Mario Rohrhofer, BSc MSc

Dynamic Simulation of a Domestic Refrigeration Appliance

DOCTORAL THESIS

in fulfillment of the requirements for the academic degree
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

First Reviewer and Supervisor
Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Sanz
Institute of Thermal Turbomachinery and Machine Dynamics,
Graz University of Technology

Second Reviewer
Ao.Univ.-Prof. Mag.rer.nat. Dr.techn. Wolfgang Ring
Institute for Mathematics and Scientific Computing,
University of Graz

Graz, March 2017

Contents

Affidavit
Eidesstattliche Erklärung

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Graz, March 20, 2017 Mario Rohrhofer

Acknowledgement

I would like to thank you for reading this thesis. This means that the hard work was not in vain. I hope that the explanations are written clear enough so that you can benefit.

Raimund Almbauer for the scientific talks and teaching me thermodynamics. Furthermore, for creating a pleasant environment for research and letting us freely elaborate the topics of the research programs which improved my autonomy a lot.

Erhard Perz who made me become a much better programmer, encouraged me to develop the Echelon Analysis and always was there to help me whenever I got stuck.

My supervisor Wolfgang Sanz for letting me freely carry out research and giving me advice during the writing process such that the quality of this thesis surely increased.

Wolfgang Ring for improving my writing and teaching me a lot about mathematics. This not only includes theorems and proofs but also how to approach problems. Furthermore, for co-supervising this thesis and verifying that the mathematical deductions are correct.

My colleagues from the research group "thermodynamics" for the advice and whiteboard sessions whenever things were unclear and for all good moments we had.

My family and friends for the unrestricted support all along the way. Especially, I would like to thank my mother for all the energy and work she invested into her sons.

And finally, my girlfriend Conny who helped me realizing my goals - not only in research.

This thesis was written as part of the ECO-COOL project. This research project was financed by the FFG (Österreichische Forschungsförderungsgesellschaft), SFG (Steirische Wirtschaftsförderung), KWF (Kärntner Wirtschaftsförderungsfonds) and Standortagentur Tirol. Furthermore I would like to thank the partners in industry Secop Austria GmbH, Liebherr-Hausgeräte Lienz GmbH as well as SimTech GmbH.

Abstract

In practice, domestic refrigeration appliances are investigated and optimized with respect to energy efficiency performing expensive and time consuming experiments in the laboratory. In order to shorten this process and reduce the costs for experiments, a simulation tool can be deployed. Therein, the setup of the model or values of parameters can be changed more easily. Therefore, in the ECO-COOL research project at the Graz University of Technology, dynamic mathematical models of domestic refrigeration appliances were developed, implemented, dynamically validated and investigated. A deeper insight into the considered appliances was gained and parameter studies showed potentials to increase the energy efficiency. However, the downside of this research code, in which the appliance model is coupled with the numerical solver and only given on a source code level, is the difficult operability for people who did not participate in the model and code development process.

To obtain a sustainable simulation tool applicable also for manufacturers, the models are transferred into the commercial software IPSEpro from SimTech GmbH. These models form a system of Differential-Algebraic Equations (DAEs). In this thesis, established numerical methods for solving DAEs are given and the problems of consistent initialization and regularization of the initialization system are discussed. Further, the component and flow sheet models implemented in IPSEpro are presented in detail. For the dynamic model validation, a semi-automatized parameter fitting procedure is applied and the appliance model is compared with dynamic measurements and the results from the original research code. Finally, a detailed insight into the simulation results of IPSEpro is given and a new parameter study, which investigates the effect of the chosen number of revolutions of the compressor on the energy consumption, is presented.

Kurzfassung

In der Praxis werden Haushaltskühlgeräte mittels teuren und zeitaufwendigen Experimenten untersucht, um in Hinblick auf ihre Energieeffizienz optimiert zu werden. Um diesen Prozess zu verkürzen und die Kosten für Experimente zu reduzieren, können stattdessen Simulationswerkzeuge verwendet werden. Darin lassen sich der Modellaufbau und Werte für Parameter leichter verändern. Daher wurden im Forschungsprojekt ECO-COOL an der Technischen Universität Graz dynamische mathematische Modelle von Haushaltskühlgeräten entwickelt, implementiert, dynamisch validiert und untersucht. Ein tieferer Einblick in die betrachteten Geräte konnte gewonnen werden und Parameterstudien zeigten Potentiale, um die Energieeffizienz zu steigern. Jedoch liegt ein Nachteil dieses universitären Forschungscodes, in dem die Gerätemodelle mit dem numerischen Löser gekoppelt und nur im Quellcode gegeben sind, in der schwierigen Bedienbarkeit für Personen, die nicht in die Modell- und Codeentwicklung eingebunden waren.

Um ein nachhaltiges Simulationswerkzeug, das auch von Geräteherstellern verwendet werden kann, zu schaffen, werden die Modelle in die kommerzielle Software IPSEpro von SimTech GmbH portiert. Die Modelle bilden dabei ein System von differential-algebraischen Gleichungen. In dieser Arbeit werden etablierte numerische Methoden zum Lösen solcher Systeme vorgestellt und die Probleme von konsistenter Initialisierung und der Regularisierung des Initialisierungssystems behandelt. Weiters werden die Komponenten- und Fließbildmodelle, wie sie in IPSEpro aufgebaut sind, detailiert präsentiert. Für die dynamische Validierung wurde eine semiautomatische Parameteranpassung gewählt und das Gerätemodell wird mit dynamische Messdaten und Resultaten des ursprünglichen Forschungscodes verglichen. Schlussendlich wird ein detailierter Einblick in die Simulationsresultate von IPSEpro gegeben und eine neue Parameterstudie, die den Einfluss der gewählten Kompressordrehzahl auf den Energieverbrauch untersucht, präsentiert.

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neuronal Network
COP	Coefficient of Performance
DAE	Differential-Algebraic Equation
DASSL	Differential Algebraic System Solver
DLL	Dynamic Link Library
BDF	Backward Differentiation Formula
FPD	Frictional Pressure Drop
HTC	Heat Transfer Coefficient
IPSEpro	Integrated Process Simulation Environment
IVP	Initial Value Problem
MDK	Model Development Kit
MDL	Model Description Language
ODE	Ordinary Differential Equation
NOR	Number of Revolutions
PDE	Partial Differential Equation
PSE	Process Simulation Environment
VBA	Visual Basic for Applications

List of Notations

Roman Letters

A	Area	$[\mathrm{m}^2]$
A	Linear mapping	
c_p	Isobaric heat capacity	$[\mathrm{J/(kg\,K)}]$
D	Diagonal matrix	
d	Diameter	$[\mathrm{m}]$
d_H	Hydraulic diameter	$[\mathrm{m}]$
e	Euler number	
F	System of functions	
f	Function	
G	Mass flow per area	$[\mathrm{kg/(m^2\,s)}]$
G	System of functions	
g	Acceleration due to gravity	$[\mathrm{m/s^2}]$
g	Function	
H	Enthalpy	$[\mathrm{J}]$
h	Specific enthalpy	$[\mathrm{J/kg}]$
h	Step size	
I	Identity	
J	Jacobian matrix	
k	Thermal transmittance	$[\mathrm{W/(m^2\,K)}]$
l	Length	$[\mathrm{m}]$
M	Matrix corresponding to an elementary row transformation	
M	Molar mass	$[\mathrm{kg/mol}]$
m	Integer	
m	Mass	$[\mathrm{kg}]$
$\dot{m}$	Mass flow	$[\mathrm{kg/s}]$
n	Integer	
n	Number of revolutions	$[\mathrm{1/s}]$
Nu	Nusselt number	$[-]$
P	Permutation matrix	
P	Power	$[\mathrm{W}]$
p	Pressure	$[\mathrm{Pa}]$
Pr	Prandtl number	$[-]$
Q	Permutation matrix	
$\dot{Q}$	Heat flux	$[\mathrm{W}]$
$\dot{Q}_0$	Cooling capacity	$[\mathrm{W}]$
r	Radius	$[\mathrm{m}]$
r_H	Half the hydraulic diameter	$[\mathrm{m}]$
r	Residual	
R	Vector of residuals	
Re	Reynolds number	$[-]$
S	Entropy	$[\mathrm{J/K}]$
S	Sensitivity matrix	
s	Specific entropy	$[\mathrm{J/(kg\,K)}]$
T	Temperature	$[\mathrm{K}]$
t	Time	$[\mathrm{s}]$
U	Inner energy	$[\mathrm{J}]$
u	Specific inner energy	$[\mathrm{J/kg}]$
V	Volume	$[\mathrm{m}^3]$
v	Velocity	$[\mathrm{m/s}]$
$\dot{V}$	Volume flow	$[\mathrm{m^3/s}]$
W	Matrix of weights	
x	Scalar or vector	
x	Vapor quality	$[-]$
$\dot{x}$	Flow quality	$[-]$
y	Scalar or vector	
z	Scalar or vector	

