closed, the oil in the pipes must also be included in the model. Each component in the cycle (pump, tank, heater, and evaporator) can be considered as control volume, and this will allow us to write transient energy and mass balance equations inside the components. First, let's start with the mass balance equations[8]:

$$\frac{dm_c v}{dt} = \dot{m}_i - \dot{m}_e \tag{2.1}$$

where $\dot{m}$ is the overall mass change rate inside the control volume, and $\dot{m}_i$, and $\dot{m}_e$ represent the mass transfers at the inlet and exit of the control volume, respectively.

By using Equation 2.2, the mass balance equation can be modified as volumetric flow rate balance. This formulation is more useful than the mass balance equation because the volumetric flow rate is constant across the cycle.

$$\dot{m} = \dot{V}\rho \tag{2.2}$$

$$\frac{d(V_{cv}\rho)}{dt} = \dot{V}_i\dot{\rho}_i - \dot{V}_e\dot{\rho}_e \tag{2.3}$$

where $\dot{V}$ is the volumetric flow rate, and $\dot{\rho}$ is the density.

Furthermore, the transient energy balance equation can be written in the control volumes (See Equation 2.4).

$$\frac{dE_{cv}}{dt} = \dot{Q} - \dot{W}_{cv} + \dot{m}_i(h_i + \frac{v_i^2}{2} + gz_i) - \dot{m}_e(h_e + \frac{v_e^2}{2} + gz_e) \tag{2.4}$$

where $E_{cv}$ is the energy inside the control volume, and with the derivative, the left side of the equation corresponds to the energy change in the control volume. The $\dot{Q}$ is the heat transferred to the control volume, $\dot{W}_{cv}$ is the work done by the control volume, $\dot{m}$ is the mass flow rate, the $h$ is the enthalpy, $v$ is the velocity of the fluid, $gz$ is the product of the gravitational acceleration and the relative height,

and the subscripts $i$ and $e$ indicate the fluid at the inlet and exit, respectively. Together $\dot{m}(h + \frac{v^2}{2} + gz)$ means energy that enters or exits the control volume with the fluid flow at the inlet or exit. The kinetic and potential effects in this equation are negligibly small, so the equation can be reduced to Equation 2.5.

$$\frac{dE_{cv}}{dt} = \dot{Q} - \dot{W}_{cv} + \dot{m}_i h_i - \dot{m}_e h_e \tag{2.5}$$

Another important governing equation is the heat transfer equation, as can be seen in Equation 2.6. This equation can be used to calculate the temperature change when heat is stored inside the material. That is why $E_{st}$ indicating energy stored in the material is used instead of $Q$.

$$E_{st} = mc(T_2 - T_1) \tag{2.6}$$

where $m$ is the mass of the material, $c$ is the specific heat of the material, and $T_1$ and $T_2$ are the initial and final temperatures of the material, respectively. This equation can be used to calculate the temperature change of a material by knowing the energy stored in the material. However, this equation is not transient. That is why this equation must be differentiated with respect to time. Note that $T_2$ transformed to $T$ because the final temperature is unknown and desired to be calculated.

$$\dot{E_{st}} = \frac{d[mc(T_2 - T_1)]}{dt} \tag{2.7}$$

$$\dot{E_{st}} = \dot{m}c(T - T_1) + mc\frac{dT}{dt} \tag{2.8}$$

The heat transfer equation and energy balance equations are related in a way. The energy balance inside a control volume determines the energy change in the control volume, which is, in our case, stored inside the oil. This relation can be shown by Equation 2.9. Afterward, the equation can be transformed to:

$$\dot{E_{st}} = \dot{E_{cv}} \tag{2.9}$$

$$\dot{m}c(T - T_1) + mc\frac{dT}{dt} = \dot{Q} - \dot{W}_{cv} + \dot{m}_i h_i - \dot{m}_e h_e \tag{2.10}$$

Here the mass flow rate is assumed as constant for the sake of simplicity. By using Equation 2.1, we can rewrite the $\dot{m}$ and solve for $\frac{dT}{dt}$.

$$mc\frac{dT}{dt} = \dot{Q} - \dot{W}_{cv} + \dot{m}(h_i - h_e) - \dot{m}c(T - T_1) \tag{2.11}$$

To further improve the question, we need to introduce a specific heat-enthalpy relationship into consideration. The specific heat changes with temperature, but we can assume it is a constant since the specific heat does not change much within our temperature range. Therefore, the specific heat-enthalpy relation can be written as:

$$c_p(T_2 - T_1) = h_2 - h_1 \tag{2.12}$$

Implementing above equation into Equation 2.11 results in:

$$mc\frac{dT}{dt} = \dot{Q} - \dot{W}_{cv} + \dot{m}c(T_1 - T) - \dot{m}c(T - T_1) \tag{2.13}$$

The above equation can be simplified further. Note that there is no work done in any of the oil cycle's components except in the pump, which is neglected because it does a negligible effect on the oil's temperature.

$$\frac{dT}{dt} = \frac{\dot{Q}}{mc} + 2\frac{\dot{m}}{m}(T_1 - T) \tag{2.14}$$

Overall, the mass balance equation, control volume energy balance equation, and stored energy equations can be used as governing equations to create a model that can determine temperature change based on initial conditions.

# Chapter 3

# Methodology

In order to test the proposed solution, a digital twin of the oil cycle is created with a process composed of three parts. Matlab/Simulink software is selected to model the system. For modeling, mathematical equations[9] and experimental results are used. The experiment was done in 2022 by Ertugrul Altun for his thesis given in [5].

First, the governing equations, which are derived in the previous chapter, are used to create a Simulink model. All variables are determined, and missing values are calculated from experimental data by curve fitting. Secondly, free variables are fitted to the experimental data with a manual training process. Finally, the model is validated by experimental data that is reserved only for validation.

Free variables:

- $UA$
- $m_r c_r$
- $Q_{loss}$

## 3.1 Modeling

The system is composed of four main components:

1. Heater
2. Pump
3. Evaporator
4. Tank

Each component has a governing differential equation that needs to be solved to give the current temperature. Therefore three differential equations are defined in three subsystems with Simulink blocks for

each component. The differential equation for the pump is not defined since the pump has no effect on the oil properties. It only enables the continuity of the flow.

The volumetric flow rate is taken as constant through the entire system and for all temperatures because it barely changes through desired temperature change and for each component if the outliers are neglected. This can be seen in Figure 3.1.

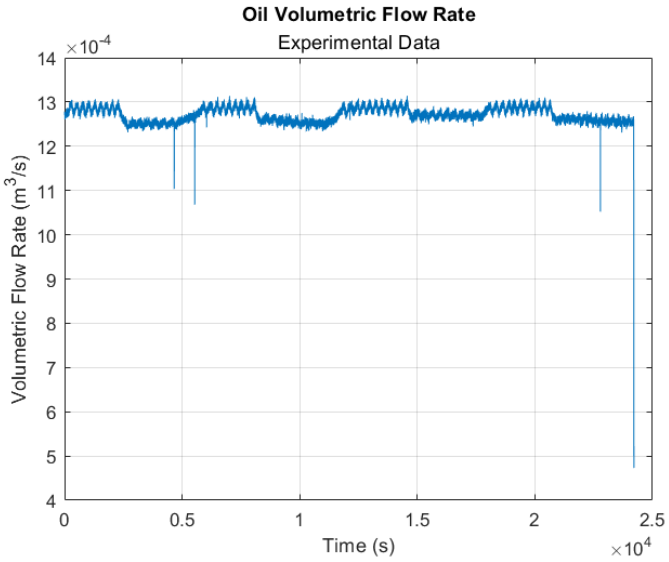



Figure 3.1: Oil's volumetric flow rate during the experiment.

Volumetric flow rate is equal to $v = 0.001278 m^3/s$.

Below assumptions are made to solve the system:

- For all components, the temperature inside the component is assumed to be equal to the temperature at the exit of that component.
- Total heat loss in the system is assumed to be lost at the tank.

### 3.1.1 Heater

The heater is the place where the oil temperature is controlled. There is a thermostat that provides an interface for users to set the desired temperature. Then, the heater heats the oil up to the desired temperature. It uses an on-off temperature control unit supplying electrical power to a resistor. This resistor then transfers heat to the flowing oil.

Because of two main reasons, oscillation occurs in the temperature of the oil. One is that this system is highly dependent on the measured oil temperature in the heater, and there is a hysteresis of 2.6 °C in the thermocouple that measures the oil's temperature. This creates a lag between the real temperature and the readings. The second reason is the thermal energy storage of resistors. Because the temperature control unit only can be fully open or closed, resistors are always at different temperatures than the oil. Thus, when the heater is turned off, resistors are still hot enough to increase the oil's temperature further above the desired level. Heater parameters are provided in Table 3.1.

Table 3.1: Heater parameters.

| Symbol | Property | Unit |
|---|---|---|
| $T_2$ | Oil temperature in heater | Kelvin |
| $m_2$ | Oil mass in heater | kg |
| $c_2$ | Oil's specific heat in heater | kJ/kg*K |
| $Q$ | Heat transferred to oil from resistor | kJ |
| $W$ | Electrical work input | kW |
| $\dot{m}_1$ | Oil mass flow rate at heater enterence | kg/s |
| $\dot{m}_2$ | Oil mass flow rate at heater exit | kg/s |
| $h_1$ | Oil enthalpy at heater enterence | kJ/kg |
| $h_2$ | Oil enthalpy at heater exit | kJ/kg |
| $U$ | Resistor overall heat transfer coefficient | W/m$^2$K |
| $A$ | Resistor surface area | m$^2$ |
| $T_r$ | Resistor temperature | Kelvin |
| $m_r$ | Resistor mass | kg |
| $c_r$ | Resistor specific heat | kJ/kg*K |

Equation 3.1 represents the governing differential equation for the heater. Every parameter in that equation needs to be found to achieve a solution.

$$\frac{dT_2}{dt} = \frac{1}{m_2 c_2}(Q + \dot{m}_1 h_1 - \dot{m}_2 h_2) \tag{3.1}$$

For representation purposes, the heater model created in Simulink is provided in Figure 3.2. The controller decides if the power will be provided to the resistor. Resistor's temperature is determined by its governing differential equation, which is the combination of Equation 3.4 and 3.5. Further detail can be investigated in the appendix section (See Appendix A).

