Doctoral Thesis

CFD MODEL OF VEHICLE CONDENSER

University of Pardubice
Faculty of Transport Engineering

Ing. Michal Schmid

Pardubice, Month Dayth, 2023

Department of Mechanics, Materials and Machine Parts

Supervisor:
doc. Ing. Petr Tomek, Ph.D.
University of Pardubice, Faculty of Transport Engineering
Supervisor specialist:
RNDr. Mgr. Václav Mácha , Ph.D.
Institute of Mathematics of the Czech Academy of Sciences

Programme of Study:

P3710 Technique and Technology in Transport and Communications

Branch of Study:

3706V005 - Transport Means and Infrastructure

Supervisors:

doc. Ing. Petr Tomek, Ph.D.

RNDr. Mgr. Václav Mácha , Ph.D.

Dissertation Title:

CFD model of vehicle condenser

The doctoral dissertation has arisen at the supervising:

Educational and Research Centre in Transport (ERCT)

Declaration of authorship

Abstract

The AC condenser plays a vital role in HVAC systems (Heating Ventilating Air Conditions). In vehicles, it is typically located within a cooling pack, alongside other heat exchangers. The dissipation of heat from the AC condenser directly affects surrounding components. Therefore, accurate modeling of heat transfer between the refrigerant and air is crucial for vehicle development, especially nowadays for battery electric vehicles.

The proposed model enhances the spatial distribution of heat transfer, resulting in improved air temperature predictions. The proposed model is based on the well-known and established $\epsilon - NTU$ approach and iterative takes into account the appropriate relation according to the refrigerant phase during the condensation process occurring in the condenser. Additionally, the approach reduces the amount of required input data and relies on directly measured condenser characteristics, leading to more generic approach.

In the thesis, a dedicated test equipment was developed for input data measurement and model verification. The proposed model was tested under two distinct conditions and compared with measurements. The model exhibited good agreement with the measurements in predicting refrigerant inlet and outlet temperatures, as well as relatively good agreement in air outlet temperature prediction. Furthermore, additional development tasks were identified as well.

Keywords:
AC Condenser, CFD, Heat Transfer, Measurement

Acknowledgements

I am deeply thankful to my supervisors prof. Ing. Petr Paščenko, Ph.D., doc. Ing. Petr Tomek, Ph.D. and RNDr. Mgr. Václav Mácha, Ph.D.,and my family, for their guidance, invaluable suggestions, technical expertise, and moral support.

Contents

List of Figures

List of Tables

Nomenclature

AC	Air Conditioning
CFD	Computational Fluid Dynamics
HTX	Heat Exchanger
$HVAC$	Heating, Ventilating, and Air Conditioning
$LGMD$	Log Mean Temperature Difference

α	Convective heat transfer coefficient	$(W/(m^2K))$
χ	Vapor/Refrigerant quality	$(-)$
Δ	Difference	$(-)$
δ	Convergence tolerance	$(-)$
$\dot{m}$	Mass flow rate	(kg/s)
$\dot{Q}$	Heat transfer rate	(W)
ϵ	Effectiveness	$(-)$
λ	Thermal conductivity	$(W/(mK))$
θ	Sensitivity coefficient	$(-)$
ϑ	Degrees of freedom	$(-)$
A	Area, Heat transfer area, Cell surface area	(m^2)
B	Bias deviation	$(-)$
C	Heat capacity	(W/K)
c_p	Specific heat capacity at constant pressure	$(J/(kgK))$
Cr	Heat capacity ratio	$(-)$
dx, dy, dz	Cell size in x,y,z direction	(m)

F_G	Geometric correction factor	$(-)$
G_i	Geometrical Parameter	$(m;-)$
h	Specific enthalpy	(J/kg)
L	Characteristic length	(m)
N	Cell count	$(-)$
Ns	Number of samples	$(-)$
NTU	Number of transfer unit	$(-)$
Nu	Nusselt number	$(-)$
p	Static pressure	(Pa)
Pr	Prandtl number	$(-)$
R_e	Reynolds number	$(-)$
R_w	Conduction resistance	(K/W)
$R_{specific}$	Specific gas constant	$(J/(kgK))$
S	Sample standard deviation	$(-)$
s	Specific entropy	$(J/(kgK))$
S_t	Student distribution standard deviation	$(-)$
T	Temperature	(K)
$t_{\vartheta-\%}$	Student distribution t-score	$(-)$
U	Overall heat transfer coefficient	$(W/(m^2K))$
u	Velocity magnitude	(m/s)
UI	Uniformity Index	$(-)$
Un	Uncertainty	$(-)$
V	Volume	(m^3)
X	Data point	$(-)$
$Xcells, Ycells, Zcells$	Cell count in x,y,z direction	$(-)$

Subscripts

aux	Auxiliary fluid
ave	Average
$global$	Heat exchanger domain level
i, j, k	x,y,z axis index
in	Inlet
$local$	Local level (macro, cell)
m	Arithmetic mean
max	Maximum
min	Minimum
n	Sample point
out	Outlet
$prim$	Primary fluid
$table$	Input table level

1 Introduction

Nowadays, virtual prototyping has become an integral part of modern vehicle development. It encompasses a wide range of activities, including numerical simulations, drawings, and product lifecycle management. The simulations themselves encompass various physical phenomena, ranging from electromagnetism, fluid dynamics, and thermal dynamics to structural analyses. The prevalent numerical methods used in modern simulations are the Finite Volume Method (FVM) and the Finite Element Method (FEM). However, other numerical approaches such as Lattice Boltzmann or Smooth Particle Hydrodynamics are gaining popularity. In recent years, computational resources have significantly expanded, enabling the creation of comprehensive numerical models that utilize hundreds or thousands of CPUs (Central Processing Units) per simulation. This utilization of High-Performance Computing (HPC) is not limited to CPUs alone; graphics cards are also being employed in HPC applications. From an engineering perspective, virtual prototyping allows for a reduction in the number of physical prototypes and measurements required. Furthermore, simulation plays a crucial role in vehicle optimization and design exploration. As a result, virtual prototyping is indispensable in the current automotive industry due to increasingly stringent emissions regulations and other vehicle design requirements. Consequently, these demands have led to higher requirements for simulation accuracy, computational time reduction, and other factors. Computational Fluid Dynamics (CFD) is particularly vital in the development of vehicle thermal management and overall vehicle design, such as external aerodynamics. The accuracy of simulations is even more critical for modern battery electric vehicles (BEVs) due to their lower energy reserves compared to conventional combustion engine (ICE) vehicles. Each watt of energy must be carefully considered to extend the vehicle's range. Additionally, the heat dissipation within the system is lower in BEVs, leading to lower system temperatures. Therefore, accurately capturing small temperature differences becomes essential in this context.

One of the crucial components in vehicle thermal management is the HVAC (Heating, Ventilating, and Air Conditioning) system. The HVAC system plays a crucial role in ensuring cabin comfort, occupant safety, as well as the cooling of the engine or battery pack in the case of BEVs. The significance of the HVAC system in the context of vehicle electrification is extensively discussed in König et al. (2022). The HVAC unit typically comprises various elements, including a cabin heater, AC (Air Conditioning) evaporator, condenser, ducting, filters, blower, compressor, and others. As a major contributor to vehicle fuel or electric energy consumption, optimizing the energy efficiency of the HVAC system holds significant importance in modern vehicle development. Generally, the heat generated by the vehicle, such as the cabin and the battery pack, dissipates into the surrounding air within the studied heat exchangers.

In modern vehicles, various heat exchangers (HTX) are integrated primarily for the purpose of dissipating heat into the ambient air. Examples of these HTX include the engine coolant HTX (radiator), Charged Air Cooler (CAC), as well as coolers for steering, transition oil, fuel, and the aforementioned AC condenser. These HTX are typically combined with a cooling fan and shroud to form a so-called cooling pack. Consequently, each HTX has an impact on the others within this assembly. Computational Fluid Dynamics (CFD) simulations focused on this area are commonly referred to as under bonnet/underhood thermal management (UTHM) or front-end simulations. The CFD modeling of HTX not only affects the prediction of coolant, oil, and fuel temperatures (auxiliary fluids) but also influences other under-bonnet components (exhaust, battery, brackets, filters, etc.) due to the airflow heating. The accurate prediction of airflow rate and fluid temperatures relies directly on the proper modeling of HTX heat transfer.
This thesis specifically focuses on enhancing the heat transfer model of the AC condenser within a comprehensive underhood CFD simulation of a full vehicle. The objective is to improve prediction of heat dissipation from the AC condenser. The proposed model enhancement aims to accurately capture the thermodynamic characteristics of each section of the AC condenser, considering both phase change and single-phase regions. The appropriate heat transfer model should be implemented based on the operating conditions and boundary conditions of the AC condenser. By improving the 1D AC condenser model, the accuracy of airflow rate and temperature predictions in full vehicle CFD simulations should be increased as well. The standard physical measurements of HTX need to be modified to obtain suitable input data for the suggested numerical model. Furthermore, the proposed model and measurement technique are more general compared to existing approaches for AC condenser modeling. Additionally, a verification measurement is conducted to validate and demonstrate the benefits of the developed AC condenser model.
The initial thesis sections discuss the calculations, modeling, and measurement related to the AC condensers. Furthermore, it suggests objectives based on current state-of-the-art research. The thesis Sections 4, 5, 6, 7, and 8 presents the suggested and developed AC condenser model. In Section 9, 10, 11, and 12 the thesis describes the measurement devices and test equipment developed for obtaining input data for the model and verifying model accuracy. The subsequent part of the thesis presents the results of the measurements and the calculations performed using the AC condenser model. Finally, the results are compared, discussed, and summarized in the conclusions. The conclusions also outline future steps based on the findings and conclusions of the thesis.

2 The current state of research

Within the chapter, the current state of research in the area in question is discussed and summarized. The first part, Section 2.1, focuses on the general heat exchanger calculations and calculations in the context of full vehicle simulations. General measurement of heat exchangers is summarized in the following Section 2.2. Furthermore, the current state of the art in AC condenser modeling and measurement is discussed within the Section 2.3.

2.1 Full Vehicle Heat Exchanger Calculations

Flow arrangement is a essential specification for heat exchangers (HTX), with common types including counter flow, parallel flow, and cross-flow HTXs. Another important consideration is the heat flux arrangement, distinguishing between mixed and unmixed flows. In an unmixed flow, there is no heat transfer in the direction perpendicular to the flow, whereas a completely mixed flow results in a constant fluid temperature in the normal flow direction. This definition is outlined in Taborek (1983). In the context of vehicle heat exchangers, a typical configuration is the cross-flow HTX with both fluids being unmixed. Figure 2 from Incropera et al. (2011) illustrates a cross-flow HTX with both fluids unmixed, as well as the scenario where the primary fluid is mixed. The figure also provides nomenclature for the working fluids. The cross-flow direction typically refers to the primary side (commonly air), while the tube flow pertains to the auxiliary side (such as coolant or refrigerant). The focus of the thesis is on the heat transfer between these two fluids within the AC condenser. Figure 2 from Kim and Kim (2008) presents a schematic representation of the cooling pack, including the AC condenser, within the underhood area of a vehicle.

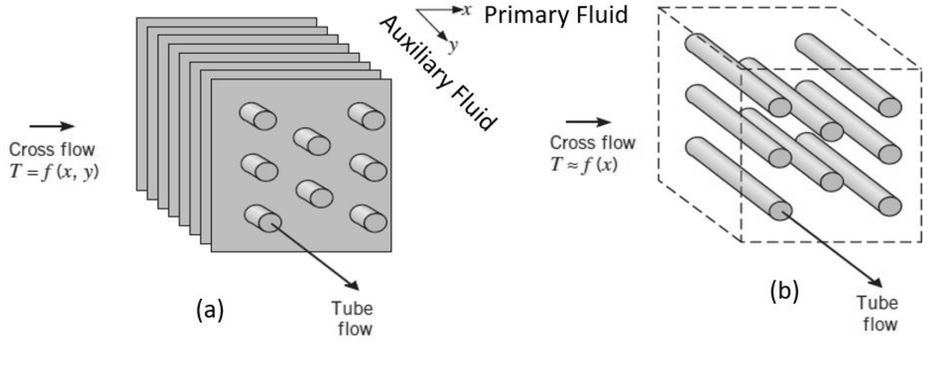


Figure 1: Cross-Flow HTX. (a) both fluid unmixed (b) primary fluid mixed Incropera et al. (2011)

The HTX is commonly integrated into full-vehicle 3D CFD models using a porous media approach. This method is preferred over directly modeling the complex geometry of the tube and fins, as it is computationally more efficient. By implementing porous media, the pressure drop across the HTX can be modeled using the Darcy-Forchheimer law with appropriate inertia and viscous coefficients Lu and qiang Lu (2018). In some cases, a multi-scale detailed sub-modeling approach has been utilized for a more comprehensive analysis, such as in Shui-Chang et al. (2014). However, this approach may result in increased computational demands. Underhood studies pertaining to full vehicles have been extensively discussed in previous research of Ding et al. (2006), Kim and Kim (2008), Mao et al. (2010). Additional aspects, such as transient modeling of auxiliary fluid (coolant) was studied by Pang et al. (2012) or consideration of drag coefficient have also been examined by Saab et al. (2013). Specific investigations on HTX performance in underhood simulations can be found in Crippa et al. (2011). Similarly, the dynamic response of the HTX core has been explored in previous work of Gao et al. (2014).

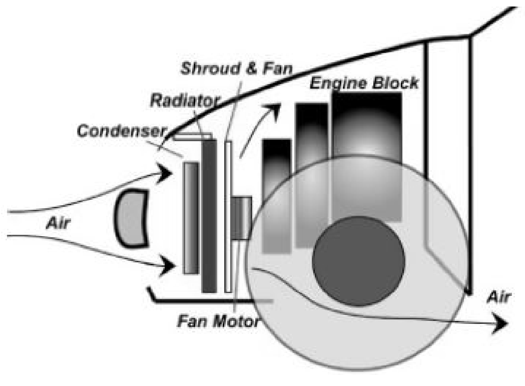


Figure 2: Vehicle underhood scheme Kim and Kim (2008)

In the modeling of HTX core heat transfer, various methods can be used, including the $\epsilon - NTU$ method, Log Mean Temperature Difference ($LMTD$), and Arithmetic Mean Temperature Difference ($AMTD$) Huang et al. (2014). These methods are implemented to define the overall heat transfer of the HTX. The $\epsilon - NTU$ method, $LMTD$, and $AMTD$ approaches assume steady-state conditions and constant material properties. However, there is a fundamental difference between the $\epsilon - NTU$ method and the Mean Temperature methods, which is the requirement of inlet and outlet temperatures for both the primary and auxiliary fluids in the Mean Temperature methods. On the other hand, the $\epsilon - NTU$ method only requires the inlet temperatures of the fluids and the fluid configuration curves (known as $\epsilon - NTU$ relations) Incropera et al. (2011), Kays and London (1984), Webb (2007). These methods are described in more detail in the sections below.

Pressure drop is a critical parameter in HTX analysis, and it is closely linked to heat transfer performance. While the focus of the thesis lies on heat transfer, it is essential to acknowledge the significant role of pressure drop in HTX design and operation. The quantitative relationship between pressure drop and temperature drop in an HTX is discussed in previous studies of Bejan (1978) and Liang et al. (2015). However,the developed model in this thesis does not directly calculate the flow field and, as a result, does not predict pressure drop. Instead, pressure drop is assumed as input data, which can be obtained through measurements or coupled with a separate 3D Computational Fluid Dynamics (CFD) analysis. Examples of 1D HTX solver coupling with airflow 3D CFD simulations, could be found in Kumar et al. (2010), or Minovski et al. (2015).

2.1.1 eNTU Method

In scenarios where only inlet temperatures are known for heat exchanger calculations, the $\epsilon - NTU$ method-based approaches are commonly used, as the log mean temperature approach cannot be directly applied without an iterative technique Incropera et al. (2011). The effectiveness ϵ is defined as the ratio of the actual heat transfer rate to the theoretically maximum possible heat transfer rate (infinite heat exchanger core length) as shown in Equation 1 Kays and London (1984). The Number of Transfer Units (NTU) represents the "heat transfer size" and is a dimensionless parameter as defined in Equation 2 Kays and London (1984). It is important to note that maintaining a constant overall heat transfer coefficient U over the entire surface area A is assumed in the analysis. Within the context of 3D CFD, this assumption is satisfied as the heat exchanger volume is divided into multiple cell volumes. Where each cell has a constant U value over its local area.

$$\epsilon = \frac{\dot{Q}}{\dot{Q}_{max}} = \frac{\dot{Q}}{C_{mins}(T_{in,aux} - T_{in,prim})} \tag{1}$$

$$NTU = \frac{1}{C_{min}} \int_0^A U \, dA = \frac{UA}{C_{min}} \tag{2}$$

In Equations 1 and 2, the $\dot{Q}$ refers to the heat transfer rate (W), U refers to the overall heat transfer coefficient ($W/(m^2K)$), and A represents the heat transfer area (m^2). The overall heat transfer coefficient, denoted by U, takes into account the convective heat transfers as well as solid wall conduction. It can be expressed as shown in Equation 3 Incropera et al. (2011). In this equation, α represents the convective heat transfer coefficient ($W/(m^2K)$), and the solid conduction is represented by the conduction resistance R_w (K/W) Incropera et al. (2011).

$$\frac{1}{UA} = \frac{1}{(\alpha A)_{prim}} + R_w + \frac{1}{(\alpha A)_{aux}} \tag{3}$$

The definition of the minimal heat capacity, denoted as $C_{min}(W/K)$, is given by Equation 4 Kays and London (1984). It represents the minimum heat capacity rate between the auxiliary fluid and the primary fluid, or alternatively, the maximum heat capacity rate. Kays and London (1984)

$$C_{min/max} = min/max\{\dot{m}_{auxiliary}c_{p,auxiliary}; \dot{m}_{primary}c_{p,primary}\} \tag{4}$$

In general, the parameter ϵ depends on the variables NTU, C_r, and the flow arrangement (e.g., cross-flow heat exchanger), as mentioned in Equation 5 Incropera et al. (2011). The ratio C_r is defined as the ratio of C_{min} to C_{max}, as shown in Equation 6 Incropera et al. (2011).

$$\epsilon = f(NTU, C_r, FlowArrangement) \tag{5}$$

$$C_r = \frac{C_{min}}{C_{max}} \tag{6}$$

The relationship between ϵ and NTU for a unmixed fluid cross-flow heat exchanger (e.g. AC condenser) is described by Equation 7 Incropera et al. (2011). However, this equation is strictly accurate only when $C_r = 1$. For other values of ϵ and the NTU relationship, a detailed and exact solution was derived by Triboix (2009). Nevertheless, Equation 7 can still be used as a reasonable approximation for $0 < C_r <= 1$ Incropera et al. (2011). For more complex arrangements, an iterative approach to obtain the ϵ-NTU relationship is suggested in Navarro and Cabezas-Gomez (2007). Additionally, a study on the performance of multi-pass parallel cross-flow heat exchangers based on the NTU method is conducted in Silaipillayarputhur and Mughanam (2018).

$$\epsilon = 1 - exp[-\frac{NTU^{0.22}}{C_r}(1 - e^{-C_r NTU^{0.78}})] \tag{7}$$

For heat exchangers with one fluid mixed, an exact relationship between ϵ and NTU exists, which is defined in Equations 8 and 9 Incropera et al. (2011). However, it is important to note that while the thermodynamic definition is rigorous, the actual physical geometry of a heat exchanger can sometimes lead to partially mixed conditions in certain scenarios. The uncertainty in defining the $\epsilon - NTU$ relationship and the proposed methods for dealing with partially mixed conditions are discussed in Digiovanni and Webb (1989). In general, heat exchanger designs with tubes and fins are considered to fall under the category of fluid unmixed heat exchangers Incropera et al. (2011) Digiovanni and Webb (1989). However, it is worth noting that the specific shapes of tubes and fins can vary among heat exchanger manufacturers, potentially leading to partially mixed or mixed designs.

$$C_{max}(mixed)\, C_{min}(unmixed): \quad \epsilon = \frac{1}{C_r}(1 - exp\{-C_r[1 - exp(-NTU)]\}) \tag{8}$$

$$C_{min}(mixed)\, C_{max}(unmixed): \quad \epsilon = (1 - exp\{-C_r^{-1}[1 - exp(-C_r(NTU))]\}) \tag{9}$$

2.1.2 Log Mean Temperature Difference Method

The log mean temperature difference ($LMTD$) approach is employed to calculate the average temperature in the heat balance equation, as shown in equation 10 Incropera et al. (2011). The average temperature is determined using a logarithmic temperature profile across the length of the heat exchanger (HTX) core, assuming a counter-flow arrangement, as shown in equation 11 Incropera et al. (2011). The equation 11 Incropera et al. (2011) is written below under the assumption that the auxiliary fluid's inlet temperature is higher than the primary fluid's inlet temperature. This approach enables the estimation of the average temperature difference, which is a key parameter in heat transfer calculations.

$$\dot{Q} = U \times A \times LMTD \tag{10}$$

$$LMTD = \frac{(T_{in,aux} - T_{out,prim}) - (T_{out,aux} - T_{in,prim})}{ln\frac{T_{in,aux}-T_{out,prim}}{T_{out,aux}-T_{in,prim}}} \tag{11}$$

Equation 11 in Incropera et al. (2011) is derived specifically for a simple counter-flow heat exchanger. However, in the case of cross-flow heat exchangers, the temperature profiles of the primary and auxiliary fluids vary along their respective flow directions. This is illustrated in Figure 3 Kays and London (1984).

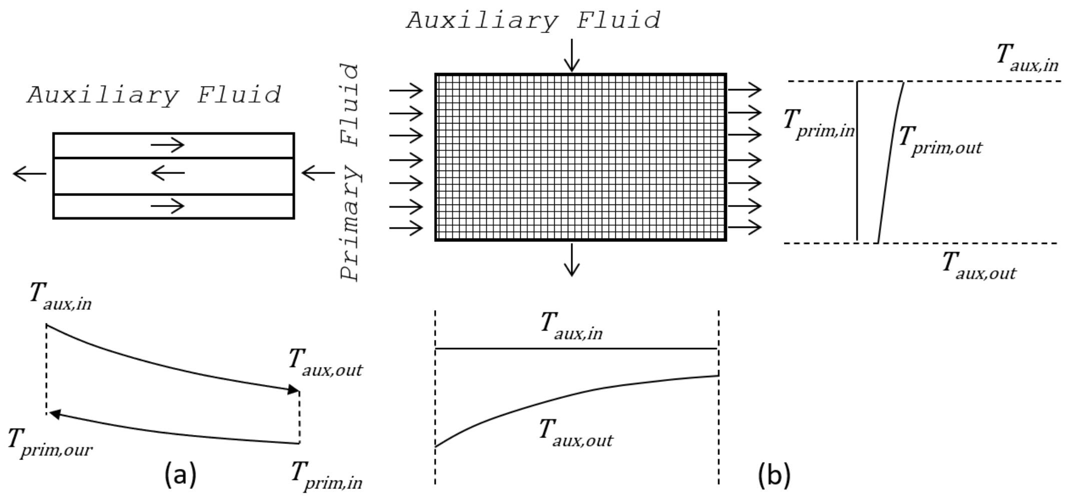


Figure 3: HTX temperature profile (a) Counter-flow (b) Cross-flow Kays and London (1984)

The non-dimensional factor F_G is used to modify Equation 11 Kays and London (1984) in order to account for different flow arrangements in heat exchangers. It is commonly referred to as the geometric correction factor. The modified equation, known as the log mean rate equation, is given by Equation 12 Kays and London (1984). The factor F_G takes a value of unity for the counter-flow arrangement, while for other flow arrangements, it is less than one. The values of F_G for various flow arrangements and operating temperatures can be found in Bowman et al. (1940). The evaluation of the cross-flow arrangement factor has been studied in detail by Smith (1934) and Mason and John (1955). Examples of in-house and general numerical implementations of the arithmetic mean temperature difference method can be found in Ribando et al. (1997).

$$\dot{Q} = U \times A \times F_G \times LMTD \tag{12}$$

2.1.3 3D CFD Codes Implementation

The chapter focuses on different utilization of Computational Fluid Dynamics (CFD) codes for modeling heat exchangers (HTX). Specifically, two common commercial CFD tools, namely $Simcenter\ STAR-CCM+$ and $Ansys\ Fluent$, were selected for comparison within the chapter. While both CFD codes offer simpler models for heat exchangers, the chapter primarily focuses on the most detailed models available. These detailed models rely on a provided performance table known as the $Q-Table$. The $Q-Table$ is typically obtained through experimental measurement and contains data on HTX heat dissipation for various primary and auxiliary mass flow rates, assuming constant inlet temperatures. From the $Q-Table$, the overall heat transfer coefficient or the ϵ values are extracted. These values are used to characterize the heat transfer performance of the heat exchanger. In cases where flow rates are not directly specified in the $Q-Table$, interpolation methods such as linear interpolation are employed to estimate the corresponding values in the performance space.

Ansys Fluent

In the $Ansys\ Fluent$ software, the $\epsilon-NTU$ method is implemented through a so-called $macro-based$ model. The $\epsilon-NTU$ approach, as described in Section 2.1.1, is applied to multiple local macros or computational cells within the heat exchanger. The heat transfer properties are scaled based on Equation 13 Fluent (2009). In this equation, V represents the appropriate volume (m^3) ($local$ stands for $macro/cell$ and $global$ for s whole HTX). The $\epsilon-NTU$ relation used in the $Ansys\ Fluent$ implementation is the same as Equation 7 Incropera et al. (2011). Therefore, it should be noted that the $Ansys\ Fluent$ implementation is valid only for unmixed fluids, cross-flow heat exchangers without any phase change Fluent (2009)

$$NTU_{local} = NTU_{global}\frac{V_{local}C_{min,global}}{V_{global}C_{min,local}} \tag{13}$$

The $local(macro)$ heat transfer in the $Ansys\ Fluent$ implementation is calculated using Equation 14 Fluent (2009). The final temperature is determined through a steady-state energy equation, as described in Equation 15 Incropera et al. (2011). Equation 14 Fluent

(2009) is derived from Equation 1 Incropera et al. (2011) mentioned earlier. If the *macro* consists of multiple computational cells, the heat transfer within each cell is expressed by Equation 16, and the total heat transfer within the heat exchanger is given by Equation 17 Fluent (2009).
The $Q-Table$ is utilized to generate the global ϵ and NTU values Fluent (2009). However, since Equation 7 Incropera et al. (2011) cannot be directly rearranged to express NTU as a function of effectiveness ϵ Incropera et al. (2011), the $\epsilon - NTU$ relation is solved iteratively using the Newton-Raphson method. This iterative approach allows for the determination of NTU based on the given effectiveness ϵ Incropera et al. (2011).

$$\dot{Q}_{local} = \epsilon_{local} C_{min,local} (T_{in,aux} - T_{in,prim}) \tag{14}$$

$$\dot{Q}_{local} = \dot{m} c_p \Delta T \tag{15}$$

$$\dot{Q}_{cell} = \dot{Q}_{local} \frac{V_{cell}}{V_{local}} \tag{16}$$

$$\dot{Q}_{global} = \sum \dot{Q}_{local} \tag{17}$$

Simcenter STAR-CCM+
In contrast to the $\epsilon - NTU$ approach used in *Ansys Fluent*, $Simcenter\ STAR-CCM+$ employs a modified average temperature difference method for heat exchanger analysis. In this method, the term *local* refers directly to a computational cell volume, as $Simcenter\ STAR-CCM+$ does not utilize a $macro-based$ model. A further difference between *Ansys Fluent* and $Simcenter\ STAR-CCM+$ is the implementation of auxiliary fluid flow phenomena. In the $macro-basedmodel$ of *Ansys Fluent*, it assumes a uniform distribution of auxiliary fluid flow and does not directly simulate the flow field. On the other hand, $Simcenter\ STAR-CCM+$ incorporates only a more general technique called the *Dual Cell* or *Dual Stream* approach, where the auxiliary fluid flow is explicitly modeled in a separate overlapping domain. It's worth mentioning that this approach is also available in *Ansys Fluent*.
The $Q-Table$ is utilized to generate the overall heat transfer coefficient U based on the table's inlet temperature difference, as described by Equation 18 STAR-CCM+ (2019). While the literature often favors the log mean temperature difference ($LMTD$) method including the correction factor for geometrical configuration for heat exchanger analysis, $Simcenter\ STAR-CCM+$ employs the arithmetic mean temperature difference ($AMTD$) method, as shown in Equation 19 STAR-CCM+ (2019). To determine the

global overall heat transfer coefficient, the average value of the inlet temperature difference from each computational cell is used. However, it is important to note that this method is valid when the computational cells are sufficiently small. Each individual HTX cell represents a local heat sink or source (auxiliary or primary) and is defined using Equation 21 STAR-CCM+ (2019).

$$UA_{table} = \frac{\dot{Q}_{table}}{T^{table}_{in,aux} - T^{table}_{in,prim}} \tag{18}$$

$$UA_{global} = \frac{UA_{table}(T^{ave}_{in,aux} - T^{ave}_{in,prim})}{\Delta T_{ave} N} \tag{19}$$

. . . where:

$$\Delta T_{ave} = \frac{\sum(T_{in,aux} - T_{in,prim})V_{cells}}{\sum V_{cells}} \tag{20}$$

The main advantage of the method used in $Simcenter\ STAR-CCM+$ is that it eliminates the need for recalculation of the $\epsilon - NTU$ relationship, making it a more general approach.

$$\dot{Q}_{local} = \frac{UA_{local}(T_{in,aux} - T_{in,prim})V_{cells}}{\frac{1}{N}\sum V_{cells}} \tag{21}$$

2.2 Heat Exchanger Measurement

A schematic of a heat exchanger (HTX) measurement facility is presented in Figure 4 Zhao (1995). The facility consists of two separate flow circuits: one for the cooling air (primary side) and another for the auxiliary side fluid (such as water). The measurements conducted can focus on obtaining characteristic quantities of the measured HTX, such as the $Q - Table$, which represents the heat transfer performance of the heat exchanger. Alternatively, the facility can be used for model verification purposes. In both cases, specific quantities typically measured to assess the performance of the heat exchanger are listed below: Khot et al. (2012)

- heat dissipation on the primary and auxiliary fluid side,
- inlet/outlet temperature difference,
- primary and auxiliary side flow rate,
- primary and auxiliary pressure drop.

Consequently, various physical quantities are measured, including temperature, velocity, flow rate, and pressure. The accuracy of these measurements is crucial, especially since heat dissipation is calculated based on flow and temperature assessments rather than being directly measured.

In order to ensure accurate measurements, it is suggested in Khot et al. (2012) to have a temperature measurement accuracy of at least $\pm 1°C$ and a flow meter accuracy of at least $\pm 2\%$. Pressure measurements using a column gauge should have an accuracy of at least $1mm$. However, it is important to note that even with lower accuracy probes, as shown in Tatara and Lupia (2011), the uncertainty in the overall heat transfer rate can be as high as 9.5%. In certain cases, such as those outlined in EN1643033 (2011), even higher measurement accuracies are required. For example, a temperature accuracy of $0.1K$ and a flow accuracy of 1% are recommended to meet the standards specified in the EN (European Norms) guidelines.

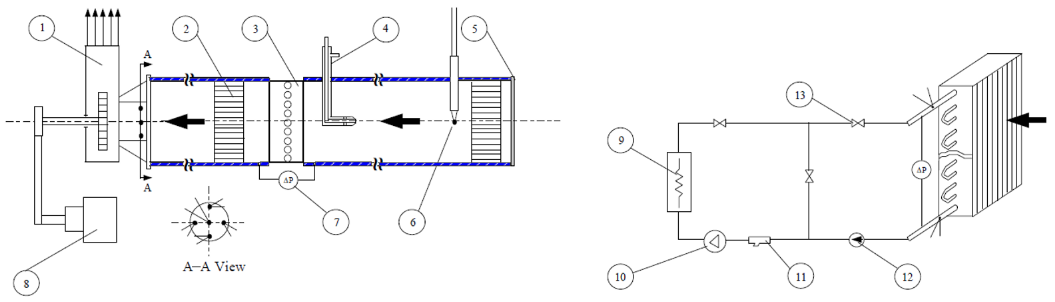


Figure 4: Schematic of a HTX measurement facility: 1-blower; 2-flow straightener; 3-heating coil; 4-pitot-static tube; 5-screen; 6-thermocouple; 7-differential pressure gauge; 8-motor; 9-electrical heater; 10-water pump; 11-strainer; 12-flow meter; 13-valve; Zhao (1995)

Given the relatively small sample size for temperature and flow measurements, it is not suitable to rely on error analysis based on the assumption of a normal distribution. Instead, the more appropriate approach is to utilize the Student's t-distribution. The sample standard deviation is defined in formula 22 Tatara and Lupia (2011). The confidence level is determined using the Student's t-distribution, as indicated by formula 23 Tatara and Lupia (2011).

$$S = \sqrt{\frac{\sum(X_n - X_m)^2}{Ns - 1}} \tag{22}$$

$$S_t = t_{\vartheta - \%} S \tag{23}$$

The final combined uncertainty, denoted as U_n, is obtained by combining the random uncertainty, described by formula 23 Tatara and Lupia (2011), and the predictable part of deviation. The predictable part, so-called bias B, accounts for factors such as instrument calibration and data acquisition errors. The equation 24 Tatara and Lupia (2011) below incorporates the sensitivity coefficient θ, which takes into consideration the effects of uncertainty dependencies. When calculating a quantity based on measured variables, such as the head dissipation of a heat exchanger, the final combined uncertainty is expressed using equation 25 Tatara and Lupia (2011) Moffat (2014).

$$Un = \sqrt{\sum(\theta B)^2 + \sum(\theta S)^2} \tag{24}$$

$$Un_{\dot{Q}} = \sqrt{(\theta Un)^2} = \sqrt{[\frac{\partial \dot{Q}}{\partial \dot{m}} Un_{\dot{m}}]^2 + [\frac{\partial \dot{Q}}{\partial c_p} Un_{c_p}]^2 + [\frac{\partial \dot{Q}}{\partial T_{in/out}} Un_{T_{in/out}}]^2} \tag{25}$$

2.3 Vehicle AC Condenser

As aforementioned, the condenser is a crucial component of any Air Conditioning (AC) system, responsible for dissipating the heat from the refrigerant fluid into the surrounding air. The AC condenser is a specific type of heat exchanger (HTX), which involves both dual-phase (condensing) and single-phase heat transfer between the refrigerant (auxiliary fluid) and the surrounding air (primary fluid) in its operation. From a geometric perspective, a typical condenser is a multiple pass, tube-and-fins, micro-channel cross-flow heat exchanger (Figure 5). Pervaiz et al. (1997).