Greek Letters

α	Heat transfer coefficient	$[\mathrm{W/(m^2\,K)}]$
Δp	Pressure drop	[Pa]
η	Dynamic viscosity	$[\mathrm{N\,s/m^2}]$
η	Efficiency	$[-]$
λ	Thermal conductivity	$[\mathrm{W/(m\,K)}]$
ρ	Density	$[\mathrm{kg/m^3}]$
σ	Surface tension	[N/m]

Miscellaneous Symbols

$\diamondsuit$	End of an example
$\frac{\partial f}{\partial x}$	Partial derivative of f wrt. x
$\mathbb{C}$	Complex numbers
$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{R}^+$	Positiv real numbers
$\mathbb{R}^n$	Real, n-dimensional vector space
$\mathbb{R}^{n\times m}$	Space of $n \times m$ matrices with real coefficients
$\mathcal{A}$	Set of active variables
$\mathcal{B}$	Set of visited variables
$\mathcal{C}^k$	Hölder space of k times continuously differentiable functions
$\mathcal{E}$	Set of indices (equations)
$\mathcal{F}$	Feasible set
$\mathcal{H}$	Set of hidden constraints
$\mathcal{I}$	Set of inactive variables
$\mathcal{L}$	Set of indices
$\mathcal{N}$	Set of not explored indices (variables/equations)
$\mathcal{O}$	Landau symbol
$\mathcal{R}$	Set of removable settings
$\mathcal{R}_s$	Set of structurally removable settings
$\mathcal{S}$	Set of settable variables
$\mathcal{S}_s$	Set of structurally settable variables
$\mathcal{T}$	Set of calculable variables
$\mathcal{V}$	Set of indices (variables)
∇	Gradient
∇^2	Hessian matrix
$\|\cdot\|$	Norm
$\|\cdot\|_2$	Euclidian norm

Subscripts

$drain$	At outlet
$feed$	At inlet
l	At the lower two-phase boundary
sat	At saturation
v	At the upper two-phase boundary
x	Partial derivative with respect to x

Superscripts

$'$	Derivative
$^\top$	Transpose

I. Introduction

The worldwide energy consumption is increasing. Studies, see [48], reveal that approximately 15 % of the electric energy in households within the European Union is spent for cooling. In [47], the authors claim that the average energy consumption of a domestic refrigeration appliance is about 1 kWh per day. For Germany, having about 40.7 million private households, this yields an energy consumption of 22 TWh (in 2011) per year, see [2]. Therefore, household cooling has an enormous potential to save energy.

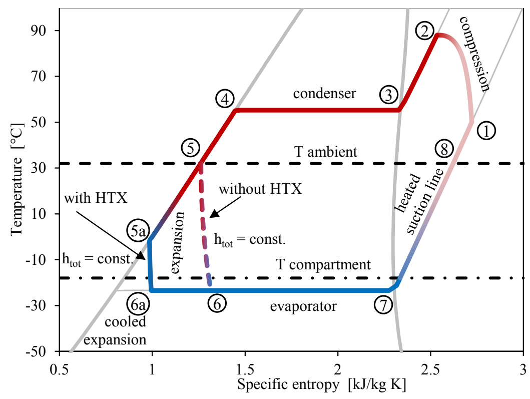
In the 1990's an energy consumption labelling system was introduced in the European Union, see [27]. The labelling system was supposed to influence the buying behavior and thus force the manufacturers to improve their appliances. Additionally, refrigeration devices shall be forbidden from being sold if their energy efficiency is too poor. Originally, the scale included the Roman letters A to G where A indicates the least energy consumption. Due to technical improvements and a considerably better energy efficiency many new appliances reached the category A. As a consequence, this category was successively subdivided to guarantee an easier comparability of the energy consumption. These additional categories are marked with A and have one or multiple "+" attached. Nowadays, in domestic refrigeration appliances the category A^{+++} is already reached and it is just a matter of time until further developments will lead to the introduction of the next category.

The progression of the labelling system shows that the technical improvements exceeded the original expectations. But still, the process of increasing the energy efficiency of refrigeration appliances is far from being over. However, finding new potentials becomes more challenging and nowadays focus is laid on the dynamic behaviour of refrigeration appliances. A computer-aided simulation seems to be a promising method to reach this goal. Once the simulation is established, not only the dynamic effects and sensitivities can be better understood but also parameter variations can be performed easier and mathematical optimization methods can be applied to locate energy consumption minima.

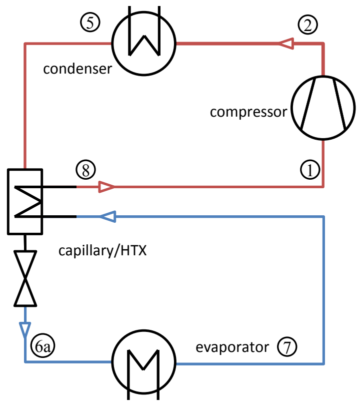
1. Refrigeration Cycle in Domestic Appliances

In this thesis the dynamic behavior of vapor-compression refrigeration cycles in domestic appliances, in which the phase of the refrigerant changes, is investigated. Since these are the only refrigeration cycles considered in this thesis, the term vapor-compression is omitted in the remaining text. For a general review of the research in this field we refer to [4]. Typically, the refrigeration cycle consists of the four main components: *compressor*, *condenser*, *expansion device* and *evaporator*. In the considered appliance the expansion device is a *capillary* and the refrigerant is *Isobutane (R600a)*. We explain the refrigeration cycle by means of figure I.1 where a Ts-diagram of an idealized cycle in steady-state and the flow sheet are depicted.

In the investigated cycle, the superheated refrigerant is sucked by the compressor and compressed to a higher pressure ($1 \to 2$). The superheated vapour is discharged into the condenser with a temperature above ambient. There the refrigerant condenses ($2 \to 5$) passing the two-phase region ($3 \to 4$). Within this region the temperature of the refrigerant, referred to as saturation temperature, remains equal if the pressure drop is neglected. Additionally, the heat transfer to the ambiance increases significantly, see [103]. Next, the subcooled refrigerant passes the capillary. The component of the capillary can also be used as an internal heat exchanger which shifts the capillary outlet from (6) to ($6a$). On the one hand, this increases the cooling capacity of the evaporator, defined in equation (1.1) below, and on the other hand, shall prevent that non-superheated refrigerant enters the compressor. For a detailed discussion we refer to [38]. In the capillary the pressure is decreased such that the saturation temperature is below the desired compartment temperature. This allows the refrigerant to absorb heat in the evaporator ($6 \rightarrow 7$) and thus cool the compartment. Finally, the internal heat exchanger is passed again ($7 \rightarrow 8$) and the refrigerant returns to the compressor where it exchanges heat with the compressor shell ($8 \rightarrow 1$) before it is compressed.



(a) Ts-diagram, [38].



(b) Flow sheet.

Figure I.1.: Schematic illustration of the refrigeration cycle.

If different refrigeration cycles have to be compared in terms of efficiency, it is not sufficient to measure their total energy consumption. Since the energy consumption depends *e.g.* on the size of the appliance, a different quantity has to be chosen. Following [38], this quantity is the dimensionless Coefficient of Performance (COP) ε which is the ratio between the output and input of a process. In refrigeration the output is the *cooling capacity* $\dot{Q}_0$ [W]. In the steady-state case the cooling capacity is defined as

$$\dot{Q}_0 = \dot{m}(h_7 - h_6) \tag{1.1}$$

where $\dot{m}$ is the mass flow and h_6, h_7 are the enthalpies at the respective points in figure I.1, *i.e.* at evaporator inlet and outlet. The input is the electric power P_{el} [W] consumed by the compressor. For a steady-state refrigeration cycle this yields

$$\varepsilon_s = \frac{\dot{Q}_0}{P_{el}}. \tag{1.2}$$

The coefficient is bounded from above by the COP of the theoretical *Carnot*-cycle ε_{carnot} where compression and expansion is isentropic and the heat is transferred at constant temperature. The Carnot-COP reads as

$$\varepsilon_{carnot} = \frac{T_{evap}}{T_{cond} - T_{evap}} = \frac{1}{\eta_{carnot}} \tag{1.3}$$

where T_{cond} and T_{evap} are the saturation temperatures in the condenser and evaporator, respectively, and η_{carnot} is the Carnot-efficiency. The above formula implies that the COP can be increased if the temperature difference is decreased. However, T_{cond} has to exceed the ambient temperature to enable heat exchange and T_{evap} has to stay below the desired compartment temperature. Thus, the Carnot-COP is bounded from above as well.

In contrast to the above assumptions, a domestic refrigeration appliance is never at steady-state except it is turned off. Nowadays there are two ways how refrigeration appliances operate. First, the compressor is able to change the Number of Revolutions (NOR) depending on the demanded cooling power.