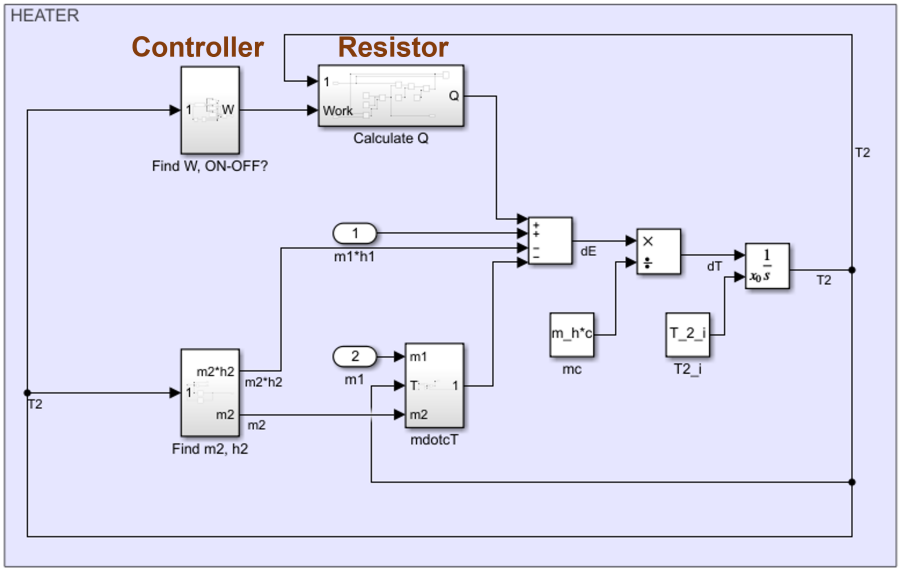



Figure 3.2: Heater Simulink model.

In order to calculate the mass flow rate and mass ($\dot{m}_2$ and $m_2$) inside the heater, volume, and density are required according to Equations 3.3 and 3.2.

$$\dot{m} = \rho * \upsilon \tag{3.2}$$

$$m = \rho * V \tag{3.3}$$

The approximate value of the heater's volume is calculated by on-site measurements and it is equal to $v_h = 0.29m^2$. The density of the oil changes according to the instant temperature. To account for this change, a lookup table is entered into the Simulink model, which determines the density of the oil and makes linear interpolation if necessary. Specific heat ($c_2$) and enthalpy values ($h_1$ and $h_2$) are also determined by a lookup table since it also changes with temperature. These tables are provided in Figure 3.3.

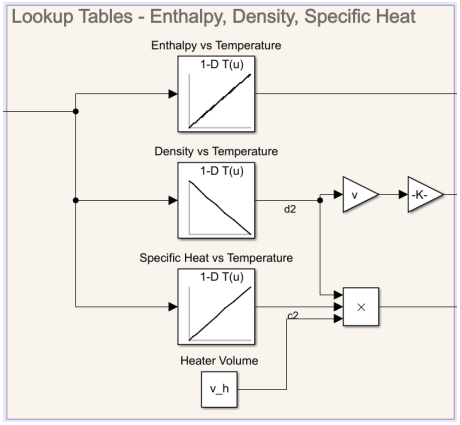



Figure 3.3: Lookup tables Simulink model.

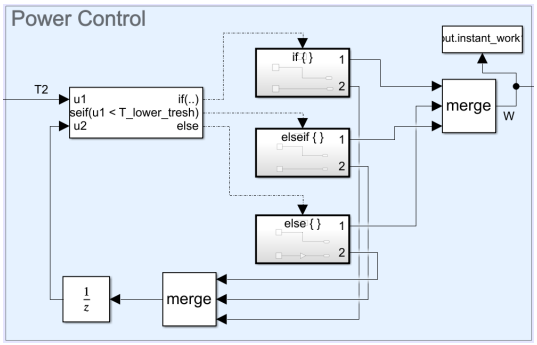



Figure 3.4: Power control Simulink model.

Heat transferred to the oil ($Q$) is determined by carrying out the heat transfer calculations between the resistor and oil. 100 kW of electrical power is given to the resistor causing it to dissipate heat. Heat transfer[10] to the oil from the resistor is calculated by Equation 3.4.

$$Q = UA(T_r - T_2) \tag{3.4}$$

$$\frac{dT_r}{dt} = \frac{1}{m_r c_r}(W - Q) \tag{3.5}$$

Resistor's temperature ($T_r$) is obtained by solving the differential equation given by Equation 3.5. In this equation, the value of the power changes from 0 to 100 kW according to the state of the controller. When the temperature of the oil at the heater exit exceeds an upper threshold, the power is off; when it falls under a lower threshold, the power is on. This behavior is enabled by using if-action blocks of Simulink, as seen in Figure 3.4.

Previously mentioned thresholds are determined in a Matlab script and fed to the Simulink model according to the set temperature ($T_{set}$). Not only the thresholds but also the initial values for integral blocks are specified accordingly. Set temperature can be chosen by the user in the same Matlab script with the interface shown in Figure 3.5.

In Figure 3.6, the heat transfer from the resistor to oil is shown in a simplified manner. It is entered into Simulink as shown in Figure 3.7.

Figure 3.8 shows the temperature variations of the resistor, oil, and on-off status of the controller as input work.

User Interface:

```
T_set =  80  + 273;    % set temperature (K)
Rate =  25   ;         % bypass rate percentage
```

Figure 3.5: Matlab user interface.

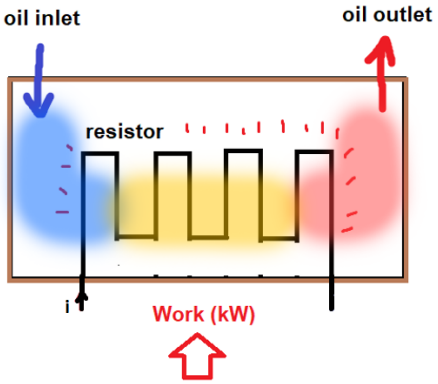



Figure 3.6: Resistor schematic.

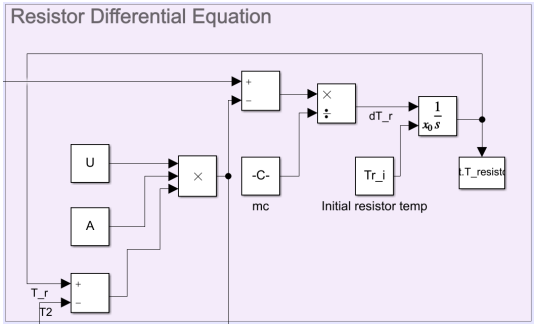



Figure 3.7: Resistor Simulink model.

Also, in Figure 3.9, the energy flow in this system can be seen clearly. Electrical energy is transformed into internal energy in the resistor. Then the oil's temperature is increased by the resistor, and finally, in the evaporator, R134A is heated up.

All parameters are found to solve for the temperature at the heater exit except for the resistor's heat transfer coefficient, surface area, mass, and specific heat. These four parameters are treated as two free variables as $UA$ and $m_r c_r$. These two variables are optimized to give minimum root-mean-square error in Section 3.2.

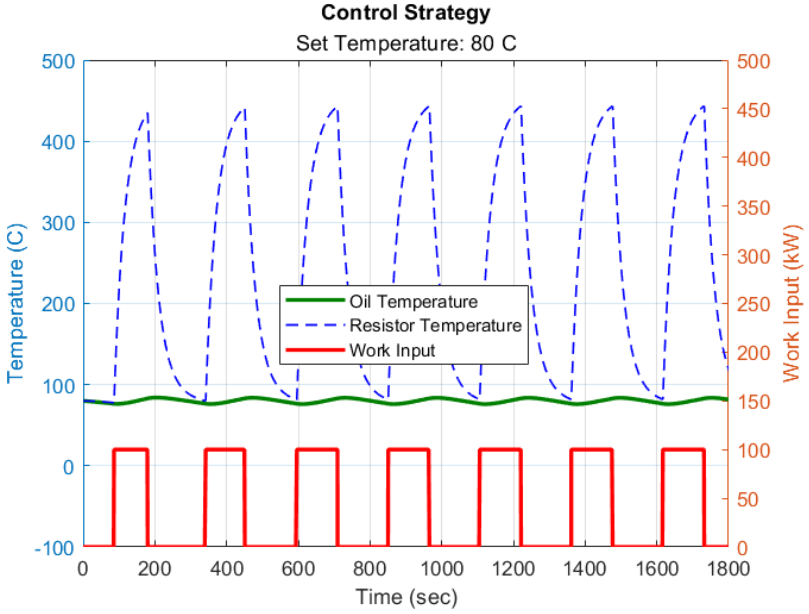



Figure 3.8: Temperature variations with control strategy.

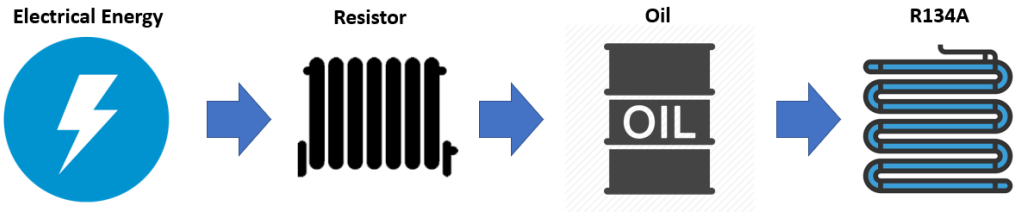



Figure 3.9: Energy flow through the system.