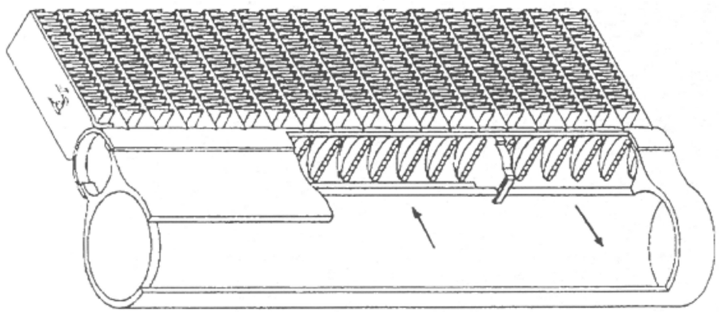


Figure 5: Detail of an AC condenser Pervaiz et al. (1997)

As illustrated in the Figure 5, the refrigerant flows through micro-channels in multiple passes. The flow direction in a typical condenser with multiple passes is demonstrated in Figure 6 Ye et al. (2009). In some cases, a baffle may be replaced by a distributor, which permits some flow between separate tanks through a hole in the baffle. These distributors contribute to a more uniform temperature distribution within an AC condenser Ye et al. (2009). From a thermodynamic perspective, the AC condenser is a component where part of the reverse Rankine Cycle takes place. In the AC circuit, organic substances are used as auxiliary fluids to meet specific application requirements, particularly those related to low boiling points. This variation of the Rankine Cycle is commonly known as the Organic Rankine Cycle (ORC). The ORC is based on the Ideal Rankine Cycle (IRC), where expansion is isentropic and evaporation/condensation processes is isobaric. Figure 7 illustrates the IRC, with the condenser stage taking place between points 2 and 3 Nozicka (2008). However, in reality, the ORC represents a Real Rankine Cycle, where the processes are not completely reversible, and the efficiency of each process (such as

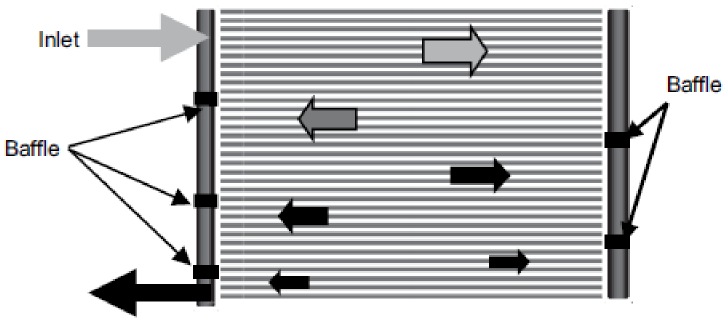


Figure 6: AC condenser flow distribution Ye et al. (2009)

the compressor) becomes a factor. The heat transfer rate of the AC condenser within the cycle is determined by equation 26 Nozicka (2008), Dai et al. (2009), Moran et al. (2014).

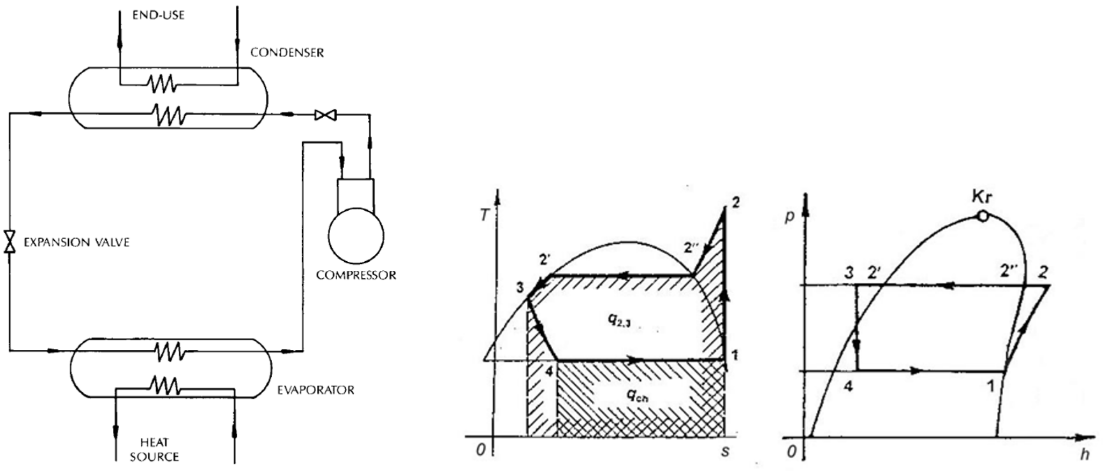


Figure 7: Function (a) Parise (1986) and thermodynamic scheme (b) of reverse ideal Rankine Cycle Nozicka (2008)

$$\dot{Q} = \dot{m}(h_2 - h_3) \tag{26}$$

The efficiency Equation 27 for a phase change AC condenser is derived by modifying equation 1. In this modification, the inlet fluid temperatures are replaced with the inlet fluid enthalpy Liang et al. (2015). The C_{min} should be in the case calculated as shown in the Equation 28 Liang et al. (2015).

$$\dot{Q} = \epsilon C_{min}(h_{in,aux} - h_{in,prim}) \tag{27}$$

$$C_{min} = min\{\dot{m}_{auxiliary}; \frac{\dot{m}_{primary}c_{p,primary}}{\frac{h_{in,aux}-h_{in,prim}}{T_{in,aux}-T_{in,prim}}}\} \tag{28}$$

2.3.1 Calculations

The 3D CFD codes mentioned earlier in Section 2.1.3 do not have a dedicated computational model specifically utilized for AC condensers. In the automotive industry, AC condensers are often simplified by using a simple heat source. However, this simplified approach fails to capture the spatial distribution of refrigerant flow, phase changes, local heat transfer, and their effects on airflow. Nevertheless, heat exchanger models available in software such as $Simcenter\ STAR-CCM+$ or $Ansys\ Fluent$ can be potentially used to simulate refrigerant flow, including phase changes, by incorporating user-defined codes. An example of an implemented AC condenser model in $Simcenter\ STAR-CCM+$ can be found in Pervaiz et al. (1997). The model proposed in the work by Pervaiz et al. (1997) incorporates phase change by employing an empirical expression for the heat transfer coefficient (α) as a function of refrigerant quality (χ).
Specialized models and approaches are available for calculating the performance of AC condensers, with many of them relying on 1D methods such as the $\epsilon - NTU$ approach, the logarithmic mean temperature difference ($LMTD$) method, or the arithmetic mean temperature difference ($AMTD$) method. These methods take into account specific condenser specifications, including micro-channel construction and refrigerant phase change, by incorporating correction factors. The determination of the (convective) heat transfer coefficient (α) plays a crucial role in these approaches. The overall or convective is obtained through measurements and correlation relations, such as Wilson, Colburn, and others Fernandez-Seara et al. (2007). Subsequently, the obtained convective heat transfer coefficient (α) is employed in the 1D models, such as the $\epsilon - NTU$ method (equations 2 and 3) Kays and London (1984), Incropera et al. (2011), or alternative methodologies. A comprehensive summary of the 1D modeling approaches for AC condensers can be found in the work by Pavlu (2012), which also includes the development and comparison of a model with extensive experimental measurements. Furthermore, the 1D models have been extended to capture the single-phase as well as phase-change region of the AC condenser. The extension of $\epsilon - NTU$ approach-based models incorporates a relation for the phase-change (two-phase) region, as indicated by equation 29 Incropera et al. (2011). This relation is based on the fact that Equation 6 is equal to zero during phase change. The combination of the phase-change and single-phase regions is implemented and compared with experimental measurements in studies such as Admiraal and Bullard (1993), Bansal and Purkayastha (1998), Huang et al. (2012), and Liang et al. (2015). Detailed investigations on the condensation heat transfer coefficient have been performed by Garcia-Cascales et al. (2010).

$$\epsilon = 1 - exp(-NTU) \tag{29}$$

The $LMTD$ and $AMTD$ based methods are commonly employed for modeling phase change heat exchangers (HTX), including condensers. In Equation 12 Kays and London (1984), the geometry correction factor F_G for the condenser is equal to unity in the case of phase change Saari (2010). A model of this type was developed and compared with experimental measurements by Garcia (1998), which also considered the effect of vapor quality as discussed in McLinden and Radermacher (1987). Iterative models encompassing the entire refrigerant circuit have been presented in Parise (1986), Rice and Sand (1990), Miyara et al. (1993), and Haselden and Chen (1994). These iterative models incorporate both single-phase and two-phase regions by accounting for varying heat transfer coefficients based on refrigerant quality (χ).

The number of passes, which refers to the construction of the AC condenser, is an important factor that determines the spatial flow of the refrigerant within the condenser (Figure 6 Ye et al. (2009)). Consequently, it significantly impacts the spatial distribution of results and overall heat transfer, as investigated in Silaipillayarputhur and Al-Mughanam (2018). An analytical solution considering multiple passes based on a known NTU value was developed by Bes (1996). In Martinez-Ballester and Gonzalvez-Macia (2013) and Gonzalvez-Macia (2013), a numerical model incorporating an empirical heat transfer coefficient was applied and compared against experimental measurements. The predicted condenser heat capacity and outlet refrigerant temperature exhibited accuracy levels below 5% and $2K$, respectively. Other models have been developed by Stoitchkov and Dimitrov (1998), Schwentker et al. (2006), Singh et al. (2009), Martínez-Ballester et al. (2011), and Liang et al. (2015). A comprehensive comparison of condenser models, incorporating different heat transfer coefficients from the literature, was performed by Tuo et al. (2012), demonstrating good accuracy. Detailed models that consider refrigerant flow were developed by Zou et al. (2014) and Hu et al. (2015). These models predict flow distribution between the tubes, local refrigerant quality, and are compared against experimental measurements.

The condenser model implemented in the $Gamma\ Technologies\ GT-SUITE\ COOL3D$ utilizes regression analysis to fit the measured performance data by adjusting coefficients in Equation 32 Gamma Technologies (2014). This Equation is used in the expression for the Nusselt number as shown in Equation 31 Gamma Technologies (2014). For single-phase heat transfer, the heat transfer coefficient is defined by Equation 30 Gamma Technologies (2014). In a study conducted by Jha and Shaik (2016), this model was employed and compared against experimental data. The results showed an accuracy of approximately $\pm 2.3\%$ for overall heat dissipation. However, when the model was applied to different condenser dimensions, which were not used in the initial parameter fitting, the accuracy decreased to around $\pm 8.1\%$.

$$\alpha = 0.23 Re^{0.8} Pr^{0.4} \frac{\lambda}{L} \tag{30}$$

$$Nu = 0.23 Re^{0.8} Pr^{0.4} \tag{31}$$

$$\alpha = Nu[(1-\chi)^{0.8} + \frac{3.8\chi^{0.76}(1-\chi)^{0.04}}{Pr^{0.38}}]\frac{\lambda}{L} \tag{32}$$

Additionally, the 1D models are often coupled with 3D CFD simulations of the entire vehicle to capture three-dimensional effects. An example of such coupling, utilizing the commercial tools *KULI soft* and *Ansys Fluent*, for a simplified cooling pack can be found in Long et al. (2014). The Wang et al. (2016) work contains a 1D model of an AC condenser using *MathWorks Matlab* software with $\epsilon - NTU$ approach, including transient simulation. It should be noted that *MathWorks Matlab* provides a 1D model for AC condensers, which includes a simplified spatial distribution. The schematic in Figure 8 (Wang et al. (2016)) illustrates the computational cells used in the AC condenser model, while the figure below in Figure 9 (Wang et al. (2016)) shows the 3D CFD results from overall underhood coupling simulation.

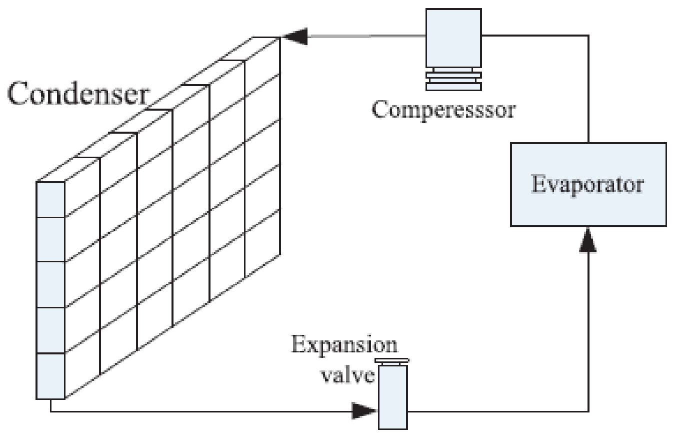


Figure 8: Schematic diagram of air conditioning modelWang et al. (2016)

In Shah (2018), a coupling of 1D and 3D solvers using the tools *Magna KULI software* and *Simcenter STAR − CCM+* is demonstrated, with a focus on the condenser. The condenser model implemented in the *Magna KULI software* utilizes the Nusselt number expression given by Equation 33. The correlation coefficients $Coeff$, $Coeff1$, and $Coeff2$ in the expression are functions of the Reynolds number and the type of heat exchanger (such as condenser or evaporator). Shah (2018)

$$Nu = Coeff Re^{Coeff1} Pr^{Coeff2} \tag{33}$$

2.3.2 Measurement

An experimental setup utilized for measuring the heat performance of an AC condenser is shown in Figure 10 Zheng et al. (2014). The typical AC condenser measurement is

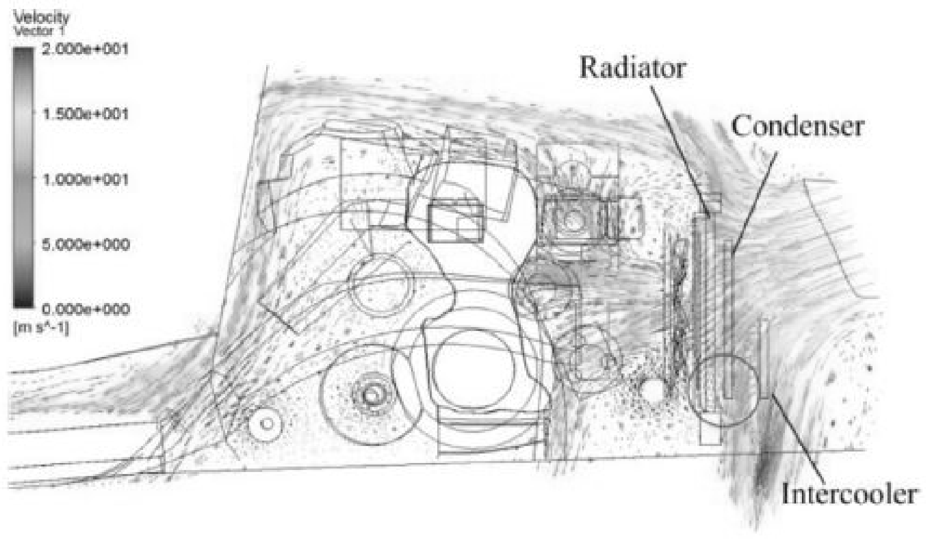


Figure 9: 3D CFD underhood with condenser coupled with 1D Wang et al. (2016)

focused on assessing the overall thermal performance parameters such as heat dissipation, inlet/outlet temperatures, and pressure losses. Important for AC condenser modeling is the measurement of the convective condensation heat transfer coefficient. The Wilson plot method and its variations (e.g. Colburn) are commonly used for this purpose Fernandez-Seara et al. (2007). These methods involve fitting the experimental data with correlation equations, such as equations 33 and 31 Shah (2018) Gamma Technologies (2014). Extensive measurements were conducted in Vist and Pettersen (2004) or Zou and Hrnjak (2013), specifically focusing on the vapor fraction within individual tubes under various conditions. Comprehensive convective condensation heat transfer measurements for micro-channels could be found inCavallini et al. (2003) or Bandhauer et al. (2006).

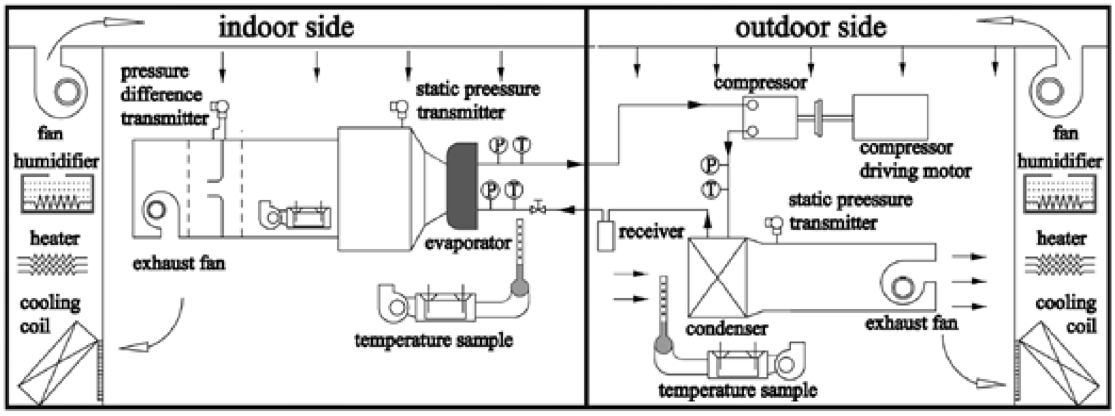


Figure 10: A layout of the experimental system apparatus Zheng et al. (2014)

As aforementioned in Section 2.3.1, a crucial aspect of condenser measurements involves gathering heat transfer coefficients or correction factors for condenser models Dittus and Boelter (1985) Kim et al. (1999) Kim and Bullard (2002). Detailed measurements of the heat transfer coefficient during phase change (within tube specimens) are described in Panchal et al. (1992). The suggested heat transfer coefficient from Panchal et al. (1992) is utilized in a heat pump simulation model and compared with the overall pump measurements in Ouazia and Snelson (1994). Another example of experimental heat transfer coefficient measurements can be found in Kondou et al. (2014), or in work of Park and Jacobi (2009), which includes multiple conditions and specifications of the Colburn factor. Detailed measurements of internal refrigerant flow and quality were performed by Ye et al. (2009), with a specific focus on enhancing condenser efficiency through modifications of baffles in the tanks into distributors (i.e., baffles with orifices) (Figure 6) Ye et al. (2009).

3 Objectives

The primary aim of this research is to enhance the accuracy of AC condenser modeling within a complex full-vehicle 3D CFD simulation, where implementing a detailed heat exchanger model is not feasible. The proposed model seeks to improve the prediction of local heat transfer between the refrigerant and cooling air within the AC condenser. By achieving enhanced heat transfer prediction, the accuracy of airflow simulation should be improved as well, leading to more precise predictions for other vehicle components and fluids, such as coolant and oil. Moreover, the proposed model and measurement approach simplify and generalize the current procedures (outlined in Section 2.3) used for calculating local heat transfer.

The enhancement of the proposed model relies on accurately defining local heat transfer based on the phase or quality of the refrigerant. This is achieved by locally defining the $\epsilon - NTU$ relationship, similar to the approaches adopted by other researchers discussed in Section 2.3.1. However, unlike the current 1D AC condenser calculation methods, which rely on the estimation of convective heat transfer coefficients, the suggested model directly measures the overall heat transfer coefficient. This integration and modification of conventional 3D CFD methods (as described in Section 2.1.3) and the latest 1D AC condenser models result in a more comprehensive approach.

Consequently, the measurement of the overall heat transfer coefficient in the suggested model combines and modifies the methodologies used for single-phase heat exchangers (as discussed in Section 2.2) and phase change heat exchangers (as detailed in Section 2.3.2). The gathering of input data for the suggested AC condenser model is expected to be more general and applicable compared to the current state-of-the-art approaches.

By incorporating these advancements, the proposed model aims to improve the accuracy and generalizability of AC condenser modeling, enabling more precise predictions of heat transfer in various refrigerant phases and enhancing the overall performance of the system.

To successfully achieve the aforementioned objectives, several critical tasks need to be accomplished. The first task involves developing a single-phase heat exchanger (HTX)

model that incorporates the current state-of-the-art approaches. This model will serve as the foundation for further extensions to account for phase change phenomena. The second task involves suggesting and conducting measurements to gather appropriate input data required for the developed AC condenser model. These measurements should encompass relevant parameters that influence heat transfer, such as temperature differentials, flow rates, and pressure. In addition to model development and data gathering, a crucial task is to verify the proposed AC condenser model through physical measurements.
The key tasks are listed below:

- **Propose the AC condenser model**
  The model should be able to capture local refrigerant quality and select appropriate heat transfer (single vs. dual-phase).
  Define the domain discretization, mass and heat. transfer
  Section 4, 5, 6, and 7.

- **Create single-phase HTX calculation algorithm**
  Based $\epsilon - NTU$ approach
  Verify model via comparing with CFD code from Section 2.1.3
  Section 8.1

- **Create HTX calculation algorithm including phase change**
  Section 8.2

- **Test equipment development**
  The test equipment should be capable of input data gathering as well as of verification measurement
  Section 9, 10 and 11

- **Verification**
  The developed AC condenser model results should be verified by comparing with measurement
  Section 12 and 13

4 Suggested Model Philosophy

The proposed model integrates the implementation of the $\epsilon - NTU$ approach used in *Ansys Fluent* (described in Section 2.1.1 and 2.1.3) with the recent advancements made to 1D AC condenser models (Section 2.3.1). Consequently, it requires modification of standard input data ($Q - Table$) measurement.

The primary concept behind the suggested and developed model is the incorporation of the spatial distribution of the $\epsilon - NTU$ relationship to account for the refrigerant phase in an AC condenser. Figure 11 (a) Nozicka (2008) illustrates a temperature-entropy and pressure-enthalpy (T-s and p-h) diagram with an AC circuit. The highlighted section between points 2 and 3 refers to the AC condenser. A schematic representation of the refrigerant flow within the AC condenser is shown in Figure 11 (b), highlighting the relevant sections. The red-marked region indicates the gas phase of the refrigerant. At the saturation temperature denoted as $2''$ in the diagram, a phase change occurs, and the refrigerant transition into a dual-phase state starts, represented by the orange color. The $\epsilon - NTU$ relationship needs to be modified accordingly, utilizing the dual-phase Equation 29 Incropera et al. (2011), Liang et al. (2015). Once the refrigerant dissipates its latent heat into the surrounding air, the condensation process is completed, and it fully transforms into a liquid phase (blue region). This blue section is referred to as overcooling. Consequently, the $\epsilon - NTU$ relationship should revert to Equation 7 Incropera et al. (2011), employing liquid input characteristic data ($Q - Table$). Within each section (red, orange, and blue), specific measured $Q - Tables$ for overall heat transfer should be utilized to accurately characterize the heat transfer process.

The proposed model follows an iterative approach similar to the works of Rice and Sand (1990), Admiraal and Bullard (1993), or Bansal and Purkayastha (1998) (Section 2.3.1). Within the same section, it was mentioned that Pervaiz et al. (1997) implemented a spatial distribution of heat transfer coefficient in the $Simcenter\ STAR - CCM+$ model. Additionally, Jha and Shaik (2016) employed a similar approach using the $Gamma\ Technologies\ GT - SUITE\ COOL3D$ model Gamma Technologies (2014). However, all these models rely on convective heat transfer coefficients obtained from literature, specimen measurements, or fitted correlation equations (fitting to the whole HTX measurement).

The proposed method directly measures the overall heat transfer coefficient (respectively the $Q - table$), which encompasses various heat transfer mechanisms such as tube conduction, phase-change convection, and air-side convection etc. This approach effectively reduces the number of required input parameters, as demonstrated in Figure 12.

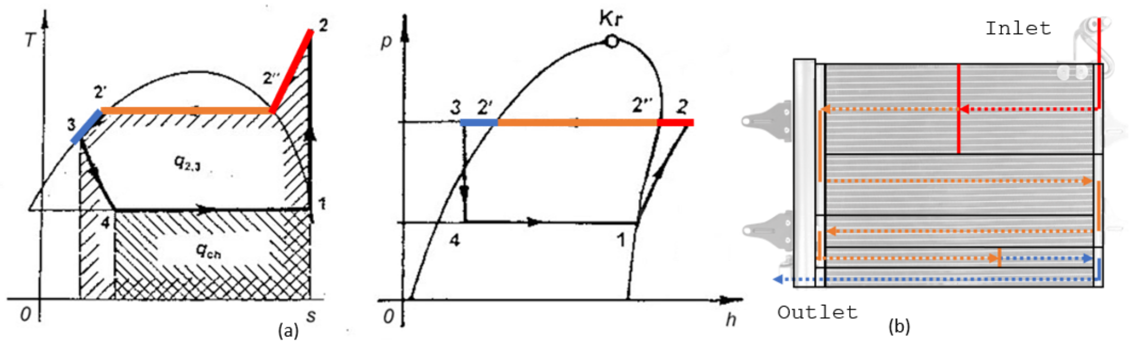


Figure 11: (a) T-s and p-s diagrams Nozicka (2008); AC condenser sections (b)

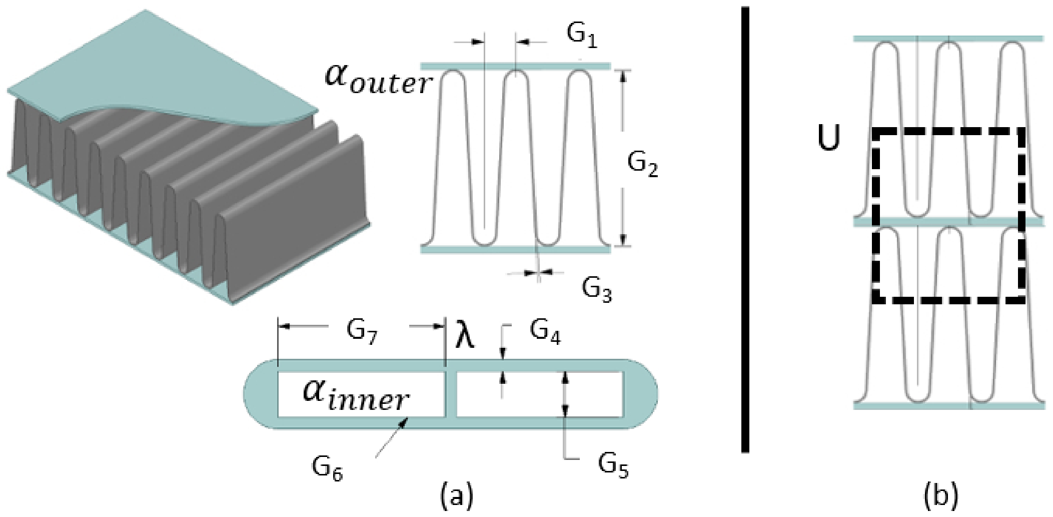


Figure 12: (a) *Gamma Technologies GT−SUITE COOL3D* model inputs Gamma Technologies (2014); suggested model input (b)

5 Computational Domain and Discretization

The computational domain, shown in Figure 13 (a), represents an AC condenser in the standard Cartesian coordinate system for vehicles J902 (2011). The Cartesian axis indexing and the structural mesh of cell volumes are illustrated in Figure 13 (b). The domain (simple block) is discretized using a uniform division with a constant interval, similar to the approach shown in Figure 8 Wang et al. (2016). Consequently, each cell (control volume) has identical size and an equal distance between the centroids of adjacent cells. In Figure 13 (b), an example cell is shaded, with the points representing the cell centroids. The dimensions of each cell are defined as dx, dy, and dz. It is important to note that the cell centroid represents a simple 1D element, wherein the values denoting the cell volume are calculated and stored.

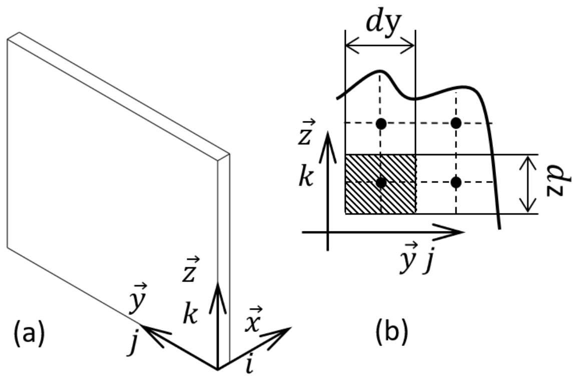


Figure 13: Computational domain (a) and mesh details (b)

6 Mass Flow Rate

The suggested condenser model focuses solely on predicting heat transfer and does not simulate fluid flow. Therefore, the mass flow rates of the fluids are considered as boundary conditions based on the measured data. However, it is possible to couple or integrate the suggested condenser model with a 3D CFD code, where the fluid flow can be simulated (Section 2.3.1). This would allow for a more comprehensive analysis of the condenser performance by considering the simulated mass flow rates.

6.1 Primary Fluid Flow

The primary flow, which corresponds to the airflow, is considered a measured boundary condition and is obtained from the test device described in Section 9.2. The device measures the magnitude of the airflow velocity and the temperature in the direction parallel to the heat exchanger surface (the x direction, as defined in Section 13).

To calculate the mass flow rate of the airflow, the ideal gas law for dry air (Nozicka (2008)), the measured ambient pressure, and the airflow frontal area are utilized. The air mass flow rate is determined by multiplying the measured velocity magnitude by the airflow frontal area, while assuming a negligible pressure difference between the ambient pressure and the pressure at the inlet surface of the AC condenser.
The uniformity of airflow distribution plays a significant role in the performance of the heat exchanger. More in-depth analysis on the importance of airflow uniformity distribution can be found in Moradi et al. (2020) and Schmid et al. (2021).

$$\dot{m}_{cell} = \frac{p_{ambient}}{R_{specific} \cdot T_{cell}} A_{cell} \cdot u_{i\_cell} \tag{34}$$

6.2 Auxiliary Fluid Flow

The refrigerant fluid flow is modeled as a one-dimensional tube flow in the direction of the passes (i.e., through the micro-channel tubes). The mass flow rate of the refrigerant liquid at the AC condenser inlet is prescribed based on the measured value obtained from the experimental data.
Figure 14 illustrates the distribution of the auxiliary fluid (refrigerant) flow between the passes. Equation 35 is used to calculate the flow rate for each computational cell. The defined refrigerant flow rate is divided equally among the cells within the cross-sectional area of a particular pass.

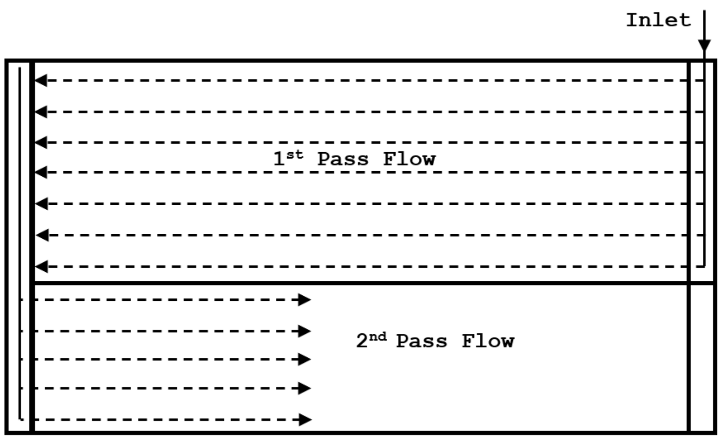


Figure 14: The refrigerant flow distribution scheme

$$\dot{m}_{cell} = \frac{\dot{m}_{aux}}{[Xcells \cdot Zcells]_{Pass}} \tag{35}$$

The distribution of flow within each pass is assumed to be uniform among the parallel tubes. This assumption is based on the Bernoulli equation (Nozicka (2008)). However, in real AC condensers and cross-flow heat exchangers in general, the flow distribution within the pass tubes can be nonuniform due to three-dimensional effects. The uniformity of the auxiliary fluid flow is an important design parameter in heat exchangers and is balanced against pressure restrictions Zou and Hrnjak (2014). The uniformity of flow is influenced by factors such as tank and tube shape, fluid properties, and mass flow rate. The uniformity of the refrigerant flow in the evaluated condenser (Section 9.3) is deemed valid based on the geometry of the tanks and tubes, particularly the size ratio between these components. This conclusion is supported by studies such as Minqiang et al. (2009), Mohammadi et al. (2013), Pistoresi et al. (2015), and Wei et al. (2015). However, in the case of nonuniform flow distribution, an alternative approach such as the *Dual Cell* or *Dual Stream* method described in Section 2.1.3, as applied in Pervaiz et al. (1997), could be employed. Nonetheless, the proposed heat transfer approach remains unchanged.

7 Heat Transfer Model

Heat transfer between primary (air) and auxiliary (refrigerant) fluid in the AC condenser is modeled by $\epsilon - NTU$ approach (Section 2.1.1 and 2.3.1) as discussed in Sections 3 and 4. This approach assumes that both fluids are unmixed, meaning that heat transfer occurs only in one direction parallel to the airflow. This assumption is based on the thermodynamic definition of unmixed fluids Taborek (1983). Additionally, it is assumed that the specific heat capacity and density of the fluids are constant, based on the average temperature. This assumption simplifies the calculations and is commonly used in heat transfer analysis.

The schematic for obtaining the NTU value from measured data (represented by the $Q - Table$) is shown in Figure 15. In this measurement process, multiple flow rates of fluids are typically tested in order to gather a range of heat exchanger characteristic data. During the measurement, the inlet temperature for each fluid is kept constant. For the suggested model, it is necessary to measure the $Q - Table$ for each phase of the refrigerant. This means that the heat transfer performance needs to be evaluated separately for both the single-phase and dual-phase regions of the AC condenser. More detailed information regarding the measurement procedure and data gathering can be found in Section 11.

The first step in the calculation is to determine the heat exchanger efficiency (ϵ) using Equation 1 Kays and London (1984). Once the efficiency is known, the corresponding NTU value can be calculated using an appropriate $\epsilon - NTU$ relationship. For cases where there is no phase change in the measured data, Equation 7 Incropera et al. (2011) is used. On the other hand, for cases involving phase change, Equation 29 Incropera et al. (2011) is employed. It is important to note that when expressing NTU using Equation 7 Incropera et al. (2011), an iterative approach is required. The developed approach utilizes the Newton-Raphson iterative method, which is similar to the implementation in $AnsysFluent$ (as discussed in Section 2.1.3).