For models using a so-called variable speed compressor we refer to [54, 102]. Second, the NOR is fixed and the compressor operates intermittently such that the compartment temperature stays within a certain range. In this thesis, only the second compressor type is investigated. In the considered domestic refrigeration appliance, the compressor operates only with an a priori prescribed NOR and the temperature in the compartment is measured by a sensor. If the lower or upper temperature threshold is reached, the compressor is turned off or on, respectively. Thus, the Ts-diagram in figure I.1a depends on the operating point and varies over time. The evolution of the Ts-diagram in the validated model for one compressor on-off-cycle is illustrated in figures V.5 and V.6. Due to the operating mode, also the concept of energy efficiency has to be adapted. For dynamic refrigeration cycles the COP is defined as

$$\varepsilon_d = \frac{\int \dot{Q}_0 dt}{\int P_{el} dt}. \tag{1.4}$$

2. Computer-aided Simulation

In practice, domestic refrigeration appliances are investigated and optimized with respect to energy efficiency performing expensive and time consuming experiments in the laboratory. The experiments are typically done in a climatic chamber where first the appliance and measuring instruments have to be set up. If the ambient temperature does not change and the compartment door is never opened, the refrigeration cycle of the appliance reaches a (time) periodic cycle in which the compressor on-off-duration remains equal and the measured values repeat for each cycle. Only after a periodic cycle evolved in the appliance, the measurements are recorded and energy efficiency is investigated. However, the time it takes to reach a periodic cycle depends on the initial states in the appliance and may take up to days.

In order to shorten this process and reduce the costs for experiments, a simulation tool can be deployed. Therein, the setup of the model or values of parameters are changed more easily, and the simulation time, at least for the models in this thesis, is considerably less than real time. Therefore, since the late 1970s, simulation tools have been implemented to support the research in refrigeration systems and since the late 1980s also in domestic refrigeration appliances. Since the compressor operates intermittently in the investigated appliance, a steady-state model is not sufficient to predict the dynamic behavior and energy efficiency. Thus we focus only on dynamic models. Simultaneously with the performance of the computers also the complexity and accuracy of the models increased.

In the historical development of refrigeration system simulation, the heat exchangers turned out to be the crucial part and three model types evolved. First, in the lumped parameter models the heat exchangers are assumed to be 0-dimensional point masses. These simple models are applied in [16, 20, 67, 91, 106] (latest in 2005). Second, in the moving boundary models the heat exchangers are subdivided into three parts of variable size which represent the liquid, two-phase and vapour region, see [37, 49, 81, 116] (latest in 2002). Third, the most sophisticated models use a distributed parameter approach where the heat exchangers are 1-dimensional spatially discretized. These more complex models were applied in [14, 64, 47, 83, 117] (latest in 2008). The compressor models are usually given by simple empirical correlations such as proposed in [50, 63, 76] (latest in 2012). Even nowadays, a fully 3-dimensional spatially discretized model which also incorporates the movement of the valves and the piston is too time consuming and therefore not applicable in refrigeration appliance simulation. Often also the capillary models use simplified empirical models as presented in [45, 46, 68, 119] (latest in 2010). Historical overviews of models and simulation can be found in [5, 21, 47, 81].

3. The ECO-COOL Research Project

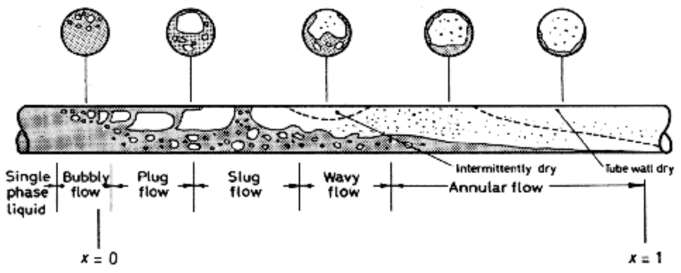
The *ECO-COOL* research project is carried out from July 2013 until June 2017 at the *Graz University of Technology*. Not only three institute of the University, namely the *Institute for Internal Combustion and Thermodynamics*, the *Institute of Thermal Turbomachinery and Machine Dynamics* and the *Electric Drives and Machines Institute* are involved in this project but also partners from industry which are *Secop Austria GmbH*, *Liebherr Hausgeräte Lienz GmbH*, *Infineon Technologies Austria AG* and *SimTech GmbH*. The aim of the project is the "development of the first fully integrated and controlled cooling cycle for the usage in household cooling appliance"[1]. To reach this goal, one objective is to develop and investigate dynamic models of the refrigeration cycle and a refrigeration appliance model wich is validated with measurements. In the following the component models and numerical solvers elaborated in the research project but also their limitations are outlined shortly. This also provides the basis of this thesis.

In the ECO-COOL research project, the component models are implemented in Visual Basic for Applications (VBA), *C* and *Matlab* and are merged to a cycle simulation tool in VBA. A distributed parameter approach using the finite volume method is applied to the heat exchangers, see [6, 7, 8]. The compressor is split into three components: shell, compression and the oil which is used to lubricate and cool the compressor. The compression is described by empirical correlations, see [84, 85], and the oil as in [81]. The capillary is simulated using an Artificial Neuronal Network (ANN) based on a complex one-dimensional model, see [38, 41, 42]. For the compartment a lumped parameter model is chosen which neglects the temperature stratification and the insulation is spatially discretized. The individual component models were validated with measurements. The validated cycle simulation and results are presented in [39, 40, 43]. Two different appliances are investigated therein and it was shown that the models fit the dynamic measurements well. Furthermore, parameter studies are performed to identify potentials to increase the energy efficiency.

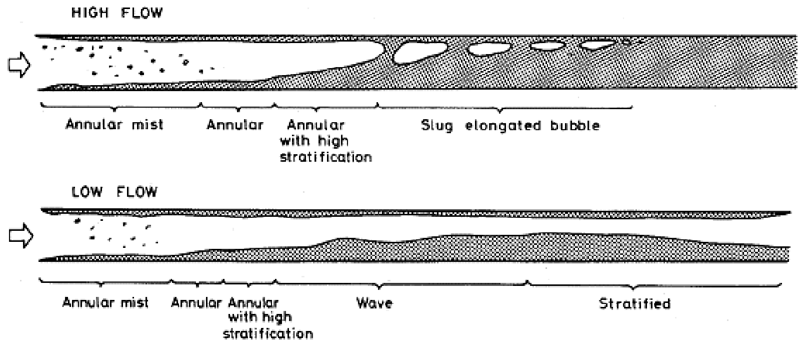
The by far most complex models in the VBA simulation tool are the heat exchangers, *i.e.* condenser and evaporator. The reason lies in the operating points of the heat exchangers which are mostly in the two phase region and the fact that the two-phase boundaries are crossed multiple times during the simulation. The heat exchangers are spatially discretized by the finite volume method, the theory of which is presented in *e.g.* [61]. In contrast to other publications, see [47, 107, 118] (latest in 2008), the pressure drop was not neglected. Since the velocity changes of the refrigerant are small in a domestic refrigeration appliance, the momentum balance reduces to an algebraic equation describing the pressure drop caused by friction, also referred to as Frictional Pressure Drop (FPD). In the literature, the Heat Transfer Coefficient (HTC) in the two-phase region between the refrigerant and the pipe wall is often determined by empirical correlations, *e.g.* [81] (2002) applies correlations from Shah proposed in [98] (1988), [47] (2008) uses correlations from [52] (2003) and [114] (2000) for the two-phase HTC for condensation and evaporation, respectively. In the VBA models, in the single phase region the FPD and the HTC between the refrigerant and the pipe wall, appearing in the energy balance of the refrigerant, are obtained from empirical correlations, see [28]. These correlation are substituted by highly complex models in the two-phase region. These complex models are based on flow patterns. In tubes, the geometry or topology of the flow, referred to as *flow pattern* or *flow regime*, depends on the tube geometry, mass flow, vapor quality and other physical properties. Two-phase flow patterns as they appear in heat exchangers are illustrated in figure I.2. In the literature, see [36, 89, 112] (earliest in 2003), so called flow pattern maps, see figure I.3, have been developed. For the map, transition curves between patterns are proposed which depend on physical properties, tube geometry, mass velocity G and vapour quality x. A given point (G, x) then defines a unique point in the flow pattern map and the corresponding pattern. For each flow pattern, models for the HTC and FPD have been established in the literature, see [90, 103, 113] (earliest in 2003), which are not

[1]Project title of the ECO-COOL research project. Homepage: https://ecocool.tugraz.at/index.html (March 20, 2017)

necessarily continuous across the transition curves. To guarantee an at least continuous model some sort of interpolation has to be applied wherever jump discontinuities appear. This also includes the two-phase boundaries.



(a) In horizontal evaporator tubes.



(b) In horizontal condenser tubes.

Figure I.2.: Two-phase flow patterns, [3].

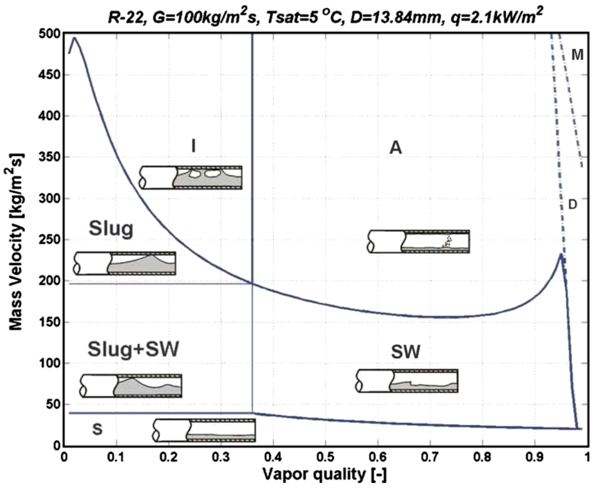


Figure I.3.: Two-phase flow pattern map, [89].