### 3.1.2 Evaporator

The evaporator model is very similar to the heater model in many respects. Parameters for the evaporator are provided in Table 3.2.

Table 3.2: Evaporator parameters.

| Symbol | Property | Unit |
|---|---|---|
| $T_4$ | Oil temperature in evaporator | Kelvin |
| $m_4$ | Oil mass in evaporator | kg |
| $c_4$ | Oil's specific heat in evaporator | kJ/kg*K |
| $Q_e$ | Heat transferred to R134A from oil | kJ |
| $\dot{m}_3$ | Oil mass flow rate at evaporator entrance | kg/s |
| $\dot{m}_4$ | Oil mass flow rate at evaporator exit | kg/s |
| $h_3$ | Oil enthalpy at evaporator entrance | kJ/kg |
| $h_4$ | Oil enthalpy at evaporator exit | kJ/kg |

The governing differential equation for the evaporator, which is solved to obtain the temperature at the exit of the evaporator, is given in Equation 3.6.

$$\frac{dT_4}{dt} = \frac{1}{m_4 c_4}(Q_e + \dot{m}_3 h_3 - \dot{m}_4 h_4) \tag{3.6}$$

In this equation, mass, specific heat, mass flow rates, and enthalpy values are found with the same method provided for the heater using lookup tables. The mass here represents the oil mass inside the evaporator. This mass can be calculated by calculating the volume of the oil inside the evaporator. According to Ertugrul's thesis [5], the plates of the evaporator have sinusoidal shapes having a perimeter of 7,64 mm, the area of a sinus wave is $A = 4mm^2$ per wave, and the heat transfer area is $7.4m^2$. This means that the total height of the sinusoidal-shaped heat transfer areas must be $h = \frac{7.4m^2}{7{,}64mm}$, and the oil volume inside the evaporator can be calculated as $V = hA = \frac{7.4m^2}{7{,}64mm} 4mm^2 = 0.387x10^{-3}m^3$.

Heat transfer between R134A and oil ($Q_e$) is found using experimental values. The temperature of the oil at the inlet and outlet of the evaporator is known. Combining this knowledge with the mass flow rate of the oil, Equation 3.7 is obtained, which gives the experimental heat transfer amounts for each temperature.

$$Q_e = \dot{m}c(T_4 - T_3) \tag{3.7}$$

Heat transfer value ($Q_e$) needs to be updated every second according to the current temperature and for all possible temperature ranges. To do that, experimental values are plotted against temperature,

and four different curves are fitted to the plot for four regions. Then, the coefficients of these curves' equations are interpolated for unknown temperature values. This process is explained in Figure 3.10.

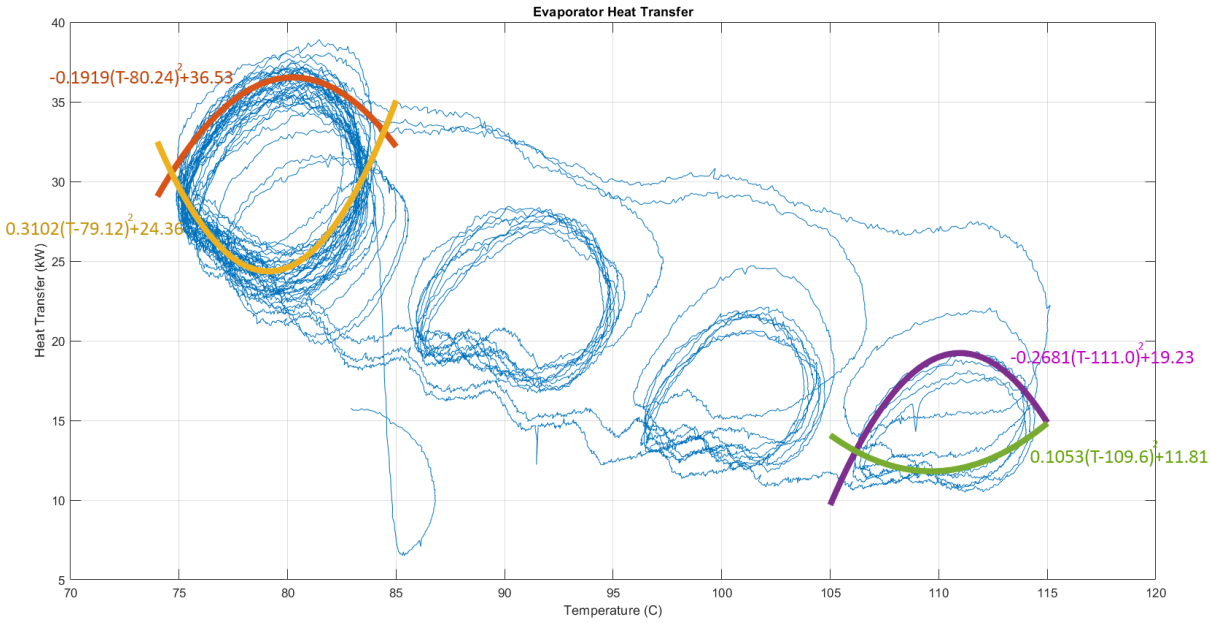



Figure 3.10: Evaporator heat transfer curve fitting to experimental data.

Second-degree polynomials are fitted either in concave or convex format to represent the circular behavior of the data. General format of the polynomial is $Q_e(T) = a(T+b)^2+c$. Obtained coefficients are provided in Table 3.3.

| Type | a | b | c |
|---|---|---|---|
| Concave, 80 $°C$ | -0.1919 | -80.24 | 36.53 |
| Concave, 110 $°C$ | -0.2681 | -111.0 | 19.23 |
| Convex, 80 $°C$ | 0.3102 | -79.12 | 24.36 |
| Convex, 110 $°C$ | 0.1053 | -109.6 | 11.81 |

Table 3.3: Coefficients of fitted polynomials.

### 3.1.3 Tank

The tank acts as a pressure stabilizer. The total oil volume in the system changes according to the temperature because temperature changes density. This change in volume is compensated in the tank. Parameters for the tank are provided in Table 3.4.

Tank is also governed by a similar differential equation provided in Equation 3.8.

$$\frac{dT_1}{dt} = \frac{1}{m_1 c_1}(Q_{loss} + \dot{m}_4 h_4 - \dot{m}_1 h_1) \tag{3.8}$$

Heat loss ($Q_e$) represents the total heat loss in the system. Heat loss normally occurs in every part, such as pipes, pump, heater, and evaporator, but it is introduced in the tank for ease of solution. It is also another free variable since it is unknown.

Table 3.4: Tank parameters.

| Symbol | Property | Unit |
|---|---|---|
| $T_1$ | Oil temperature in tank | Kelvin |
| $m_1$ | Oil mass in tank | kg |
| $c_1$ | Oil's specific heat in tank | kJ/kg*K |
| $Q_{loss}$ | Total heat loss in the system | kJ |
| $\dot{m}_4$ | Oil mass flow rate at tank entrance | kg/s |
| $\dot{m}_1$ | Oil mass flow rate at tank exit | kg/s |
| $h_4$ | Oil enthalpy at tank entrance | kJ/kg |
| $h_1$ | Oil enthalpy at tank exit | kJ/kg |

## 3.2 Training

There are three free variables:

- Resistor's overall heat transfer coefficient and surface area: $UA$
- Resistor's mass and specific heat: $m_r c_r$
- Total heat loss from the system: $Q_{loss}$

Before training the model, experimental data is divided into two parts: one for training and the other for validation. This is because, for the reliability of validation, training data needs to be different from the validation data. This separation is shown in Figure 3.11.

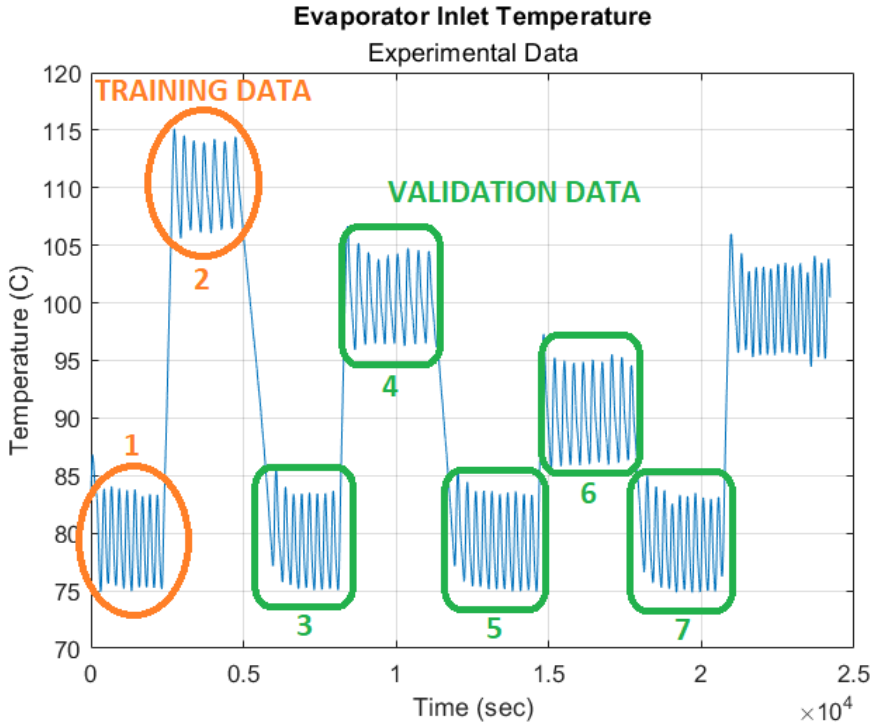



Figure 3.11: Experimental oil temperature data at evaporator inlet.

As seen in Figure 3.11, in the first section of the experiment temperature is set to 80 °C. Then, it is set to 110 °C, which is the maximum temperature in the experiment. Oscillation frequency and amplitude

change when set temperature is changed. Two far ends are selected as training data to get the nature of these oscillation frequency and amplitude changes.