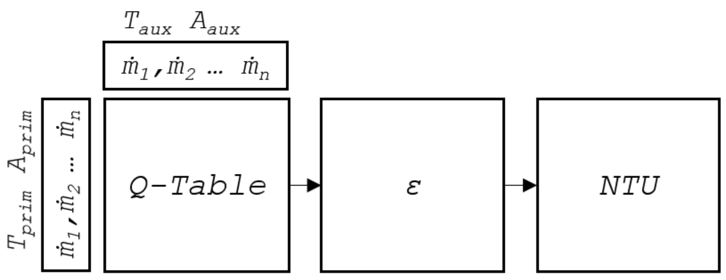

T_{aux} A_{aux}
$\dot{m}_1, \dot{m}_2 \dots \dot{m}_n$
T_{prim} A_{prim}
$\dot{m}_1, \dot{m}_2 \dots \dot{m}_n$
Q-Table
ε
NTU

Figure 15: Measured data processing workflow

Once the *NTU* values are determined for the entire heat exchanger, the next step is to evaluate the operating point for each individual cell within the heat exchanger. In the suggested model, where constant inlet conditions are assumed, the operating point for each cell is calculated only at the beginning of the simulation, as described in Section 8. In the case of changing flow conditions, such as varying inlet temperatures or flow rates, the operating point for each cell would need to be calculated iteratively. However, in the suggested model with constant inlet conditions, this iterative calculation is not necessary, and the operating points can be determined at the start of the simulation based on the initial conditions. The operating point for each cell within the heat exchanger is determined using the following expressions. Firstly, the mass flow rate throughout the cell is scaled to the entire dimension of the heat exchanger, resulting in the *global* flow rate, as defined in Equation 36 Fluent (2009). This global flow rate is used to interpolate the corresponding *global NTU* value for each cell from the *NTU* table, using linear interpolation. Based on the *global NTU* value, the *local NTU* value for each cell is defined using Equation 37, following a similar approach as in Fluent (2009). The efficiency of each cell is then calculated based on the local *NTU* value, utilizing the relations provided in Equation 7 Incropera et al. (2011) for non-phase-change conditions, and Equation 29 Incropera et al. (2011) for phase-change conditions. Finally, the heat dissipated by each cell is determined using Equation 14 Incropera et al. (2011) or for phase-change Equation 27 Liang et al. (2015), and the outlet temperatures are calculated using Equation 15 Incropera et al. (2011). These expressions allow for the evaluation of the heat transfer characteristics and outlet temperatures at the cell level within the heat exchanger.

$$\dot{m}_{cell}^{global} = \dot{m}_{cell} \frac{A_{Q-Table}}{A_{cell}} \tag{36}$$

$$NTU_{cell}^{local} = NTU_{cell}^{global} \frac{V_{cell} C_{min,global}}{V_{global} C_{min,cell}} \tag{37}$$

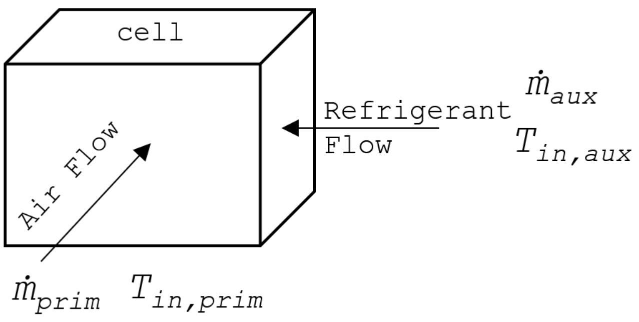


Figure 16: Computational cell flow scheme

7.1 Tanks' Flow Mixing

The assumption of ideal mixing inside the tanks of the AC condenser, and with considered uniform mass flow within the HTX core, leads to the assumption of uniform outlet temperatures from the tanks (Section 14). To calculate the uniform outlet temperature or enthalpy from the tanks, the average of the inlet temperature or enthalpy of the tanks is taken. Similarly, the refrigerant phase, specifically the quality χ, is calculated based on the average inlet values for each tank. The refrigerant vapor quality χ ranges from 0 to 1 and is generally defined by Equation 38 and illustrated in Figure 17 Nozicka (2008). However, in the AC condenser model, the vapor quality χ is not treated as a continuous parameter, but rather as discrete ranges/values defined in Equation 39. Based on the value of χ, the appropriate heat transfer calculation is applied in the model.

$$\chi = \frac{m_{vapour}}{m_{vapour} + m_{liquid}} \tag{38}$$

$$Q - Table = \begin{cases} Liquid\ Phase, & \text{if } \chi = 0. \\ DualPhase, & \text{if } 0 < \chi < 1. \\ GasPhase, & \text{if } \chi = 1. \end{cases} \tag{39}$$

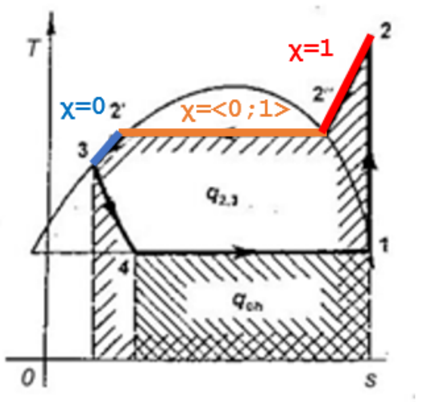


Figure 17: T-s diagram with vapour quality Nozicka (2008)

8 Algorithm

The iterative solver adjusts the auxiliary fluid inlet temperature of the heat exchanger until multiple conditions are met. This adjustment is carried out using the bisection method, as described in Stepan (2012). Similar approaches have been implemented by Rice and Sand (1990), Admiraal and Bullard (1993), or Bansal and Purkayastha (1998) (Section 2.3.1).
The first subsection of this chapter presents a simple heat exchanger model based on the implementation of *Ansys Fluent* Fluent (2009) (Section 2.1.3). This model is compared with other commercial tools in Section 2.1.3 to initially verify the code.
The second subsection of the chapter describes the proposed algorithm for calculating the AC condenser model, which has been developed in this thesis.

8.1 Simple Heat Exchanger

The algorithm for the heat exchanger (HTX) model is described in Figure 18, where the indexes are defined as shown in Figure 13. This model is based on a fixed heat rejection and an iterative adjustment of the auxiliary fluid inlet temperature. This approach is more suitable for typical vehicle applications, such as known heat rejection from an engine.
In Figure 8, relation (a) represents the HTX with single-phase flow, while relation (b) represents the HTX with exclusive dual-phase (phase change) flow. It is important to appropriately combine these two scenarios in the AC condenser model, as both single-phase and dual-phase regimes occur in real condensers (Section 8.2). The single-phase HTX model was implemented in *GNU Octave* 5.2.0 and verified against commercial tools in the following subsection.

8.1.1 Verification

To verify the implementation of the simple heat exchanger model without phase change, a test case is compared against commercial CFD tools, namely *Ansys Fluent* and *Simcenter* $STAR-CCM+$.
As mentioned earlier, this thesis focuses on heat transfer, and thus, the flow in the test case is assumed to be inviscid. For the sake of simplicity, no pressure drop is considered throughout the domain. The test case involves a conventional air-to-water heat exchanger (HTX) selected for verification.
The dimensions of the HTX test case domain are $(x, y, z) = (0.04, 0.7, 0.6)m$. The domain is numerically decomposed into $(8, 70, 60)$ grid cells along the coordinate system axes. The auxiliary fluid flow direction is aligned with the positive z-axis, while the primary flow direction is along the positive x-axis, indicating a single-pass HTX configuration.
The heat performance characteristics, represented by the $Q-Table$, of the selected HTX

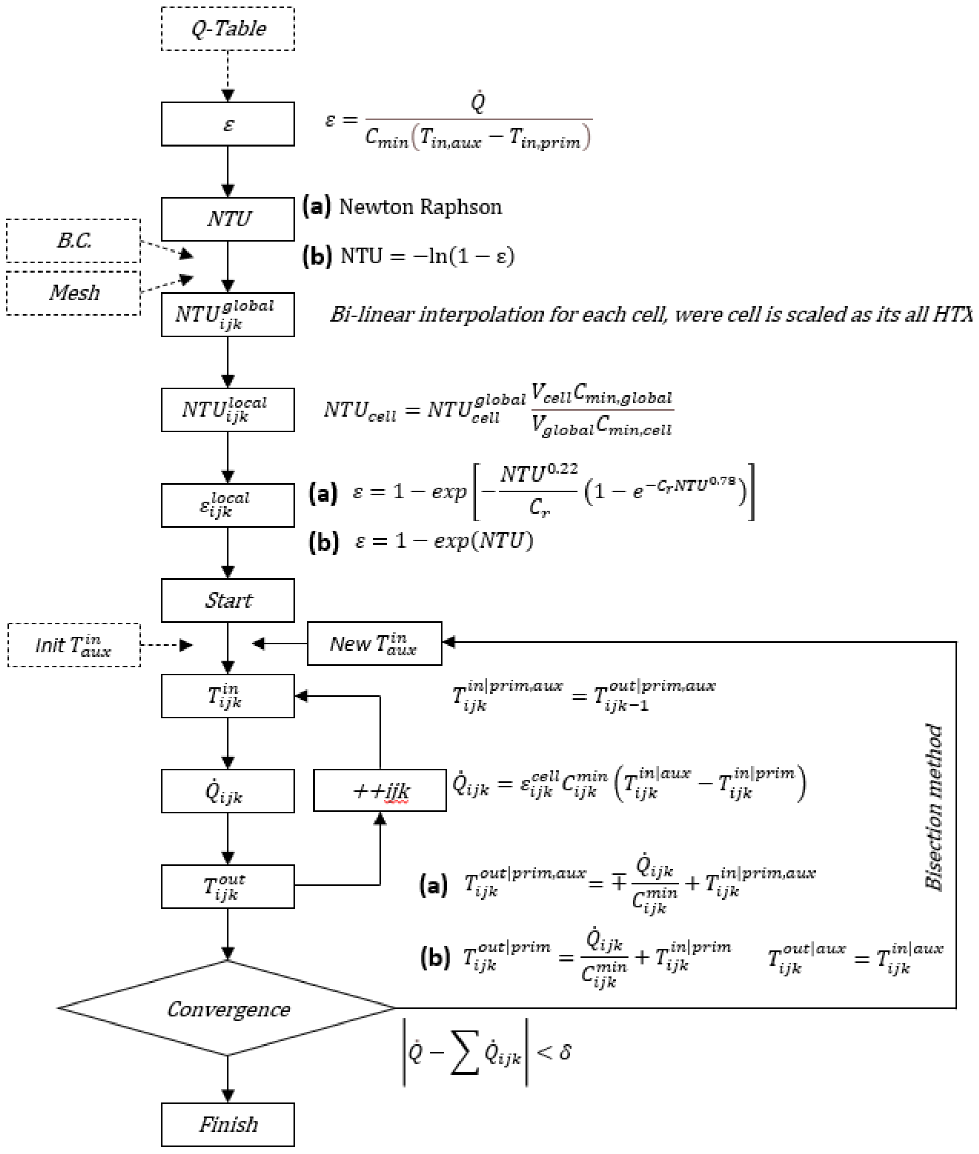

Q-Table
ε
$\varepsilon = \frac{\dot{Q}}{C_{min}(T_{in,aux} - T_{in,prim})}$
NTU
(a) Newton Raphson
(b) NTU = $-\ln(1-\varepsilon)$
B.C.
Mesh
NTU_{ijk}^{global}
Bi-linear interpolation for each cell, were cell is scaled as its all HTX
NTU_{ijk}^{local}
$NTU_{cell} = NTU_{cell}^{global} \frac{V_{cell} C_{min,global}}{V_{global} C_{min,cell}}$
$\varepsilon_{ijk}^{local}$
(a) $\varepsilon = 1 - exp\left[-\frac{NTU^{0.22}}{C_r}\left(1 - e^{-C_r NTU^{0.78}}\right)\right]$
(b) $\varepsilon = 1 - exp(NTU)$
Start
Init T_{aux}^{in}
New T_{aux}^{in}
T_{ijk}^{in}
$T_{ijk}^{in|prim,aux} = T_{ijk-1}^{out|prim,aux}$
$\dot{Q}_{ijk}$
$++ijk$
$\dot{Q}_{ijk} = \varepsilon_{ijk}^{cell} C_{ijk}^{min} \left(T_{ijk}^{in|aux} - T_{ijk}^{in|prim}\right)$
T_{ijk}^{out}
(a) $T_{ijk}^{out|prim,aux} = \mp \frac{\dot{Q}_{ijk}}{C_{ijk}^{min}} + T_{ijk}^{in|prim,aux}$
(b) $T_{ijk}^{out|prim} = \frac{\dot{Q}_{ijk}}{C_{ijk}^{min}} + T_{ijk}^{in|prim}$ $T_{ijk}^{out|aux} = T_{ijk}^{in|aux}$
Bisection method
Convergence
$\left|\dot{Q} - \sum \dot{Q}_{ijk}\right| < \delta$
Finish

Figure 18: Simple heat exchanger. (a) single-phase variant (b) dual-phase variant

are shown in Figure 1. The $Q - Table$ performance data was obtained using an auxiliary fluid flow temperature of $388.71K$ and a primary fluid flow temperature of $322.04K$ Fluent (2009).

Table 1: Heat exchanger performance characteristic data Fluent (2009)

Heat rejection (W)			
Coolant flow (kg/s)	2.535	3.169	3.803
Air Flow (kg/s)	-	-	-
0.567	26187	26639	26494
0.945	40891	41355	41677
1.512	56177	57128	57792
2.268	70569	72143	73249
3.024	81529	83677	85196
3.780	90793	93501	95428

The test case simulations were conducted with a heat dissipation of $40.0W$ within the HTX. The auxiliary fluid flow rate was set to $3.169kg/s$, and the primary fluid inlet temperature to $319.15K$.
For the auxiliary fluid, water-like material properties were used, including a density of $1000kg/m^3$ and a specific heat capacity of $4000J/kgK$. The properties of the primary fluid are based on dry air.
Two different air inlet velocity profiles were tested, as shown in Figure 19. Profile A represents a uniform air inlet velocity magnitude, while Profile B represents a sinusoidal function-based velocity profile.

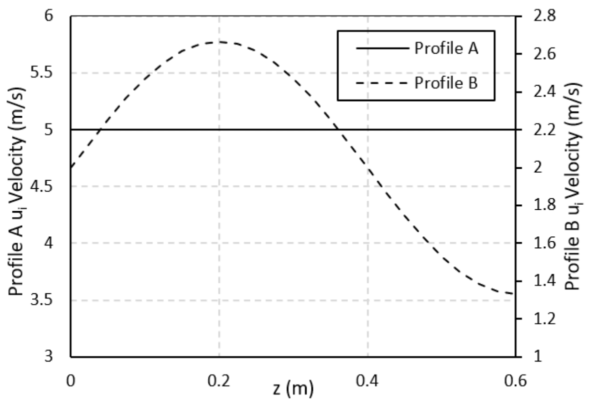


Figure 19: Air inlet velocity profiles

The verification results are summarized in Figure 20, where a comparison is made between the outcomes obtained from the implementation in *GNU Octave* 5.2.0. and the commercial tools (*Ansys Fluent* and *Simcenter* $STAR-CCM+$). This comparison is based on the air outlet temperature contour and the water inlet temperature. It is observed that the air temperature contour profile and the water inlet temperatures show a satisfactory agreement with the results obtained from the CFD codes. As a result, the implementation of the simple HTX in *GNU Octave* 5.2.0. is assumed valid for the development of the AC condenser model (Section 8.2).

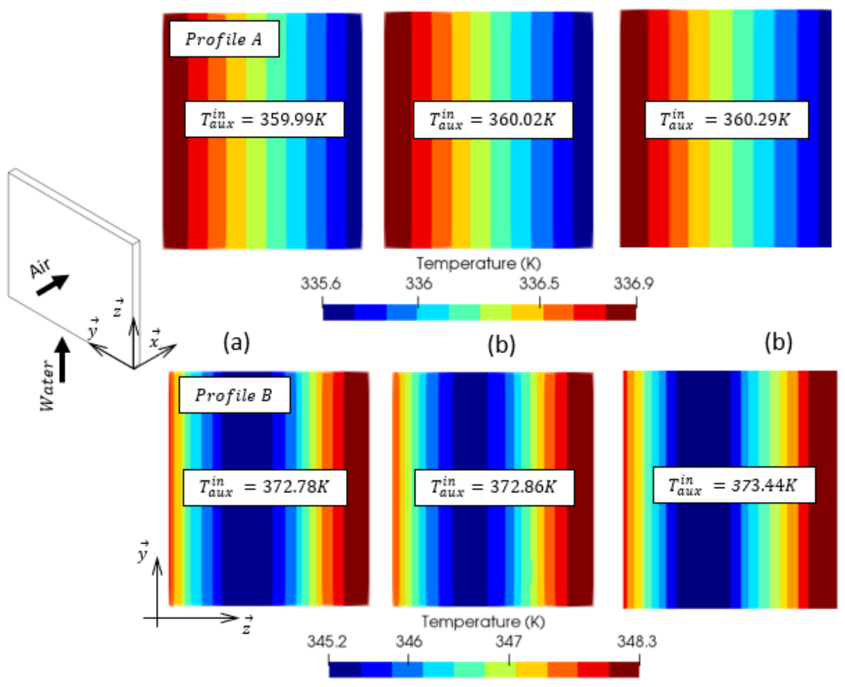


Figure 20: Result for Octave (a), Fluent (b), and STAR-CCM+ (c)

8.2 AC Condenser

The proposed AC condenser model algorithm is presented in Figure 21. This model combines the single-phase and dual-phase heat exchanger (HTX) models based on input data and refrigerant characteristics, as shown in the T-s diagram shown in Figure 17 Nozicka (2008).

The AC condenser model iteratively determines the refrigerant subcooling, phase-change, and overcooling regions based on prescribed boundary conditions. It applies the appropriate $\epsilon - NTU$ relation and utilizes the measured $Q - Table$ for each region, as described in Section 4.

The specific script for implementing the AC condenser model in *GNU Octave* 5.2.0. can be found in Appendix D.

9 Measurement Devices

The selection of devices used for the measurements should align with the requirements for gathering input data for the $Q - Table$ as well as for verification measurements (Section 2.2, 2.3.2, 3, 4, and 8.2). This section describes the developed test bench, including sensor specifications, and provides a description of the tested AC condenser.

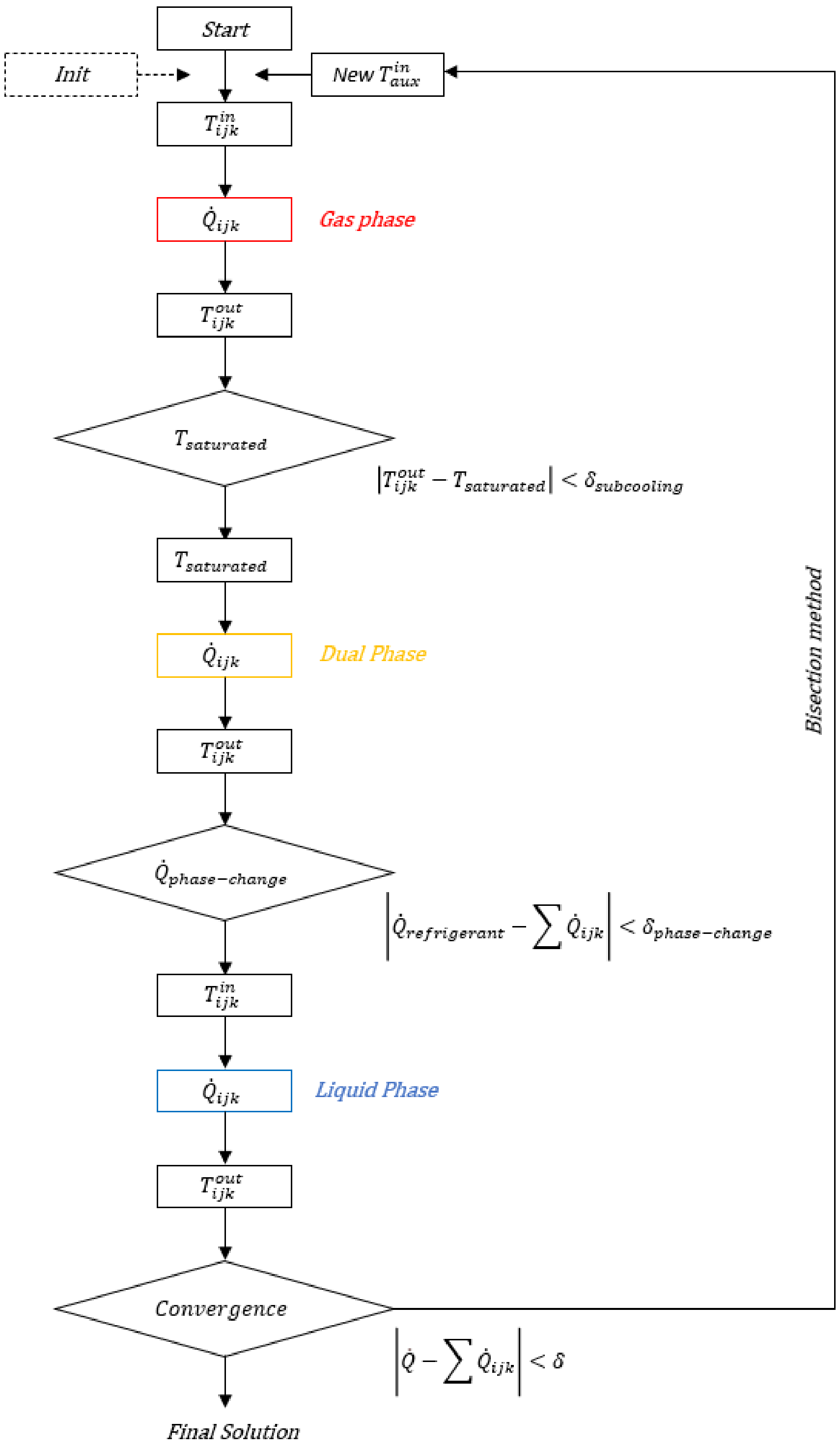


Figure 21: Proposed AC condenser model

It is important to note that in the case of industrial applications, the test procedure needs to be fine-tuned to achieve more accurate and repeatable results. Additionally, the test setup should be capable of operating under a wider range of conditions to meet the requirements of industrial applications.

9.1 Test Equipment

The test equipment schematic is presented in Figure 22, showing a typical automotive AC thermodynamic circuit (Figure 7 Nozicka (2008)). The test equipment consists of three distinct and separate sections. The first section encompasses the refrigerant circuit, which includes the tested AC condenser (highlighted in red), evaporator, compressor, and thermal expansion valve. Pressure sensors and a flow meter are integrated within the refrigerant circuit for measurement purposes. The Figure 23 illustrates the test equipment, highlighting specific labeled components.
The second section pertains to the air side of the AC condenser, where cooling air is sucked into the AC condenser by an axial fan. This section represents the surrounding air of a vehicle or engine bay. To ensure a uniform flow throughout the AC condenser, the fan is positioned within a straight ducting configuration. Achieving flow uniformity is crucial for accurately measuring the performance of the heat exchanger ($Q - Table$), which serves as a crucial input for the developed model (Section 4).
The third section is physically separated and encompasses an evaporator, heater, and axial fan, representing the vehicle cabin.
Additional photographs of the test equipment can be found in Appendix A.

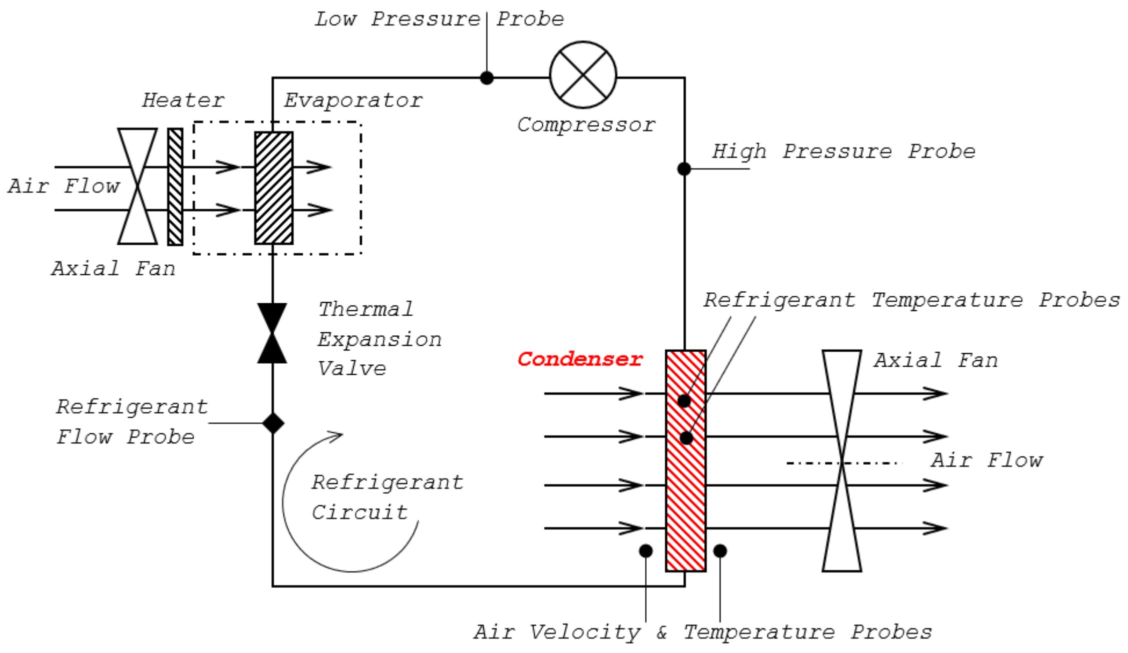


Figure 22: Test equipment scheme

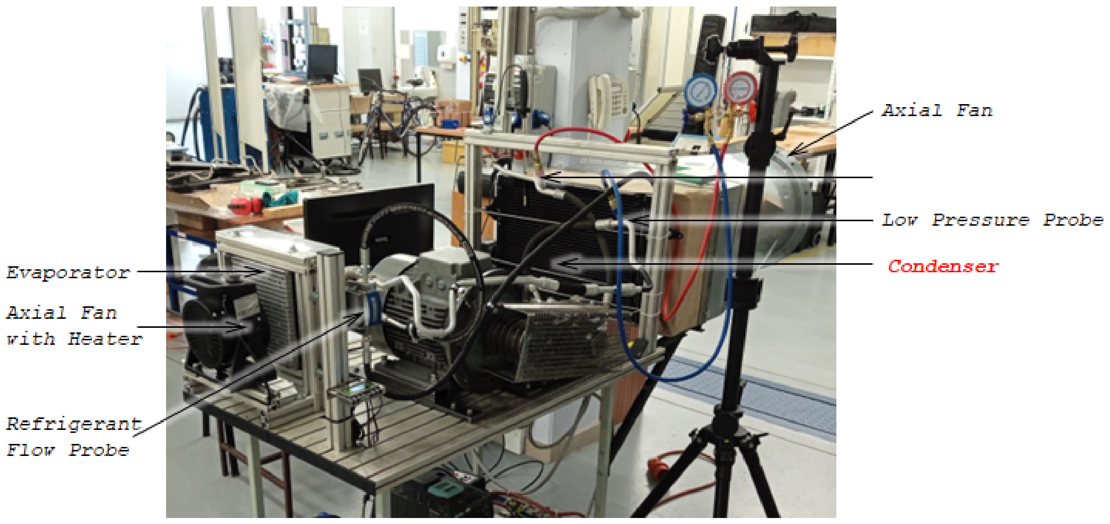


Figure 23: Test equipment photo

9.2 Sensor and Probes

The following list provides the measurement devices along with their respective systematic errors as provided by the manufacturers and data sampling frequencies.

- **Refrigerant flow**
  Flowmeter KROHNE DK34
  Maximum flow (kg/h) error of measured value is 4%
  Continuous measurement

- **Refrigerant pressure**
  Manometer CT-466
  Maximum pressure (Psi) measuring error of measured value is 1.6% of span
  Continuous measurement

- **Air Flow**
  Anemometer VOLTCRAFT PL-135
  Maximum velocity (m/s) measuring error of measured value is $5\% + 0.01$
  Maximum temperature $(°C)$ measuring error of measured value is $\pm 1°C$
  Sampling frequency 1Hz

- **Refrigerant and air temperature**
  Thermocouple OMEGA HH309A with grade K
  Maximum temperature $(°C)$ measuring error of measured value is $\pm 1°C$
  Sampling frequency 1Hz

- **Refrigerant temperature**
  Thermal camera Testo 875

Maximum temperature ($°C$) measuring error of measured value is $\pm 2°C$
Continuous measurement

9.3 Tested AC Condenser

The evaluated AC condenser is originally used in Skoda Auto Roomster passenger vehicle. The AC condenser basic dimensions as well as passes division are shown in Figure 24 below. The AC condenser core thickness is $160mm$. Photos of the evaluated condenser and decomposition are shown in Appendix B. The decomposition was used with the aim to determine pass flow within the condenser.

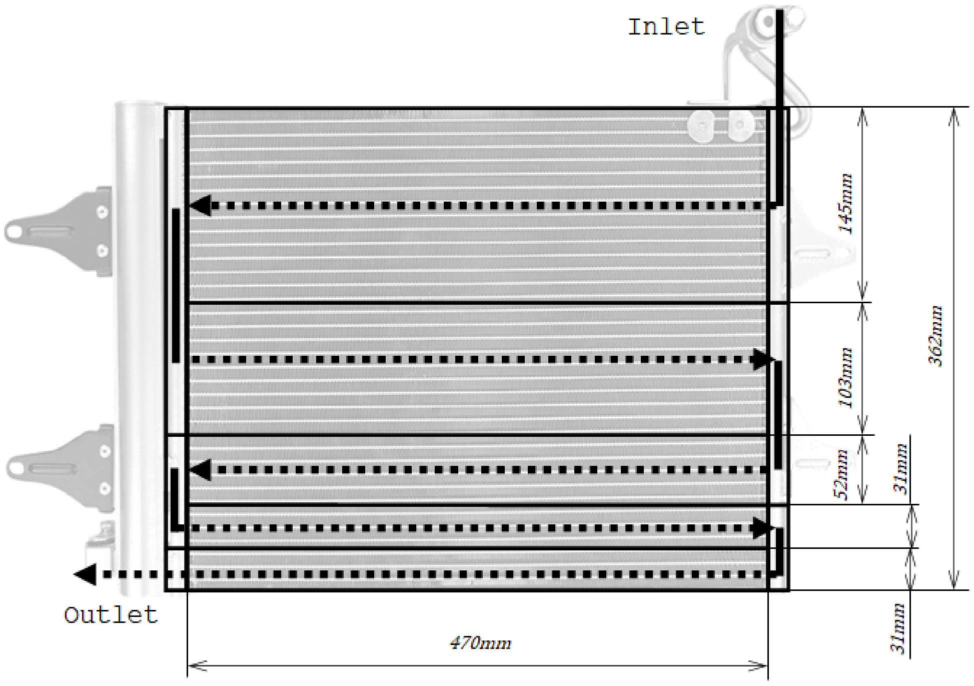


Figure 24: Evaluated AC condenser

9.4 Refrigerant

The test equipment refrigerant circuit (Figure 22) is filled with R13a refrigerant. Detailed chemical specifications and material properties of the used refrigerant could be found in Morrison and Ward (1991). The R134a refrigerant properties described by an empirical function could be found in Oliveira and Wakeham (1993) or in the book of Poling et al. (2001). Appendix C contains material properties applied in the AC condenser model and measurement evaluation in combination with online properties available from Freon et al. (2023).

10 Air Flow Measurement

Airflow measurement is crucial for evaluating the air mass flow rate and determining the air inlet and outlet temperatures. It plays a vital role in both the gathering of characteristic data (Section 11) and the evaluation of the AC condenser model (Section 12).
To measure the airflow, the tested AC condenser frontal area in the x-axis direction is divided into a grid of 25 measurement points, as illustrated in Figure 25. At each measurement point, the air inlet velocity, inlet temperature, and outlet temperature are measured. The VOLTCRAFT PL-135 sensor is used for measuring the air inlet velocity and temperature. Additionally, the air outlet temperature is measured using the OMEGA HH309A thermocouple with grade K (Section 9.2)

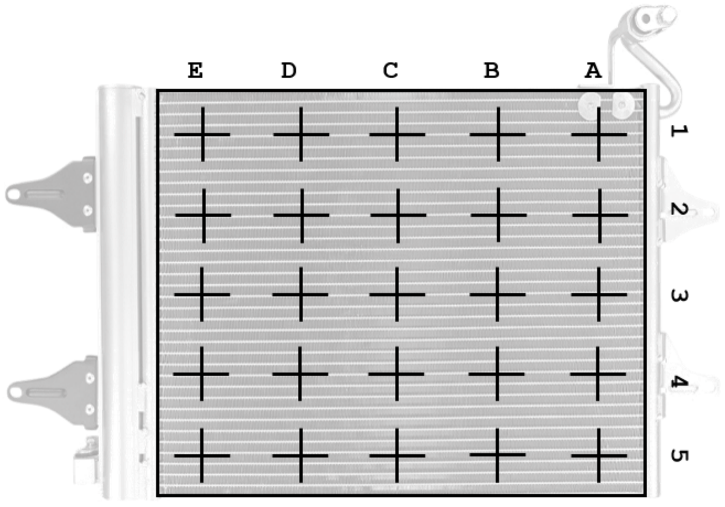


Figure 25: Measurement locations

The uniformity index (UI) is a significant parameter that determines the air inlet conditions. It plays a crucial role in the overall performance of the heat exchanger, as studied in Schmid et al. (2021). Achieving uniform air inlet conditions is essential for accurate characteristic data measurement (Section 11). The UI is defined in Equation 40 Tischer et al. (2001), including its discrete form used in measurement or simulation evaluations. For temporal error estimation, the normal distribution is used. However, for spatial error estimation, the T-Student distribution is preferred, and the values from Hahn and Hendrickson (1971) (Section 2.2) are utilized.

$$.UI = 1 - \frac{\int_0^A |u - \bar{u}| \, dA}{2|\bar{u}|A} = 1 - \frac{\sum_{jk} |u_{1jk} - \bar{u}| \, A_{1jk}}{2|\bar{u}| \sum A_{1jk}} \tag{40}$$

11 Characteristic Data Measurement

Characteristic data measurement focuses on gathering input data for the AC condenser model. This procedure involves measuring multiple physical quantities with the aim of calculating the performance table via equation 15 Incropera et al. (2011) (so-called $Q - Table$).

The dissipated heat within each phase region of the AC condenser, along with the corresponding region volume as highlighted in Figure 26, are crucial information for the AC condenser model. The volume is required for equation 37 Fluent (2009) used in the model. Determination of each section is based on identifying the onset of condensation, such as the saturated temperature of the refrigerant. To determine the region areas shown in Figure 26, measurement devices such as the thermal camera Testo 87 and Thermocouple OMEGA HH309A with grade K (Section 9.2) were used. Additionally, simultaneous measurements of air and refrigerant mass flow rates, as well as refrigerant pressure, are essential. The air mass flow rate can be calculated based on air velocity measurements discussed in the previous Section 10 and equation 34 Nozicka (2008). It should be noted that the refrigerant temperature is not measured directly, but rather on the outer surface of the AC condenser. This simplification is based on assuming minimal temperature gradients between the refrigerant inside microchannels or tubes and the outer surfaces. This assumption is justified by the thinness of the tubes (approximately 1mm) and their high thermal conductivity, as they are made of aluminum.