Furthermore, some physical properties are not defined in the two phase region, *e.g.* the thermal conductivity λ, the isobaric heat capacity c_p or the dynamic viscosity η. If these properties are calculated from pressure p and temperature t, a jump discontinuity emerges at the saturation temperature t_{sat} for any given pressure less than the critical pressure. Since these properties are used in the HTC and FPD models, the model developer has to make sure that a continuous description is achieved such that no convergence issues emerge when the model is solved.

Besides the model development also mathematical solvers were elaborated for the VBA cycle simulation tool which shall be explained shortly. A description of the calculation schemes can be found in [40] for the cycle simulation and in [7] for the heat exchangers. In figure I.4 the refrigeration appliance model flow sheet and the calculation procedure are depicted. In the VBA implementation, during each integration step each component model is solved independently using boundary conditions from the other components. These boundary conditions are taken from the previous integration step. Although not implemented, this procedure allows a parallel treatment of the components at each time step.

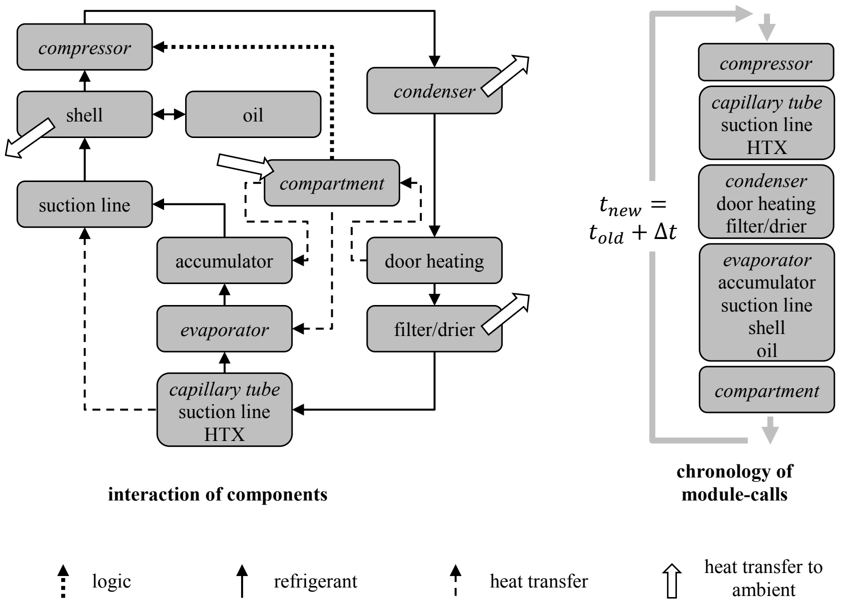

compressor
shell
oil
condenser
compartment
suction line
accumulator
door heating
evaporator
filter/drier
capillary tube suction line HTX
interaction of components
$t_{new} = t_{old} + \Delta t$
compressor
capillary tube suction line HTX
condenser door heating filter/drier
evaporator accumulator suction line shell oil
compartment
chronology of module-calls
logic
refrigerant
heat transfer
heat transfer to ambient

Figure I.4.: Schematic of the VBA cycle simulation tool, [40].

In the VBA tool, the component models compressor, capillary and compartment are calculated fairly straight forward and therefore not presented in detail here. However, the heat exchangers turned out to be more challenging. In each integration step a boundary value problem of Partial Differential Equations (PDEs) coupled with algebraic equations has to be solved which is handled by a shooting method. For a deeper insight into this kind of problems and available numerical methods, we refer to [18]. In the VBA model, the chosen boundary values for the heat exchanger are the mass flows $\dot{m}_{in}$, $\dot{m}_{out}$ at inlet and outlet, respectively, and the specific enthalpy h_{in} at the inlet. Let p_{in} be the pressure at the inlet, $\varphi : \mathbb{R}^3 \to \mathbb{R}$ with $\varphi(p_{in}, h_{in}, \dot{m}_{in}) = \dot{m}_{out}$ and let the given boundary conditions be marked with a superscript b. Then we look for a root of the function $\Phi : \mathbb{R} \to \mathbb{R}$ with

$$\Phi(p_{in}) = \varphi(p_{in}, h^b_{in}, \dot{m}^b_{in}) - \dot{m}^b_{out}. \tag{3.1}$$

In the VBA implementation, the pressure p_{in} is iterated using the bisection method until $|\Phi(p_{in})| < \varepsilon$ for $\varepsilon > 0$ is satisfied. Hence, for given boundary conditions $h^b_{in}, \dot{m}^b_{in}$ at the inlet, the pressure p_{in} at the inlet is adjusted until the mass flow $\dot{m}_{out}$ at the outlet fits the boundary condition $\dot{m}^b_{out}$. For one evaluation of φ, the finite volume model of the heat exchanger has to be solved. Since the triple $(p_{in}, h^b_{in}, \dot{m}^b_{in})$ is given, the volumes can be treated sequentially. The inlet of the heat exchanger is also the inlet of the first finite volume, *i.e.* $(p_{in}, h^b_{in}, \dot{m}^b_{in}) = (p^1_{in}, h^1_{in}, \dot{m}^1_{in})$ where the superscript indicates the index of the volume. Hence, the inlet of the first volume is fully defined and solving the equations for the first volume yields $(p^1_{out}, h^1_{out}, \dot{m}^1_{out})$ at the outlet which equals the inlet of the second volume $(p^2_{in}, h^2_{in}, \dot{m}^2_{in})$. Repeating this for all volumes leads to $(p_{out}, h_{out}, \dot{m}_{out})$ at the outlet of the heat exchanger.

Above for each volume, a system Ψ of coupled nonlinear equations has to be solved . Since the coupling of the variables within Ψ is sufficiently weak, a simple sequential algorithm is applied. Let $\Psi = (\psi_1, \dots, \psi_n) : \mathbb{R}^n \to \mathbb{R}^n$ and the variables be $x \in \mathbb{R}^n$. In the heat exchanger models, the system for a volume has the structure

$$\begin{pmatrix} \psi_1(x_1) \\ \psi_2(x_1, x_2) \\ \vdots \\ \psi_{n-1}(x_1, \dots, x_{n-1}) \\ \psi_n(x_1, \dots, x_n) \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \\ 0 \end{pmatrix}. \tag{3.2}$$

By fixing x_1, the equations can be evaluated sequentially from ψ_1 to ψ_n. Thus the above system is solved by iterating x_1 using the regular falsi method until $|\psi_n(x_1, \dots, x_n)| < \varepsilon$ for given $\varepsilon > 0$. All together, this approach allows to evaluate the equations successively and needs no derivatives.

Although, in the first place the shooting method may seem to be more easily comprehensible, especially when implementing complex models for the HTC and FPD, the huge disadvantage is that the kind of boundary condition have to be known a priori. If the boundary condition alters, *i.e.* different variables are selected, the shooting method may have to be reformulated and the structure of the system Ψ, which is ordered manually in the VBA simulation tool, may change. Furthermore, the numerical method and the model equations are coded within the same procedure which can make it harder to understand the model for someone who was not part of the development team.

In the VBA tool, an appropriate graphical user interface, in which the flow sheet of the refrigeration cycle model is illustrated, is missing. The corresponding graphic from figure I.4 was created manually. The connections between the component models are only visible on a source code level. Even a small change in the structure of the model may require major adaptations in the code. Additionally, due to the sequential formulation of the numerical method, reversion of the direction of the mass flow, models consisting of parallel heat exchangers and splitter-mixer problems can not be implemented. Therefore, in order to obtain a sustainable tool applicable for refrigeration appliance manufacturers, porting the models into a commercial software which does not exhibit these disadvantages is necessary.

4. IPSEpro

The commercial software Integrated Process Simulation Environment (IPSEpro), developed and distributed by SimTech GmbH from Graz, Austria, was chosen since it fulfils the demanded requirements and the company participates in the ECO-COOL research project. Other possible software tools are *Matlab Simulink* from *The MathWorks Inc.* or *Dymola* (based on *Modelica*) from *Dassault Systèmes*. IPSEpro consists of several modules of which the Model Development Kit (MDK) and Process Simulation Environment (PSE) are the two most important. Screenshots of these modules are illustrated in figures III.2 and III.3. In IPSEpro, the mathematical description of the component models, setting up a flow sheet model and numerical solution are well separated. First, in MDK the components and their mathematical models are described, second, in PSE the component models are arranged in a flow sheet and the boundary conditions are specified and third, in the mathematical solver kernel the system of equations is solved using appropriate numerical methods.

Until the beginning of the ECO-COOL research project only steady-state simulations together with validation and optimization problems could be handled in IPSEpro. Since simulating the dynamic behaviour of the refrigeration cycle is the goal, first an appropriate numerical solver had to be incorporated and the framework for dynamic simulations had to be developed. The framework includes, among others, data management, performance, visualizations of the time evolution of variables, time tables for forcing terms and their processing, error handling and of course adaptations in the graphical user interface. These elements have been implemented in IPSEpro during the ECO-COOL research project but will not be explained in detail in this thesis.