Many tests are carried out to determine the variables. Comparison plots are created, and RMSE values are calculated for each test. Plots are investigated to prevent under-fitting and over-fitting behaviors because if only the RMSE value is looked at, wrong results may be obtained.

Variables to be determined can be divided into two categories. $UA$ and $m_r c_r$ are almost constant for all temperature values, and they are assumed as constant. Besides, $Q_{loss}$ significantly changes with the varying temperature, and it accounts for the previously mentioned oscillation frequency and amplitude change with the temperature. That's why at first, two variables $UA$ and $m_r c_r$ are determined from the test results. Then, the last variable $Q_{loss}$ fits the data both at higher temperatures and lower temperatures.

Final values of the variables:

- $UA = 0.27 \ W/K$
- $m_r c_r = 8 \ W/K$
- $Q_{loss} = -7.4074 * 10^{-06} * (T_{set} - 110)^4 - 8 \ W$

The final version of the plots that are used in the optimization of the variables is provided in Figures 3.12 and 3.13.

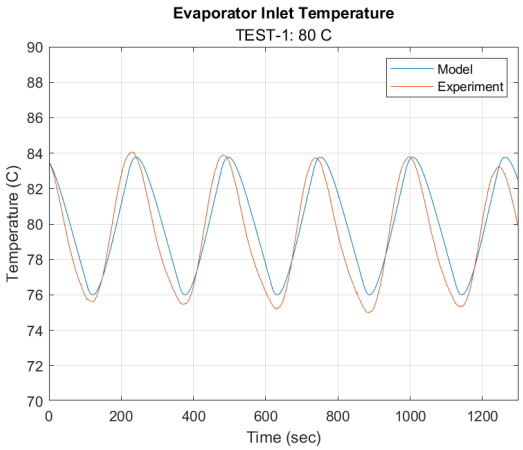



Figure 3.12: Test-1.

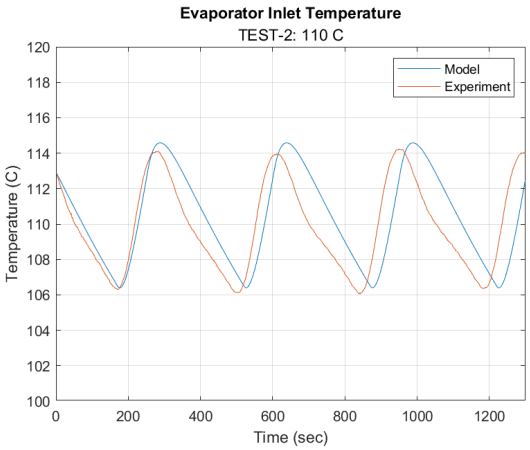



Figure 3.13: Test-2.

## 3.3 Proposed Solution

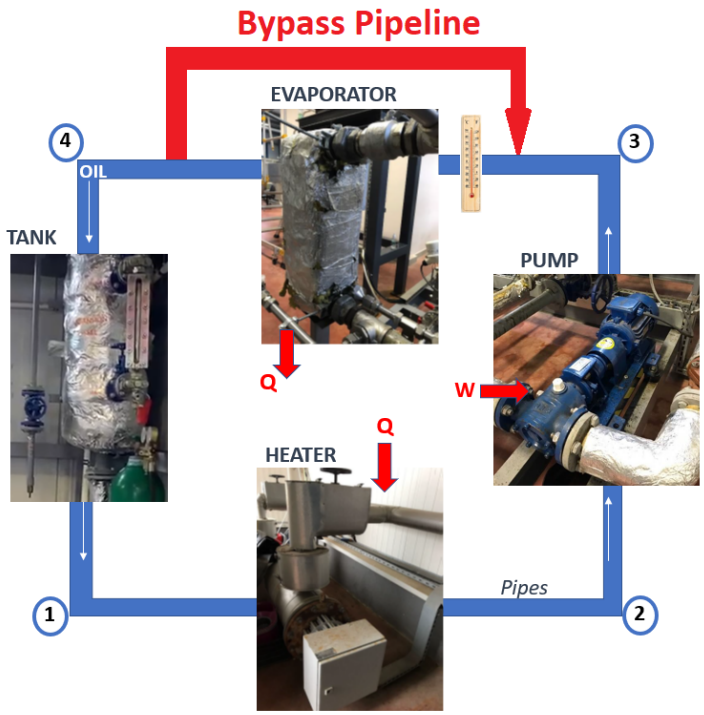



Figure 3.14: Oil cycle with suggested method.

The main goal of this project is to reduce the oil's temperature oscillations in the evaporator by applying mechanical solutions. A lower oscillation portion of the oil can be mixed with the mainstream oil just before the evaporator inlet. In Figure 3.14, our suggested method is implemented on the oil cycle.

According to the investigations of the experimental data, it is found that the oil's temperature oscillations are lower at the evaporator exit. Also, there already exists a bypass line between the inlet and exit of the evaporator. Thus, the proposed solution is feeding back the exiting oil to the inlet of the evaporator via this bypass line. This method is implemented in Simulink as seen in Figure 3.15.

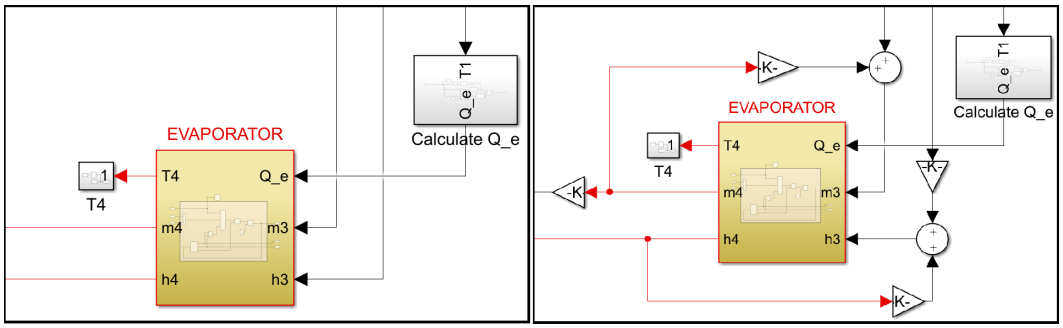



Figure 3.15: Evaporator model before and after the proposed solution is implemented.

# Chapter 4

# Experimental Setup

There are some crucial components needed in order to carry out the experiment of the proposed solution. Note that in the oil cycle part of ORC, there is an already installed bypass line. This bypass line can be used to reduce the cost of implementing the proposed solution.

1. First of all, a pump is needed to establish a flow in the reverse direction of the cycle.
2. Second, a check valve is needed to ensure fluid flow only in one direction in the bypass line.
3. Third, a 3-way valve and an actuator must be employed at the entrance of the bypass line to control the mass flow rate that will flow inside the bypass pipeline
4. Lastly, a power source for the actuator is needed.

## 4.1 Concept Setup of the Proposed Solution

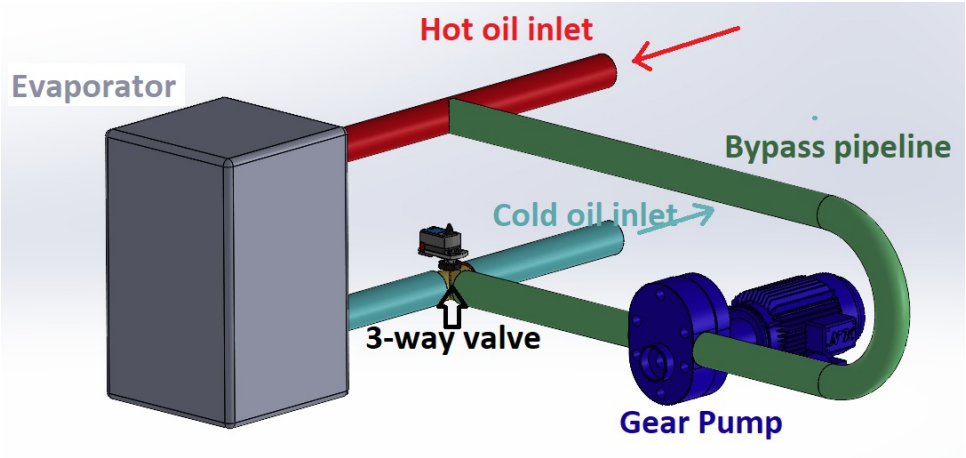



Figure 4.1: Concept Setup of the Proposed Solution

The bypass pipeline connects the exit of the evaporator to the inlet of the evaporator. Since there is a height difference between the inlet and exit of the evaporator and there is a pressure loss inside the

evaporator, it is not possible to constitute a flow from exit to inlet without a pump. Therefore, an appropriate pump is needed to be installed in order to employ a reverse flow. There are some crucial criteria for the pump. The pump must endure temperatures as high as 140°. Additionally, the pump must not corrode by oil. Generally, in high-temperature oil applications, gear pumps that are capable of enduring high-temperature oil are used. The head a gear pump provides is not much important since there is a small height difference between the inlet and the exit, and the pressure loss inside the evaporator is negligibly low, according to Ertugrul's thesis. A gear pump is approximately 5500 TLs.


Figure 4.2: A Gear Pump

Moreover, a proper check valve must be installed as well to ensure a one-direction-only flow. The same criteria are valid for the check valve: the valve must endure high-temperature oil. For this use, a disc type of check valve made up of stainless steel can be installed since it can endure high temperatures and will not corrode because of its material. This type of check valve is around 500 TLs.

Figure 4.3: A Disc Check Valve

Additionally, a 3-way valve will be used to establish an adjustable fluid flow direction. There are a couple of options for the type of 3-way valve, but since there is an available 3-way ball valve in the BURET lab, there is not much reason to purchase a new valve. The available 3-way ball valve is Siemens VB61.50-40 DN50 which is a brass valve with a stainless steel ball. The ball can be rotated by rotating the rectangular-shaped pin above the ball.