The equation 25 is derived and normalized by the dissipated heat. Assuming a zero uncertainty in specific heat capacity, as stated in Tatara and Lupia (2011), equation 41 is obtained. This uncertainty equation is valid for both the air side and the refrigerant side. However, it should be noted that the heat dissipation measured on the air side and the refrigerant side will never be exactly the same. This discrepancy is due to measurement uncertainties and non-uniform flow on the air side. Therefore, the heat dissipation should be calculated using equation 42 Tatara and Lupia (2011). The uncertainty in heat dissipation can then be calculated as shown in equation 43 Tatara and Lupia (2011).

$$Un_{\dot{Q}} = \dot{Q}\sqrt{[\frac{U_m}{\dot{m}}]^2 + [\frac{U_{T_{in}}}{(T_{in} - T_{out})}]^2 + [\frac{U_{T_{out}}}{(T_{in} - T_{out})}]^2} \tag{41}$$

$$\dot{Q}_{Overall} = \frac{\dot{Q}_{air}Un_{air}^2 + \dot{Q}_{ref}Un_{ref}^2}{Un_{air}^2 + Un_{ref}^2} \tag{42}$$

$$Un_{\dot{Q}_{Overall}} = \dot{Q}_{Overall}\frac{\dot{Q}_{ref}}{\dot{Q}_{air}}\sqrt{[\frac{U_{mAir}}{\dot{m}}]^2 + [\frac{U_{T_{in}Air}}{(T_{in} - T_{out})}]^2 + [\frac{U_{T_{out}Air}}{(T_{in} - T_{out})}]^2 + Refrigerant\ side} \tag{43}$$

For temporal error estimation, the normal distribution is used. However, for spatial error

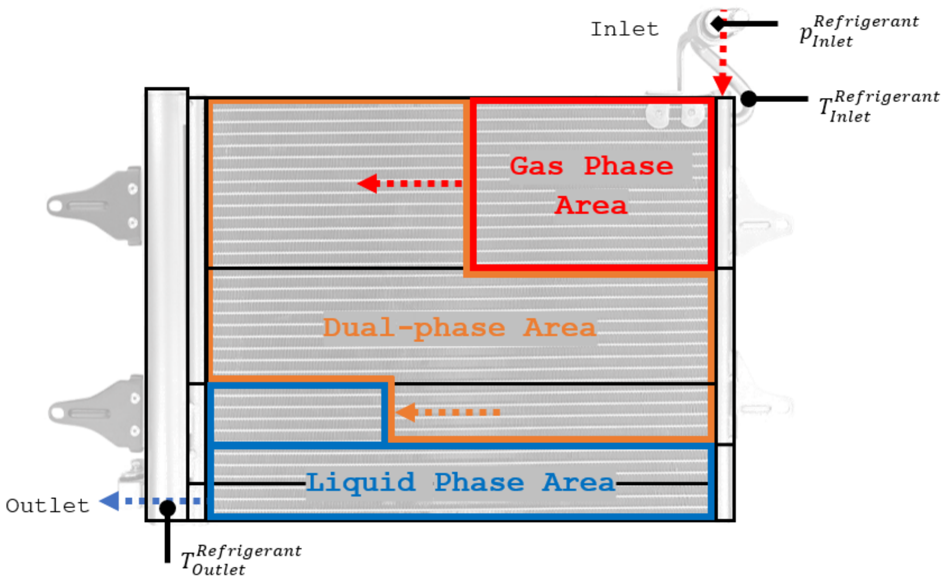

Inlet
$p_{Inlet}^{Refrigerant}$
$T_{Inlet}^{Refrigerant}$
Gas Phase Area
Dual-phase Area
Liquid Phase Area
Outlet
$T_{Outlet}^{Refrigerant}$

Figure 26: Characteristic data measurement scheme

estimation, the T-Student distribution is preferred, with values obtained from Hahn and Hendrickson (1971) (Section 2.2), as well as for Section 10.

11.1 Gas and Liquid Phase

For the gas and liquid regions indicated in Figure 26, equations 41, 42, and 43 Tatara and Lupia (2011) can be directly used. The only difference lies in the inlet and outlet temperatures. In the gas region, the inlet temperature corresponds to the AC condenser inlet temperature, while the outlet temperature corresponds to the saturation temperature of the refrigerant at the measured pressure. In the liquid region, the inlet temperature is the saturated refrigerant temperature, and the outlet temperature is the AC condenser outlet temperature.

11.2 Dual Phase

However, for the dual-phase condition, the heat dissipation on the refrigerant side cannot be calculated using equation 15 Incropera et al. (2011) since the inlet and outlet temperatures are the same. Instead, equation 26 Liang et al. (2015) should be used. In this equation, the specific enthalpy is derived from the measured pressure using data from Section 9.4. Therefore, the uncertainty of the pressure measurement is included in the calculation, as shown in quotation 44. The remaining calculations follow the same methodology as described in equations 42 and 43 Tatara and Lupia (2011).

$$Un_{\dot{Q}} = \dot{Q}\sqrt{[\frac{U_m}{\dot{m}}]^2 + [\frac{U_p}{\dot{p}}]^2} \tag{44}$$

12 Verification Measurement

Verification measurements are crucial for evaluating the AC condenser model. These measurements comprise a combination of previously mentioned data. The air inlet flow data obtained in accordance with Section 10 is used as a boundary condition. Additionally, the characteristic data from Section 11 are incorporated into the model. The AC condenser model also requires overall heat dissipation as an input. The measured pressure is utilized for determining specific refrigerant enthalpy and saturated temperature, which pertain to the material properties of the refrigerant. The requirements for the model are summarized in Section 4. Finally, the air outlet temperature measurement, as well as the refrigerant inlet and outlet temperatures, are used for model verification.

13 Results

In order to validate the proposed AC condenser model that has been developed, two separate sets of measurements were conducted, referred to as Set A and Set B. Each set of measurements was carried out under distinct but constant and steady-state conditions and repeated tree times. For each measurement set, the model's input data ($Q - Table$), model boundary conditions, and verification data were recorded. This comprehensive approach eliminates the need for interpolating characteristic data ($Q - Table$) for various refrigerants and airflow rates, a requirement that may arise in industrial applications.
Raw measured data are provided in Appendix E.

13.1 Air Flow Measurement

The airflow measurement provides the input data for the AC condenser model as well as verification data. For the input, the model uses the airflow inlet velocity and temperature, while for model verification, the air outlet temperatures. The model aims to accurately predict the air outlet temperatures based on the given inlet velocity and temperature. The measurement locations are shown in Figure 25.

13.1.1 Air Flow Inlet

The measurement of airflow inlet serves as the boundary condition for the developed AC condenser model, specifically determining the air inlet mass flow rate and temperature. The direct measurement of air mass flow rate is not performed; instead, it is calculated based on the magnitude of air velocity using the equation 34 Nozicka (2008). Additionally, the airflow measurement is utilized to determine the characteristic data of the AC condenser as outlined in Section 11.

Table 2 and 3 present the results of the first set (Set A) of air flow inlet measurements, accompanied by the corresponding uncertainties. The Figure 27 shows contour of the measured inlet profiles for the Set A. The average inlet velocity is calculated to be $1.669 m/s$ with a uniformity index of 0.982. The average inlet air temperature is $24.42°C$ with a uniformity index of 0.999.

Similarly, Table 4 and 5 present the results of the second set (Set B) of air flow inlet measurements, along with the corresponding uncertainties. The Figure 28 shows contour of the measured inlet profiles for the Set B. The average inlet velocity for this set is determined to be $1.265 m/s$ m/s, with a uniformity index of 0.977. The average inlet air temperature is recorded as $18.23°C$, with a uniformity index of 0.999.

The mean refers to the area-weighted average for both quantities and for both measurement sets. The uniformity index is relatively high, which is crucial for the characteristic data measurement results in the next Section 13.2.

Table 2: Air inlet velocity measurement Set A

Air inlet velocity (m/s)					
Loc.	A	B	C	D	E
1	1.599±0.101	1.625±0.187	1.627±0.141	1.622±0.152	1.570±0.101
2	1.761±0.107	1.767±0.140	1.680±0.175	1.721±0.133	1.695±0.172
3	1.757±0.115	1.604±0.101	1.699±0.106	1.695±0.140	1.700±0.146
4	1.768±0.244	1.696±0.128	1.567±0.142	1.792±0.122	1.648±0.160
5	1.573±0.169	1.583±0.109	1.657±0.177	1.726±0.138	1.593±0.131

Table 3: Air inlet temperature measurement Set A

Air inlet temperature (°C)					
Loc.	A	B	C	D	E
1	24.245±1.016	24.392±1.012	24.203±1.040	24.746±1.196	24.635±1.076
2	23.900±1.014	24.100±1.027	25.126±1.920	25.432±1.089	24.999±1.046
3	23.876±1.014	24.040±1.013	24.559±1.145	24.445±1.051	25.026±1.081
4	23.869±1.066	23.802±1.010	24.393±1.046	24.345±1.072	25.700±1.175
5	23.657±1.057	23.452±1.021	23.816±1.063	24.069±1.149	25.683±1.183

Table 4: Air inlet velocity measurement Set B

Air inlet velocity (m/s)					
Loc.	A	B	C	D	E
1	1.143±0.085	1.288±0.171	1.280±0.089	1.122±0.087	1.139±0.109
2	1.325±0.126	1.193±0.130	1.240±0.099	1.136±0.104	1.270±0.130
3	1.239±0.114	1.209±0.136	1.267±0.099	1.292±0.126	1.315±0.127
4	1.311±0.093	1.328±0.087	1.263±0.097	1.337±0.098	1.352±0.090
5	1.387±0.082	1.338±0.141	1.239±0.101	1.276±0.090	1.324±0.094

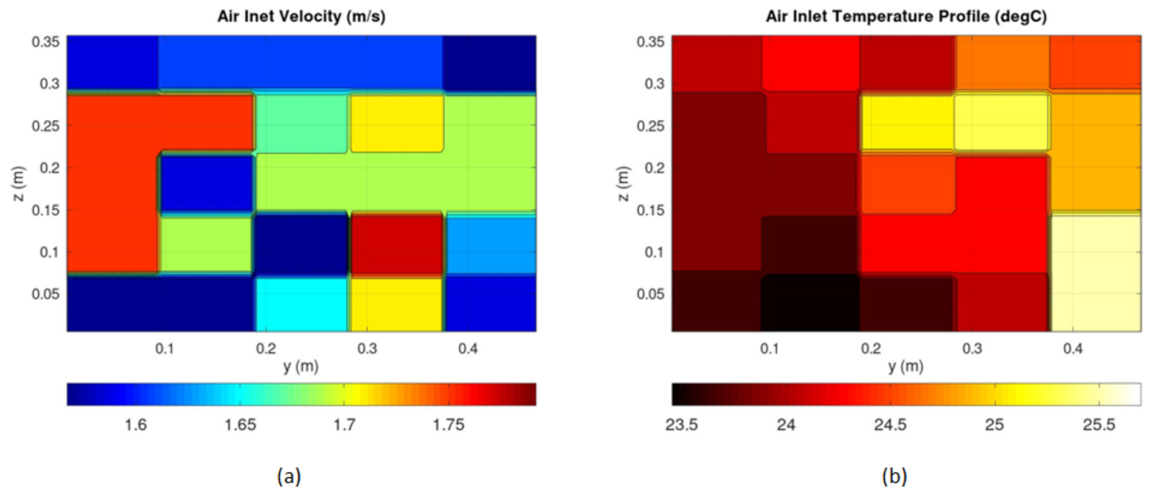


Figure 27: Air inlet (a) velocity (m/s); (b) temperature (°C) profile Set A

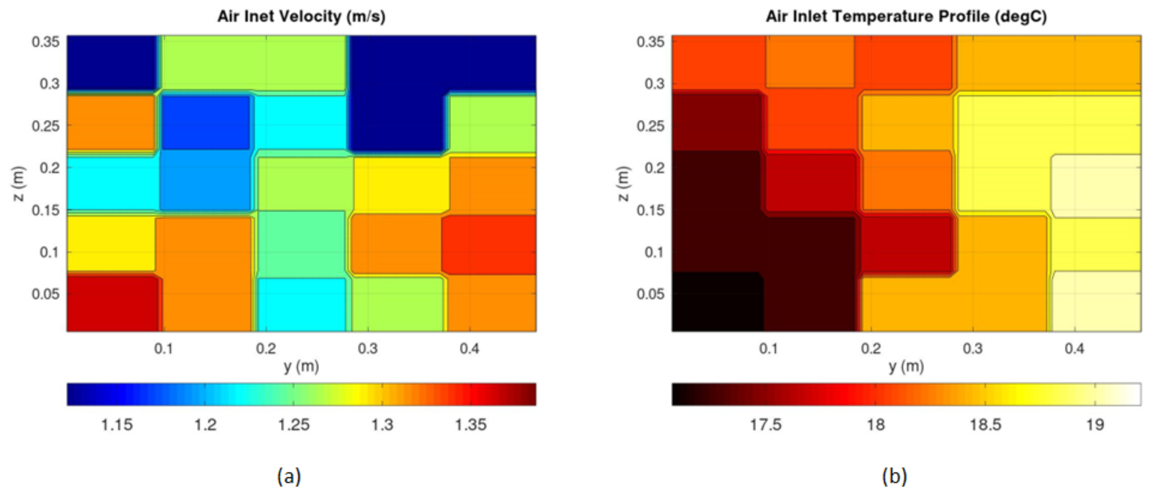


Figure 28: Air inlet (a) velocity (m/s); (b) temperature (°C) profile Set B

Table 5: Air inlet temperature measurement Set B

Air inlet temperature (°C)					
Loc.	A	B	C	D	E
1	18.215±1.157	18.244±1.009	18.154±1.007	18.502±1.008	18.517±1.045
2	17.484±1.106	18.079±1.077	18.480±1.036	18.853±1.030	18.856±1.045
3	17.399±1.000	17.730±1.026	18.262±1.026	18.881±1.061	19.101±1.032
4	17.312±1.012	17.438±1.012	18.408±1.009	18.541±1.052	19.006±1.047
5	17.073±1.014	17.421±1.053	18.612±1.071	18.533±1.050	19.209±1.036

13.1.2 Air Flow Outlet

The air outlet temperature measurements for Sets A and B are listed in Tables 6 and 7, respectively. The corresponding uncertainties are provided within the tabels.

Table 6: Air outlet temperature measurement Set A

Air outlet temperature (°C)					
Loc.	A	B	C	D	E
1	37.246±1.065	28.359±1.075	28.795±1.036	29.702±1.377	29.394±1.102
2	36.726±1.032	29.243±1.012	28.474±1.305	29.350±1.032	28.835±1.249
3	29.239±1.125	28.596±1.272	27.777±1.024	29.220±1.105	27.979±1.489
4	29.685±1.009	28.850±1.031	27.783±1.070	29.094±1.321	28.085±1.279
5	29.485±1.011	29.400±1.187	28.389±1.038	28.863±1.358	28.652±1.401

Table 7: Air outlet temperature measurement Set B

Air outlet temperature (°C)					
Loc.	A	B	C	D	E
1	28.921±1.059	25.510±1.027	25.523±1.027	25.456±1.089	25.815±1.047
2	29.047±1.034	25.778±1.071	25.509±1.088	25.719±1.037	25.862±1.036
3	25.872±1.058	25.259±1.044	25.584±1.020	25.839±1.131	25.881±1.205
4	25.492±1.052	25.671±1.165	25.304±1.029	25.686±1.080	25.915±1.045
5	25.352±1.068	25.637±1.034	25.631±1.060	25.702±1.046	25.647±1.204

13.2 Characteristic Data Measurement

The measurement of characteristic data provides vital information on various parameters, including the refrigerant mass flow rate, refrigerant inlet pressure, and refrigerant inlet and outlet temperatures, for both Set A and Set B measurements. Furthermore, as discussed in Section 11, the frontal area for each phase is measured. This allows for the acquisition of essential input data for equation 37 Fluent (2009), which is utilized in the developed AC condenser model.

In Set A, the measured refrigerant mass flow rate is determined to be $0.018 + / - 0.001 (kg/s)$, with a measurement uncertainty of 4.038% relative to the mass flow rate value. The refrigerant AC condenser inlet and outlet temperatures are recorded as $40.626+/-1.291°C$) and $27.684+/-1.272°C$, respectively. The refrigerant inlet pressure is measured at $105.2+/-8.010(psi)$, with a measurement uncertainty of 7.031% relative to the pressure value.

In Set B, the measured refrigerant mass flow rate is determined to be $0.015+/-0.001(kg/s)$, with a measurement uncertainty of 4.031% relative to the mass flow rate value. The refrigerant AC condenser inlet and outlet temperatures are recorded as $29.190+/-1.242°C$ and $24.637+/-1.154°C$, respectively. The refrigerant inlet pressure is measured at $90.267+/-8.001(psi)$, with a measurement uncertainty of 7.614% relative to the pressure value.

The temperature measurement curves are shown for the both measurement sets in the Figure 29. The time development of these curves serves as evidence of the steady-state conditions within the measurements.

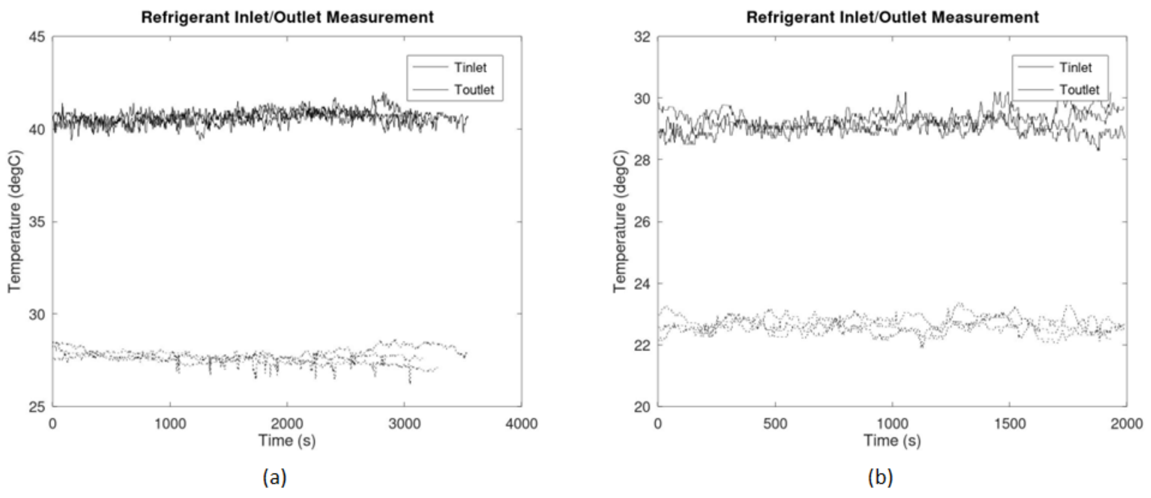


Figure 29: Refrigerant inlet and outlet temperature (°C) (a) Set A; (b) Set B

Based on the aforementioned measured values, the heat release for each refrigerant section, as well as the overall heat dissipation, is calculated following the guidelines provided in Section 11. The heat dissipation values on the refrigerant side are presented in the Table 8. It is important to note that the accuracy of measuring the total heat dissipation within the AC condenser is 16.441% for Set A measurements and 27.653% for Set B. Material properties used within the calculation are based on Freon et al. (2023) and values shown in Appendix C. As indicated in Table 8, both measurements did not reach the overcooling section (liquid section). The inability to measure the overcooling section is attributed to the refrigerant saturated temperature falling within the uncertainties of the refrigerant outlet measurements. However, achieving overcooling in the AC condenser can be accomplished in a integrated dryer connected to the AC condenser's refrigerant outlet.

The calculation of heat dissipation on the air side is performed using Equation 15 Incr-

Table 8: Measured heat dissipation on refrigerant side

Heat dissipation (W)		
Region	Set A	Set B
Gas	247.382 +/-32.893	98.187 +/-22.030
Dual	3175.714 +/-21.321	2712.147 +/-9.561
Liquid	N/A	N/A
Overall	3423.096 +/-562.792	2810.334 +/-777.127

opera et al. (2011), along with the measured values of airflow from Section 13.1. The obtained results are presented in Table 9.

Table 9: Measured heat dissipation on air side

Heat dissipation (W)		
Region	Set A	Set B
Gas	327.632 +/-20.068	268.357 +/-20.636
Dual	1362.951 +/-16.860	1737.456 +/-17.315
Liquid	N/A	N/A
Overall	1690.583 +/-16.876	2005.813 +/-17.308

The refrigerant state (gas, dual-phase and liquid) regions size for measurement Set A and Set B are specified in Table 10. Additionally, Figure 30 illustrates the thermal camera measurement utilized to determine the area as described in Section 11.

Table 10: Refrigerant state region areas

Area (m^2)		
Region	Set A	Set B
Gas	0.010	0.007
Dual	0.160	0.163
Liquid	N/A	N/A

Based on the expressions 42 and 43, the Table 11 is generated, representing the combined heat dissipation. Regarding the higher uncertainty in the heat dissipation on the refrigerant side compared to the air side, the combined heat dissipation aligns more closely with the refrigerant side. However, it's important to note that the uncertainty in the air side does not account for the sensitivity to the number of measured points in Figure 25. This aspect should be further investigated, especially in the case of industrial applications. The combined heat dissipation value should be applied in the setup of the AC condenser model. However, the refrigerant side heat dissipation could be preferred over the combined or air heat dissipation due to the inherent uncertainties associated with the number of measured points in the air side.

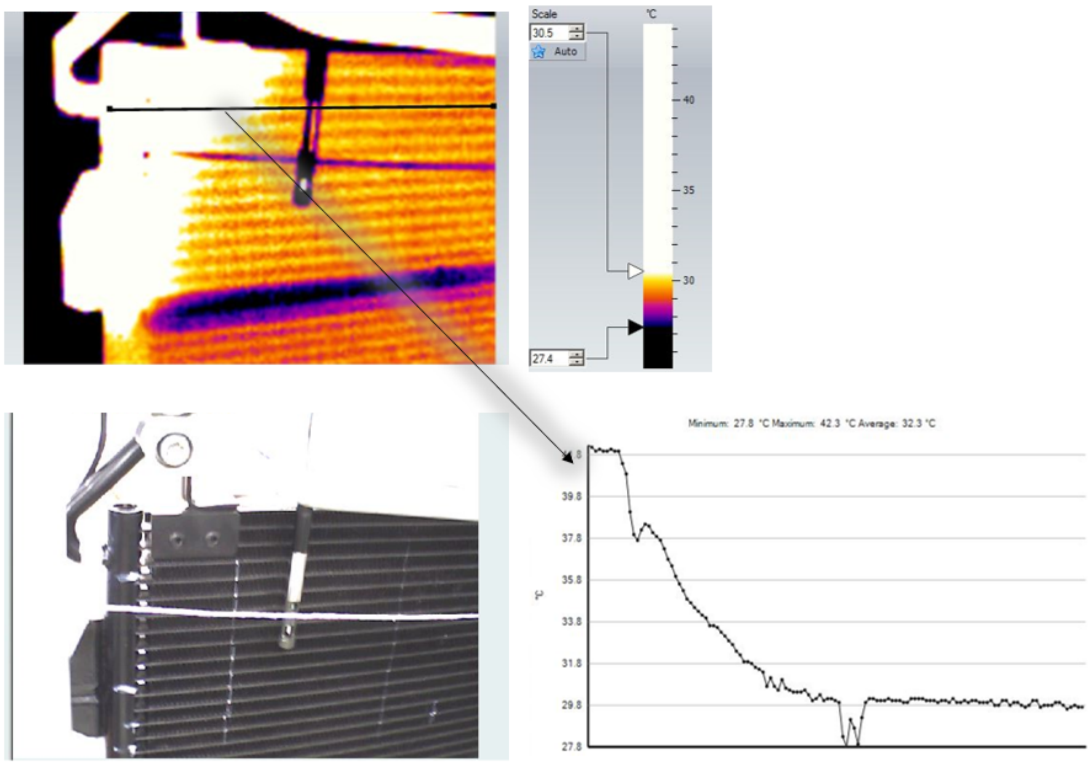


Figure 30: Refrigerant phase section measurement

Table 11: Measured heat dissipation

Heat dissipation (W)		
Region	Set A	Set B
Gas	269.150 +/-24.725	177.718 +/-20.801
Dual	2478.280 +/-20.627	1965.202 +/-11.817
Liquid	N/A	N/A
Overall	3421.540 +/-455.746	2809.935 +/-520.697

13.3 Verification Measurement

The model's verification is performed via comparing the results of the AC condenser model with the measured air outlet temperature (Section 13.1.2) and the refrigerant inlet and outlet temperature (Section 13.2)

13.4 AC Condenser Model

The computational domain, as shown in Figure 13, has been discretized into a numerical grid consisting of $(8, 75, 35)$ cells along the coordinate system. The size of the domain has been determined based on the dimensions of the evaluated AC condenser (Figure 24). The resulting computational mesh is presented in Figure 31.
The mesh size was chosen following a mesh sensitivity study, as illustrated in Figure 32. The plot shows the resulting auxiliary inlet temperature against the cell count. The

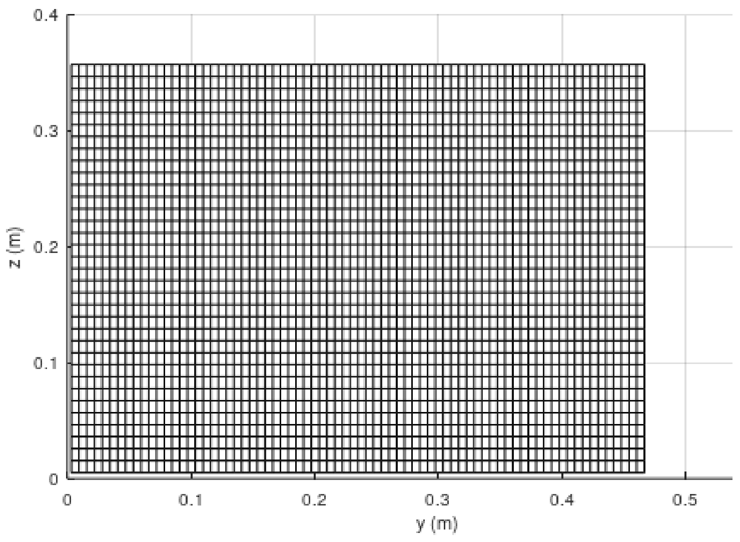


Figure 31: Computational mesh in $y - z$ area

highlighted mesh size is selected , as further increasing the mesh decomposition does not significantly affects the results.

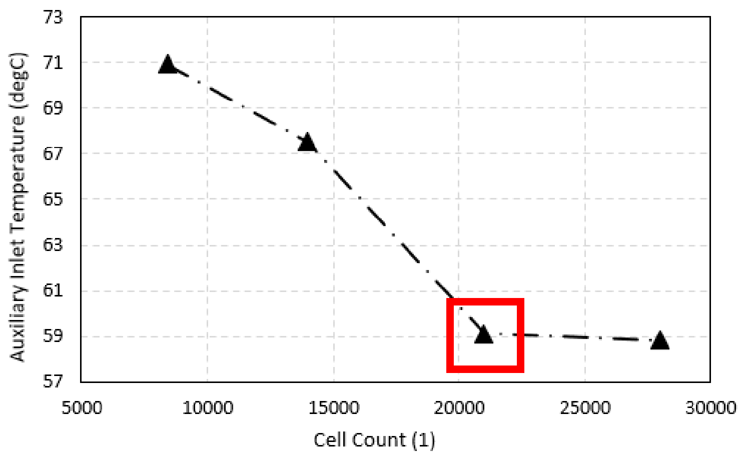


Figure 32: Mesh sensitivity study

The boundary condition for the air inlet mass flow rate is determined using air flow velocity and temperature measurements, as described in Section 6.1, and utilizing data from Section 13.1.1. For both setup Sets A and B, constant material properties were defined based on appropriate average temperature or measured pressure, and these properties are listed in Table 12. In both calculations, the convergence tolerance δ was set to $1W$. The convergence of overall heat dissipation in setup Set A and Set B during the solution process is illustrated in Figure 33. The upper limit for the bisection method was established at $200K$ above the saturated temperature, specific to each respective Set.

The characteristic data and heat dissipation values for the entire AC condenser are derived from the findings presented in Section 13.2. Specifically, Table 8 provides the relevant information. This table was preferred over Table 11 because the uncertainty of the dissipation on the air side is affected by the spatial distribution of measured points, a parameter that was not determined in this thesis. Therefore, further investigation is required to address this aspect. Nevertheless, the difference between the refrigerant side and the combined heat dissipation is negligible for the both measurements sets.

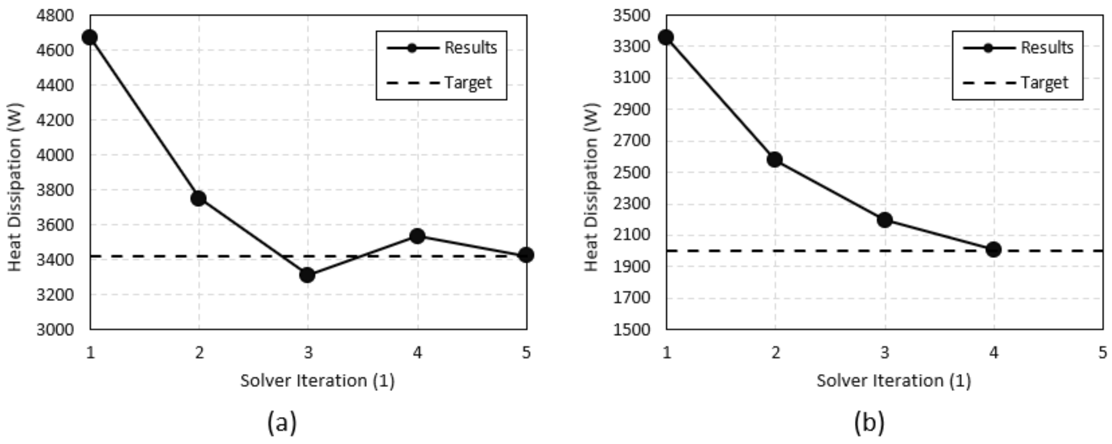


Figure 33: Convergence (a) Set A; (b) Set B

Table 12: Constant material properties

Material properties specific values		
Constant	Set A	Set B
Air Cp (J/kg-K)	1005.743	1004.533
Refrigerant Cp Gas (J/kg-K)	1010.704	1057.234
Refrigerant Cp Liquid (J/kg-K)	1432.144	1410.101
Refrigerant Latent Heat (J/kg)	187424.159	153541.046
Refrigerant Saturated Temperature (°C)	27.876	23.043

The contour plots displaying the resultant air outlet temperature for both setup Sets A and B, are presented in Figures 34 and 35. The refrigerant inlet and outlet temperatures are $60.884°C$ and $27.876°C$, respectively, for Set A, while for Set B, the corresponding values are $41.793°C$ and $23.043°C$. The refrigerant temperature contour is shown in Figure 36. The model results show a significant tendency to overpredict the verification measurements (Section 13.3). However, it is important to acknowledge that the input data, especially concerning heat dissipation, exhibits relatively high uncertainty. This uncertainty allows for adjustments to the input parameters, enabling the model to align better with the measured values. A more detailed discussion of this aspect is presented in the conclusion Section 14.

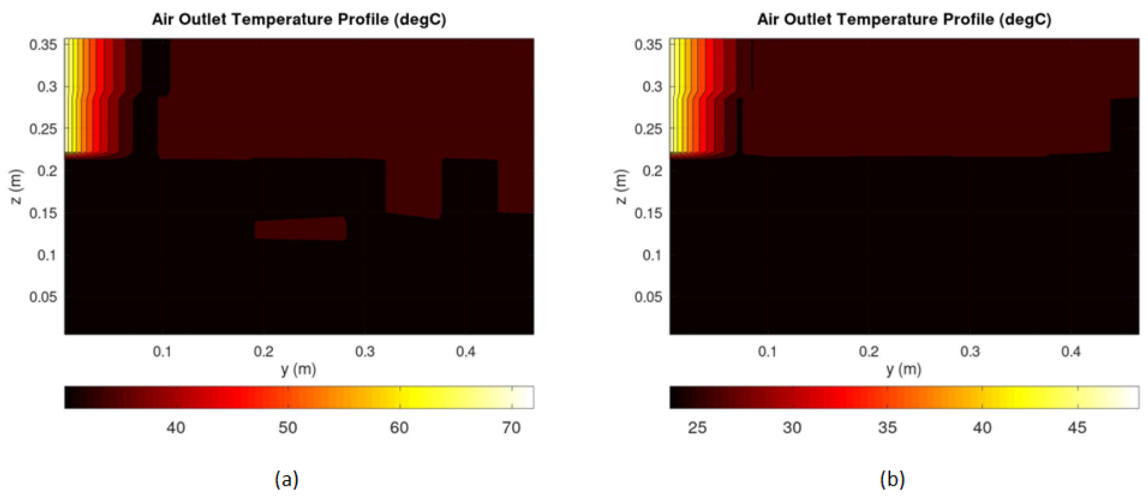


Figure 34: Air outlet temperature (°C) (a) Set A; (b) Set B

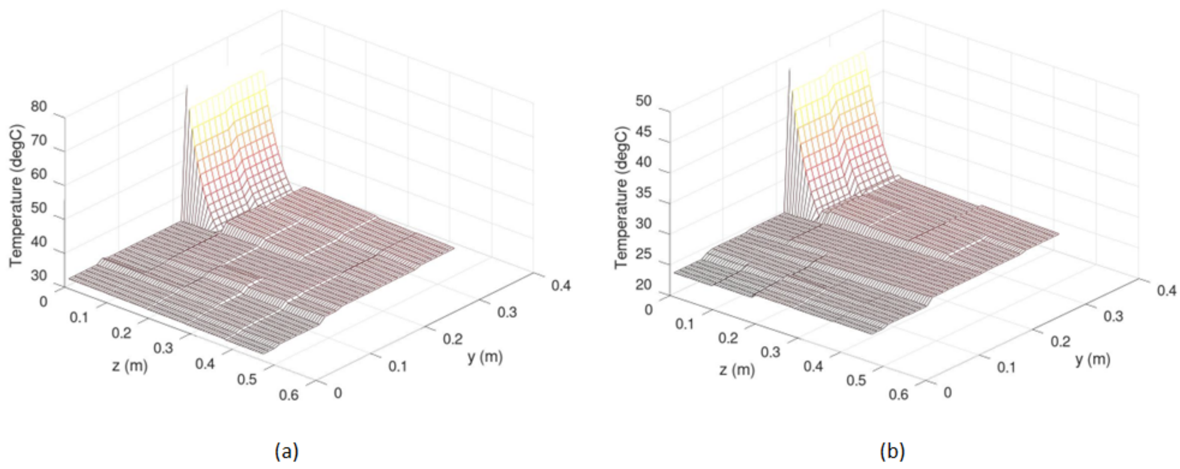


Figure 35: Air outlet temperature (°C) (a) Set A; (b) Set B

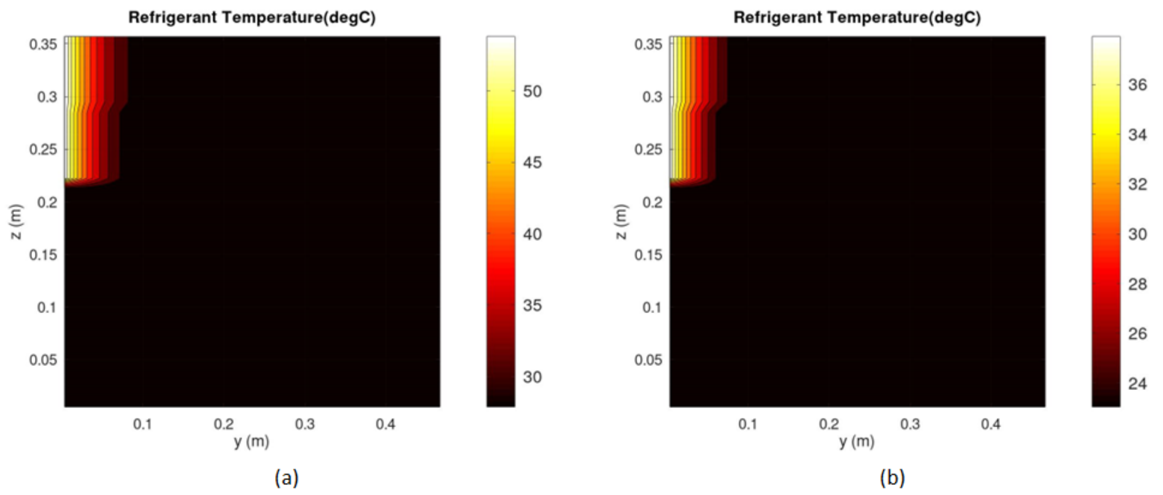


Figure 36: Refrigerant temperature (°C) (a) Set A; (b) Set B

14 Conclusions

Within the thesis, a comprehensive study of AC condenser heat transfer modeling and measurement was conducted. The primary focus was on developing a simplified model suitable for complex 3D computational fluid dynamics (CFD) simulations.
The philosophy underlying the suggested model is to iteratively utilize directly measured overall heat transfer coefficients instead of relying on literature or separately measured convective/condensation heat transfer coefficients. These separately measured coefficients are specific to the micro-channel heat exchanger design, the refrigerant used, and the operating conditions. Thus, the proposed approach offers greater generality while significantly reducing the amount of required input information. However, it should be noted that this approach necessitates the modification of input data measurements, as outlined in Section 11.
The developed model utilized the widely recognized $\epsilon - NTU$ approach, as elaborated in Section 4. Through this approach, the AC condenser model iteratively predicts locally refrigerant phase and appropriately selects heat transfer models and measured characteristics, resulting in an accurate spatial distribution of heat dissipation. Enhanced predictions of refrigerant and air temperatures generally lead to improved predictions of other components within a vehicle. This is particularly advantageous, especially for battery electric vehicles (BEVs), where the requirements for thermal management are becoming increasingly stringent. In the context of BEVs, every watt of energy must be carefully considered and efficiently managed to optimize the performance and range of the vehicle.
In Section 13.4, the suggested and developed model's results for the refrigerant inlet temperature were found to exhibit significant over-prediction. For Set A, the model over-predicted the refrigerant temperature by more than $18.967K$, while for Set B, the over-prediction was above $12.603K$. Consequently, the air outlet temperature was also over-predicted.
However, when the input heat dissipation was reduced by 8% for Set A and by 7% for Set B, the resulting refrigerant inlet temperature fell within the uncertainty bounds of the measurement data, as shown in Figure 37. Similarly, the refrigerant outlet temperatures of $27.876°C$ for Set A and $23.043°C$ for Set B were also within the uncertainty range of the corresponding measurements, which were $27.684 \pm 1.272°C$ and $22.637 \pm 1.154°C$, respectively.
The reduced prescribed heat dissipation is within the range of uncertainty for the measured heat dissipation (Table 8), the model achieved improved agreement also with the measured air outlet temperatures. The contours of the air outlet temperature with the reduced heat dissipation prescribed in the model are shown in Figure 38, and a comparison with the measurement data is presented in Figures 39, 40, 41, 42, and 43. It was observed that the predicted values were in good agreement with the measurement data

for Set B, while in Set A, there was a slight over-prediction.
One noteworthy observation in Figure 39 was a significant change in the air outlet temperature, highlighted by a red box. This change was attributed to the refrigerant being cooled to the saturation temperature, altering the heat transfer behavior.
Overall, the findings highlighted the sensitivity of the model to the prescribed heat dissipation and its ability to match the measurement data when within the range of uncertainty. The results presented in the thesis demonstrate the feasibility of the developed AC condenser model and measurement approach for industrial application, while keeping the aforementioned benefits.