When a dynamic model such as the refrigeration cycle has to be solved, a system of PDEs coupled with algebraic equations emerges. In IPSEpro, only first order derivatives are provided which we always interpret as the derivative with respect to time. This implies that the spatial derivatives have to be handled by the model developer. Consequently, the numerical solver in the kernel has to deal with a system of Differential-Algebraic Equations (DAEs).

5. Objectives and Contents of this Thesis

First, we summarize the objectives of this thesis. The first and most general objective is to

1. build an easily useable simulation tool for a better understanding of the refrigeration appliance.

We define three other objectives which are necessary to achieve this general objective. Each of the following objectives is handled in one chapter and they are presented in the corresponding order.

2. a solver for DAEs has to be incorporated into the commercial software IPSEpro,

3. the refrigeration appliance models from the VBA model have to be transferred into IPSEpro and

4. the dynamic models have to be validated with measurements.

In *chapter II*, established numerical methods to solve Initial Value Problems (IVPs) involving DAEs are presented. In the literature, see *e.g.* [11, 55, 59, 62, 69, 79, 115] (latest in 2012), the problem of finding consistent initial values is addressed. In this thesis a similar approach as proposed in [115] is chosen and the initialization system F is solved once at the initial time before the DAE-solver is called. Typically, the component models in MDK are underdetermined, *i.e.* more variables than equations are defined, which allows a more general application of the component models. Consequently, the flow sheet model in PSE is underdetermined as well and boundary conditions, referred to as *settings*, have to be prescribed with given values such that F is solvable. In large systems this may become a nontrivial task. Therefore, a regularization method, referred to as Echelon Analysis, for underdetermined systems is elaborated, see section II.3. Regularization in this context means formulating the flow sheet model (structure and boundary conditions) in such a way that the numerical method in the kernel is able to handle the emerging system of equations not considering convergence or accuracy issues. This implies that F has to be square and the Jacobian non-singular. The Echelon Analysis finds in underdetermined systems the variables which can be set, *i.e.* serve as a boundary condition, such that the degree of freedom is reduced successively and the Jacobian does not become globally singular. In particular this regularization method can be applied to dynamic but also to steady-state systems.

Either by not following the Echelon Analysis or on purpose, the settings can lead to a (partially) overdetermined system. In this case no further setting would lead to a regular system which implies that the Echelon Analysis is not applicable. To handle this case, the Adjoint Echelon Analysis, see section II.3.2, is designed. In a (partially) overdetermined system this analysis returns the settings which can be removed so that the degree of overdetermination is reduced by one and the system does not become globally singular.

In *chapter III*, the commercial software IPSEpro is shortly presented and its model structure is explained. Then the component models used for the simulation of the domestic refrigeration appliance are given. As in the VBA model the heat exchangers turned out to be the more challenging components. In particular, numerous discontinuities in the models of the HTC and FPD had to be identified and treated appropriately. In section III.8 these problems are discussed more detailed. Since the complex models of the HTC and FPD could not be implemented in MDK directly, they are outsourced in a Dynamic Link Library (DLL) which is written in C++. The object oriented structure is described in section III.7. For a comprehensive introduction to programming in C++, we recommend [25, 26, 70, 99]. Finally, a strategy for setting up the flow sheet model of the refrigeration appliance is proposed and the model itself is presented.

When the component and flow sheet models are developed and the dynamic simulation can be performed successfully for a sufficiently large domain of boundary conditions and parameters, the refrigeration cycle model has to be compared with measurements. If the results are not satisfying, either the uncertain parameters appearing as settings or the model setup have to be adjusted. This task is referred to as *model validation* and is done in *chapter IV.* Since the compressor operates intermittently, the validation has to be done in a dynamic way. This means that the time evolution of the respective variables has to coincide with the measurements which significantly increases the challenge of the task. Since the compressor operates intermittently the on-off-durations are also of importance, implying that the logic of the appliance and the temperature sensor in the compartment have to be modelled correctly. In the VBA model the dynamic validation was performed manually and took several months. Within this thesis a semi-automatized approach for fitting the parameters is applied, *i.e.* a subset of the parameters is fitted automatically to steady-state measurements and the few remaining ones are adjusted manually such that the results coincide with the dynamic measurements.

Finally, in chapter V the results of the refrigeration appliance model set up in IPSEpro are presented for the periodic on-off-cycle. The local distribution of refrigerant as well as the heat transfers are discussed. Further, the time-evolution of the Ts-diagram and the behaviour of the appliance model for various ambient temperatures are shown and results of the one dimensional spatially discretized heat exchangers are investigated.

Once a model is validated, the optimization process can start. Primarily, the energy consumption of the refrigeration appliance is of importance, but also the manufacturing costs should not be disregarded. A complete and comprehensive optimization of the investigated appliance would go beyond the scope of this thesis and is left as an outlook for future work. Still, in section V.2 a parameter study was performed in which the NOR of the compressor was varied.

II. Solving Differential-Algebraic Equations

Differential-Algebraic Equations (DAEs) arise naturally in mathematical models describing physical systems. DAEs are systems which consist of differential and algebraic equations. The differential part describes the time evolution of the dependent variables. The algebraic part either can emerge from constraints on the differential variables or from equations describing variables whose derivatives do not appear explicitly in the system. Systems in which the algebraic part is significantly higher than the differential part arise *e.g.* from the models developed in chapter III. The DAE endowed with initial values forms an Initial Value Problem which is solved by an appropriate numerical method.

Within this chapter, first basic definitions and properties of DAEs are presented. For a general overview and introduction into the field we refer to [1, 11, 35, 92]. After introducing numerical methods to solve these systems, the problem of consistent initialization is discussed and methods are presented to obtain a regular initialization system. Of course, also the stability of a solution is of interest but a full analysis of this topic would go beyond the scope of this thesis. Therefore, we refer to [57] in which the classical concepts of stability of Ordinary Differential Equations (ODEs) are generalized to DAEs.

1. Differential-Algebraic Equations

Definition 1.1 *The general (nonlinear) Differential-Algebraic Equation (DAE) system is defined as*

$$F(t, x, x') = 0 \tag{1.1}$$

where $F : \mathbb{R}^{2n+1} \to \mathbb{R}^n$*,* $t \in \mathbb{R}$ *is the independent variable,* $x, x' : \mathbb{R} \to \mathbb{R}^n$ *are the dependent variables and* $\partial F / \partial x'$ *is (potentially) singular. For reasons of clarity we write* x *instead of* $x(t)$ *which will only be used if the time dependence has to be emphasized. Endowing* (1.1) *with initial values* (x_0, x'_0) *is referred to as Initial Value Problem (IVP), i.e.*

$$\begin{aligned} F(t, x, x') &= 0, \\ x(t_0) &= x_0, \\ x'(t_0) &= x'_0. \end{aligned} \tag{1.2}$$

In many applications not all time derivatives of the variables x_i appear in the DAE system explicitly and some equations are free of derivative. Then the variables and equations can be separated into two parts: The *differential variables* $y \in \mathbb{R}^{n_d}$ and *algebraic variables* $z \in \mathbb{R}^{n_a}$ with $n_d + n_a = n$ which together yield the vector $x = (y, z)$ and the *differential equations* $f(t, y, y', z) : \mathbb{R}^{2n_d+n_a+1} \to \mathbb{R}^p$ and *algebraic equations* $g(t, y, z) : \mathbb{R}^{n_d+n_a+1} \to \mathbb{R}^q$ with $p + q = n$. Then (1.1) can be rewritten as

$$F(t, x, x') = \begin{bmatrix} f(t, y, y', z) \\ g(t, y, z) \end{bmatrix} = 0. \tag{1.3}$$

If the system in (1.1) can be written as

$$A(t, x)x' + \varphi(t, x) = 0 \tag{1.4}$$

with $A, \varphi : \mathbb{R}^{n+1} \to \mathbb{R}^n$, the system is denoted as *linearly implicit.* Furthermore, the DAE is referred to as *semi-explicit nonlinear* if it has the form

$$\begin{aligned} y' &= f(t, y, z), \\ 0 &= g(t, y, z). \end{aligned} \tag{1.5}$$

1.1. Solvability and the Index

From [57] we have the following definition of a solution.

Definition 1.2 *Let* (1.1) *with sufficiently smooth F be given. Then the function $x : \mathbb{R} \to \mathbb{R}^n$ is a* solution *of* (1.1) *if $x \in \mathcal{C}^1(\mathbb{R}, \mathbb{R}^n)$ and x satisfies* (1.1) *pointwise. The function x is a* solution of the initial value problem (1.2) *if it is a solution of* (1.1) *and satisfies the initial values in* (1.2)*.*

Definition 1.3 *An initial value (x_0, x_0') is denoted as* consistent *if it satisfies* (1.1) *at time t_0 and the corresponding IVP* (1.2) *has at least one solution.*

In [59], the following definitions for the solvability of a general DAE are given which guarantees the existence of solutions x of solvable DAEs, that the value $x_0 = x(t_0)$ uniquely defines x and that no bifurcation of the solutions, see *e.g.* [110], occurs.