Figure 4.4: 3-way ball valve

Additionally, an actuator is needed to control the fluid flow direction by rotating the ball inside the valve. There is also an available actuator in the BURET lab, which is Siemens GLB161.9E. The actuator is coherent with the 3-way ball valve, so it can rotate the pin on the valve. Furthermore, there must be a 24V power supply for the actuator to rotate.

Figure 4.5: Actuator - Siemens GLB161.9E

In Table 4.1, the components required for applying our solution method are listed with the corresponding costs.

| **Components** | **Cost** |
|---|---|
| Three-way valve | 4400 TL |
| Gear pump | 5500 TL |
| Disc check valve | 500 TL |
| Rotary actuator | 2300 TL |
| *Total cost* | 12700 TL |

Table 4.1: Components and costs.

Since the bypass pipeline, 3-way ball valve, and actuator are already available, the initial cost of setting this experiment up is around 6000 TLs. Due to the cost limitations, however, a more comprehensive experiment that consists of a pump and a check valve is not done. Instead, our experimental setup will cover controlling an actuator connected to a 3-way ball valve which will control the mass flow rate in the bypass pipeline.

## 4.2 Setup of the Experiment

The setup of the experiment consists of three main parts, which are a 3-way ball valve, an actuator, and a power supply. There are also some cables, a resistance, and a potentiometer used. The purpose of the experiment was to establish a setup that can control the actuator's angle, which represents how much the ball valve supplies fluid to the bypass pipeline. The actuator has four cables connected to it which are the power inlet, the ground, the control, and the feedback cables. A 24V power supply is added to the setup to power the actuator. The power inlet and ground cables of the actuator are connected to this power supply.

By controlling the voltage across the control and feedback cables, the actuator can be rotated between 0° which represents that the ball valve is closed, meaning there will be no flow to the bypass pipeline, and 90° which represents the ball valve closed the inlet of the evaporator and fully opened the bypass pipeline. Since a closed evaporator inlet is not desired in the experiment, the actuator should never be rotated to 90°.

To control the voltage across the control and feedback cables, a simple electronic circuit must be established. The rotation of the actuator is related to the voltage difference between the feedback and control cables. If the feedback's voltage is more than the control cables, then the actuator rotates toward 90°. If the voltage across the control is more than the feedback's, then the actuator rotates toward 0°. The actuator stops when the voltage across the control and feedback cables are nearly equal to each other. The mechanism of the actuator requires that an adjustable resistance should be placed to control the voltage. Therefore, a potentiometer is placed between the control cable and the ground to establish an adjustable voltage across the control cable. Additionally, the feedback cable must be connected to a resistor before the ground. Otherwise, the voltage across the feedback will always be bigger than the control's voltage.

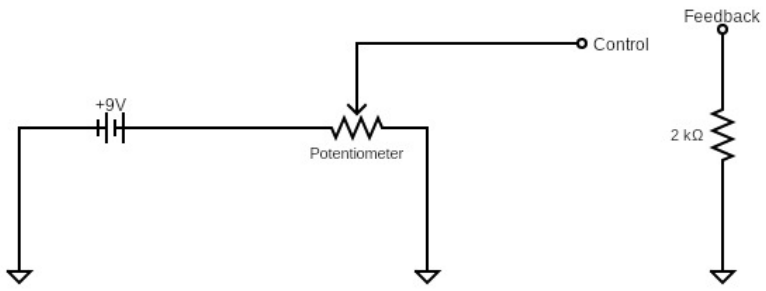



Figure 4.6: Circuit Diagram of the Experiment Setup

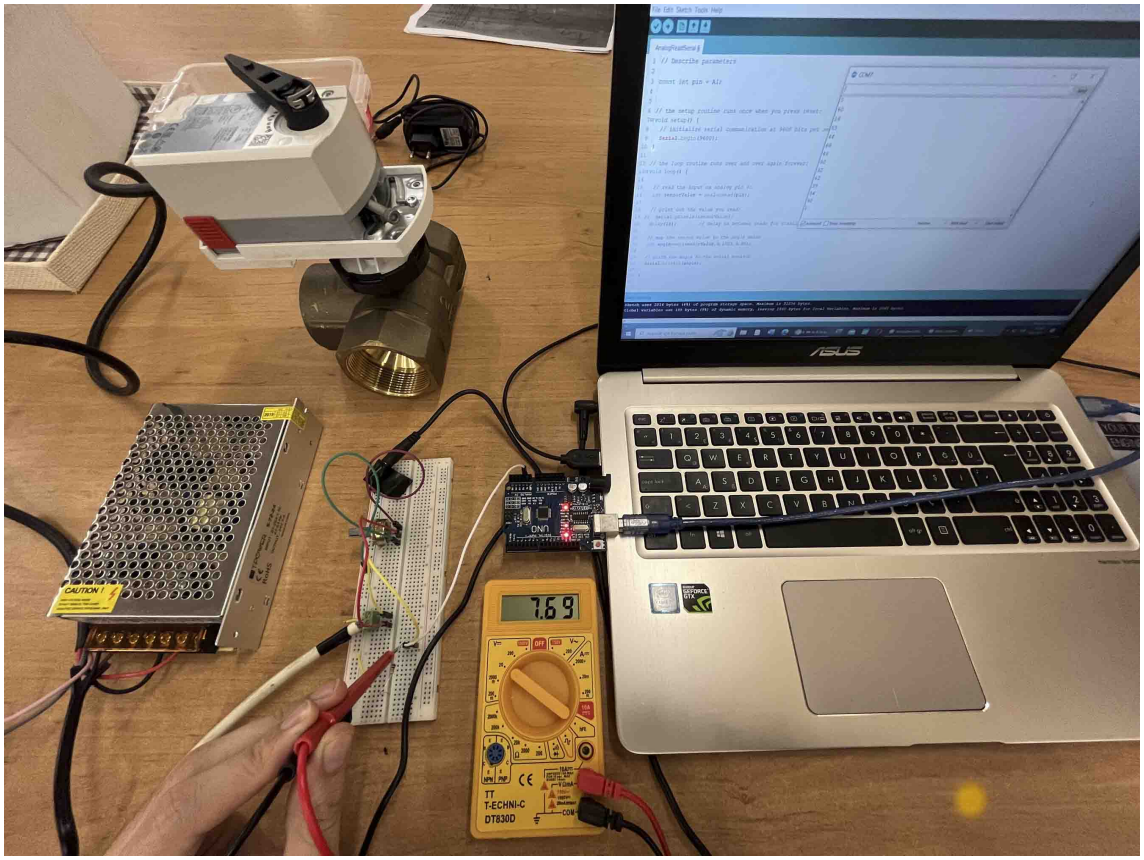



Figure 4.7: Actuator motor control.

In case of a possible real-world implementation, the damper actuator is tested. It is connected to the three-way valve, and then the valve is opened and closed respectively. In order to do that, all connections of the actuator are made. It is connected to a 24V DC power supply, its control signal is connected to a 12V DC power supply, and its voltage is controlled by a 10k potentiometer. The feedback signal is connected to the Arduino's A1 analog input port. The incoming signal is converted to the angle and read from the serial monitor. Several positions are tested, and a video recording is completed during this process.

This damper actuator control setup is shown in Figure 4.7

# Chapter 5

# Results & Discussion

## 5.1 Validation

Variables are determined in the previous section. In this section, by using these variables and the model, five different plots regarding the different set temperatures and trials are generated to compare the model's results with the experimental data.

Validation plots are provided below in Figures 5.1, 5.2, 5.3, 5.4 and 5.5.

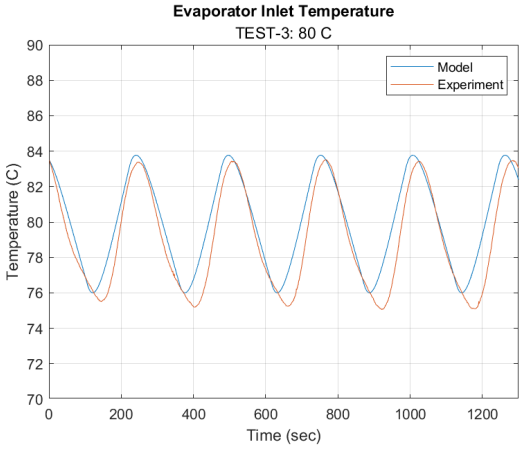



Figure 5.1: Test-3.

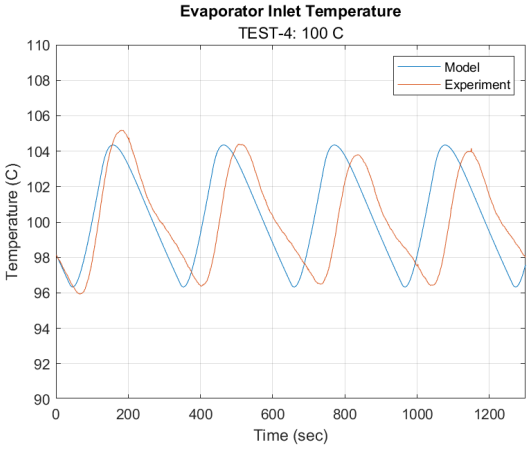



Figure 5.2: Test-4.

As can be seen, the results of the simulations are highly coherent with the experiment results. Tests 3 and 7 show that the model produces better results at low set temperatures, except for test 5. Test 4 and 6 demonstrates that the model data has a slightly low period than the experiment data when the set temperature is high.

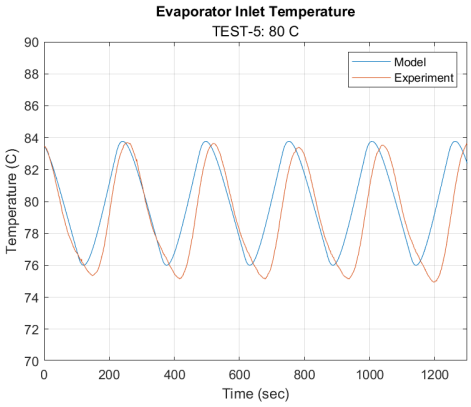



Figure 5.3: Test-5.