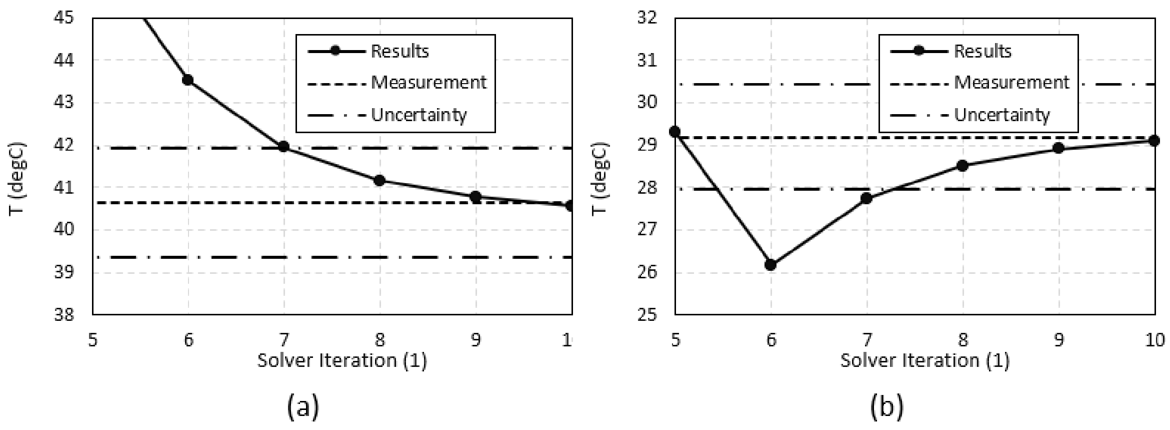


Figure 37: Refrigerant inlet temperature (°C) (a) Set A ; (b) Set B

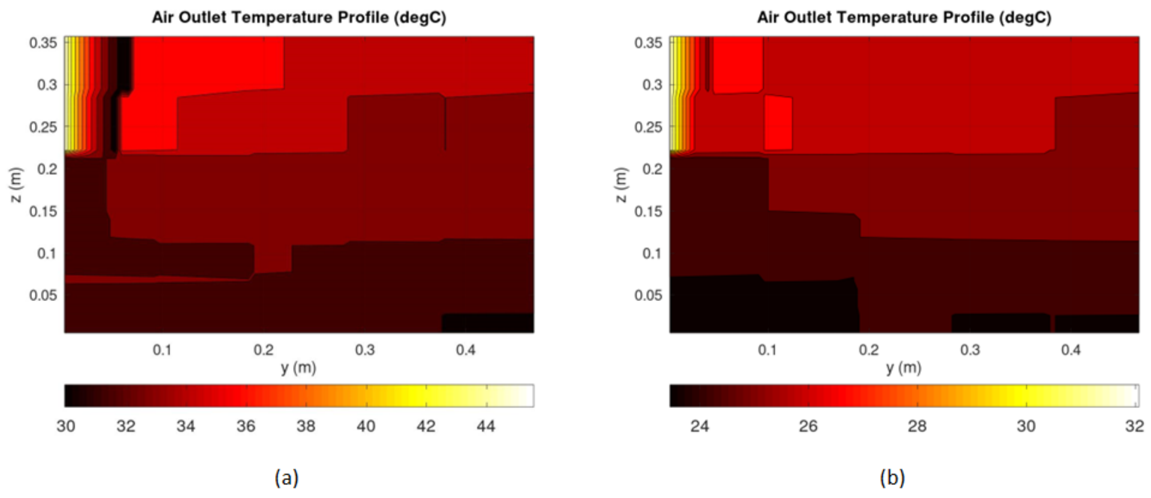


Figure 38: Air outlet temperature (°C) (a) Set A; (b) Set B

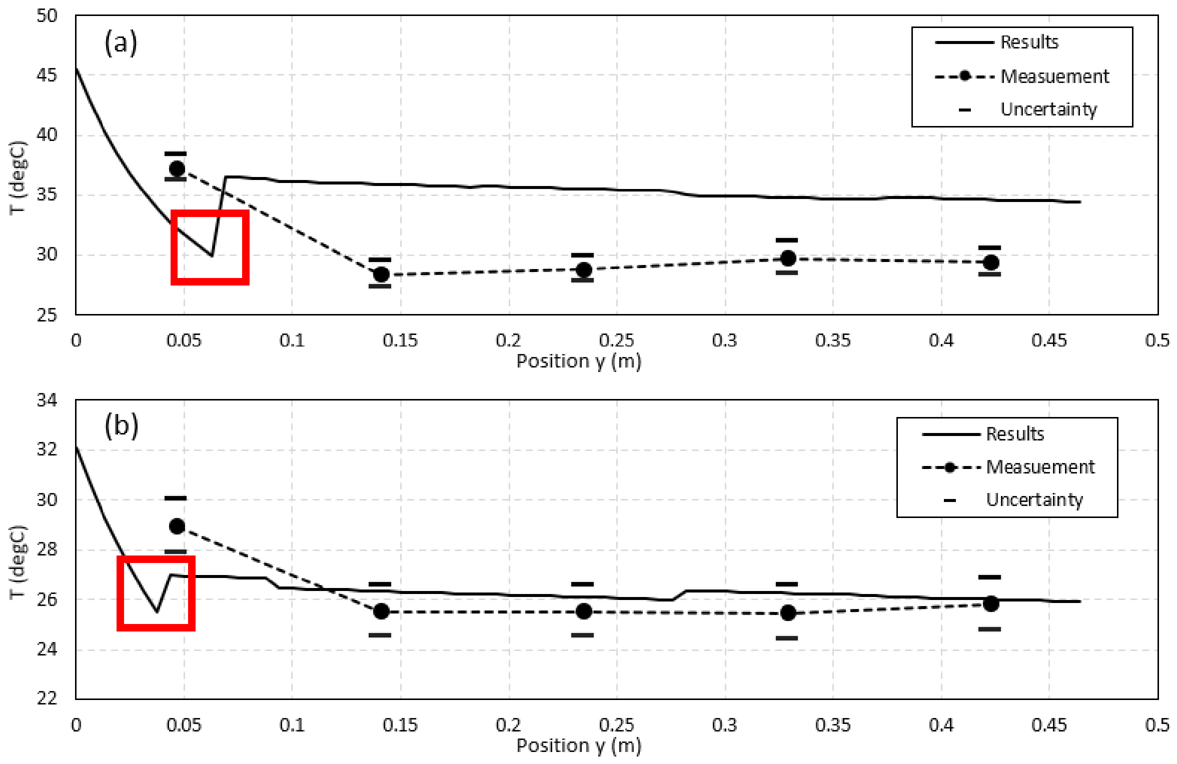


Figure 39: Air outlet temperature A1;B1;C1;D1;E1 (°C) (a) Set A; (b) Set B

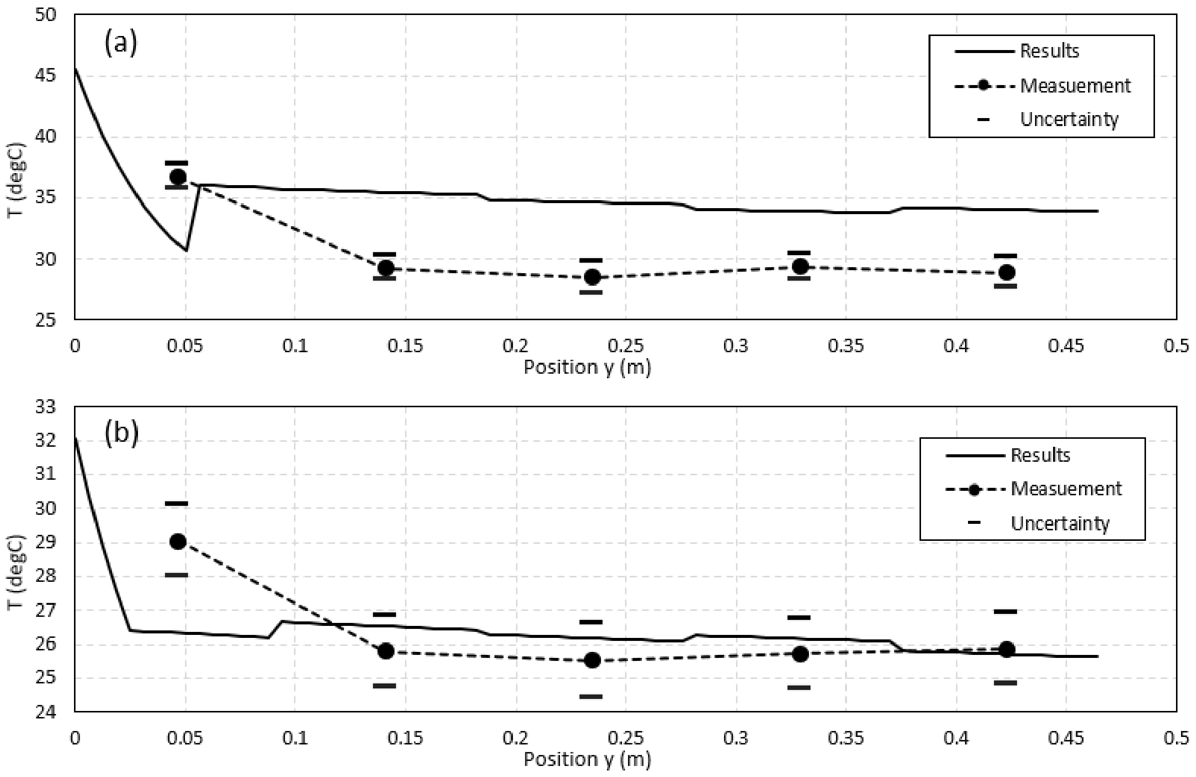


Figure 40: Air outlet temperature A2;B2;C2;D2;E2 (°C) (a) Set A; (b) Set B

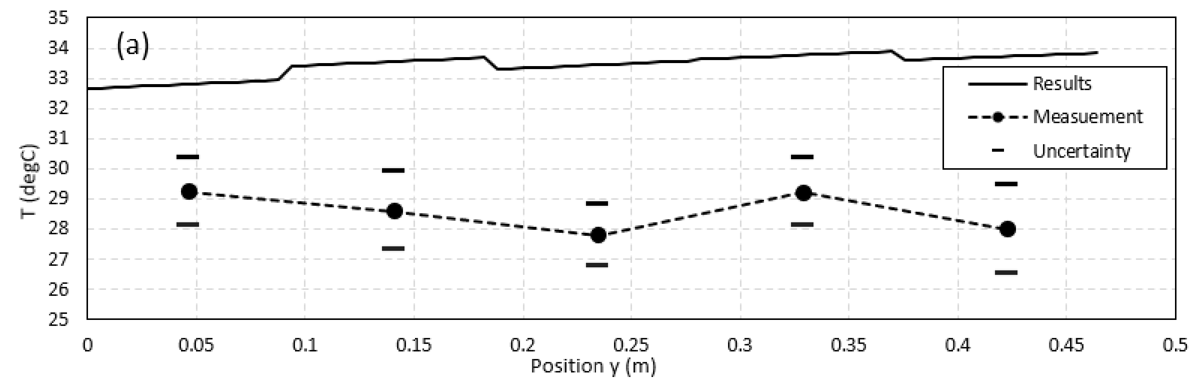


Figure 41: Air outlet temperature A3;B3;C3;D3;E3 (°C) (a) Set A; (b) Set B

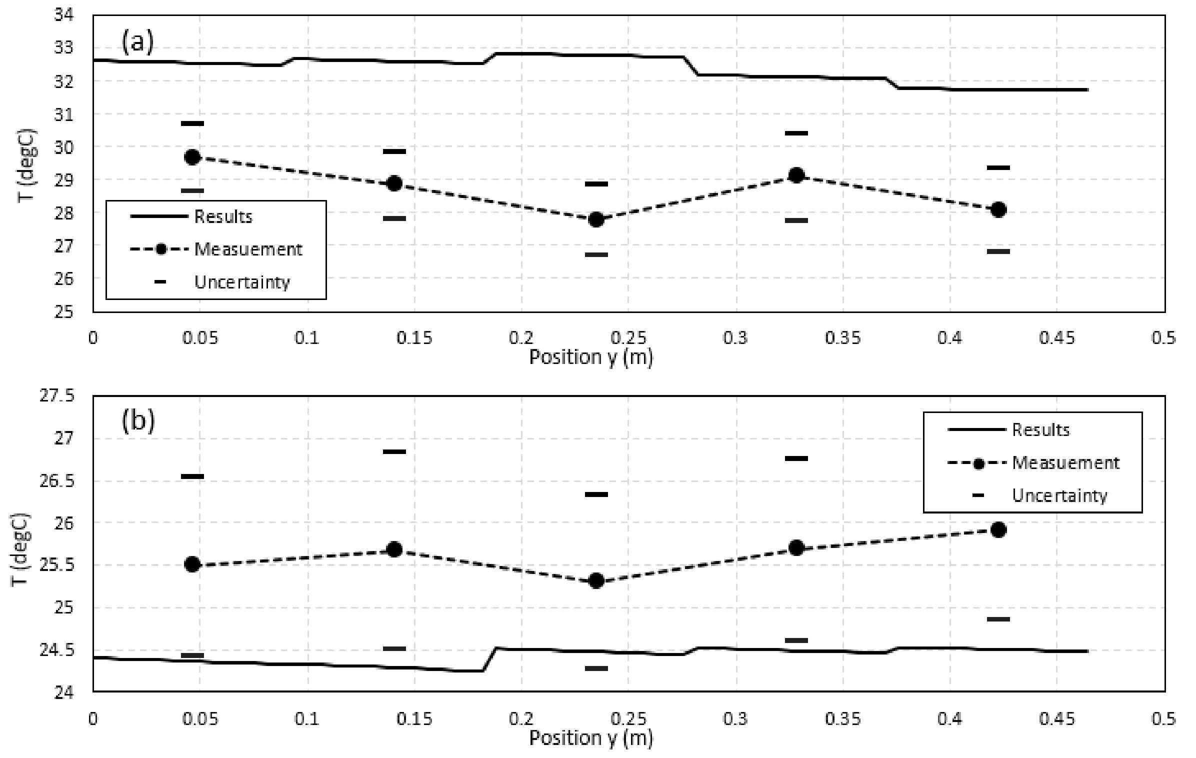


Figure 42: Air outlet temperature A4;B4;C4;D4;E4 (°C) (a) Set A; (b) Set B

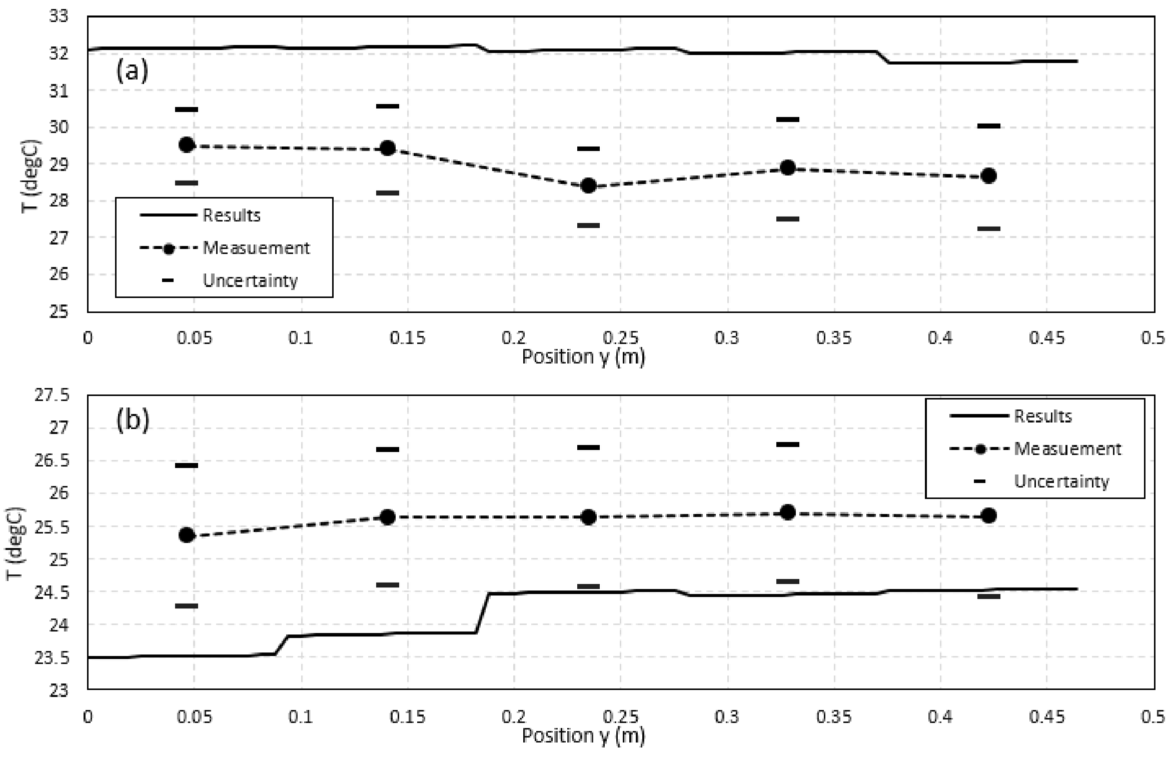


Figure 43: Air outlet temperature A5;B5;C5;D5;E5 (°C) (a) Set A; (b) Set B

14.1 Future Research

As discussed in the results Section 13, further investigation is needed for model industrialization and validation. Key tasks include improving measurement accuracy by increasing measuring points on the air side, and enhancing temperature measurement precision as per standard EN1643033 (2011).

For industrial application, it is essential to measure more characteristic data and operating points to ensure comprehensive coverage of various scenarios. Additionally, coupling the model with 3D CFD and investigating its influence on the underhood compartments are crucial steps for evaluating the developed model impact.

Furthermore, it would be appropriate to conduct studies focusing on comparing the results of the developed model with other approaches under a wider range of operating conditions. With the aim of quantifying the model's limitations and benefits.

15 Contributions

First-author

Schmid, M. (2019). Windshield Defrost Simplified CFD Model. *Production Engineering Archives*, **25**(25), 8-11.

Schmid, M., Pascenko, P., Bozkurt, F. and Petrzela, P. (2021). Importance of three- dimensional vehicle heat exchanger modeling, *Journal of Thermal Science and Engi- neering Applications* **14**(7): 7

Schmid, M., Tomek, P., and Hanus, P. (2022). Multi-physical contact simulation in vehicle applications. *Production Engineering Archives*, **28**(4), 369-374.

Co-author

Schmidova, E., Bozkurt, F., Schmid, M., and Culek, B. (2016). LOCAL ELASTIC-PLASTIC RESPONSE OF WELDING JOINTS OF DOMEX700MS STEEL. *In METAL 2016: 25th International Conference on Metallurgy and Materials.* TANGER, spol. s ro.

Kaya, U., Schmidova, E., Schmid, M., and Culek, B. (2016). Rolling-Contact Hardening Evaluated by Indentation.*In Defect and Diffusion Forum*,(Vol. **2368**, pp. 11-14). Trans Tech Publications Ltd.

Schmidova, E., Hojka, P., Culek, B., Klejch, F., and Schmid, M. (2019). Dynamic strength and anisotropy of DMLS manufactured maraging steel. *Komunikacie: Communications (Scientific Letters of the University of Zilina)*, volume **21**, issue: 3.

Schmidova, E., Kumar, M. S., Schmid, M., and Bozkurt, F. (2020). Role of Nb in the failure of dual-phase steel in heterogeneous welds. *Engineering Failure Analysis*, **116**, 104708.

Pascenko, P., Schmidova, E., Culek, B., and Schmid, M. (2021). Premature failures of railway axles after repeated pressing. *Engineering Failure Analysis*, **123**, 105253.

Projects

Narodni centrum kompetence Josefa Bozka, č. TN 0100 0026; (2019-2022)

Eliminace provoznich poruch naprav kolejovych vozidel, projekt TACR, c. TH 02010542; (2017-2019)

Prediktivni udrzba kolejove dopravni cesty, DOPRAVA 2020+, č. CK02000177 (2021-2024)

Narodni centrum kompetence inzenyrstvi pozemnich vozidel Josefa Bozka, č. TN02000054 (2023-2028)

References

Admiraal, D. M. and Bullard, C. W. (1993). *Heat Transfer in Refrigerator Condensers and Evaporators*, University of Illinois.

Bandhauer, T. M., Agarwal, A. and Garimella, S. (2006). Measurement and modeling of condensation heat transfer coefficients in circular microchannels, *Journal of Heat Transfer* **128**: 1050–1059.

Bansal, P. K. and Purkayastha, B. (1998). An ntu-e model for alternative refrigerants, *International Journal of Refrigeration* **21**(5): 381–397.

Bejan, A. (1978). General criterion for rating heat-exchanger performance, *International Journal of Heat and Mass Transfer* **21**(5): 655–658.

Bes, T. (1996). Thermal performances of codirected cross-flow heat exchangers, *Heat and mass transfer* **31**(4): 215–222.

Bowman, R. A., Mueller, A. C. and Nagle, W. M. (1940). Mean temperature difference in design, *AME* **62**(4): 283–294.

Cavallini, A., Censi, G., Col, D. D., Doretti, Longo, G. A., Rossetto, L. and Zilio, C. (2003). Experimental investigation on condensation heat transfer coefficient inside multi-port minichannels, *International Conference on Nanochannels, Microchannels, and Minichannels* **36673**: 691–698.

Crippa, L., Mattei, M., Polidoro, F., Kleinclaus, C., abd B Xu, Y. R. and Alajbegovic, A. (2011). Design and analysis of computer experiments (dace) and 3d computational fluid dynamics (cfd) simulations for agricultural tractor cooling concept, *Vehicle Thermal Management System Conference and Exhibition* **VTMS10**: 459–471.

Dai, Y., Wang, J. and Gao, L. (2009). Parametric optimization and comparative study of organic rankine cycle (orc) for low grade waste heat recovery, *Energy Conversion and Management* **50**(28): 576–582.

Digiovanni, M. A. and Webb, R. L. (1989). Uncertainty in effectiveness-ntu calculations for crossflow heat exchangers, *Heat Transfer Engineering* **10**(3): 61–70.

Ding, W., Williams, J. and Karanth, D. (2006). Cfd application in automotive front-end design, *SAE Technical Paper* **115**(6): 244–252.

Dittus, F. W. and Boelter, L. M. K. (1985). Heat transfer in automobile radiators of the tubular type, *International communications in heat and mass transfer* **12**(1): 3–22.

EN1643033 (2011). *Fan assisted radiators, convectors and trench convectors Part 3 Test method and rating of cooling capacity*, Standard.

Fernandez-Seara, J., Uhia, F. J., Sieres, J. and Campo, A. (2007). A general review of the wilson plot method and its modifications to determine convection coefficients in heat exchange devices, *Applied Thermal Engineering* **27**(17-18): 2745–2757.

Fluent, A. (2009). *Theory Guide Release 12.0*, Ansys.

Freon, 134a, Refrigerant and ONLINE (2023). Freon 134a refrigerant (r-134a) thermodynamic properties (si units), *https://www.freon.com/en/-/media/files/freon/freon-134a-si-thermodynamic-properties.pdf?rev=7519d264dfd74fe68c04c9e119f7361f* .

Gamma Technologies, I. (2014). *GT-Suite User's Manual*, Gamma Technologies, Inc. Westmont IL.

Gao, T., Geer, J. and Sammakia, B. (2014). Nonuniform temperature boundary condition effects on data center cross flow heat exchanger dunamic performance, *International Journal of Heat and Mass Transfer* **79**: 1048–1058.

Garcia, C. J. M. M. M. (1998). Modelling of plate finned tube evaporators and condensers working with r134a, *International Journal of Refrigeration* **21**(4): 273–284.

Garcia-Cascales, J. R., Vera-Garcia, F., Gonzalvez-Macia, J., Corberan-Salvador, J. M., Johnson, M. W. and Kohler, G. T. (2010). Compact heat exchangers modeling: Condensation, *International Journal of Refrigeration* **33**(1): 135–147.

Gonzalvez-Macia, M. B. S. J. M. C. J. (2013). Numerical model for microchannel condensers and gas coolers: Part ii–simulation studies and model comparison, *International journal of refrigeration* **36**(1): 191–202.

Hahn, G. J. and Hendrickson, R. W. (1971). A table of percentage points of the distribution of the largest absolute value of k student t variates and its applications, *Biometrika* **58**(2): 323–332.

Haselden, G. G. and Chen, J. (1994). A computer simulation program for mixed-refrigerant air conditioning, *Rev. Int. Froid* **17**(5): 334–342.

Hu, H., Zhang, R., Zhuang, D., Ding, G. and Xiang, L. (2015). Numerical model of two-phase refrigerant flow distribution in a plate evaporator with distributors, *Applied Thermal Engineering* **75**: 167–176.

Huang, L., Aute, V. and Radermacher, R. (2012). A generalized effectiveness-ntu based variable geometry microchannel heat exchanger model, *International Refrigeration and Air Conditioning* **Conference**(1206).

Huang, L., Aute, V. and Radermacher, R. (2014). A model for air-to-refrigerant microchannel condensers with variable tube and fin geometries, *International Journal of Refrigeration* **40**: 269–281.

Incropera, F. P., Dewitt, D. P., Bergman, T. L. and Lavine, A. S. (2011). *Fundamentals of Heat and Mass Transfer 7th edition*, John Wiley & Sons ISBN 13 978-0470-50197-9.

J902, S. (2011). *Surface Vehicle Recommended Practice*, SAE International.

Jha, K. K. and Shaik, I. (2016). Scaling model of heat exchangers in automotive air conditioning systems, *SAE Technical Paper* **No. 2016-01-0227**.

Kays, W. M. and London, A. L. (1984). *Compact Heat Exchangers 3th Edition*, Krieger Publishing Company, ISBN 1-57524-060-2.

Khot, A. R., Thombare, D. G., Gaikwad, S. P. and Adadande, A. S. (2012). Overview of radiator performance evaluation and testing, *Journal of Mechanical and Civil Engineering (IOSR-JMCE)* **2**: 07–14.

Kim, H. J. and Kim, C. J. (2008). A numerical analysis for the cooling module related to automobile air-conditioning system, *Applied Thermal Engineering* **28**(14-15): 1896–1905.

Kim, M.-H. and Bullard, C. W. (2002). Air-side performance of brazed aluminum heat exchangers under dehumidifying condition, *International Journal of refrigeration* **25**(7): 924–934.

Kim, N. H., Youn, B. and Web, R. L. (1999). Air-side heat transfer and friction correlations for plain fin-and-tube heat exchangers with staggered tube arrangements, *Journal of Heat Transfer* **131**(2): 662–667.

Kondou, C., Mishima, F., Liu, J. and Koyama, S. (2014). Condensation and evaporation of r134a, r1234ze (e) and r1234ze (z) flow in horizontal microfin tubes at higher temperature, *International Refrigeration and Air Conditioning Conference* .

Kumar, V., Shendge, S. A. and Baskar, S. (2010). Underhood thermal simulation of a small passenger vehicle with rear engine compartment to evaluate and enhance radiator performance, *No. 2010-01-0801* **SAE Technical Paper**.

König, A., Mayer, S., Nicoletti, L., Tumphart, S. and Lienkamp, M. (2022). The impact of hvac on the development of autonomous and electric vehicle concepts, *Energies* **15**: 441.

Liang, Y. Y., Liu, C. C., Li, C. Z. and Chen, J. P. (2015). Experimental and simulation study on the air side thermal hydraulic performance of automotive heat exchangers, *Applied Thermal Engineering* **87**: 305–315.

Long, Y., Li, W., Liu, J., Ca, J. and Jiang, Z. (2014). 1d/3d coupling analysis of engine cooling system, *Journal of Vibroengineering* **16**(5): 2519–2526.

Lu, J. and qiang Lu, W. (2018). Review: heat and mass transfer in porous medium, mathematic/numerical models and research directions, *International Journal of Petrochemical Science & Engineering* **14**(8).

Mao, S., Feng, Z. and Michaelides, E. E. (2010). Off-highway heavy duty truck underhood thermal analysis, *Applied Thermal Engineering* **30**(13): 1726–1733.

Martinez-Ballester, S. and Gonzalvez-Macia, J. M. C. J. (2013). Numerical model for microchannel condensers and gas coolers: Part i–model description and validatio, *International journal of refrigeration* **36**(1): 173–190.

Martínez-Ballester, S., Gonzalvez-Macia, J. M. C. J. and Domanski, P. A. (2011). Impact of classical assumptions in modelling a microchannel gas cooler, *international journal of refrigeration* **34**(8): 1898–1910.

Mason and John, L. (1955). Heat transfer in crossflow, *ASME* pp. 801–803.

McLinden, M. O. and Radermacher, R. (1987). Methods for comparing the performance of pure and mixed refrigerants in the vapour compression cycle, *International Journal of Refrigerants* **10**(6): 10.

Minovski, B. B., Lofdahl, L. and Gullberg, P. (2015). A 1d method for transient simulations of cooling systems with non-uniform temperature and flow boundaries extracted from a 3d cfd solution, *No. 2015-01-0337* **SAE Technical Paper**.

Minqiang, P. A. N., Dehuai, Z., Yong, T. and Dongqing, C. (2009). Cfd-based study of velocity distribution among multiple parallel microchannels, *Journal of Computers* **4**(11): 1133–1138.

Miyara, A., Koyama, S. and Fujii, T. (1993). Performance evaluation of a heat pump cycle using narms by a simulation with equations of heat transfer and pressure drop, *Rev. Int. Froid* **16**(3): 161–168.

Moffat, R. J. (2014). The measurement chain and validation of experimental measurements, *Acta IMEKO* **3**(1): 16–18.

Mohammadi, M., Jovanovic, G. N. and Sharp, K. V. (2013). Numerical study of flow uniformity and pressure characteristics within a microchannel array with triangular manifolds, *Computers & Chemical Engineering* **52**: 134–144.

Moradi, I., Arash, K., Masoud, A., Zhixiong, L. and Quang-Vu, B. (2020). Three-dimensional numerical simulation of external fluid flow and heat transfer of a heat exchanger in a wind tunnel using porous media mode, *Journal of Thermal Analysis and Calorimetry* **141**(5): 1647–1667.

Moran, M. J., Shapiro, H. N., Boettner, D. D. and Bailey, M. B. (2014). *Fundamentals of engineering thermodynamics*, Wiley.

Morrison, G. and Ward, D. K. (1991). Thermodynamic properties of two alternative refrigerants: 1, 1-dichloro-2, 2, 2-trifluoroethane (r123) and 1, 1, 1, 2-tetrafluoroethane (r134a), *Fluid Phase Equilibria* **62**(1-2): 65–86.

Navarro, H. A. and Cabezas-Gomez, L. C. (2007). Effectiveness-ntu computation with a mathematical model for cross-flow heat exchangers, *Brazilian Journal of Chemical Engineering* **24**: 509–521.

Nozicka, J. (2008). *Zaklady Termomechanicky*, Ceska technika - nakladatelstvi CVUT.

Oliveira, C. M. B. P. and Wakeham, W. A. (1993). The viscosity of liquid r134a, *International journal of thermophysics* **14**: 33–44.

Ouazia, B. and Snelson, W. K. (1994). Predicting system performance of alternative refrigerants using a water-water heat pump, *ASHRAE Transactions* **CONF-9406105**: 100(2), 8.

Panchal, C. B., France, D. M. and Bell, K. J. (1992). Experimental investigation of single-phase, condensation, and flow boiling heat transfer for a spirally fluted tube, *Heat Transfer Engineering* **13**(1): 42–52.

Pang, S. C., Kalam, M. A., Masjuki, H. H., Badruddin, I. A., Ramli, R. and Hazrat, M. A. (2012). Underhood geometry modification and transient coolant temperature modelling for robust cooling neworks, *International Journal of Mechanical and Materials Engineering* **7**(3): 251–258.

Parise, J. A. R. (1986). Simulation of vapour-compression heat pumps, *Simulation* **46**(2): 71–76.

Park, Y.-G. and Jacobi, A. M. (2009). Air-side heat transfer and friction correlations for flat-tube louver-fin heat exchangers, *Journal of Heat Transfer* **131**(2).

Pavlu, J. (2012). Vyvoj vypocetniho modelu a metodiky pro vypocet kondenzatoru s minikanalky.