Definition 1.4 *The DAE in* (1.1) *is* solvable *on a subset $\Omega \subset \mathbb{R}^{n+1}$ if, for some $r \geq 1$, there is an r-parameter family of solutions $x(c)$ with $t \in \mathbb{R}$ and $c \in \mathbb{R}^r$ such that*

1. *$(t, x(c)) \in \Omega$,*
2. *If $\hat{x}$ is any solution whose traces lies in Ω, then $\hat{x} = x(\hat{c})$ for some $\hat{c}$,*
3. *The graph of x(c) is an $(r+1)$-dimensional manifold in $\mathbb{R}^{n+1}$.*

Definition 1.5 *The DAE in* (1.1) *is* smoothly solvable *on $[t_0, T]$ if there exists a $m \in \mathbb{N}$ such that F has m continuous derivatives and the nonlinear system of "derivative equations"*

$$\begin{aligned} F(t, \xi_0, \xi_1) &= 0, \\ \frac{d}{dt} F(t, \xi_0, \xi_1, \xi_2) &= 0, \\ &\vdots \\ \left(\frac{d}{dt}\right)^m F(t, \xi_0, \xi_1, \cdots, \xi_{m+1}) &= 0 \end{aligned} \tag{1.6}$$

when viewed as relating the independent symbols $t, \xi_0, \xi_1, \cdots, \xi_{m+1}$ can be solved on $[t_0, T]$ for ξ_1 in terms of ξ_0 and t, i.e. $\xi_1 = \phi(t, \xi_0)$ where the mapping ϕ is Lipschitz continuous. Here $\frac{d}{dt}$ indicates the total (or material) derivative.

In [11], the following characterization for the solvability for one type of DAEs is given.

Theorem 1.6 *The* linear constant coefficient *DAE*

$$Ax' + Bx = \varphi(t) \tag{1.7}$$

with $A, B \in \mathbb{R}^{n \times n}$ and $\varphi : \mathbb{R} \to \mathbb{R}^n$ is solvable if and only if $f(\lambda) = \det(\lambda A + B)$ is not identically zero.

Next we define the *index* of a DAE. The index enables a classification of DAEs and plays an important role when numerical methods are applied. Loosely speaking, the (differentiation) index can be seen as a measure of the distance from the DAE to an ODE. Over the years multiple indices have been defined from which we only present two, the differentiation and perturbation index. For a review on all indices, we refer to [66]. From [11] we cite

Definition 1.7 *Let the DAE system be given by* (1.1). *Then the* differentiation index ν_d *is the number of times that all or a part of the equations must be differentiated with respect to time in order to determine x' as a continuous function of x and t.*

In particular the differentiation index equals the integer m in definition 1.5. From [35] we cite

Definition 1.8 *Let the DAE system be given by* (1.1). *Then it has* perturbation index ν_p *along a solution $x(t)$ for $t \in [0, t_{max}]$, if ν_p is the smallest integer such that for all functions $\hat{x}(t)$ having a defect*

$$F(t, \hat{x}, \hat{x}') = \delta(t), \tag{1.8}$$

there exists an estimate on $[0, t_{max}]$

$$\|\hat{x}(t) - x(t)\| \leq C \left(\|\hat{x}(0) - x(0)\| + \sum_{i=0}^{v_p - 1} \max_{0 \leq \xi \leq t} \| \left(\frac{d}{dt} \right)^i \delta(\xi) \| \right) \tag{1.9}$$

whenever the right hand side is sufficiently small.

The perturbation index measures the sensitivity of the solution with respect to perturbations. In [31], the relation between these two indices was investigated. For general DAEs we obtain $\nu_d \leq \nu_p \leq \nu_d + 1$. If the derivatives in the system result from total differentials of functions, *e.g.* conservation of mass or energy, then $\nu_d = \nu_p$ holds. From this point on, only the differentiation index ν_d of a DAE is investigated and the short notation index ν is used instead.

Definition 1.9 *The index is referred to as* uniform *if it is independent of the evaluation point.*

Next, we characterize index 1 systems. The implicit function theorem plays an important role in several proofs. We cite this theorem from [53].

Theorem 1.10 (Implicit Function) *Let X, Y, Z be finite dimensional vector spaces. Let $\phi \in \mathcal{C}^1(U, Z)$ with a neighbourhood $U \subset X \times Y$ of a root (a, b) of ϕ. Let $\frac{\partial \phi}{\partial Y}(a, b)$ be invertible. Then there are neighbourhoods $U' \subset X$ of a and $U'' \subset Y$ of b and a $\psi \in \mathcal{C}^1(U', U'')$ such that the set of roots of ϕ which lies within $U' \times U'' \subset U$ equals the graph of ψ, i.e.*

$$\phi(x, y) = 0,\ (x, y) \in U' \times U'' \quad \Leftrightarrow \quad y = \psi(x),\ x \in U'. \tag{1.10}$$

Theorem 1.11 *Let the semi-explicit DAE be given by* (1.5). *Then the system has index* 1 *if and only if $\partial g / \partial z$ is regular. In particular, the system can be reduced to an ODE for the variable y only.*

Proof. The index is the number of differentiations w.r.t. time of a subset of the equations necessary to obtain an explicit ODE. Differentiation of the second row in (1.5) yields

$$0 = g_t(t, y, z) + g_y(t, y, z) y' + g_z(t, y, z) z'.$$

If and only if $g_z = \partial g/\partial z$ is regular, the above equation can be rewritten to

$$z' = -g_z(t,y,z)^{-1}\left(g_t(t,y,z) + g_y(t,y,z)f(t,y,z)\right).$$

Above we used the first row of (1.5) to replace y'. Consequently, one differentiation was necessary to obtain an ODE. The second statement is proved using the implicit function theorem. Since $\partial g/\partial z$ is regular, a mapping ψ exists such that the second row of (1.5) can be replaced with $z = \psi(y)$. Finally, substituting this result into the first row, we obtain

$$y' = f(t, y, \psi(y)).$$

□

THEOREM 1.12 *Let the implicit DAE be given by* (1.3)*. Then the system has index* 1 *if $\partial f/\partial y'$ and $\partial g/\partial z$ are regular.*

Proof. If $\partial f/\partial y'$ is regular, the implicit function theorem can be applied. A mapping ψ exists such that the first row of the system in (1.3) can be rewritten to

$$\begin{aligned} y' &= \psi(t,y,z) \\ 0 &= g(t,y,z). \end{aligned}$$

Hence the implicit system was reduced to a semi-explicit system and theorem 1.11 yields that it is index 1 if $\partial g/\partial z$ is regular which is an assumption. □

From [32] we cite the following theorem without proof.

THEOREM 1.13 *Let the implicit DAE be given by* (1.1) *and $A = \partial F/\partial x'$ and $B = \partial F/\partial x$ where $A, B \in \mathbb{R}^{n\times n}$. Let A be singular and suppose a nonsingular $T \in \mathbb{R}^{n\times n}$ exist such that*

$$TA = \begin{bmatrix} A_1 \\ 0 \end{bmatrix} \tag{1.11}$$

with a $p \times n$ matrix A_1 of rank p. Furthermore let the matrix

$$\begin{bmatrix} A_1 \\ B_2 \end{bmatrix} \tag{1.12}$$

be nonsingular where

$$TB = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}. \tag{1.13}$$

Then the system has index 1.

EXAMPLE 1.14 In [79], the system

$$\begin{aligned} x' + y' &= a(t), \\ x + y^2 &= b(t) \end{aligned} \tag{1.14}$$

with $x, y \in \mathbb{R}$ and $a, b : \mathbb{R} \to \mathbb{R}$ is presented. This system has index 1 almost everywhere. First, we apply theorem 1.13 to prove that $\nu = 1$ almost everywhere. Let $T \in \mathbb{R}^{2\times 2}$ be the identity. Then

$$\begin{bmatrix} A_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2y \end{bmatrix}. \tag{1.15}$$

This matrix has full rank for $y \in \mathbb{R}\backslash\{\frac{1}{2}\}$. Hence, for all points $(x, y) \in \mathbb{R} \times \mathbb{R}\backslash\{\frac{1}{2}\}$ the system has index 1. Second, we show the same result by differentiating the second equation with respect to time, *i.e.*

$$x' + 2yy' = b'(t). \tag{1.16}$$

Replacing the second row in (1.14) and rewriting the system yields

$$\begin{aligned} x' &= a(t) - \frac{b'(t) - a(t)}{2y - 1}, \\ y' &= \frac{b'(t) - a(t)}{2y - 1}. \end{aligned} \tag{1.17}$$

The variables (x', y') are uniquely determined for all $(x, y) \in \mathbb{R} \times \mathbb{R}\backslash\{\frac{1}{2}\}$. This yields the same result as above. ♢

In the above example, the index was determined with moderate effort. For larger systems it is not applicable anymore to do this by hand. Therefore algorithms have be developed to determine the index of arbitrary DAEs. We mention two which work on a structural (symbolic) level. This means that they can be performed a priori and without knowledge of numeric values. In [79], the developed graph-theoretical algorithm was initially designed to determine consistent initial values of the system, a problem which will be discussed in section 2.1. The algorithm identifies subsets S of equations which have to be differentiated since the so gained, additional equations have to be fulfilled by the initial values as well. A side effect is the determination of the structural index ν_s. In [105], a structural algorithm is developed to analyse the DAE and one result is the structural index as well. Both algorithms operate on a structural level, thus the structural index is a lower bound of the respective differentiation index. In [79], an example of the form (1.5) is presented in which the Jacobian $\partial g/\partial z$ becomes numerically, but not structurally singular which implies that z' can not be determined uniquely and thus at least one more differentiation is necessary. This numerical singularity can not be recognized by structural algorithms, hence $\nu_s \leq \nu$ holds. In [88], a structural analysis method for DAEs is presented, namely the Σ-method. If the method succeeds, an upper bound of the index is obtained.