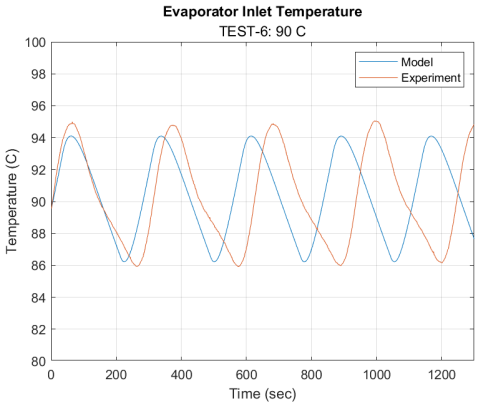



Figure 5.4: Test-6.

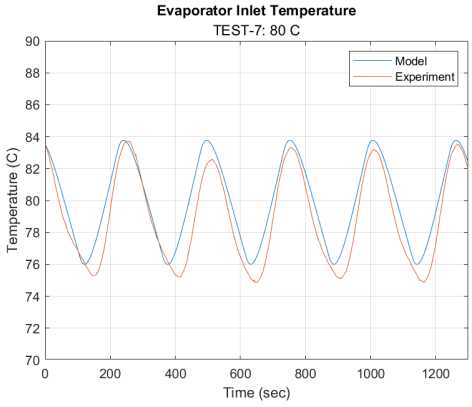



Figure 5.5: Test-7.

## 5.2 Result of the Bypass Pipeline

To see if the proposed solution works or not, the model is tested by setting a bypass rate. Then the results are compared with the older results without bypass line usage. The decrease in the oscillation amplitude is measured. The resulting plot is provided in Figure 5.6.

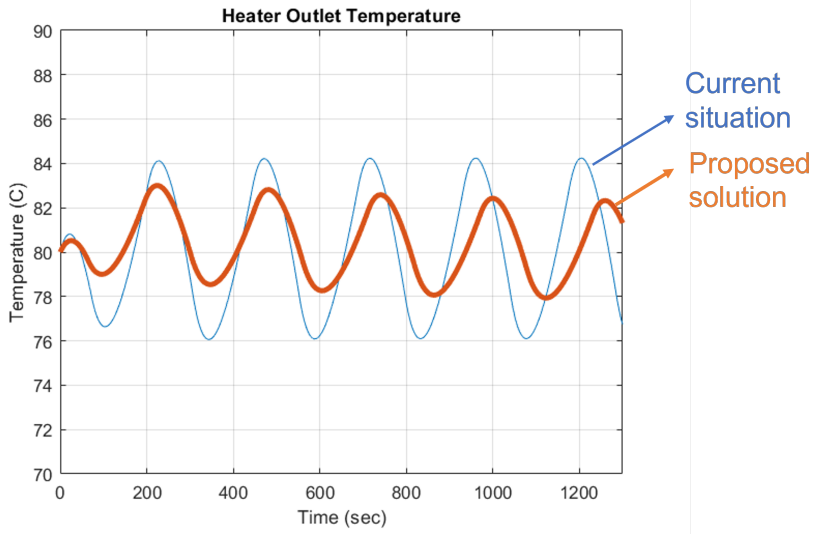



Figure 5.6: Comparison of temperature oscillation results with the solution.

As seen in Figure 5.6, several things are changed in the system's time response. Firstly, the oscillation amplitude is decreased as expected. Furthermore, the oscillation period is increased, which has no effect on the cycle, either positive or negative. Thus, results are satisfactory in the sense of decreasing the oscillation magnitude.

On the other hand, there are several things to be noted. When the bypass line is used, the set temperature needs to be set to a higher temperature. Also, the oil flowing in the cycle will be at higher temperatures to compensate for the temperature decrease in the evaporator inlet due to mixing with cooler oil. Higher temperatures overall can lead to more heat loss and, eventually, more power consumption. Additionally, higher temperatures in the components can increase material wearing. Lastly, the construction of this bypass system requires some amount of funding which should be taken into account as well.

The system is tested at its limits to find out how much reduction can be maintained by specified bypass percentages and set temperatures. Two different tests are carried out by setting bypass rates of 50% and 75%. Results are provided in Tables 5.1 and 5.2:

| | |
|---|---|
| Set temperature (°C): | 93 |
| Bypass rate (%): | 50 |
| Oscillation amplitude before (°C): | 7.0783 |
| Oscillation amplitude after (°C): | 4.9918 |
| Oscillation decrease percentage (%): | 30 |

Table 5.1: Limits of the system when bypass rate is 50%.

| | |
|---|---|
| Set temperature (°C): | 120 |
| Bypass rate (%): | 75 |
| Oscillation amplitude before (°C): | 7.0783 |
| Oscillation amplitude after (°C): | 3.0159 |
| Oscillation decrease percentage (%): | 57 |

Table 5.2: Limits of the system when bypass rate is 75%.

As a result, it is seen that the maximum decrease in oscillations can be achieved by 57% with the maximum operating temperature for 80 °C and 75% bypass rate. Thus, considering the drawbacks mentioned above, this reduction is seen as not sufficient enough to construct the real system. Still, this model can be used for other investigation purposes in the oil cycle. Furthermore, this modeling process can be a guideline for researchers who want to model a power cycle in Matlab/Simulink.

# Chapter 6

# Conclusion

The purpose of this report is to address the issue of oscillation in the temperature of the oil in an ORC (organic rankine cycle) system. This oscillation is caused by the heater in the oil cycle overshooting or undershooting the desired temperature. To solve this problem, a dynamic model was developed using Matlab/Simulink, which incorporates thermodynamic and heat transfer equations to simulate the various components of the oil cycle. This model was then validated by comparing it to experimental data, which showed that the model accurately represents the real system. To address the issue of oscillation, a solution was proposed in which a bypass pipeline is created between the inlet and exit of the oil cycle, allowing for reverse flow. This solution was simulated using the validated model, and the results showed that the oscillations could be significantly reduced by adjusting the flow through the bypass pipeline. However, since the resistor can not endure too high temperatures, there is a limit to the set temperature, which bounds the mass flow rate that can flow through the bypass line. Therefore, the oscillations cannot be reduced a lot when the set temperature is high. To test the proposed solution in a real system, an experimental concept setup was designed using a pump, a check valve, a 3-way valve, and an actuator. However, the actual experiment only utilized a 3-way valve and an actuator to demonstrate that the flow direction of the fluid could be controlled using these two components.

# References


[1] Matteo Marchionni, Giuseppe Bianchi, Apostolos Karvountzis-Kontakiotis, Apostolos Pesiridis, and Savvas A Tassou. Dynamic modeling and optimization of an orc unit equipped with plate heat exchangers and turbomachines. *Energy Procedia*, 129:224–231, 2017.

[2] Donghong Wei, Xuesheng Lu, Zhen Lu, and Jianming Gu. Dynamic modeling and simulation of an organic rankine cycle (orc) system for waste heat recovery. *Applied Thermal Engineering*, 28(10):1216–1224, 2008.

[3] Ying Zhang, Shuai Deng, Li Zhao, Shan Lin, Mengjie Bai, Wei Wang, and Dongpeng Zhao. Dynamic test and verification of model-guided orc system. *Energy Conversion and Management*, 186:349–367, 2019.

[4] Sylvain Quoilin, Adriano Desideri, Jorrit Wronski, Ian Bell, and Vincent Lemort. Thermocycle: A modelica library for the simulation of thermodynamic systems. In *10th international Modelica conference*, 2014.

[5] Erturul Altun. *Dynamic Modeling of Organic Rankine Cycle and Its Experimental Verification.* PhD thesis, 2018.

[6] Gianluca Carraro, Sergio Rech, Andrea Lazzaretto, Giuseppe Toniato, and Piero Danieli. Dynamic simulation and experiments of a low-cost small orc unit for market applications. *Energy Conversion and Management*, 197:111863, 2019.

[7] Muhammad Akmal and Brendan Fox. *Modelling and Simulation of Underfloor Heating System Supplied From Heat Pump.* 04 2016.

[8] MICHAEL J. MORAN. *Fundamentals of Engineering Thermodynamics.* WILEY, 2020.