Pervaiz, M. M., Brewster, R. A., Ross, F., Bauer, W. and Reister, H. (1997). Numerical methodology for automotive radiator and condenser simulations, *SAE Transactions JOURNAL OF PASSENGER CARS* pp. 2475–2501.

Pistoresi, C., Fan, Y. and Luo, L. (2015). Numerical study on the improvement of flow distribution uniformity among parallel mini-channels, *Chemical Engineering and Processing: Process Intensification* **95**: 63–71.

Poling, B. E., Prausnitz, J. M. and O'Connell, J. P. (2001). *The properties of gases and liquids*, McGraw-Hill Education.

Ribando, R. J., W, O. G. and Susan, C.-S. (1997). General numerical scheme for heat exchanger thermal analysis and design, *Inc. Comput Appl Eng Educ 5* **5**(4): 231–242.

Rice, C. K. and Sand, J. B. (1990). Initial parametric results using cycles an lmtd-specified, lorenz-meutzner cycle refrigerator-freezer model, *International Refrigeration and Air Conditioning Conference* **CONF-900742-1**: 130.

Saab, S., Hetet, J.-F., Maiboom, A. and Charbonnelle, F. (2013). Impact of the under-hood opening area on the drag coefficient and the thermal performance of a vehicle, *SAE Technical Paper* **2013-01-0869**.

Saari, J. (2010). *Heat exchanger dimensioning*, Lappeenranta University of Technology.

Schmid, M., Pascenko, P., Bozkurt, F. and Petrzela, P. (2021). Importance of three-dimensional vehicle heat exchanger modeling, *Journal of Thermal Science and Engineering Applications* **14**(7): 7.

Schwentker, R. A., Winkler, J. M., Aute, V. C. and Radermacher, R. (2006). A simulation and design tool for flat tube, louvered-fin heat exchangers, *SAE Technical Paper* **No. 2006-01-1451**.

Shah, S. (2018). Integration of 1d and 3d cfd software for cabin cool down simulation, *SAE Technical Paper* **No. 2018-01-0773**.

Shui-Chang, L., Li-Fu, L. and Yong, Z. (2014). Vehicle radiatorsṕerformance calculation and improvement based on the coupling of multi-scale models simulations, *The Open Mechanical Engineering Journal* **8**(1): 636–642.

Silaipillayarputhur, K. and Al-Mughanam, T. (2018). Performance charts for multi-pass parallel cross-flow heat exchangers, *International Journal of Mechanical Engineering and Robotics Research* **7**(5): 478–482.

Silaipillayarputhur, K. and Mughanam, T. A. (2018). Performance charts for multi-pass parallel cross-flow heat exchangers, *International Journal of Mechanical Engineering and Robotics Research* **7**(5): 478–482.

Singh, V., Aute, V. and Radermacher, R. (2009). A heat exchanger model for air-to-refrigerant fin-and-tube heat exchanger with arbitrary fin sheet, *International journal of refrigeration* **32**(7): 1724–1735.

Smith, D. M. (1934). Mean temperature difference in cross flow, *Engineering (London) illustrated weekly journal* **138**(479): 374.

STAR-CCM+, S. S. (2019). *Theory Guide Release 13.04.010*, Siemens.

Stepan, B. D. G. (2012). Bisection method in higher dimensions and the efficiency number, *Periodica Polytechnica Mechanical Engineering* **56**(2): 81–86.

Stoitchkov, N. J. and Dimitrov, G. I. (1998). Effectiveness of crossflow plate heat exchanger for indirect evaporative cooling, *International journal of refrigeration* **21**(6): 463–471.

Taborek, J. (1983). *Handbook of Heat Exchanger Design*, Hemisphere.

Tatara, R. A. and Lupia, G. M. (2011). Assessing heat exchanger performance data using temperature, *International Journal of Engineering, Science and Technology* **3**(8): 1–12.

Tischer, S., Correa, C. and Deutschmann, O. (2001). Transient three-dimensional simulations of a catalytic combustion monolith using detailed models for heterogeneous and homogeneous reactions and transport phenomena, *Catalysis Today* **69**(1-4): 57–62.

Triboix, A. (2009). Exact and approximate formulas for cross flow heat exchangers with unmixed fluids, *International Communications in Heat and Mass Transfer* **36**(2): 121–124.

Tuo, H., Bielskus, A. and Hrnjak, P. (2012). Experimentally validated model of refrigerant distribution in a parallel microchannel evaporator, *SAE International Journal of Materials and Manufacturing* **5**(2): 365–374.

Vist, S. and Pettersen, J. (2004). Two-phase flow distribution in compact heat exchanger manifolds, *Experimental Thermal and Fluid Science* **28**(2-3): 209–215.

Wang, G., Gao, Q., Zhang, T. and Wang, Y. (2016). A simulation approach of underhood thermal management, *Advances in Engineering Software* **100**: 43–52.

Webb, R. L. (2007). Heat exchanger design methodology for electronic heat sinks, *Journal of Heat Transfer* pp. 899–901.

Wei, M., Fan, Y., Luo, L. and Flamant, G. (2015). Cfd-based evolutionary algorithm for the realization of target fluid flow distribution among parallel channels, *Chemical Engineering Research and Design* **100**: 341–352.

Ye, L., Tong, M. W. and Zeng, X. (2009). Design and analysis of multiple parallel-pass condensers, *International journal of refrigeration* **32**(6): 1153–1161.

Zhao, X. (1995). *PERFORMANCE OF A SINGLE-ROW HEAT*, Department of Aerospace and Mechanical Engineering.

Zheng, W., Chen, Y., Hua, N., Zhong, T. and Gong, Y. (2014). Comparative performance of an automotive air conditioning system using micro-channel condensers with and without liquid-vapor separation, *The 6th International Conference on Applied Energy* **61**: 1646–1649.

Zou, Y. and Hrnjak, P. S. (2013). Refrigerant distribution in the vertical header of the microchannel heat exchanger–measurement and visualization of r410a flow, *International Journal of Refrigeration* **36**(8): 2196–2208.

Zou, Y. and Hrnjak, P. S. (2014). Effects of fluid properties on two-phase flow and refrigerant distribution in the vertical header of a reversible microchannel heat exchanger–comparing r245fa, r134a, r410a, and r32, *Applied Thermal Engineering* **70**(1): 966–976.

Zou, Y., Tuo, H. and Hrnjak, P. S. (2014). Modeling refrigerant maldistribution in microchannel heat exchangers with vertical headers based on experimentally developed distribution results, *Applied thermal engineering* **64**(1-2): 172–181.

A AppendixA

Appendix A contains a photograph of the developed and used test equipment.

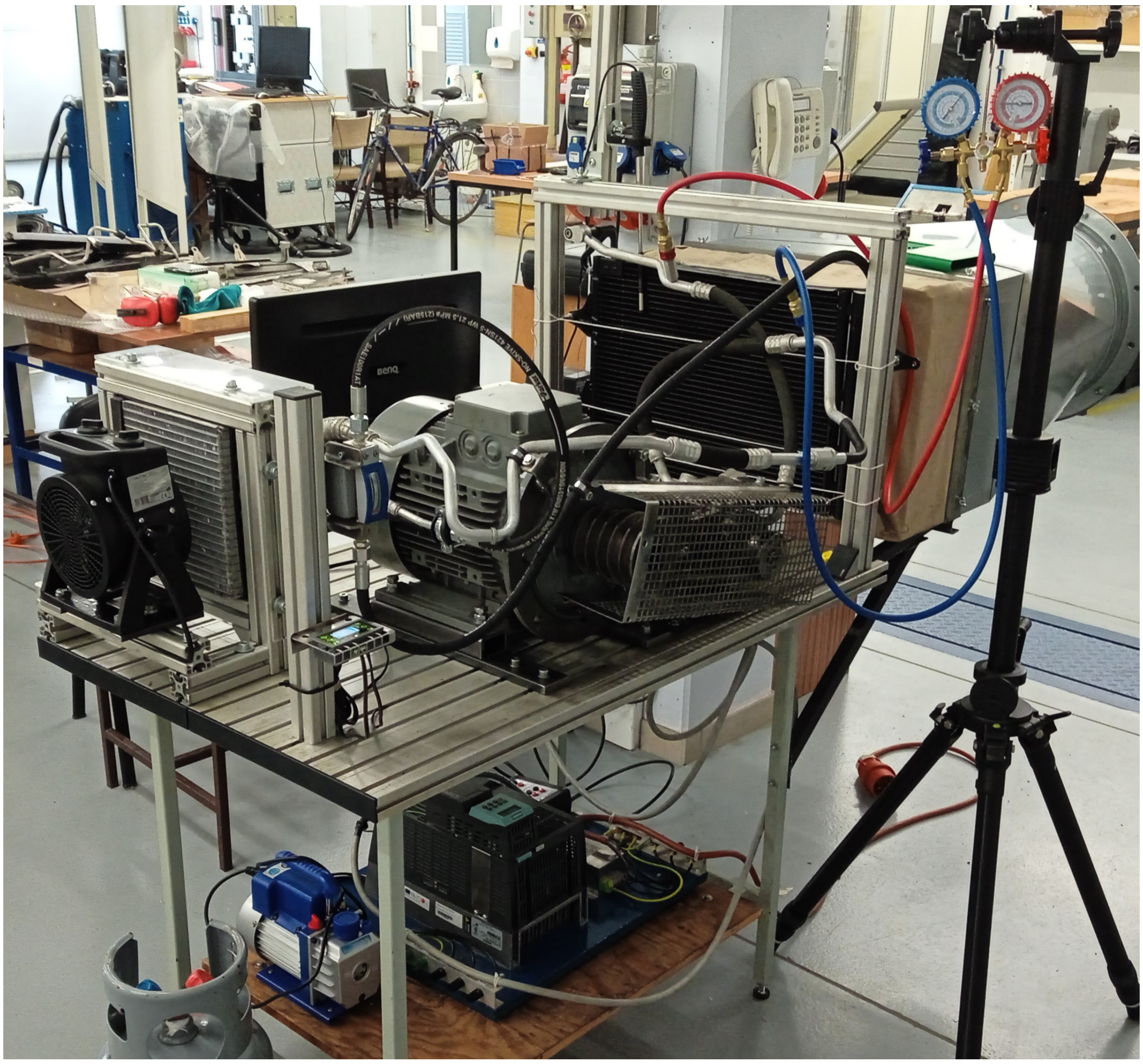


Figure 44: Test equipment view 1

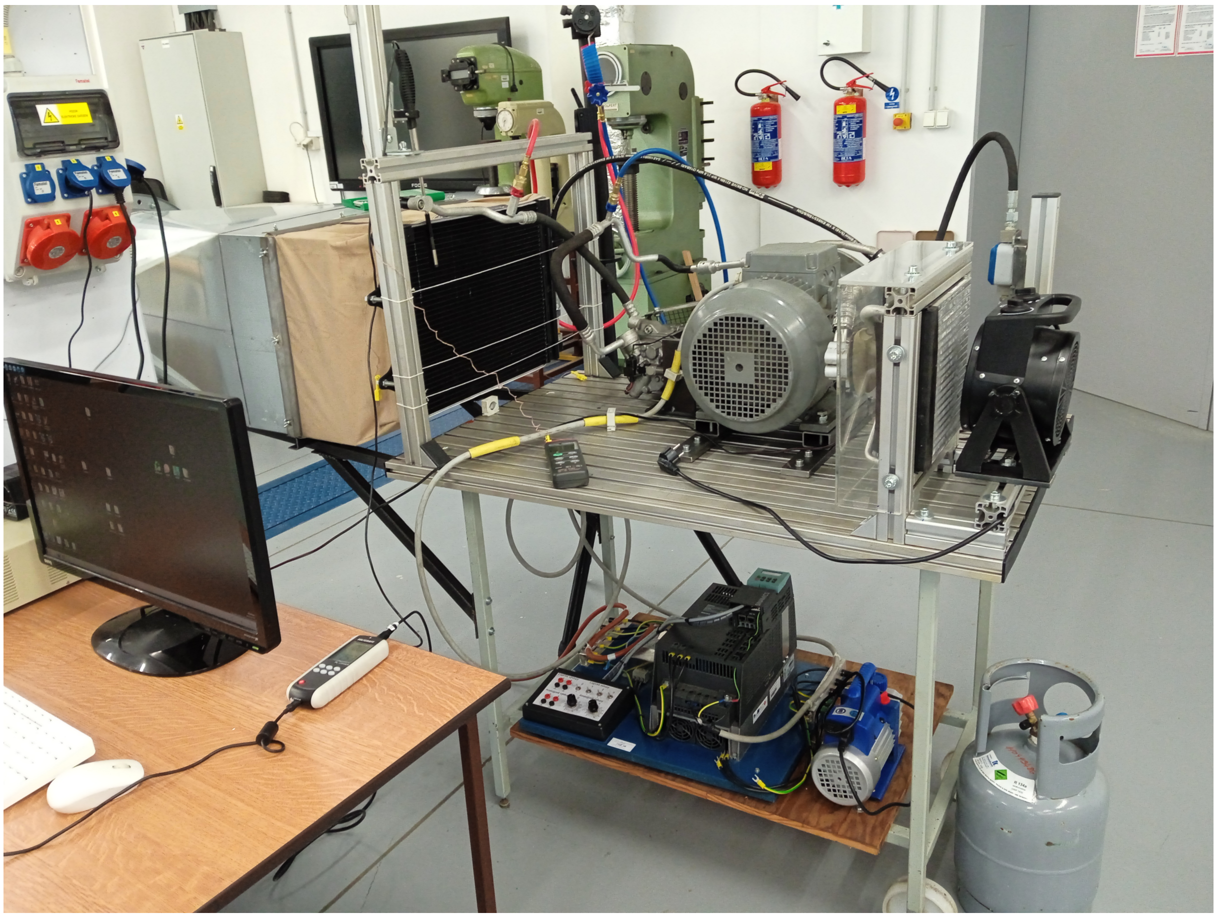


Figure 45: Test equipment view 2

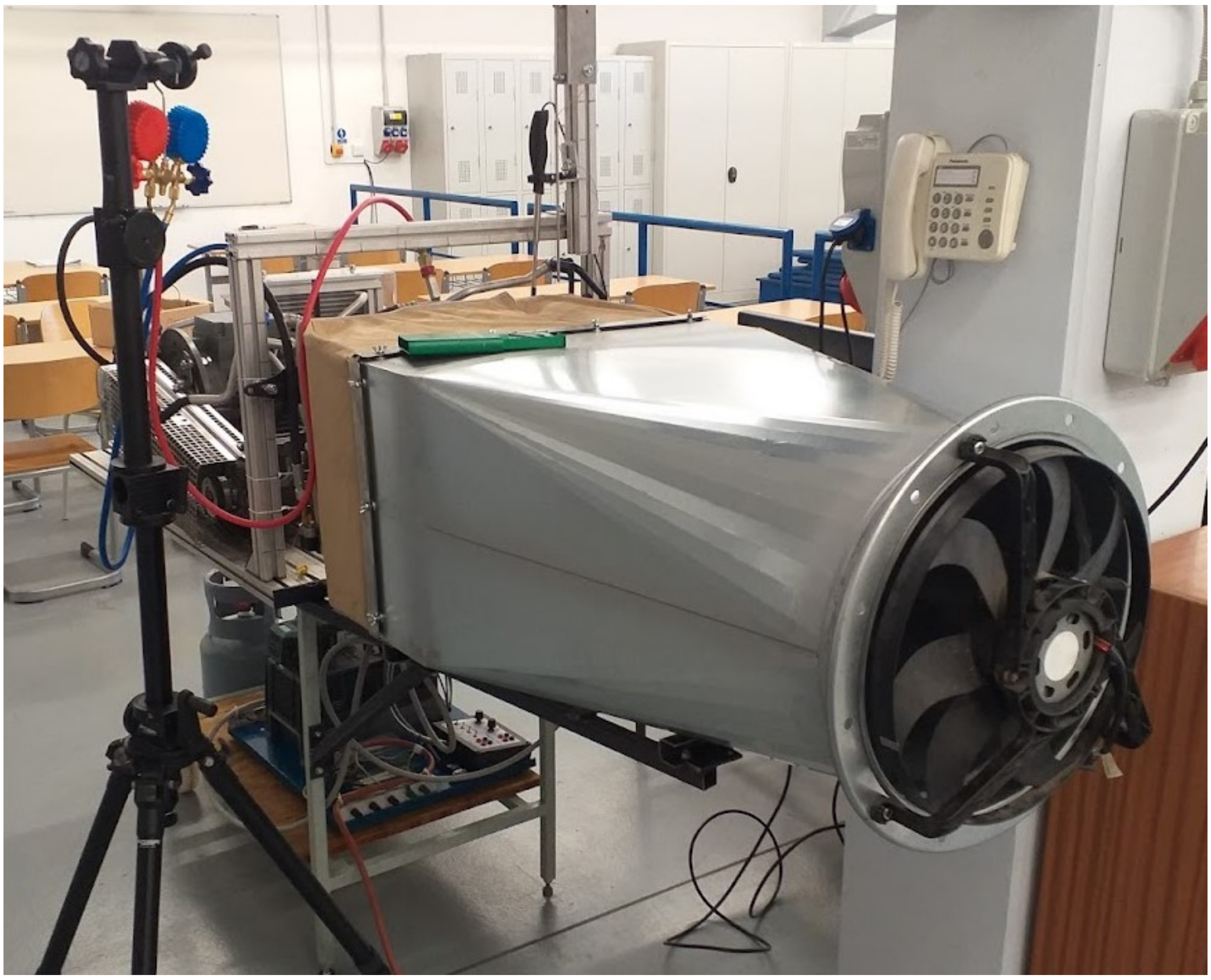


Figure 46: Test equipment view 3

B AppendixB

Appendix B contains a photograph of the evaluated type of condenser and its decomposition, which was conducted to determine the dimensions of the passes and the internal flow arrangement of refrigerant.

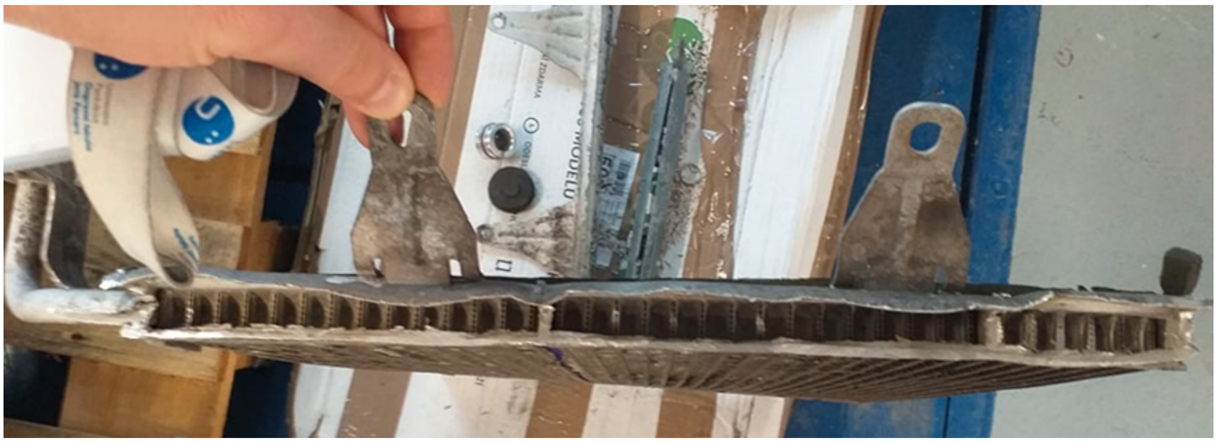


Figure 47: Condenser decomposition

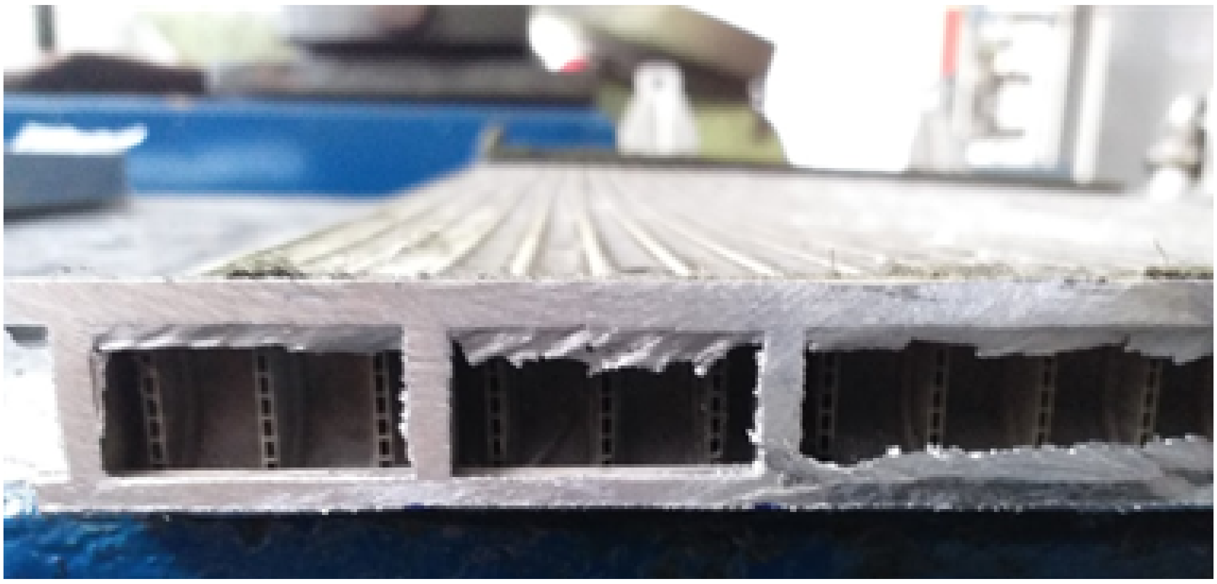
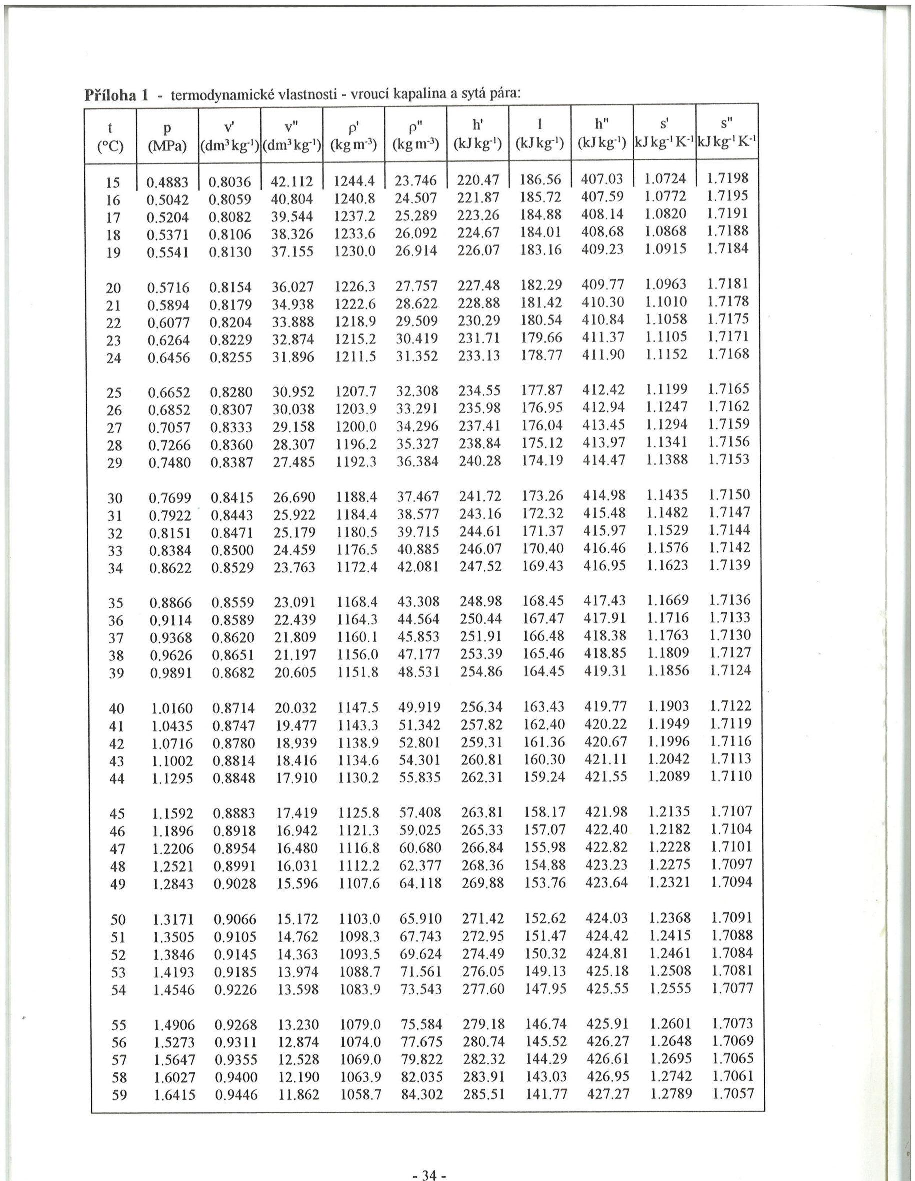


Figure 48: Condenser decomposition detail

C AppendixC

Appendix C contains the thermodynamic properties of R134a refrigerant, as shown in Figure 49 Nozicka (2008).



Příloha 1 - termodynamické vlastnosti - vroucí kapalina a sytá pára:

t (°C)	p (MPa)	v' ($dm^3 kg^{-1}$)	v" ($dm^3 kg^{-1}$)	ρ' ($kg\,m^{-3}$)	ρ" ($kg\,m^{-3}$)	h' ($kJ\,kg^{-1}$)	l ($kJ\,kg^{-1}$)	h" ($kJ\,kg^{-1}$)	s' $kJ\,kg^{-1}K^{-1}$	s" $kJ\,kg^{-1}K^{-1}$
15	0.4883	0.8036	42.112	1244.4	23.746	220.47	186.56	407.03	1.0724	1.7198
16	0.5042	0.8059	40.804	1240.8	24.507	221.87	185.72	407.59	1.0772	1.7195
17	0.5204	0.8082	39.544	1237.2	25.289	223.26	184.88	408.14	1.0820	1.7191
18	0.5371	0.8106	38.326	1233.6	26.092	224.67	184.01	408.68	1.0868	1.7188
19	0.5541	0.8130	37.155	1230.0	26.914	226.07	183.16	409.23	1.0915	1.7184
20	0.5716	0.8154	36.027	1226.3	27.757	227.48	182.29	409.77	1.0963	1.7181
21	0.5894	0.8179	34.938	1222.6	28.622	228.88	181.42	410.30	1.1010	1.7178
22	0.6077	0.8204	33.888	1218.9	29.509	230.29	180.54	410.84	1.1058	1.7175
23	0.6264	0.8229	32.874	1215.2	30.419	231.71	179.66	411.37	1.1105	1.7171
24	0.6456	0.8255	31.896	1211.5	31.352	233.13	178.77	411.90	1.1152	1.7168
25	0.6652	0.8280	30.952	1207.7	32.308	234.55	177.87	412.42	1.1199	1.7165
26	0.6852	0.8307	30.038	1203.9	33.291	235.98	176.95	412.94	1.1247	1.7162
27	0.7057	0.8333	29.158	1200.0	34.296	237.41	176.04	413.45	1.1294	1.7159
28	0.7266	0.8360	28.307	1196.2	35.327	238.84	175.12	413.97	1.1341	1.7156
29	0.7480	0.8387	27.485	1192.3	36.384	240.28	174.19	414.47	1.1388	1.7153
30	0.7699	0.8415	26.690	1188.4	37.467	241.72	173.26	414.98	1.1435	1.7150
31	0.7922	0.8443	25.922	1184.4	38.577	243.16	172.32	415.48	1.1482	1.7147
32	0.8151	0.8471	25.179	1180.5	39.715	244.61	171.37	415.97	1.1529	1.7144
33	0.8384	0.8500	24.459	1176.5	40.885	246.07	170.40	416.46	1.1576	1.7142
34	0.8622	0.8529	23.763	1172.4	42.081	247.52	169.43	416.95	1.1623	1.7139
35	0.8866	0.8559	23.091	1168.4	43.308	248.98	168.45	417.43	1.1669	1.7136
36	0.9114	0.8589	22.439	1164.3	44.564	250.44	167.47	417.91	1.1716	1.7133
37	0.9368	0.8620	21.809	1160.1	45.853	251.91	166.48	418.38	1.1763	1.7130
38	0.9626	0.8651	21.197	1156.0	47.177	253.39	165.46	418.85	1.1809	1.7127
39	0.9891	0.8682	20.605	1151.8	48.531	254.86	164.45	419.31	1.1856	1.7124
40	1.0160	0.8714	20.032	1147.5	49.919	256.34	163.43	419.77	1.1903	1.7122
41	1.0435	0.8747	19.477	1143.3	51.342	257.82	162.40	420.22	1.1949	1.7119
42	1.0716	0.8780	18.939	1138.9	52.801	259.31	161.36	420.67	1.1996	1.7116
43	1.1002	0.8814	18.416	1134.6	54.301	260.81	160.30	421.11	1.2042	1.7113
44	1.1295	0.8848	17.910	1130.2	55.835	262.31	159.24	421.55	1.2089	1.7110
45	1.1592	0.8883	17.419	1125.8	57.408	263.81	158.17	421.98	1.2135	1.7107
46	1.1896	0.8918	16.942	1121.3	59.025	265.33	157.07	422.40	1.2182	1.7104
47	1.2206	0.8954	16.480	1116.8	60.680	266.84	155.98	422.82	1.2228	1.7101
48	1.2521	0.8991	16.031	1112.2	62.377	268.36	154.88	423.23	1.2275	1.7097
49	1.2843	0.9028	15.596	1107.6	64.118	269.88	153.76	423.64	1.2321	1.7094
50	1.3171	0.9066	15.172	1103.0	65.910	271.42	152.62	424.03	1.2368	1.7091
51	1.3505	0.9105	14.762	1098.3	67.743	272.95	151.47	424.42	1.2415	1.7088
52	1.3846	0.9145	14.363	1093.5	69.624	274.49	150.32	424.81	1.2461	1.7084
53	1.4193	0.9185	13.974	1088.7	71.561	276.05	149.13	425.18	1.2508	1.7081
54	1.4546	0.9226	13.598	1083.9	73.543	277.60	147.95	425.55	1.2555	1.7077
55	1.4906	0.9268	13.230	1079.0	75.584	279.18	146.74	425.91	1.2601	1.7073
56	1.5273	0.9311	12.874	1074.0	77.675	280.74	145.52	426.27	1.2648	1.7069
57	1.5647	0.9355	12.528	1069.0	79.822	282.32	144.29	426.61	1.2695	1.7065
58	1.6027	0.9400	12.190	1063.9	82.035	283.91	143.03	426.95	1.2742	1.7061
59	1.6415	0.9446	11.862	1058.7	84.302	285.51	141.77	427.27	1.2789	1.7057

- 34 -

Figure 49: R134a Nozicka (2008)