Next, we briefly discuss systems with index $\nu > 1$. Such systems are referred to as *higher index* systems. In [11], the following proposition is given.

Proposition 1.15 *Assume that $F(t, x, x')$ is solvable and has index ν. Then*

$$\begin{aligned} x' &= z, \\ 0 &= F(t, x, z) \end{aligned} \tag{1.18}$$

is semi-explicit and has index $\nu + 1$.

Proof. The first row determines x'. Since $F(t, x, x')$ has index ν, ν differentiations are required to determine z uniquely. Thus, differentiating once more yields z'. □

Example 1.16 (Plane Pendulum) In [59], the plane pendulum with one mass swinging on a rigid shaft is described by the system of equations

$$\begin{aligned} x'' &= \lambda x, \\ y'' &= -\lambda y - g, \\ 1 &= x^2 + y^2. \end{aligned} \tag{1.19}$$

Here, g represents the gravitational acceleration, $(x, y) \in \mathbb{R}^2$ the position of the mass and $\lambda \in \mathbb{R}$ is a Lagrange multiplier. This system has index $\nu = 3$. ♢

From a numerical point of view a DAE with smaller index is easier to handle. In [30], a method is described to reduce the index of a system. Intuitively this is achieved by differentiation of equations identified by the algorithms of [79]. Unfortunately, then the original constraints do not have to be fulfilled exactly anymore. Let a general system of the form (1.1) be given. Then we solve an extended, semi-explicit system instead, *i.e*

$$\begin{aligned} x' &= \tilde{f}(t,x) + g_x^\top(t,x)\mu, \\ 0 &= g(t,x). \end{aligned} \tag{1.20}$$

Above, $\tilde{f}$ is the transformed system gained from the original system (1.1) by differentiating a subset of equations in order to reduce the index, g represents the differentiated equations in a non-differentiated form and μ is a Lagrange multiplier. We cite the following two results without proof from [11].

THEOREM 1.17 *Let the system be given by* (1.20) *and* $\partial g/\partial x$ *have full rank. Then the system has index* 2*.*

THEOREM 1.18 *Let the system be given by* (1.20)*,* $\partial g/\partial x$ *have full rank and let* $g(t,x) = 0$ *characterize the solution manifold of* (1.1)*. Then the only solutions of* (1.20) *are* $\mu = 0$ *and* x *which is a solution of*

$$\begin{aligned} x' &= \tilde{f}(t,x), \\ 0 &= g(t,x). \end{aligned} \tag{1.21}$$

The last assumption in the above theorem implies, that a solution $\hat{x}$ at time $\hat{t}$ satisfying (1.1) also satisfies $g(\hat{t},\hat{x}) = 0$ for fixed $\hat{t}$. For further readings concerning index reduction methods, we refer to [10, 35, 79, 100, 105]. In the next section we elaborate numerical methods to solve the IVP from (1.2).

1.2. Numerical Methods

Within this section we present the linear multistep methods. First, we derive the methods on equidistant grids for ODEs and then extend the results to DAEs with index 1 and arbitrary grids. We follow [18]. Let the explicit ODE endowed with initial conditions be given by

$$\begin{aligned} x' &= f(t,x), \\ x_0 &= x(t_0) \end{aligned} \tag{1.22}$$

where $t \in I \subset \mathbb{R}$, $x \in \mathbb{R}^n$ and $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$. Let $x(t) \in \mathcal{C}^1(I, \mathbb{R}^n)$ be a solution of (1.22). Let the discrete times be $t_k \in I$ for $k = 0, ..., N$ and the step size $h = t_k - t_{k-1}$ be uniform for all time steps. Let $D = \{t_0, ..., t_N\}$, then we look for a discrete function $x_h(t) : D \to \mathbb{R}^n$ that approximates the solution $x(t)$ on the grid, *i.e.*

$$x_h(t_k) \approx x(t_k) \quad \text{for } t_k \in D. \tag{1.23}$$

In the remainder we use the shorthand notations $x_h(t_k) = x_k$ and $f(t_k, x_k) = f_k$. The *general linear multistep method* with d steps is defined as

$$\sum_{j=0}^{d} \alpha_{d-j} x_{k-j} = h \sum_{j=0}^{d} \beta_{d-j} f_{k-j}. \tag{1.24}$$

The coefficients $\alpha_j, \beta_j \in \mathbb{R}$ for $j = 0, ..., d$ are fixed and we require $\alpha_d \neq 0$ and $|\alpha_0| + |\beta_0| > 0$. If $\beta_d = 0$ the method is referred to as *explicit*, otherwise *implicit*.

Example 1.19 The simplest methods are the one-step explicit and implicit *Euler*-methods. In the explicit case we have $\alpha_d = 1$, $\alpha_{d-1} = -1$, $\beta_d = 0$, $\beta_{d-1} = 1$. This yields the recursion formula

$$x_k = x_{k-1} + hf_{k-1}. \tag{1.25}$$

For the implicit method, the coefficients are $\alpha_d = 1$, $\alpha_{d-1} = -1$, $\beta_d = 1$, $\beta_{d-1} = 0$ and we obtain

$$x_k = x_{k-1} + hf_k. \tag{1.26}$$

$\diamondsuit$

We cite the following first result from [18], see *Satz* 7.3 for the proof which is an application of the *Banach fixed-point theorem.*

Theorem 1.20 (Existence and Uniqueness) *Let $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be Lipschitz continuous, i.e.*

$$\|f(t, x) - f(t, \tilde{x})\| \leq L\|x - \tilde{x}\| \tag{1.27}$$

for all $x, \tilde{x} \in \mathbb{R}^n$ and $t \in \mathbb{R}$. Then there exists for

$$h = \frac{|\alpha_d|}{|\beta_d|L} \tag{1.28}$$

and arbitrary initial values $x_0, ..., x_d \in \mathbb{R}^n$ a unique $x_h(t) : D \to \mathbb{R}^n$ which fulfills (1.24).

From a numerical method we expect that the error of the approximation decreases with the step size and vanishes in the limit. This motivates the following definition.

Definition 1.21 (Convergence) *Let $x(t) \in \mathcal{C}^1(I, \mathbb{R}^n)$ be a solution of* (1.22). *A multistep method* converges *to a solution if*

$$\lim_{h \to 0} x_h(t_k) = x(t_k) \tag{1.29}$$

for all grid points $t_k \in D$ and the initial values $x_0, ..., x_d \in \mathbb{R}^n$ of the multistep method satisfy

$$\lim_{h \to 0} x_h(t_0 + ih) = \lim_{h \to 0} x_i = x_0 \tag{1.30}$$

for all $i = 0, ..., d - 1$. It is convergent *if it converges for arbitrary IVPs with sufficiently smooth right hand side.*

Since the above definition may be hard to prove, appropriate conditions, easier to verify, have to be elaborated which then imply convergence. To simplify the notations, we define the *shift*-operator $Ex : I \to \mathbb{R}^n$ as

$$(Ex)(t) = x(t + h) \tag{1.31}$$

and the *characteristic polynomials*

$$\rho(\xi) = \sum_{j=0}^{d} \alpha_j \xi^j \quad \text{and} \quad \sigma(\xi) = \sum_{j=0}^{d} \beta_j \xi^j. \tag{1.32}$$

Then the general linear multistep method from (1.24) can be rewritten as

$$\rho(E)x_{k-d} = h\sigma(E)f_{k-d}. \tag{1.33}$$

The above equation is also referred to as (inhomogeneous) linear *difference equation.*

Definition 1.22 (Consistency) *If for all $x(t) \in \mathcal{C}^\infty(I, \mathbb{R}^n)$*

$$\rho(E)x(t) - h\sigma(E)x'(t) = \mathcal{O}(h^{p+1}) \tag{1.34}$$

holds uniform for all $t \in I$, $h > 0$, a linear multistep method is consistent *with order p.*

Next we consider stability. Recall the definition for ODEs.