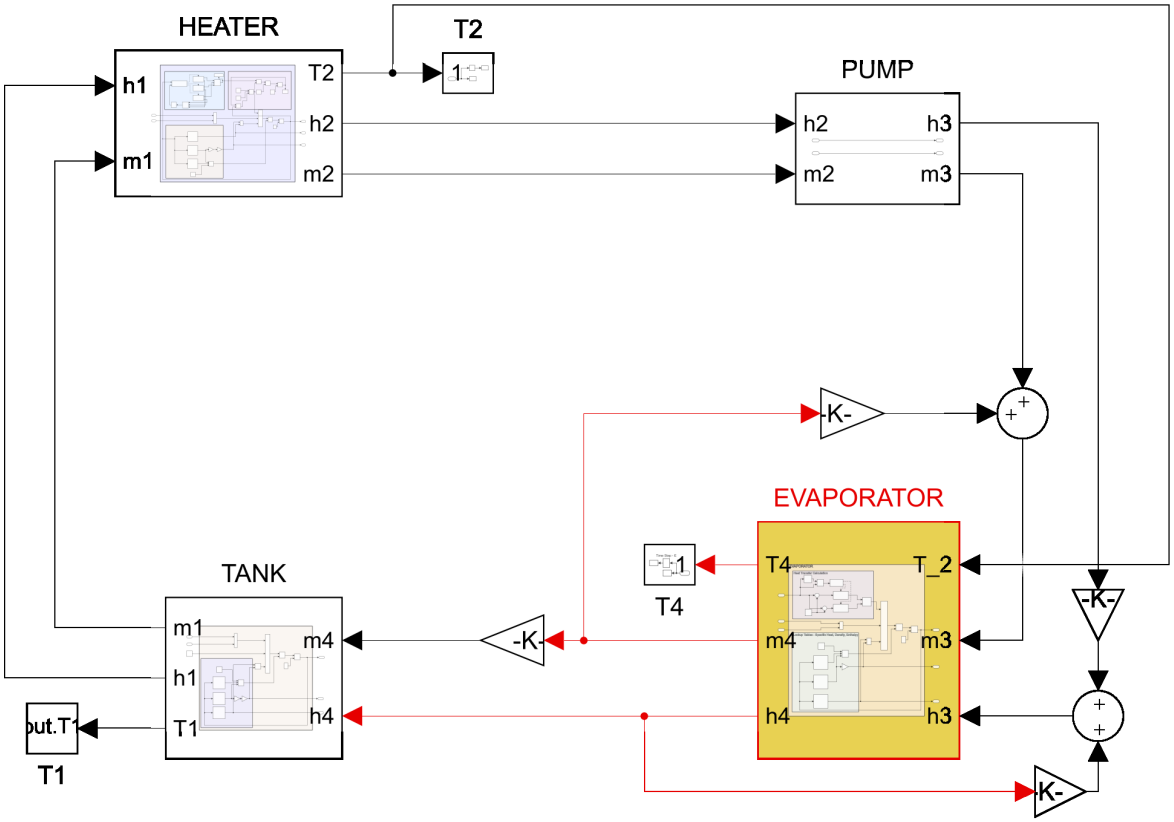
[9] Wilfrido Rivera, Karen Sánchez-Sánchez, J. Alejandro Hernández-Magallanes, J. Camilo Jiménez-García, and Alejandro Pacheco. Modeling of novel thermodynamic cycles to produce power and cooling simultaneously. *Processes*, 8(3), 2020.

[10] Frank P. Incropera and David P. DeWitt. *Fundamentals of Heat and Mass Transfer.* John Wiley & Sons, Inc., New York City, New York, 4th edition edition, 1996.