D AppendixD

Appendix D contains the source code of the developed model implemented in *GNU Octave* 5.2.0.

```
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||        Geometry - HTX Domain Mesh        ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  xcells=4;
  ycells=50;
  dx=0.004;
  dy=0.470/50;
  dz=0.010342857;
  %Dimensions y=0.470 z=0.362 x=0.016

  zcells_Pass1=14;
  zcells_Pass2=10;
  zcells_Pass3=5;
  zcells_Pass4=3;
  zcells_Pass5=3;

  zcells=zcells_Pass1+zcells_Pass2+zcells_Pass3+zcells_Pass4+zcells_Pass5;
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||       Variables                          ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  HTX_VPrim= zeros (xcells,ycells,zcells);
  HTX_MdotPrim= zeros (xcells,ycells,zcells);
  HTX_DenPrim= zeros (xcells,ycells,zcells);
  HTX_CpPrim= zeros (xcells,ycells,zcells);
  HTX_TPrimIN= zeros (xcells,ycells,zcells);
  HTX_TPrimOUT= zeros (xcells,ycells,zcells);

  HTX_MdotAux= zeros (xcells,ycells,zcells);
  HTX_DenAux= zeros (xcells,ycells,zcells);
  HTX_CpAux= zeros (xcells,ycells,zcells);
  HTX_TAuxIN= zeros (xcells,ycells,zcells);
  HTX_TAuxOUT= zeros (xcells,ycells,zcells);
  HTX_hAuxIN=zeros (xcells,ycells,zcells);
  HTX_hAuxOUT=zeros (xcells,ycells,zcells);

  HTX_Cmin=zeros (xcells,ycells,zcells);
  HTX_Cmax=zeros (xcells,ycells,zcells);
  HTX_Cr=zeros (xcells,ycells,zcells);
  HTX_NTU=zeros (xcells,ycells,zcells);
  HTX_e=zeros (xcells,ycells,zcells);
  HTX_q=zeros (xcells,ycells,zcells);
  HTX_Phase=zeros (xcells,ycells,zcells);

  HTX_Cmin_GasPhase=zeros (xcells,ycells,zcells);
  HTX_Cmax_GasPhase=zeros (xcells,ycells,zcells);
  HTX_Cr_GasPhase=zeros (xcells,ycells,zcells);
  HTX_NTU_GasPhase=zeros (xcells,ycells,zcells);
  HTX_e_GasPhase=zeros (xcells,ycells,zcells);
  HTX_Cmin_LiquidPhase=zeros (xcells,ycells,zcells);
  HTX_Cmax_LiquidPhase=zeros (xcells,ycells,zcells);
  HTX_Cr_LiquidPhase=zeros (xcells,ycells,zcells);
  HTX_NTU_LiquidPhase=zeros (xcells,ycells,zcells);
  HTX_e_LiquidPhase=zeros (xcells,ycells,zcells);
  HTX_NTU_DualPhase=zeros (xcells,ycells,zcells);
  HTX_e_DualPhase=zeros (xcells,ycells,zcells);
  HTX_LatentHeat_DualPhase=zeros (xcells,ycells,zcells);

  x =zeros(xcells,1);
  y =zeros(1,ycells);
  z =zeros(1,zcells);

%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||        Material properties                ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  AuxCp_GasPhase=1010.704;
  AuxDensity_GasPhase=33.358;

  AuxCp_LiquidPhase=1432.144;
  AuxDensity_LiquidPhase=1197.954;
  AuxSpecificLatentHeat=187424.159;

  %Ideal Gas
  R=287.058;

%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||        Input Data - Measurement             ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  AuxMdot=0.018093;

  qIN=3423.0961;
  %+/-562.7923(W)

  Tsaturated=27.876+273.15;
  latentQDualPhase=AuxSpecificLatentHeat;
  AmbientPressure=101325;

  %Primary Inlet Profile
  HTX_CpPrim(:,:,:)=1005.743;

  HTX_VPrim_Measurement=[1.599,1.625,1.627,1.622,1.570;
  1.761,1.767,1.680,1.721,1.695;1.757,1.604,1.699,1.695,1.700;
1.768,1.696,1.567,1.792,1.648;1.573,1.583,1.657,1.726,1.593;];
  HTX_TPrim_Measurement=[24.245,24.392,24.203,24.746,24.635;
23.900,24.100,25.126,25.432,24.999;
23.876,24.040,24.559,24.445,25.026;
23.869,23.802,24.393,24.345,25.700;
23.657,23.452,23.816,24.069,25.683;];
  HTX_TPrim_Measurement=HTX_TPrim_Measurement.+273.15;
  HTX_VPrim_Measurement=HTX_VPrim_Measurement';
  HTX_TPrim_Measurement=HTX_TPrim_Measurement';

  Measurement_Grid_Size=size (HTX_VPrim_Measurement);
  StepY=ycells/Measurement_Grid_Size(1,1);
  StepZ=zcells/Measurement_Grid_Size(1,2);

  for j_measured=1:1:Measurement_Grid_Size(1,1)
   for k_measured=1:1:Measurement_Grid_Size(1,2)
       for j=(1+(j_measured-1)*StepY):1:(j_measured*StepY)
        for k=(zcells-(k_measured-1)*StepZ):-1:(1+zcells-(k_measured)*StepZ)
          HTX_VPrim(:,j,k)=HTX_VPrim_Measurement(j_measured,k_measured);
          HTX_TPrimIN(:,j,k)=HTX_TPrim_Measurement(j_measured,k_measured);
        endfor
      endfor
   endfor
  endfor

  HTX_DenPrim(:,:,:)= AmbientPressure./(R.*HTX_TPrimIN(:,:,:));
  HTX_MdotPrim(:,:,:)=HTX_DenPrim(:,:,:).*HTX_VPrim(:,:,:)*dy*dz;

  %Mdot distribution according to passes
  HTX_MdotAux(:,:,1:zcells_Pass5)=   AuxMdot/((xcells)*(zcells_Pass5));
  HTX_MdotAux(:,:,zcells_Pass5+1:(zcells_Pass5+zcells_Pass4))= AuxMdot/((xcells)*(zcells_Pass4));
  HTX_MdotAux(:,:,(zcells_Pass5+zcells_Pass4+1):(zcells_Pass5+zcells_Pass4+zcells_Pass3))= AuxMdot/((xcells)*(zcells_Pass3));
  HTX_MdotAux(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3+1):
  (zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2))=
  = AuxMdot/((xcells)*(zcells_Pass2));
  HTX_MdotAux(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):
  :(zcells))
  = AuxMdot/((xcells)*(zcells_Pass1));

%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||        QTables - Measurement                ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  %QTable Gas phase
  QAreaPrim_GasPhase=145*70/1000000;
  QAreaAux_GasPhase=xcells*dx*145/1000;
  QVolume_GasPhase=QAreaPrim_GasPhase*xcells*dx;
  QPrimInletT_GasPhase=24.42+273.15;
  QPrimCp_GasPhase=1005.743;
  QAuxInletT_GasPhase=40.645+273.15;
  QAuxCp_GasPhase=AuxCp_GasPhase;
  Q_GasPhase=[247.382];
  QAuxMdot_GasPhase=[0.018093];
  QPrimMdot_GasPhase=[0.01404557];

  QCmin_GasPhase=min (QAuxMdot_GasPhase*QAuxCp_GasPhase,QPrimMdot_GasPhase*QPrimCp_GasPhase);
```

```
QCmax_GasPhase=max (QAuxMdot_GasPhase*QAuxCp_GasPhase,QPrimMdot_GasPhase*QPrimCp_GasPhase);
Qeq_GasPhase=Q_GasPhase./(QCmin_GasPhase*(QAuxInletT_GasPhase-QPrimInletT_GasPhase));
QCr_GasPhase=QCmin_GasPhase./QCmax_GasPhase;

%Limmmiting Effectiveness
for i=1:1:(length(QPrimMdot_GasPhase))
  for j =1:1:(length(QAuxMdot_GasPhase))
    if (Qeq_GasPhase(i,j)>1)
      Qeq_GasPhase(i,j)=0.999;
      disp("Effectiveness_Limited_in_Gas_Phase!")
    endif
  endfor
endfor

%Newton-Raphson
NTU_GasPhase=Q_GasPhase./Q_GasPhase./100; %initialization
Converged=0;
do
  fx= 1 - exp((-1 ./ QCr_GasPhase) .* (NTU_GasPhase .^ 0.22) .* (1 - e .^ (-QCr_GasPhase .* NTU_GasPhase .^ 0.78)))
  -Qeq_GasPhase;
  fxderivated = (0.78 *e.^ (((-1.+exp(-(QCr_GasPhase.* NTU_GasPhase.^0.78))).* NTU_GasPhase.^0.22)
  ./ QCr_GasPhase.-QCr_GasPhase.*NTU_GasPhase.^0.78).* (-0.282051+0.282051.*
  exp(QCr_GasPhase.*NTU_GasPhase.^0.78).+QCr_GasPhase.* NTU_GasPhase.^0.78))./(QCr_GasPhase.*NTU_GasPhase.^ 0.78);

  NTUnext_GasPhase = NTU_GasPhase - fx ./ fxderivated;
  if (abs(NTUnext_GasPhase - NTU_GasPhase)  < 1e-3 )
    Converged = 1;
  else
    Converged = 0;
  endif
  NTU_GasPhase = NTUnext_GasPhase;
  %disp ("Iterating Newton-Rhapson for Gass Phase of Q-Table")
until (Converged==1)

%QTable Liquid phase
QAreaPrim_LiquidPhase=31*60/1000000;
QAreaAux_LiquidPhase=xcells*dx*145/1000;
QVolume_LiquidPhase=QAreaPrim_LiquidPhase*xcells*dx;
QPrimInletT_LiquidPhase=25.692+273.15;
QPrimCp_LiquidPhase=1005.7426;
QAuxInletT_LiquidPhase=27.626+273.15;
QAuxCp_LiquidPhase=148.4*1000;
Q_LiquidPhase=[1];
QAuxMdot_LiquidPhase=[0.018093];
QPrimMdot_LiquidPhase=[0.003603415];

QCmin_LiquidPhase=min (QAuxMdot_LiquidPhase*QAuxCp_LiquidPhase,QPrimMdot_LiquidPhase*QPrimCp_LiquidPhase);
QCmax_LiquidPhase=max (QAuxMdot_LiquidPhase*QAuxCp_LiquidPhase,QPrimMdot_LiquidPhase*QPrimCp_LiquidPhase);
Qeq_LiquidPhase=Q_LiquidPhase./(QCmin_LiquidPhase*(QAuxInletT_LiquidPhase-QPrimInletT_LiquidPhase));
QCr_LiquidPhase=QCmin_LiquidPhase./QCmax_LiquidPhase;

%Limmmiting Effectiveness
  for i=1:1:(length(QPrimMdot_LiquidPhase))
    for j =1:1:(length(QAuxMdot_LiquidPhase))
      if (Qeq_LiquidPhase(i,j)>1)
        Qeq_LiquidPhase(i,j)=0.999;
        disp("Effectiveness_Limited_in_Liquid_Phase!")
      endif
    endfor
   endfor

%Newton-Raphson
NTU_LiquidPhase=Q_LiquidPhase./Q_LiquidPhase./100; %initialization
Converged=0;
fx=0;

do
  fx= 1 - exp((-1 ./ QCr_LiquidPhase) .*
  (NTU_LiquidPhase .^ 0.22).*
  (1 - e .^ (-QCr_LiquidPhase .* NTU_LiquidPhase .^ 0.78)))
  -Qeq_LiquidPhase;
  fxderivated = (0.78 * e.^ (((-1.+
  exp(-(QCr_LiquidPhase.* NTU_LiquidPhase.^ 0.78))).*
  NTU_LiquidPhase.^ 0.22)./ QCr_LiquidPhase.- QCr_LiquidPhase.*
  NTU_LiquidPhase.^ 0.78).* (-0.282051 + 0.282051.*
  exp(QCr_LiquidPhase.* NTU_LiquidPhase.^ 0.78).+
  QCr_LiquidPhase.* NTU_LiquidPhase.^ 0.78)).
  / (QCr_LiquidPhase.* NTU_LiquidPhase .^ 0.78);

  NTUnext_LiquidPhase = NTU_LiquidPhase - fx ./ fxderivated;
```

```
    if (abs(NTUnext_LiquidPhase - NTU_LiquidPhase)  < 1e-5 )
      Converged = 1;
    else
      Converged = 0;
    endif
    NTU_LiquidPhase = NTUnext_LiquidPhase;
    %disp ("Iterating Newton-Rhapson for Liquid Phase of Q-Table")
  until (Converged==1)


  %QTable Dual phase

  QAreaPrim_DualPhase=0.15999;
  QAreaAux_DualPhase=xcells*dx*145/1000;
  QVolume_DualPhase=QAreaPrim_DualPhase*xcells*dx;
  QPrimInletT_DualPhase=25.6915+273.15;
  QPrimCp_DualPhase=1005.7426;
  QAuxInletT_DualPhase=Tsaturated;
  QAuxInletH_DualPhase=413.9081*1000;
  QPrimInletH_DualPhase=25.81996*1000;
  QAuxOutletH_DualPhase=238.6617*1000;
  QAuxSpecificLatentHeat=AuxSpecificLatentHeat;
  QAuxMdot_DualPhase=[0.018093];
  Q_DualPhase = [3175.714];
  QPrimMdot_DualPhase=[0.315570055];
  QCmin_DualPhase=min (QAuxMdot_DualPhase
  (QPrimMdot_DualPhase*QPrimCp_GasPhase)/((QAuxInletH_DualPhase
  QPrimInletH_DualPhase)/(QAuxInletT_DualPhase
  QPrimInletT_DualPhase)));
  QCmax_DualPhase=max (QAuxMdot_DualPhase
  (QPrimMdot_DualPhase*QPrimCp_GasPhase)/((QAuxInletH_DualPhase
  QPrimInletH_DualPhase)/(QAuxInletT_DualPhase
  QPrimInletT_DualPhase)));
  QCr_DualPhase=0;
  Qeq_DualPhase=Q_DualPhase./(QCmin_DualPhase.*(QAuxInletH_DualPhase
  QPrimInletH_DualPhase));

  %Limmmiting Effectiveness
  for i=1:1:(length(QPrimMdot_DualPhase))
    for j =1:1:(length(QAuxMdot_DualPhase))
      if (Qeq_DualPhase(i,j)>1)
         Qeq_DualPhase(i,j)=0.999;
         disp("Effectiveness␣Limited␣in␣Dual␣Phase!")
      endif
    endfor
  endfor
  %Cmax=infinity -> Cr=0
  NTU_DualPhase = -log(1-Qeq_DualPhase);

%||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%|||||||||||||||||||||||||          NTU operating point and local        ||||||||||||||||||||||||
%||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  %Gas Phase
  HTX_MdotPrimNTU_GasPhase=HTX_MdotPrim.*(QAreaPrim_GasPhase/(dy.*dz));
  HTX_MdotAuxNTU_GasPhase=HTX_MdotAux.*(QAreaAux_GasPhase/(dx.*dz));

  %Max/Min Interpolation bounds
  for k =1:1:zcells
    for i=1:1:xcells
      for j =1:1:ycells
        if ((HTX_MdotPrimNTU_GasPhase(i,j,k))<(min (QPrimMdot_GasPhase)))
          HTX_MdotPrimNTU_GasPhase(i,j,k)=min (QPrimMdot_GasPhase);
          %disp ("Min data used for Primary Inlet Gas Phase");
        endif
          if ((HTX_MdotPrimNTU_GasPhase(i,j,k))>(max (QPrimMdot_GasPhase)))
          HTX_MdotPrimNTU_GasPhase(i,j,k)=max (QPrimMdot_GasPhase);
          %disp ("Max data used for Primary Inlet Gas Phase");
        endif
      endfor
     endfor
   endfor
   for k =1:1:zcells
    for i=1:1:xcells
      for j =1:1:ycells
        if ((HTX_MdotAuxNTU_GasPhase(i,j,k))<(min (QAuxMdot_GasPhase)))
          HTX_MdotAuxNTU_GasPhase(i,j,k)=min (QAuxMdot_GasPhase);
          %disp ("Min data used for Axuliary Inlet Gas Phase");
        endif
          if ((HTX_MdotAuxNTU_GasPhase(i,j,k))>(max (QAuxMdot_GasPhase)))
          HTX_MdotAuxNTU_GasPhase(i,j,k)=max (QAuxMdot_GasPhase);
          %disp ("Max data used for Auxiliary Inlet Gas Phase");
        endif
      endfor
    endfor
  endfor

  NTUg_GasPhase=NTU_GasPhase;
  %griddata(QAuxMdot_GasPhase,QPrimMdot_GasPhase,
  NTU_GasPhase,HTX_MdotAuxNTU_GasPhase,HTX_MdotPrimNTU_GasPhase);
  %Linear Interpolation
  Cming_GasPhase=min (HTX_MdotAuxNTU_GasPhase*QAuxCp_GasPhase
  HTX_MdotPrimNTU_GasPhase*QPrimCp_GasPhase);
  Cmaxg_GasPhase=max (HTX_MdotAuxNTU_GasPhase*QAuxCp_GasPhase,
  HTX_MdotPrimNTU_GasPhase*QPrimCp_GasPhase);
  Crg_GasPhase=Cming_GasPhase./Cmaxg_GasPhase;
  eg_GasPhase=1-exp(-((NTUg_GasPhase.^0.22)./Crg_GasPhase)
  .*(1-e.^(-Crg_GasPhase.*(NTUg_GasPhase.^0.78))));
  HTX_Cmin_GasPhase=min(HTX_MdotAux.*AuxCp_GasPhase,HTX_MdotPrim.*HTX_CpPrim);
  HTX_Cmax_GasPhase=max(HTX_MdotAux.*AuxCp_GasPhase,HTX_MdotPrim.*HTX_CpPrim);
  HTX_Cr_GasPhase=HTX_Cmin_GasPhase./HTX_Cmax_GasPhase;
  HTX_NTU_GasPhase=NTUg_GasPhase.*((dx.*dy.*dz.*Cming_GasPhase)./(HTX_Cmin_GasPhase.*QVolume_GasPhase));
  HTX_e_GasPhase=1-exp(-((HTX_NTU_GasPhase.^0.22)./HTX_Cr_GasPhase).*(1-e.^(-HTX_Cr_GasPhase.*(HTX_NTU_GasPhase.^0.78))));


  %Liquid Phase
  HTX_MdotPrimNTU_LiquidPhase=HTX_MdotPrim.*(QAreaPrim_LiquidPhase/(dy.*dz));
  HTX_MdotAuxNTU_LiquidPhase=HTX_MdotAux.*(QAreaAux_LiquidPhase/(dx.*dz));

  %Max/Min Interpolation bounds
  for k =1:1:zcells
    for i=1:1:xcells
      for j =1:1:ycells
        if ((HTX_MdotPrimNTU_LiquidPhase(i,j,k))<(min (QPrimMdot_LiquidPhase)))
          HTX_MdotPrimNTU_LiquidPhase(i,j,k)=min (QPrimMdot_LiquidPhase);
          %disp ("Min data used for Primary Inlet Gas Phase");
        endif
          if ((HTX_MdotPrimNTU_LiquidPhase(i,j,k))>(max (QPrimMdot_LiquidPhase)))
          HTX_MdotPrimNTU_LiquidPhase(i,j,k)=max (QPrimMdot_LiquidPhase);
          %disp ("Max data used for Primary Inlet Gas Phase");
        endif
      endfor
    endfor
  endfor
  for k =1:1:zcells
   for i=1:1:xcells
     for j =1:1:ycells
       if ((HTX_MdotAuxNTU_LiquidPhase(i,j,k))<(min (QAuxMdot_LiquidPhase)))
         HTX_MdotAuxNTU_LiquidPhase(i,j,k)=min (QAuxMdot_LiquidPhase);
         %disp ("Min data used for Axuliary Inlet Gas Phase");
       endif
         if ((HTX_MdotAuxNTU_LiquidPhase(i,j,k))>(max (QAuxMdot_LiquidPhase)))
         HTX_MdotAuxNTU_LiquidPhase(i,j,k)=max (QAuxMdot_LiquidPhase);
         %disp ("Max data used for Auxiliary Inlet Gas Phase");
       endif
     endfor
   endfor
  endfor

  NTUg_LiquidPhase=NTU_LiquidPhase;
%griddata(QAuxMdot_LiquidPhase,QPrimMdot_LiquidPhase,
NTU_LiquidPhase,HTX_MdotAuxNTU_LiquidPhase,
HTX_MdotPrimNTU_LiquidPhase);  %Linear Interpolation
  Cming_LiquidPhase=min (HTX_MdotAuxNTU_LiquidPhase*QAuxCp_LiquidPhase,HTX_MdotPrimNTU_LiquidPhase*QPrimCp_LiquidPhase);
  Cmaxg_LiquidPhase=max (HTX_MdotAuxNTU_LiquidPhase*QAuxCp_LiquidPhase,HTX_MdotPrimNTU_LiquidPhase*QPrimCp_LiquidPhase);
  Crg_LiquidPhase=Cming_LiquidPhase./Cmaxg_LiquidPhase;
  eg_LiquidPhase=1-exp(-((NTUg_LiquidPhase.^0.22)./Crg_LiquidPhase).*(1-e.^(-Crg_LiquidPhase.*(NTUg_LiquidPhase.^0.78))));
  HTX_Cmin_LiquidPhase=min (HTX_MdotAux.*AuxCp_LiquidPhase,HTX_MdotPrim.*HTX_CpPrim);
  HTX_Cmax_LiquidPhase=max (HTX_MdotAux.*AuxCp_LiquidPhase,HTX_MdotPrim.*HTX_CpPrim);
  HTX_Cr_LiquidPhase=HTX_Cmin_LiquidPhase./HTX_Cmax_LiquidPhase;
  HTX_NTU_LiquidPhase=NTUg_LiquidPhase.*((dx.*dy.*dz.*Cming_LiquidPhase)./(HTX_Cmin_LiquidPhase.*QVolume_LiquidPhase));
  HTX_e_LiquidPhase=1-exp(-((HTX_NTU_LiquidPhase.^0.22).
  /HTX_Cr_LiquidPhase).*(1-e.^(-HTX_Cr_LiquidPhase.*(HTX_NTU_LiquidPhase.^0.78))));

  %Dual Phase
  HTX_MdotPrimNTU_DualPhase=HTX_MdotPrim.*(QAreaPrim_DualPhase/(dy.*dz));
  HTX_MdotAuxNTU_DualPhase=HTX_MdotAux.*(QAreaAux_DualPhase/(dx.*dz));
  for k =1:1:zcells
    for i=1:1:xcells
      for j =1:1:ycells
```

```
        if ((HTX_MdotPrimNTU_DualPhase(i,j,k))<(min (QPrimMdot_DualPhase)))
          HTX_MdotPrimNTU_DualPhase(i,j,k)=min (QPrimMdot_DualPhase);
          %disp ("Min data used for Primary Inlet");
        endif
          if ((HTX_MdotPrimNTU_DualPhase(i,j,k))>(max (QPrimMdot_DualPhase)))
          HTX_MdotPrimNTU_DualPhase(i,j,k)=max (QPrimMdot_DualPhase);
          %disp ("Max data used for Primary Inlet");
        endif
      endfor
     endfor
   endfor
    for k =1:1:zcells
    for i=1:1:xcells
      for j =1:1:ycells
        if ((HTX_MdotAuxNTU_DualPhase(i,j,k))<(min (QAuxMdot_LiquidPhase)))
          HTX_MdotAuxNTU_DualPhase(i,j,k)=min (QAuxMdot_LiquidPhase);
          %disp ("Min data used for Axuliary Inlet Gas Phase");
        endif
          if ((HTX_MdotAuxNTU_DualPhase(i,j,k))>(max (QAuxMdot_LiquidPhase)))
          HTX_MdotAuxNTU_DualPhase(i,j,k)=max (QAuxMdot_LiquidPhase);
          %disp ("Max data used for Auxiliary Inlet Gas Phase");
        endif
      endfor
     endfor
   endfor

  NTUg_DualPhase=NTU_DualPhase;
%griddata(QAuxMdot_DualPhase,QPrimMdot_DualPhase,
NTU_DualPhase,HTX_MdotAuxNTU_DualPhase,
HTX_MdotPrimNTU_DualPhase);         %Linear Interpolation
  Cming_DualPhase=min (HTX_MdotAuxNTU_DualPhase,(HTX_MdotPrimNTU_DualPhase*QPrimCp_DualPhase)
  /((QAuxInletH_DualPhase-QPrimInletH_DualPhase)/(QAuxInletT_DualPhase-QPrimInletT_DualPhase)));
  HTX_Cmin_DualPhase=min (HTX_MdotAux,(HTX_MdotPrim.*HTX_CpPrim)./
  (((-(HTX_TPrimIN*1005).+QAuxInletH_DualPhase))./(-HTX_TPrimIN.+QAuxInletT_DualPhase)));
  eg_DualPhase=1-exp(-NTUg_DualPhase);
  HTX_NTU_DualPhase=NTUg_DualPhase.*((dx.*dy.*dz.*Cming_DualPhase)./(HTX_Cmin_DualPhase.*QVolume_DualPhase));
  HTX_e_DualPhase=1-exp(-HTX_NTU_DualPhase);

%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%||||||||||||||||||||||||||         Solver                             ||||||||||||||||||||||||
%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

  %Solver Setup
  BisectionInterval=1000;
  BisectionUpperLimit=Tsaturated+BisectionInterval;
  BisectionLowerLimit=Tsaturated;
  AuxInletT=(Tsaturated+BisectionUpperLimit)/2;

  ConTarg=1;   %Accuracy/Error

  %Initialization
  QdualReal=0;

  HTX_TPrimOUT(:,:,:)=HTX_TPrimIN(:,:,:);
  HTX_DenAux(:,:,:)=AuxDensity_GasPhase;
  HTX_CpAux(:,:,:)=AuxCp_GasPhase;
  HTX_TAuxIN(:,:,:)= AuxInletT;
  HTX_TAuxOUT(:,:,:)= AuxInletT;
  HTX_hAuxIN(:,:,:)= QAuxInletH_DualPhase;
  HTX_hAuxOUT(:,:,:)= QAuxInletH_DualPhase;

  TAuxIN=mean(mean (HTX_TAuxIN(:,:,:)));
  TAuxOUT=mean(mean (HTX_TAuxOUT(:,:,:)));
  TPrimIN=mean(mean (HTX_TPrimIN(:,:,:)));
  TPrimOUT=mean(mean (HTX_TPrimOUT(:,:,:)));
  iteration=0;

do
  Converged = 0;
  HTX_Cmin=HTX_Cmin_GasPhase;
  HTX_Cmax=HTX_Cmax_GasPhase;
  HTX_Cr=HTX_Cr_GasPhase;
  HTX_NTU=HTX_NTU_GasPhase;
  HTX_e=HTX_e_GasPhase;
  QdualReal=0;
  HTX_Phase_Pass1=1;
  HTX_Phase_Pass2=1;
  HTX_Phase_Pass3=1;
  HTX_Phase_Pass4=1;
  HTX_qDualPhaseSum_Pass1=0;
```

```
HTX_qDualPhaseSum_Pass2=0;
HTX_qDualPhaseSum_Pass3=0;
HTX_qDualPhaseSum_Pass4=0;
HTX_qDualPhaseSum=zeros (xcells,ycells,zcells);
PhaseChangeStart_j=zeros (xcells,zcells);
TAuxOUT_Pass1=0;
TAuxOUT_Pass2=0;
TAuxOUT_Pass3=0;
TAuxOUT_Pass4=0;
hAuxOUT_Pass1=0;
hAuxOUT_Pass2=0;
hAuxOUT_Pass3=0;
hAuxOUT_Pass4=0;
HTX_Phase(:,:,:)=1;  % 1 = Gass ; 2 = Dual ; 3 = Liquid
HTX_q(:,:,:)=0;


%1st Pass
for j =1:1:ycells
  y(1,j)=j;
    for i=1:1:xcells
      x(i,1)=i;
      for k =zcells:-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1)
        z(1,k)=k;
        if (j==1)
          if (i==1)
              if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=AuxInletT;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)=-HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                  HTX_Phase(i,j:ycells,k)=2;
                  PhaseChangeStart_j(i,k)=j;
                endif
              endif
              if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=AuxInletT;
                HTX_hAuxIN(i,j,k)=QAuxInletH_DualPhase;


                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));
                HTX_qDualPhaseSum(i,j,k)=sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k)))));

                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);


                if (((HTX_qDualPhaseSum(i,j,k))/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                  HTX_Phase(i,j:ycells,k)=3;
                  QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=AuxInletT;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
              endif
          else
```

```
     if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
        HTX_CpAux(i,j,k)=AuxCp_GasPhase;

         HTX_TAuxIN(i,j,k)=AuxInletT;
         HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
         HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
         HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
         HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:ycells,k)=2;
          PhaseChangeStart_j(i,k)=j;
        endif
      endif
      if(HTX_Phase(i,j,k)==2)
        HTX_Cmin=HTX_Cmin_DualPhase;
        HTX_NTU=HTX_NTU_DualPhase;
        HTX_e=HTX_e_DualPhase;

        HTX_TAuxIN(i,j,k)=AuxInletT;
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
        HTX_hAuxIN(i,j,k)=QAuxInletH_DualPhase;

        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

        HTX_qDualPhaseSum(i,j,k)=  sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k)))));

        HTX_TAuxOUT(i,j,k)=HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

        if (((HTX_qDualPhaseSum(i,j,k))/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
          HTX_Phase(i,j:ycells,k)=3;
          QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
        endif
      endif

      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
        HTX_e=HTX_e_LiquidPhase;
        HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

        HTX_TAuxIN(i,j,k)=AuxInletT;
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
      endif
  endif
endif

if (j>1)
  if (i==1)
      if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
        HTX_CpAux(i,j,k)=AuxCp_GasPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:ycells,k)=2;
          PhaseChangeStart_j(i,k)=j;
        endif
      endif
      if(HTX_Phase(i,j,k)==2)
```

```
        HTX_Cmin=HTX_Cmin_DualPhase;
        HTX_NTU=HTX_NTU_DualPhase;
        HTX_e=HTX_e_DualPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);

        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

        HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

        HTX_qDualPhaseSum(i,j,k)=  sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k)))));

        if (((HTX_qDualPhaseSum(i,j,k))/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
          HTX_Phase(i,j:ycells,k)=3;
          QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
        endif
      endif
      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
        HTX_e=HTX_e_LiquidPhase;
        HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
       endif
else
       if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
        HTX_CpAux(i,j,k)=AuxCp_GasPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:ycells,k)=2;
          PhaseChangeStart_j(i,k)=j;
        endif
      endif
     if(HTX_Phase(i,j,k)==2)
        HTX_Cmin=HTX_Cmin_DualPhase;
        HTX_NTU=HTX_NTU_DualPhase;
        HTX_e=HTX_e_DualPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

        HTX_qDualPhaseSum(i,j,k)=  sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k)))));

        HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

        if (((HTX_qDualPhaseSum(i,j,k))/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
          HTX_Phase(i,j:ycells,k)=3;
          QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
        endif
      endif
      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
```

```
                HTX_e=HTX_e_LiquidPhase;
                HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
             endif
         endif
       endif
     endfor
   endfor
 endfor

%Tank Mixinng

HTX_Phase_Pass1=mean (mean (mean(HTX_Phase(:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells)))));
     if (HTX_Phase_Pass1 <= 1.1)
     HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2):-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1))
     =1;TAuxOUT_Pass1=mean (mean (mean(HTX_TAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells)))));
     hAuxOUT_Pass1=mean (mean (mean(HTX_hAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells)))));
     HTX_qDualPhaseSum_Pass1=0;
     endif
     if HTX_Phase_Pass1 > 1.1
      if HTX_Phase_Pass1 < 2.9
        HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2):
        :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1))=2;
        TAuxOUT_Pass1=Tsaturated;
        hAuxOUT_Pass1=mean (mean (mean(HTX_hAuxOUT(:,ycells,(zcells_Pass5+zcells_Pass4+
        +zcells_Pass3+zcells_Pass2+1):(zcells)))));
        HTX_qDualPhaseSum_Pass1=(sum (sum (sum(HTX_qDualPhaseSum(:,ycells,
        (zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells))))))/(xcells*zcells_Pass2);
        QdualReal=0;
      endif
     endif
     if HTX_Phase_Pass1 >= 2.9
     HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2):-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1))=3;
     TAuxOUT_Pass1=mean (mean (mean(HTX_TAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells)))));
     HTX_qDualPhaseSum_Pass1=0;
     endif

%Inlet Temparature OPT
TAuxIN=mean (mean (HTX_TAuxIN(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2+1):(zcells))));
%2nd Pass
for j =ycells:-1:1
  y(1,j)=j;
    for i=1:1:xcells
      x(i,1)=i;
      for k =(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2):-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)
        z(1,k)=k;
        if (j==ycells)
          if (i==1)
              if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)=-HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                  HTX_Phase(i,j:-1:1,k)=2;
                  PhaseChangeStart_j(i,k)=j;
                endif
              endif
              if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
      HTX_hAuxIN(i,j,k)=hAuxOUT_Pass1;

      HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

      HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
      HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
      HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

      if (PhaseChangeStart_j(i,k)==0)
        PhaseChangeStart_j(i,k)=j;
      endif

      HTX_qDualPhaseSum(i,j,k)= (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));


      if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass1)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
        HTX_Phase(i,j:-1:1,k)=3;
        QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
      endif

    endif
    if(HTX_Phase(i,j,k)==3)
      HTX_Cmin=HTX_Cmin_LiquidPhase;
      HTX_Cmax=HTX_Cmax_LiquidPhase;
      HTX_Cr=HTX_Cr_LiquidPhase;
      HTX_NTU=HTX_NTU_LiquidPhase;
      HTX_e=HTX_e_LiquidPhase;
      HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

      HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
      HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
      HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
      HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
    endif
else
   if (HTX_Phase(i,j,k)==1)
      HTX_Cmin=HTX_Cmin_GasPhase;
      HTX_Cmax=HTX_Cmax_GasPhase;
      HTX_Cr=HTX_Cr_GasPhase;
      HTX_NTU=HTX_NTU_GasPhase;
      HTX_e=HTX_e_GasPhase;
       HTX_CpAux(i,j,k)=AuxCp_GasPhase;

       HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
       HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
       HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
       HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
       HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

      if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
        HTX_Phase(i,j:-1:1,k)=2;
        PhaseChangeStart_j(i,k)=j;
      endif
    endif
    if(HTX_Phase(i,j,k)==2)
      HTX_Cmin=HTX_Cmin_DualPhase;
      HTX_NTU=HTX_NTU_DualPhase;
      HTX_e=HTX_e_DualPhase;

      HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
      HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
      HTX_hAuxIN(i,j,k)=hAuxOUT_Pass1;

      HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

      HTX_TAuxOUT(i,j,k)=HTX_TAuxIN(i,j,k);
      HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
      HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

       if (PhaseChangeStart_j(i,k) ==0)
        PhaseChangeStart_j(i,k)=j;
      endif

      HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));


      if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass1)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
        HTX_Phase(i,j:-1:1,k)=3;
        QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
      endif
      endif
      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
        HTX_e=HTX_e_LiquidPhase;
         HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

        HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
      endif
  endif
endif

if (j<ycells)
  if (i==1)
      if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
         HTX_CpAux(i,j,k)=AuxCp_GasPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:-1:1,k)=2;
          PhaseChangeStart_j(i,k)=j;
        endif
      endif
      if(HTX_Phase(i,j,k)==2)
        HTX_Cmin=HTX_Cmin_DualPhase;
        HTX_NTU=HTX_NTU_DualPhase;
        HTX_e=HTX_e_DualPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
        HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j+1,k);

        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

        if (PhaseChangeStart_j(i,k) ==0)
          PhaseChangeStart_j(i,k)=j;
        endif

        HTX_qDualPhaseSum(i,j,k)=(sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));

        HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

        if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass1)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
          HTX_Phase(i,j:-1:1,k)=3;
          QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
        endif
      endif
      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
        HTX_e=HTX_e_LiquidPhase;
         HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
       endif
else
       if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
```

```
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                 HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                HTX_Phase(i,j:-1:1,k)=2;
                PhaseChangeStart_j(i,k)=j;
                endif
              endif
             if(HTX_Phase(i,j+1,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif

                HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));


                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass1)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                    HTX_Phase(i,j:-1:1,k)=3;
                    QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                 HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
             endif
        endif
      endif
    endfor
  endfor
endfor


%Tank Mixinng

 HTX_Phase_Pass2=mean (mean (mean(HTX_Phase(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
:-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)))));
    if HTX_Phase_Pass2 <=1.1
    HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1))=1;
    TAuxOUT_Pass2=mean (mean (mean(HTX_TAuxOUT(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
    :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)))));
    hAuxOUT_Pass2=mean (mean (mean(HTX_hAuxOUT(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
    :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)))));
    HTX_qDualPhaseSum_Pass2=0;
    endif
    if HTX_Phase_Pass2 > 1.1
     if HTX_Phase_Pass2 < 2.9
       HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1))=2;
       TAuxOUT_Pass2=Tsaturated;
```

```
          hAuxOUT_Pass2=mean (mean (mean(HTX_hAuxOUT(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
          :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)))));
          HTX_qDualPhaseSum_Pass2=(sum (sum (sum(HTX_qDualPhaseSum(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
          :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1))))))/(xcells*zcells_Pass3);
          QdualReal=0;
         endif
        endif
        if HTX_Phase_Pass2 >= 2.9
        HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1))=3;
        TAuxOUT_Pass2=mean (mean (mean(HTX_TAuxOUT(:,1,(zcells_Pass5+zcells_Pass4+zcells_Pass3+zcells_Pass2)
        :-1:(zcells_Pass5+zcells_Pass4+zcells_Pass3+1)))));
        HTX_qDualPhaseSum_Pass2=0;
        endif


%3nd Pass
for j =1:1:ycells
  y(1,j)=j;
    for i=1:1:xcells
      x(i,1)=i;
      for k =(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1)
        z(1,k)=k;
        if (j==1)
          if (i==1)
              if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                 HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)=-HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                  HTX_Phase(i,j:ycells,k)=2;
                  PhaseChangeStart_j(i,k)=j;
                endif
              endif
              if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                HTX_hAuxIN(i,j,k)=hAuxOUT_Pass2;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif

                HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass2)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                  HTX_Phase(i,j:ycells,k)=3;
                  QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
              endif
          else
             if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;

                 HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                 HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                 HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                 HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                 HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                  HTX_Phase(i,j:ycells,k)=2;
                  PhaseChangeStart_j(i,k)=j;
                endif
              endif
              if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                HTX_hAuxIN(i,j,k)=hAuxOUT_Pass2;
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                HTX_TAuxOUT(i,j,k)=HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)=HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)=HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif

                HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass2)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                  HTX_Phase(i,j:ycells,k)=3;
                  QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass2;
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
              endif
          endif
        endif

        if (j>1)
          if (i==1)
              if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                 HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                  HTX_Phase(i,j:ycells,k)=2;
                  PhaseChangeStart_j(i,k)=j;
                endif
              endif
              if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif

                HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass2)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                  HTX_Phase(i,j:ycells,k)=3;
                  QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif

              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                 HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
               endif
        else
               if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                 HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                HTX_Phase(i,j:ycells,k)=2;
                PhaseChangeStart_j(i,k)=j;
                endif
              endif
             if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*( HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif

                HTX_qDualPhaseSum(i,j,k)= (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass2)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
```

```
                    HTX_Phase(i,j:ycells,k)=3;
                    QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
             endif
        endif
      endif
    endfor
  endfor
endfor

%Tank Mixinng

 HTX_Phase_Pass3=mean (mean (mean(HTX_Phase(:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:
 (zcells_Pass5+zcells_Pass4+1)))));
     if HTX_Phase_Pass3 <= 1.1
     HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1))=1;
     TAuxOUT_Pass3=mean (mean (mean(HTX_TAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1)))));
     hAuxOUT_Pass3=mean (mean (mean(HTX_hAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1)))));
     HTX_qDualPhaseSum_Pass3=0;
     endif
     if HTX_Phase_Pass3 > 1.1
      if HTX_Phase_Pass3 < 2.9
        HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1))=2;
        TAuxOUT_Pass3=Tsaturated;
        hAuxOUT_Pass3=mean (mean (mean(HTX_hAuxOUT
        (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1)))));
        HTX_qDualPhaseSum_Pass3=(sum (sum (sum(HTX_qDualPhaseSum(:,ycells,
        (zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1))))))/(xcells*zcells_Pass4);
        QdualReal=0;
      endif
     endif
     if HTX_Phase_Pass3 >= 2.9
     HTX_Phase(:,:,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1))=3;
     TAuxOUT_Pass3=mean (mean (mean(HTX_TAuxOUT
     (:,ycells,(zcells_Pass5+zcells_Pass4+zcells_Pass3):-1:(zcells_Pass5+zcells_Pass4+1)))));
     HTX_qDualPhaseSum_Pass3=0;
     endif



%4nd Pass
for j =ycells:-1:1
  y(1,j)=j;
    for i=1:1:xcells
      x(i,1)=i;
      for k =(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)
        z(1,k)=k;
        if (j==ycells)
          if (i==1)
              if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                 HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=TAuxOUT_Pass3;
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)=-HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                HTX_Phase(i,j:-1:1,k)=2;
          PhaseChangeStart_j(i,k)=j;
          endif
        endif
        if(HTX_Phase(i,j,k)==2)
          HTX_Cmin=HTX_Cmin_DualPhase;
          HTX_NTU=HTX_NTU_DualPhase;
          HTX_e=HTX_e_DualPhase;

          HTX_TAuxIN(i,j,k)=TAuxOUT_Pass3;
          HTX_hAuxIN(i,j,k)=hAuxOUT_Pass3;

          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

          HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
          HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

          if (PhaseChangeStart_j(i,k) ==0)
            PhaseChangeStart_j(i,k)=j;
          endif

          HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));


          if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass3)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
            HTX_Phase(i,j:-1:1,k)=3;
            QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
          endif
        endif
        if(HTX_Phase(i,j,k)==3)
          HTX_Cmin=HTX_Cmin_LiquidPhase;
          HTX_Cmax=HTX_Cmax_LiquidPhase;
          HTX_Cr=HTX_Cr_LiquidPhase;
          HTX_NTU=HTX_NTU_LiquidPhase;
          HTX_e=HTX_e_LiquidPhase;
           HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

          HTX_TAuxIN(i,j,k)=TAuxOUT_Pass3;
          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
          HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        endif
    else
       if (HTX_Phase(i,j,k)==1)
          HTX_Cmin=HTX_Cmin_GasPhase;
          HTX_Cmax=HTX_Cmax_GasPhase;
          HTX_Cr=HTX_Cr_GasPhase;
          HTX_NTU=HTX_NTU_GasPhase;
          HTX_e=HTX_e_GasPhase;
           HTX_CpAux(i,j,k)=AuxCp_GasPhase;

           HTX_TAuxIN(i,j,k)=TAuxOUT_Pass3;
           HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
           HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
           HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
           HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

          if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
            HTX_Phase(i,j:-1:1,k)=2;
            PhaseChangeStart_j(i,k)=j;
          endif
        endif
        if(HTX_Phase(i,j,k)==2)
          HTX_Cmin=HTX_Cmin_DualPhase;
          HTX_NTU=HTX_NTU_DualPhase;
          HTX_e=HTX_e_DualPhase;

          HTX_TAuxIN(i,j,k)=TAuxOUT_Pass3;
          HTX_hAuxIN(i,j,k)=hAuxOUT_Pass3;
          HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

          HTX_TAuxOUT(i,j,k)=HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
          HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

          if (PhaseChangeStart_j(i,k) ==0)
            PhaseChangeStart_j(i,k)=j;
          endif
```

```
            HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));

            if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass3)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
              HTX_Phase(i,j:-1:1,k)=3;
              QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
            endif
          endif
          if(HTX_Phase(i,j,k)==3)
            HTX_Cmin=HTX_Cmin_LiquidPhase;
            HTX_Cmax=HTX_Cmax_LiquidPhase;
            HTX_Cr=HTX_Cr_LiquidPhase;
            HTX_NTU=HTX_NTU_LiquidPhase;
            HTX_e=HTX_e_LiquidPhase;
             HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

            HTX_TAuxIN(i,j,k)=TAuxOUT_Pass1;
            HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
            HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
            HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
            HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
          endif
      endif
    endif

    if (j<ycells)
      if (i==1)
          if (HTX_Phase(i,j,k)==1)
            HTX_Cmin=HTX_Cmin_GasPhase;
            HTX_Cmax=HTX_Cmax_GasPhase;
            HTX_Cr=HTX_Cr_GasPhase;
            HTX_NTU=HTX_NTU_GasPhase;
            HTX_e=HTX_e_GasPhase;
             HTX_CpAux(i,j,k)=AuxCp_GasPhase;

            HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
            HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
            HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
            HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

            if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
              HTX_Phase(i,j:-1:1,k)=2;
              PhaseChangeStart_j(i,k)=j;
            endif
          endif
          if(HTX_Phase(i,j,k)==2)
            HTX_Cmin=HTX_Cmin_DualPhase;
            HTX_NTU=HTX_NTU_DualPhase;
            HTX_e=HTX_e_DualPhase;

            HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
            HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j+1,k);

            HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

            HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
            HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
            HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

            if (PhaseChangeStart_j(i,k) ==0)
              PhaseChangeStart_j(i,k)=j;
            endif

            HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));

            if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass3)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
              HTX_Phase(i,j:-1:1,k)=3;
              QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
            endif
          endif
          if(HTX_Phase(i,j,k)==3)
            HTX_Cmin=HTX_Cmin_LiquidPhase;
            HTX_Cmax=HTX_Cmax_LiquidPhase;
            HTX_Cr=HTX_Cr_LiquidPhase;
            HTX_NTU=HTX_NTU_LiquidPhase;
            HTX_e=HTX_e_LiquidPhase;
             HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

            HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
            HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
```

```
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
               endif
        else
               if (HTX_Phase(i,j,k)==1)
                HTX_Cmin=HTX_Cmin_GasPhase;
                HTX_Cmax=HTX_Cmax_GasPhase;
                HTX_Cr=HTX_Cr_GasPhase;
                HTX_NTU=HTX_NTU_GasPhase;
                HTX_e=HTX_e_GasPhase;
                HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                HTX_Phase(i,j:-1:1,k)=2;
                PhaseChangeStart_j(i,k)=j;
                endif
              endif
             if(HTX_Phase(i,j,k)==2)
                HTX_Cmin=HTX_Cmin_DualPhase;
                HTX_NTU=HTX_NTU_DualPhase;
                HTX_e=HTX_e_DualPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                if (PhaseChangeStart_j(i,k) ==0)
                  PhaseChangeStart_j(i,k)=j;
                endif
                HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,j:PhaseChangeStart_j(i,k),k))))));


                if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass3)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                  HTX_Phase(i,j:-1:1,k)=3;
                  QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                endif
              endif
              if(HTX_Phase(i,j,k)==3)
                HTX_Cmin=HTX_Cmin_LiquidPhase;
                HTX_Cmax=HTX_Cmax_LiquidPhase;
                HTX_Cr=HTX_Cr_LiquidPhase;
                HTX_NTU=HTX_NTU_LiquidPhase;
                HTX_e=HTX_e_LiquidPhase;
                 HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j+1,k);
                HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
             endif
        endif
      endif
    endfor
  endfor
endfor

%Tank Mixinng
 HTX_Phase_Pass4=mean (mean (mean(HTX_Phase(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)))));
     if HTX_Phase_Pass4 <= 1.1
     HTX_Phase(:,:,(zcells_Pass5):-1:1)=1;
     TAuxOUT_Pass4=mean (mean (mean(HTX_TAuxOUT(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)))));
     hAuxOUT_Pass4=mean (mean (mean(HTX_hAuxOUT(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)))));
     HTX_qDualPhaseSum_Pass4=0;
     endif
     if HTX_Phase_Pass4 > 1.1
      if HTX_Phase_Pass4 < 2.9
        HTX_Phase(:,:,(zcells_Pass5):-1:1)=2;
```

```
         TAuxOUT_Pass4=Tsaturated;
         hAuxOUT_Pass4=mean (mean (mean(HTX_hAuxOUT(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)))));
         HTX_qDualPhaseSum_Pass4=
         (sum (sum (sum(HTX_qDualPhaseSum(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1))))))/(xcells*zcells_Pass5);
         QdualReal=0;
        endif
       endif
       if HTX_Phase_Pass4 >=2.9
       HTX_Phase(:,:,(zcells_Pass5):-1:1)=3;
       TAuxOUT_Pass4=mean (mean (mean(HTX_TAuxOUT(:,1,(zcells_Pass5+zcells_Pass4):-1:(zcells_Pass5+1)))));
       HTX_qDualPhaseSum_Pass4=0;
       endif


%5nd Pass
 for j =1:1:ycells
   y(1,j)=j;
     for i=1:1:xcells
       x(i,1)=i;
       for k =(zcells_Pass5):-1:1
         z(1,k)=k;
         if (j==1)
           if (i==1)
               if (HTX_Phase(i,j,k)==1)
                 HTX_Cmin=HTX_Cmin_GasPhase;
                 HTX_Cmax=HTX_Cmax_GasPhase;
                 HTX_Cr=HTX_Cr_GasPhase;
                 HTX_NTU=HTX_NTU_GasPhase;
                 HTX_e=HTX_e_GasPhase;
                  HTX_CpAux(i,j,k)=AuxCp_GasPhase;

                 HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
                 HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                 HTX_TAuxOUT(i,j,k)=-HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                 HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

                 if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
                   HTX_Phase(i,j:ycells,k)=2;
                   PhaseChangeStart_j(i,k)=j;
                 endif
               endif
               if(HTX_Phase(i,j,k)==2)
                 HTX_Cmin=HTX_Cmin_DualPhase;
                 HTX_NTU=HTX_NTU_DualPhase;
                 HTX_e=HTX_e_DualPhase;

                 HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
                 HTX_hAuxIN(i,j,k)=hAuxOUT_Pass4;

                 HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

                 HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
                 HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
                 HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

                 if (PhaseChangeStart_j(i,k) ==0)
                   PhaseChangeStart_j(i,k)=j;
                 endif
                 HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));


                 if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass4)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                   HTX_Phase(i,j:ycells,k)=3;
                   QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                 endif
               endif
               if(HTX_Phase(i,j,k)==3)
                 HTX_Cmin=HTX_Cmin_LiquidPhase;
                 HTX_Cmax=HTX_Cmax_LiquidPhase;
                 HTX_Cr=HTX_Cr_LiquidPhase;
                 HTX_NTU=HTX_NTU_LiquidPhase;
                 HTX_e=HTX_e_LiquidPhase;
                 HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                 HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
                 HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                 HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                 HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
               endif
           else
     if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
         HTX_CpAux(i,j,k)=AuxCp_GasPhase;

         HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
         HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
         HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
         HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
         HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:ycells,k)=2;
          PhaseChangeStart_j(i,k)=j;
        endif
      endif
      if(HTX_Phase(i,j,k)==2)
        HTX_Cmin=HTX_Cmin_DualPhase;
        HTX_NTU=HTX_NTU_DualPhase;
        HTX_e=HTX_e_DualPhase;

        HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
        HTX_hAuxIN(i,j,k)=hAuxOUT_Pass4;
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

        HTX_TAuxOUT(i,j,k)=HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
        HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

        if (PhaseChangeStart_j(i,k) ==0)
          PhaseChangeStart_j(i,k)=j;
        endif

        HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

        if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass4)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
          HTX_Phase(i,j:ycells,k)=3;
          QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
        endif
      endif
      if(HTX_Phase(i,j,k)==3)
        HTX_Cmin=HTX_Cmin_LiquidPhase;
        HTX_Cmax=HTX_Cmax_LiquidPhase;
        HTX_Cr=HTX_Cr_LiquidPhase;
        HTX_NTU=HTX_NTU_LiquidPhase;
        HTX_e=HTX_e_LiquidPhase;
         HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

        HTX_TAuxIN(i,j,k)=TAuxOUT_Pass4;
        HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
      endif
  endif
endif

if (j>1)
  if (i==1)
      if (HTX_Phase(i,j,k)==1)
        HTX_Cmin=HTX_Cmin_GasPhase;
        HTX_Cmax=HTX_Cmax_GasPhase;
        HTX_Cr=HTX_Cr_GasPhase;
        HTX_NTU=HTX_NTU_GasPhase;
        HTX_e=HTX_e_GasPhase;
         HTX_CpAux(i,j,k)=AuxCp_GasPhase;

        HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
        HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
        HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
        HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

        if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
          HTX_Phase(i,j:ycells,k)=2;
          PhaseChangeStart_j(i,k)=j;
          endif
        endif
        if(HTX_Phase(i,j,k)==2)
          HTX_Cmin=HTX_Cmin_DualPhase;
          HTX_NTU=HTX_NTU_DualPhase;
          HTX_e=HTX_e_DualPhase;

          HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
          HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);

          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

          HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
          HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

          if (PhaseChangeStart_j(i,k) ==0)
            PhaseChangeStart_j(i,k)=j;
          endif

          HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));

          if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass4)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
            HTX_Phase(i,j:ycells,k)=3;
            QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
          endif
        endif
        if(HTX_Phase(i,j,k)==3)
          HTX_Cmin=HTX_Cmin_LiquidPhase;
          HTX_Cmax=HTX_Cmax_LiquidPhase;
          HTX_Cr=HTX_Cr_LiquidPhase;
          HTX_NTU=HTX_NTU_LiquidPhase;
          HTX_e=HTX_e_LiquidPhase;
           HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

          HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
          HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
         endif
  else
         if (HTX_Phase(i,j,k)==1)
          HTX_Cmin=HTX_Cmin_GasPhase;
          HTX_Cmax=HTX_Cmax_GasPhase;
          HTX_Cr=HTX_Cr_GasPhase;
          HTX_NTU=HTX_NTU_GasPhase;
          HTX_e=HTX_e_GasPhase;
          HTX_CpAux(i,j,k)=AuxCp_GasPhase;

          HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
          HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
          HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);

          if (HTX_TAuxOUT(i,j,k) <= Tsaturated)
            HTX_Phase(i,j:ycells,k)=2;
            PhaseChangeStart_j(i,k)=j;
          endif
        endif
       if(HTX_Phase(i,j,k)==2)
          HTX_Cmin=HTX_Cmin_DualPhase;
          HTX_NTU=HTX_NTU_DualPhase;
          HTX_e=HTX_e_DualPhase;

          HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
          HTX_hAuxIN(i,j,k)=HTX_hAuxOUT(i,j-1,k);
          HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);

          HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_hAuxIN(i,j,k)-(HTX_TPrimIN(i,j,k)*1.005));

          HTX_TAuxOUT(i,j,k)= HTX_TAuxIN(i,j,k);
          HTX_hAuxOUT(i,j,k)= HTX_hAuxIN(i,j,k)-HTX_q(i,j,k)/HTX_MdotAux(i,j,k);

          HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
          if (PhaseChangeStart_j(i,k) ==0)
            PhaseChangeStart_j(i,k)=j;
          endif
          HTX_qDualPhaseSum(i,j,k)=  (sum (sum (sum((HTX_q(i,PhaseChangeStart_j(i,k):j,k))))));
```

```
                  if (((HTX_qDualPhaseSum(i,j,k)+HTX_qDualPhaseSum_Pass4)/HTX_MdotAux(i,j,k)) >= (latentQDualPhase))
                    HTX_Phase(i,j:ycells,k)=3;
                    QdualReal=QdualReal+HTX_qDualPhaseSum(i,j,k);
                  endif
                endif
                if(HTX_Phase(i,j,k)==3)
                  HTX_Cmin=HTX_Cmin_LiquidPhase;
                  HTX_Cmax=HTX_Cmax_LiquidPhase;
                  HTX_Cr=HTX_Cr_LiquidPhase;
                  HTX_NTU=HTX_NTU_LiquidPhase;
                  HTX_e=HTX_e_LiquidPhase;
                  HTX_CpAux(i,j,k)=AuxCp_LiquidPhase;

                  HTX_TAuxIN(i,j,k)=HTX_TAuxOUT(i,j-1,k);
                  HTX_TPrimIN(i,j,k)=HTX_TPrimOUT(i-1,j,k);
                  HTX_q(i,j,k)=HTX_e(i,j,k)*HTX_Cmin(i,j,k)*(HTX_TAuxIN(i,j,k)-HTX_TPrimIN(i,j,k));
                  HTX_TAuxOUT(i,j,k)= -HTX_q(i,j,k)/(HTX_CpAux(i,j,k)*HTX_MdotAux(i,j,k))+HTX_TAuxIN(i,j,k);
                  HTX_TPrimOUT(i,j,k)= HTX_q(i,j,k)/(HTX_CpPrim(i,j,k)*HTX_MdotPrim(i,j,k))+HTX_TPrimIN(i,j,k);
               endif
          endif
        endif
      endfor
    endfor
  endfor

  %Tank Mixinng
  TAuxOUT=mean (mean (HTX_TAuxOUT(:,ycells,(1:zcells_Pass5))));

  TPrimIN=mean(mean (HTX_TPrimIN(1,:,:)));
  TPrimOUT=mean(mean (HTX_TPrimOUT(xcells,:,:)));


  iteration=iteration+1;


  q=sum (sum (sum(HTX_q(:,:,:))));

  if (abs(qIN - q)  < ConTarg )
    Converged = 1;
  else
    Converged = 0;
  endif

  if (q < qIN)
    BisectionLowerLimit=TAuxIN;
    AuxInletT=TAuxIN+(BisectionUpperLimit-TAuxIN)/2;
  else
    BisectionUpperLimit=TAuxIN;
    AuxInletT=TAuxIN-(TAuxIN-BisectionLowerLimit)/2;
  endif

 %Print
  disp ("******************␣Iterating␣3D␣HTX␣Solver␣******************")
  iteration
  qIN
  q
  TAuxIN=TAuxIN-273.15
  TAuxOUT=TAuxOUT-273.15
  TPrimIN=TPrimIN-273.15
  TPrimOUT=TPrimOUT-273.15
  TPrimOUT_MAX=max (max (max(HTX_TPrimOUT(:,:,:))))-273.15
  TPrimOUT_Min=min (min (min(HTX_TPrimOUT(:,:,:))))-273.15
  QdualReal
until (Converged==1)

%||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%||||||||||||||||||||||||||         Post Processing                     ||||||||||||||||||||||||
%||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

%Print CLI

disp("******************␣Setup␣Summary␣******************")
FrontalArea=dy*dz*(ycells)*(zcells)
PrimMdot=sum (sum (sum(HTX_MdotPrim(1,:,:))))
AuxMdot
qIN
Tsaturated
latentQDualPhase
qDualPhaseTheoretical=latentQDualPhase*AuxMdot
```

```
AmbientPressure
AuxCp_GasPhase
AuxDensity_GasPhase
AuxCp_LiquidPhase
AuxDensity_LiquidPhase
AuxSpecificLatentHeat


%UI
Umean=mean (mean (HTX_VPrim(1,:,:)))
Tmean=mean (mean (HTX_TPrimIN(1,:,:)))
UI1=0;
UIT1=0;
for j =1:1:ycells
  y(1,j)=j;
  for k =1:1:zcells
    z(1,k)=k;
    UI1=UI1+abs((HTX_VPrim(1,j,k)-Umean))*dy*dz;
    UIT1=UIT1+abs((HTX_TPrimIN(1,j,k)-Tmean))*dy*dz;
  endfor
endfor
PrimMdot=sum (sum (sum(HTX_MdotPrim(1,:,:))));
UI=1-UI1/(2*abs(Umean)*(dy*ycells)*(dz*zcells))
UIT=1-UIT1/(2*abs(Tmean)*(dy*ycells)*(dz*zcells))


%Export data in measurment points

HTX_TPrimOUT_Verification=zeros(Measurement_Grid_Size(1,1),Measurement_Grid_Size(1,2));

for j_measured=1:1:Measurement_Grid_Size(1,1)
  HTX_TPrimOUT_Verification(j_measured,1)=HTX_TPrimOUT(xcells,j_measured*StepY,28);
  HTX_TPrimOUT_Verification(j_measured,2)=HTX_TPrimOUT(xcells,j_measured*StepY,17);
  HTX_TPrimOUT_Verification(j_measured,3)=HTX_TPrimOUT(xcells,j_measured*StepY,9);
  HTX_TPrimOUT_Verification(j_measured,4)= HTX_TPrimOUT(xcells,j_measured*StepY,5);
  HTX_TPrimOUT_Verification(j_measured,5)=HTX_TPrimOUT(xcells,j_measured*StepY,2);
endfor

% Plot Graphs

HTX_TPrimOUT_Plot= zeros (ycells,zcells);
HTX_TPrimOUT_Plot(:,:)=HTX_TPrimOUT(xcells,:,:)-273.15;
HTX_TPrimOUT_Plot=HTX_TPrimOUT_Plot';

HTX_VPrim_Plot= zeros (ycells,zcells);
HTX_VPrim_Plot(:,:)=HTX_VPrim(1,:,:);
HTX_VPrim_Plot=HTX_VPrim_Plot';

HTX_MdotAux_Plot=zeros (ycells,zcells);
HTX_MdotAux_Plot(:,:)=HTX_MdotAux(1,:,:);
HTX_MdotAux_Plot=HTX_MdotAux_Plot';

HTX_MdotPrim_Plot=zeros (ycells,zcells);
HTX_MdotPrim_Plot(:,:)=HTX_MdotPrim(1,:,:);
HTX_MdotPrim_Plot=HTX_MdotPrim_Plot';

HTX_TPrimIN_Plot= zeros (ycells,zcells);
HTX_TPrimIN_Plot(:,:)=HTX_TPrimIN(1,:,:)-273.15;
HTX_TPrimIN_Plot=HTX_TPrimIN_Plot';

HTX_TAuxOUT_Plot=zeros (ycells,zcells);
HTX_TAuxOUT_Plot(:,:)=HTX_TAuxOUT(xcells,:,:)-273.15;
HTX_TAuxOUT_Plot=HTX_TAuxOUT_Plot';

HTX_Phase_Plot=zeros (ycells,zcells);
HTX_Phase_Plot(:,:)=HTX_Phase(xcells,:,:);
HTX_Phase_Plot=HTX_Phase_Plot';

HTX_Top_Phase_Plot=zeros (xcells,ycells);
HTX_Top_Phase_Plot(:,:)=HTX_Phase(:,:,zcells);
HTX_Top_Phase_Plot=HTX_Top_Phase_Plot';

HTX_Top_PrimT_Plot=zeros (xcells,ycells);
HTX_Top_PrimT_Plot(:,:)=HTX_TPrimOUT(:,:,zcells);
HTX_Top_PrimT_Plot=HTX_Top_PrimT_Plot';

HTX_Top_AuxT_Plot=zeros (xcells,ycells);
HTX_Top_AuxT_Plot(:,:)=HTX_TAuxOUT(:,:,zcells);
HTX_Top_AuxT_Plot=HTX_Top_AuxT_Plot';
```

```
clf;
figure(1);
  colormap ("hot");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_TPrimOUT_Plot);
  Tmax= max (max (HTX_TPrimOUT_Plot));
  Tmin=min (min (HTX_TPrimOUT_Plot));
  caxis ([Tmin Tmax]);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("y (m)");
  ylabel ("z (m)");
  grid on;
  title ("Air Outlet Temperature Profile (degC)");


figure(2);
  colormap ("jet");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_VPrim_Plot);
  Vmax= max (max (HTX_VPrim_Plot));
  Vmin=min (min (HTX_VPrim_Plot));
  caxis ([Vmin Vmax]);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("y (m)");
  ylabel ("z (m)");
  grid on;
  title ("Air Inet Velocity (m/s)");

figure(3);
  colormap ("hot");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_TPrimIN_Plot);
  Tmax= max (max (HTX_TPrimIN_Plot));
  Tmin=min (min (HTX_TPrimIN_Plot));
  caxis ([Tmin Tmax]);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("y (m)");
  ylabel ("z (m)");
  grid on;
  title ("Air Inlet Temperature Profile (degC)");

figure(6);
  colormap ("hot");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_TAuxOUT_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("y (m)");
  ylabel ("z (m)");
  grid on;
  title ("Refrigerant Temperature(degC)");

figure(10);
  colormap ("gray");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_Phase_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("z (m)");
  ylabel ("y (m)");
  grid on;
  title ("Refrigerant Phase (1)");

figure(11);
  colormap ("hot");
  mesh ((y*dy)-dy/2,(z*dz)-dz/2, HTX_TAuxOUT_Plot);
  xlabel ("z (m)");
  ylabel ("y (m)");
  grid on;
  zlabel ("Refrigerant Temperature(degC)");

 figure(12);
  colormap ("hot");
  mesh ((y*dy)-dy/2,(z*dz)-dz/2, HTX_TPrimOUT_Plot);
  xlabel ("z (m)");
  ylabel ("y (m)");
  grid on;
  zlabel ("HTX_TPrimOUT_Plot");

 figure(13);
  colormap ("viridis");
  contourf ((y*dy)-dy/2,(z*dz)-dz/2, HTX_MdotAux_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("z␣(m)");
  ylabel ("y␣(m)");
  grid on;
  title ("Refrigerant␣Mass␣Flow␣Rate␣(kg/s)");

 figure(14);
  colormap ("gray");
  contourf ((x*dx)-dx/2,(y*dy)-dy/2, HTX_Top_Phase_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("x␣(m)");
  ylabel ("y␣(m)");
  grid on;
  title ("Refrigerant␣Phase␣(1)");

 figure(15);
  colormap ("hot");
  contourf ((x*dx)-dx/2,(y*dy)-dy/2, HTX_Top_PrimT_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("x␣(m)");
  ylabel ("y␣(m)");
  grid on;
  title ("Air␣Temparature␣(degC)");

 figure(16);
  colormap ("hot");
  contourf ((x*dx)-dx/2,(y*dy)-dy/2, HTX_Top_AuxT_Plot);
  %set (gca (), "xdir", "reverse")
  colorbar ("SouthOutside");
  xlabel ("x␣(m)");
  ylabel ("y␣(m)");
  grid on;
  title ("Refrigerant␣Temperature␣(degC)");
```

E AppendixE

Appendix E contains the raw data from the conducted measurements, presented in the form of graphs for the sake of clarity. The measurement locations can be referred to in Figure 25. Multiple data sets correspond to multiple measurements.

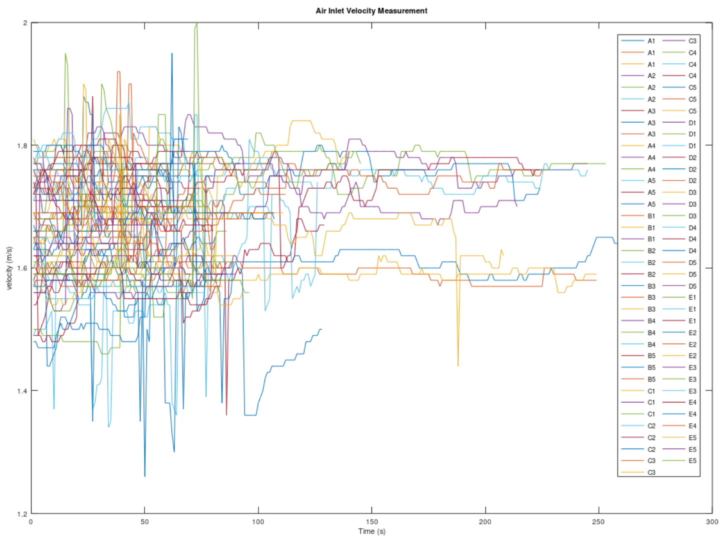


Figure 50: Air inlet velocity Set A (m/s)

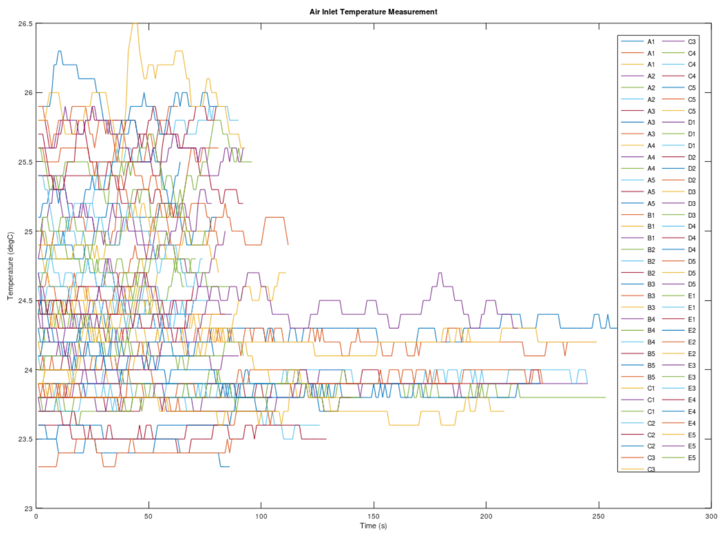


Figure 51: Air inlet temperature Set A (°C)

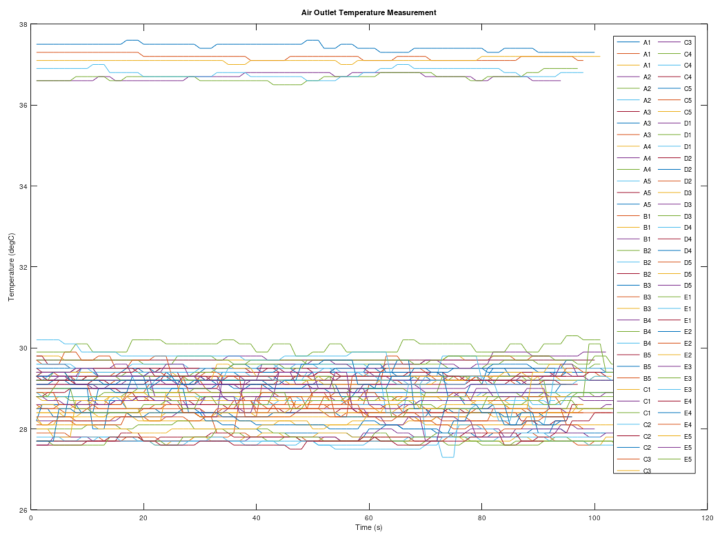


Figure 52: Air outlet temperature Set A (°C)

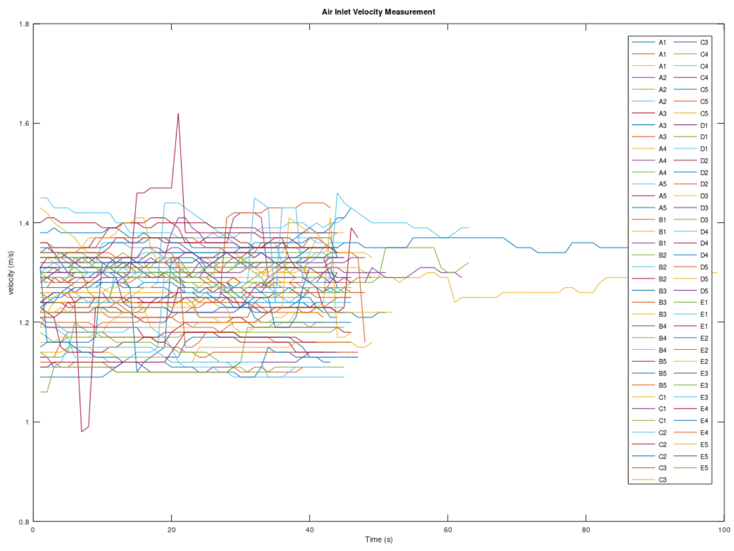


Figure 53: Air inlet velocity Set B (m/s)

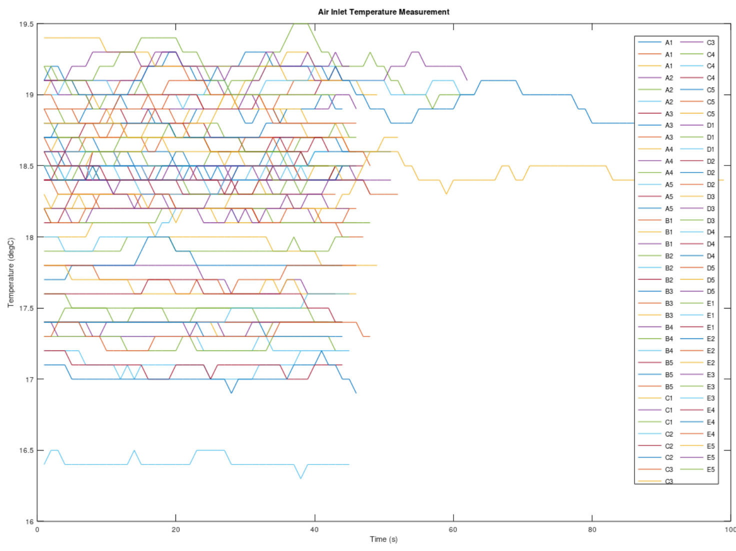


Figure 54: Air inlet temperature Set B (°C)

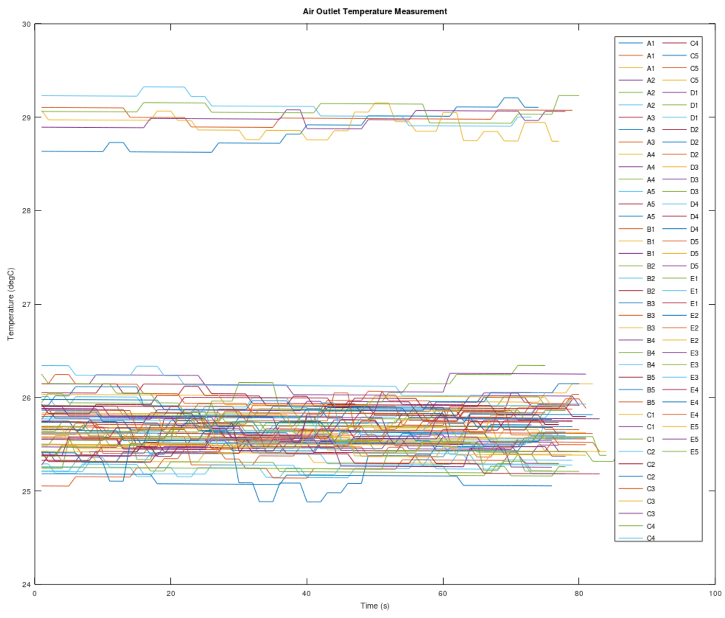


Figure 55: Air outlet temperature Set B (°C)