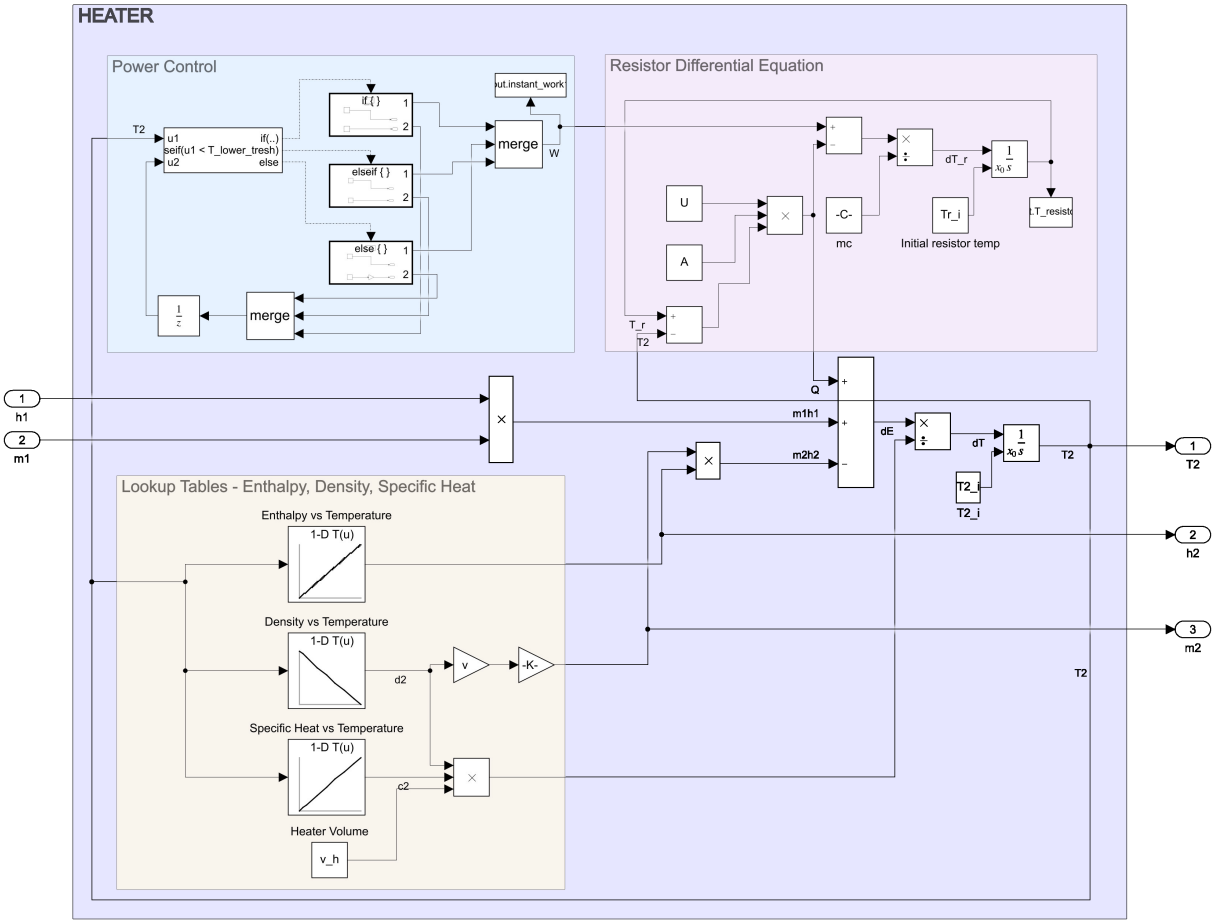
# Appendix A

# Simulink Model

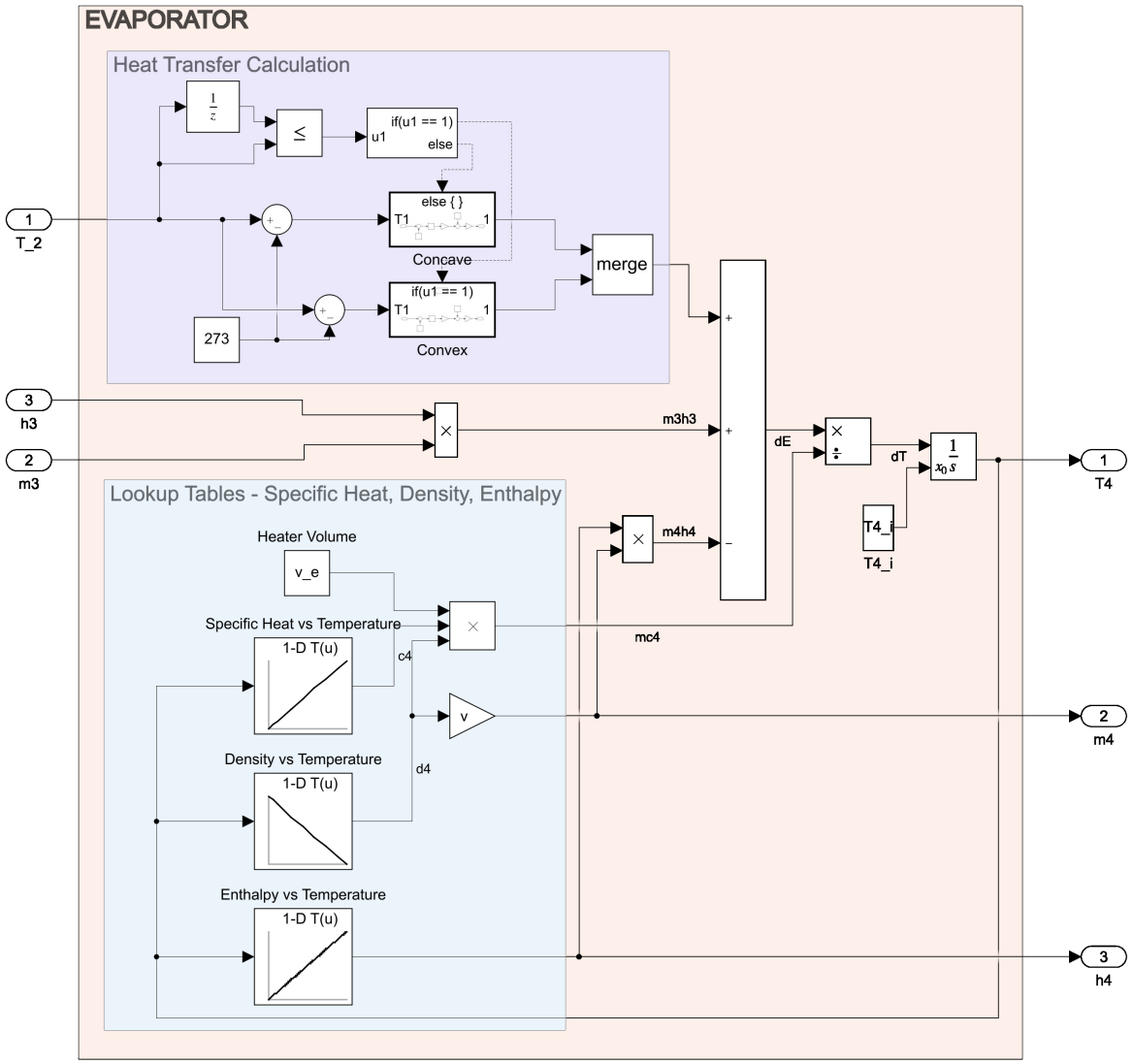
## A.1 Cycle Model



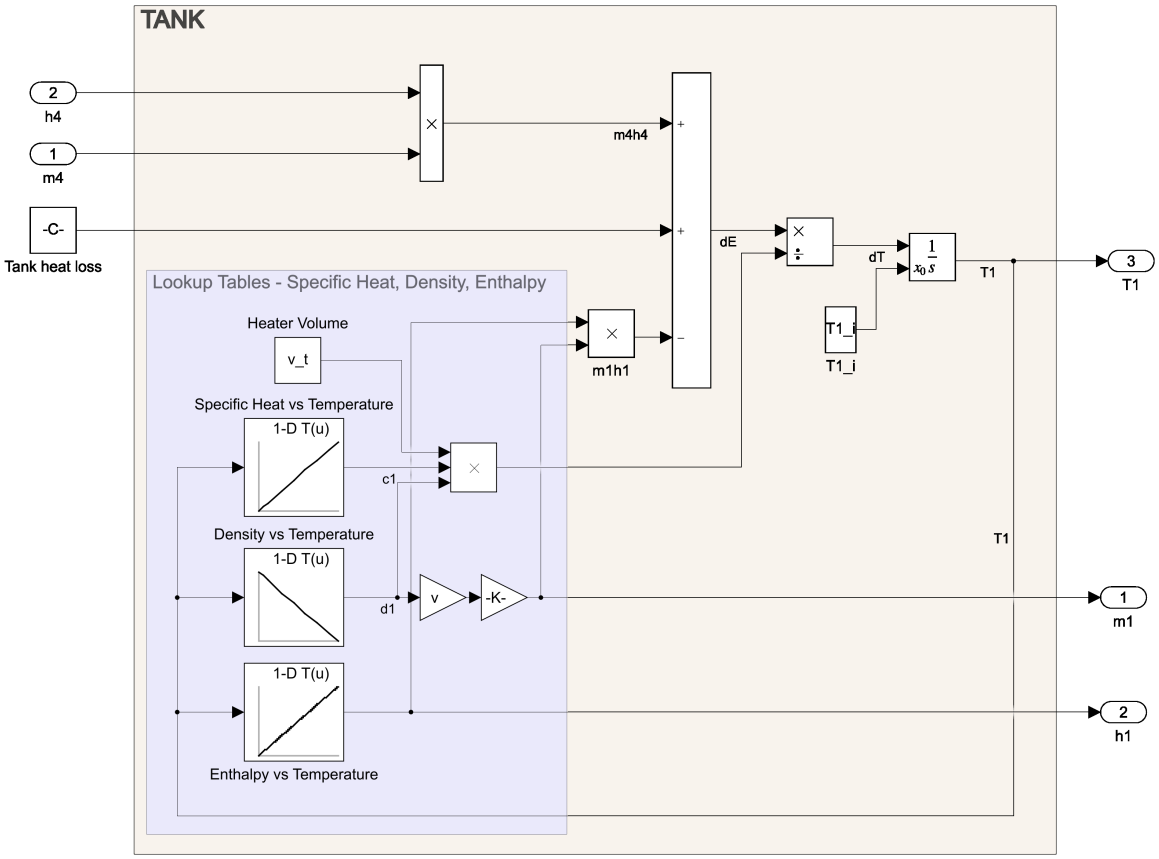
## A.2 Heater Model



## A.3 Evaporator Model



## A.4 Tank Model



# Appendix B

# Matlab Codes

## B.1 Main Live Script

*Load expirmental data:*

```
load("exp_data.mat")
```

**Select Test:**

```
test = 1; % select section of experimental data
switch test
    case 0
        % choose manual set temperature
    case 1
        T_set = 80 + 273; % test1 80
        lower_boundary = 220;
        upper_boundary = 2400;
    case 2
        T_set = 110 + 273; % test2 110
        lower_boundary = 2731;
        upper_boundary = 5000;
    case 3
        T_set = 80 + 273; % text3 80
        lower_boundary = 6300;
        upper_boundary = 8277;
    case 4
        T_set = 100 + 273; % test4 100
        lower_boundary = 8361;
        upper_boundary = 11096;
```

```
    case 5
        T_set = 80 + 273; % test5 80
        lower_boundary = 12257;
        upper_boundary = 14674;
    case 6
        T_set = 90 + 273; % test6 90
        lower_boundary = 15400;
        upper_boundary = 17700;
    case 7
        T_set = 80 + 273; % test7 80
        lower_boundary = 18188;
        upper_boundary = 20743;
end
```

## Parameters:

```
v = 0.001278; % volumetric flow rate (constant)
```

### Set Bypass Rate:

```
Rate = 0; % bypass rate percentage
bypassRate = Rate/100;
```

*If bypass:*

```
%add = (T_set-340)/(1-bypassRate)*bypassRate;
%T_set = T_set + add;
```

### Set Temperature:

```
if test == 0

    T_set =    80 + 273; % set temperature (K)

end
```

*Set thresholds:*

```
T_upper_tresh = T_set + 3;
T_lower_tresh = T_set - 3.5;
```

**Set Resistor Parameters:**

```
W =      100; % input work
m_r =    20; % resistor's mass
c_r =    0.4; % resistor's c (kJ/kgK)
U =      0.27; % overall heat transfer coefficient (kW/m2K)
A =      1; % area of the resistor (m2)
Tr_i =   T_set; % initial resistor temperature (K)
```

**Q_e fit:**

```
% concav
a_cv_int = [-0.1919 -0.2681];
b_cv_int = [-80.2397 -110.964];
c_cv_int = [36.531 19.2292];
a_cv = interp1([353 383], a_cv_int, T_set);
b_cv = interp1([353 383], b_cv_int, T_set);
c_cv = interp1([353 383], c_cv_int, T_set);

% convex
a_cx_int = [0.3102 0.1053];
b_cx_int = [-79.1199 -109.644];
c_cx_int = [24.3597 11.8066];
a_cx = interp1([353 383], a_cx_int, T_set);
b_cx = interp1([353 383], b_cx_int, T_set);
c_cx = interp1([353 383], c_cx_int, T_set);
```

**Q_loss fit:**

```
Q_tank = -7.4074e-06*(T_set-273-110)^4 - 8; % quadratic fit
```

**Density & Specific Heat Lookup Tables:**

```
T_lookup = [0 20 40 100 150 200 250 300 340] + 273;
d_lookup = [871 858 845 804 773 741 708 676 651];
c_lookup = [1.962 2.049 2.137 2.4 2.619 2.838 3.058 3.277 3.452];
```

**Heater parameters:**

```
v_h =       0.29; % oil's volume in the heater
T2_i =     T_set; % initial oil temperature in the heater
```

**Evaporator parameters:**

```
v_e =    0.00384;                % oil's volume in the evaporator
T4_i =   T_set - 13;             % initial oil temperature in evaporator
```

**Tank parameters:**

```
v_t =   0.167;                  % oil's volume in the tank + pipes
T1_i =  T_set - 13;             % initial oil temperature in tank
```

## Results:

*Simulation:*

```
out = sim('OilHeaterModel', 2300);
T1 = out.T1.Data - 273;
T2 = out.T2.Data - 273;
T4 = out.T4.Data - 273;

time = out.T1.Time;
```

*Temperature after bypass at evaporator inlet:*

```
T2_updated = T4*bypassRate + T2*(1-bypassRate);
```

*Oscillation magnitudes before and after bypass:*

```
O_magnitude_before = max(T2) - min(T2)
O_magnitude_after = max(T2_updated(70150:end)) - min(T2_updated(70150:end))
```

**Plot:**

```
figure
plot(time, T1)
grid on, hold on
title("Tank Outlet Temperature")
xlabel("Time (sec)")
ylabel("Temperature (C)")
ylim([0 100])
xlim([0 1300])

figure
plot(time, T2)
grid on, hold on
title("Heater Outlet Temperature")
xlabel("Time (sec)")
ylabel("Temperature (C)")
ylimits = [T_set-273-10 T_set-273+10];
ylim(ylimits)
xlim([0 1300])

figure
plot(time, T2_updated)
grid on, hold on
title("Evaporator Inlet Temperature")
xlabel("Time (sec)")
ylabel("Temperature (C)")
ylim(ylimits)
xlim([0 1300])

figure
plot(time, T4)
grid on, hold on
title("Evaporator Outlet Temperature")
xlabel("Time (sec)")
ylabel("Temperature (C)")
ylim([40 120])
xlim([0 1300])
```

## B.2 Training & Validation Script

```
T2_step = out.T2_step.Data;
T2_step_ein = zeros(2300,1);
j = 1;

for i=1:length(T2_step)
    if T2_step(i) > 0
        T2_step_ein(j) = T2_step(i);
        j = j + 1;
        if j > 2300
            break;
        end
    end
end
T_model_ein = T2_step_ein(1001:end) - 273;

T4_step = out.T4_step.Data;
T_step_eout = zeros(2300,1);
k = 1;

for i=1:length(T4_step)
    if T4_step(i) > 0
        T_step_eout(k) = T4_step(i);
        k = k + 1;
    end
end
T_model_eout = T_step_eout(1001:end) - 273;

T_exp_ein = T_ein(lower_boundary:upper_boundary);

initial = T_model_ein(1);

slop = T_model_ein(2)-T_model_ein(1) > 0;

min_error = 10;
starting_index = 1;
for i=1:500
    error = abs(initial - T_exp_ein(i));
    exp_slop = (T_exp_ein(i+1) - T_exp_ein(i)) > 0;
    if error < min_error && exp_slop == slop
        starting_index = i;
        min_error = error;
    end
end
T_exp_ein = T_exp_ein(starting_index:starting_index+1299);
```

```
mse = (T_model_ein - T_exp_ein)'*(T_model_ein - T_exp_ein)/1300

fig = figure;
plot(T_model_ein)
hold on, grid on
plot(T_exp_ein)
xlim([0 1300])
ylim(ylimits)
title("Evaporator Inlet Temperature", sprintf("TEST-%g: %g C", test, T_set-273))
xlabel("Time (sec)")
ylabel("Temperature (C)")
legend("Model", "Experiment")

text = 'C:\Users\ASUS\Desktop\Test-Results\TEST-' + sprintf("%g.png", test);
saveas(fig, text)
```