Definition 1.23 (Stability of the ODE) *Let the IVP from* (1.22) *be given and $x(t), \tilde{x}(t) \in \mathcal{C}^1(I, \mathbb{R}^n)$ be solutions of the ODE for initial values $x_0, \tilde{x}_0 \in \mathbb{R}^n$. If for arbitrary $\varepsilon > 0$ there exists a $\delta > 0$ such that*

$$\|x_0 - \tilde{x}_0\| < \delta \tag{1.35}$$

implies

$$\|x(t) - \tilde{x}(t)\| < \varepsilon, \tag{1.36}$$

the IVP is stable. *If there exists a $\delta > 0$ such that*

$$\lim_{t \to \infty} \|x(t) - \tilde{x}(t)\| = 0, \tag{1.37}$$

it is asymptotically stable. *Otherwise,* instable.

Definition 1.24 (Stability of the difference equation) *Let the homogeneous linear difference equation be given by*

$$\rho(E)x_k = 0 \tag{1.38}$$

with $\alpha_j \in \mathbb{R}$ for $j = 0, ..., d$ and given initial values $x_0, ..., x_{d-1} \in \mathbb{R}^n$. Then the difference equation is stable *if there exists a constant $M > 0$ such that $\|x_k\| \leq M$ for $k \in \mathbb{N}$. If*

$$\lim_{k \to \infty} x_k = 0, \tag{1.39}$$

it is asymptotically stable. *Otherwise,* instable.

We cite the following theorem from [18], see *Satz* 3.40 and *Satz* 3.33 for the proof.

Theorem 1.25 (Dahlquist Root Condition) *Let $\lambda \in \mathbb{C}$ be a root of the characteristic polynomial $\rho(\xi)$ of the linear difference equation. Then the difference equation is stable if $|\lambda| \leq 1$ holds for all λ and $|\lambda| = 1$ only for simple roots. If $|\lambda| < 1$ for any root, it is asymptotically stable.*

Definition 1.26 (Stability of the general multistep method) *The general linear multistep method is stable if the corresponding homogeneous linear difference equation is stable.*

We cite the following very important theorem. The proof is given in [18], *Satz* 7.23.

Theorem 1.27 (Lax-equivalence) *A stable, consistent linear multistep method is convergent. In particular, let* (1.22) *be given, with $f(t,x) \in \mathcal{C}^p(\mathbb{R}^{n+1}, \mathbb{R}^n)$, $p \geq 1$ and the solution $x(t) \in \mathcal{C}^1(I, \mathbb{R}^n)$. Assume that the stable linear multistep method is consistent with order p. Let the initial values be $x_0, ..., x_{d-1} \in \mathbb{R}^n$ and the starting error*

$$\varepsilon_0 = \max_{0 \leq l \leq d-1} (x_l - x(t_l)). \tag{1.40}$$

Then there exist constants $C, \hat{\varepsilon}, \hat{h} > 0$, such that for $h \leq \hat{h}$, $\varepsilon_0 \leq \hat{\varepsilon}$ the discretization error is

$$\|\varepsilon_k\| = \|x_k - x(t_k)\| \leq C(\varepsilon_0 + h^p). \tag{1.41}$$

for $k = 0, ..., N$.

Hence the order of convergence depends on the consistency, the smoothness of the right hand side and the initial values chosen to start the recursion with. Instead of proving convergence, usually stability and consistency are shown and the Lax-equivalence theorem is applied. There are two types of multistep methods, the ones named after *John Couch Adams* and the Backward Differentiation Formulas (BDFs). We only discuss the latter. The general formula for the BDFs reads as

$$\sum_{j=0}^{d} \alpha_{d-j} x_{k-j} = h f_k \tag{1.42}$$

where we used the normalization $\beta_d = 1$.

Lemma 1.28 *The d-step BDF is stable if and only if $d < 7$.*

This proposition can be shown by evaluating the roots of the corresponding characteristic polynomial $\rho(\xi)$ for $d < 7$ and is left to the reader. A proof for the instability for $d \geq 7$ can be found in [34]. The one-step BDF is the implicit Euler method, see example 1.19. In practice, a new time step $t_{k-1} \to t_k$ is often computed by a predictor-corrector scheme. In the predictor-stage the new solution x_k at time t_k is predicted by x_k^0. The value x_k^0 can be calculated *e.g.* by using information of the derivatives of pasts steps or by applying polynomial interpolation of order l to the previous solutions $x_{k-l-1}, ..., x_{k-1}$ and evaluating the polynomial at t_k. Then x_k^0 serves as the starting value for the corrector-stage where a system of (nonlinear) equations has to be solved by a Newton-type iteration. We derive the iteration for implicitly given ODEs. Let the IVP from (1.22) be written as

$$\begin{aligned} F(t, x, x') &= 0, \\ x_0 &= x(t_0) \end{aligned} \tag{1.43}$$

with $F : \mathbb{R}^{2n+1} \to \mathbb{R}^n$. We apply the BDF method to the above system. First, substituting $x'_k = f_k$ in (1.42) yields

$$\sum_{j=0}^{d} \alpha_{d-j} x_{k-j} = h x'_k. \tag{1.44}$$

Inserting this into the implicit ODE leads to

$$F\left(t_k, x_k, \frac{1}{h}\sum_{j=0}^{d} \alpha_{d-j} x_{k-j}\right) = 0 \tag{1.45}$$

which has to be solve with respect to x_k at each time t_k. The above system is also referred to as *corrector equation.* The system is a composition of the mapping $g : \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ with

$$g(x_k) = \begin{pmatrix} x_k \\ \frac{1}{h}\sum_{j=0}^{d} \alpha_{d-j} x_{k-j} \end{pmatrix} \tag{1.46}$$

and the implicit ODE. Hence, Newton's method can be applied to $F(t_k, g(x_k)) : \mathbb{R}^n \to \mathbb{R}^n$ for fixed t_k. Let the iteration index be i, then the update formula reads as

$$x_k^{i+1} = x_k^i + \Delta x^i. \tag{1.47}$$

with

$$\Delta x^i = -J_x^{-1}(F)(t_k, x_k^i) \cdot F\left(t_k, g(x_k^i)\right) \tag{1.48}$$

Applying the chain rule yields the Jacobian

$$J_x(F)(t_k, x_k^i) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial x'}\right) \cdot \begin{pmatrix} 1 \\ \frac{\alpha_d}{h} \end{pmatrix} (t_k, x_k^i) = \left(\frac{\partial F}{\partial x} + \frac{\alpha_d}{h}\frac{\partial F}{\partial x'}\right)(t_k, x_k^i). \tag{1.49}$$

The termination criteria of the iteration with $\varepsilon, \delta > 0$ read as

$$\|F(x)\|_2 < \varepsilon \quad \text{and} \quad \|\Delta x\| = \left(\sum_{\iota=1}^{n} \left|\frac{\Delta x_\iota}{x_\iota}\right|^2\right)^{\frac{1}{2}} < \delta. \tag{1.50}$$

Theorem 1.29 *Newton's method converges locally quadratic.*

A proof of the above statement can be found in [19], page 101f. The above theorem yields that the initial value x_k^0 has to lie in a ε-neighbourhood of the solution and the Jacobian has to be invertible such that convergence is achieved. Next we extend the BDFs to DAEs. We cite a convergence result of the BDF methods for index 1 DAEs from [11]. A proof can be found therein.

Theorem 1.30 *Let* (1.2) *have uniform index* 1 *on an interval* $I \subset \mathbb{R}$. *Then the d-step BDF method with fixed step size h and* $d < 7$ *converges with* $\mathcal{O}(h^p)$ *if the starting error* $\varepsilon_0 = \mathcal{O}(h^p)$ *and Newton's method at each step is solved with* $\mathcal{O}(h^{p+1})$ *accuracy.*

Convergence results are also available for semi-explicit index 1, 2 and 3 DAEs as well as for constant coefficient linear DAEs with arbitrary index. To improve the robustness of the BDF, adaptive time stepping and order selection have been proposed. For a description of a systematic way to construct variable step size formulas we refer to [96] and in [71] control mechanisms for the step size are discussed. It can be shown that variable step size BDF methods converge for index 1 DAEs if their implementation is stable for standard ODEs, see [11]. An implementation of these extensions is the Differential Algebraic System Solver (DASSL) of [80] which is used to solve the models described in chapter III. DASSL utilizes adaptive time stepping and order selection coupled with a Predictor-Corrector-Scheme and an integration error control. A detailed description is also given in [11].

But the BDFs are not the only numerical methods to solve DAEs. We highlight two more without giving any details. First, implicit Runge-Kutta methods can be applied to DAEs, see [35]. For these methods also an implementation is available in Fortran and C++, namely *RADAU5.* And second, a newer approach using Taylor series expansion presented in [73, 74, 75, 87, 88] allows to solve DAEs of arbitrary index but requires time derivatives of the equations.

2. The Initialization System

A crucial point in solving the IVP in (1.2) is already reached at the very beginning. Consistent initial values have to be provided such that the IVP has a solution. On the one hand, many DAE solvers have serious problems when inconsistent initial values are given, see [11, 115]. Finding such values might be the most challenging part of solving a system of DAEs. On the other hand, it is known that DAE-solvers tend to be robust if the initial values are consistent. This reflects the emphasis which the task of finding such values has to receive. To obtain initial values, we choose a similar approach as in [115], set up a system of equations at initial time t_0 and solve it with Newton's method. If this idea can not be applied and the integration method has to be started directly with possibly not consistent initial values, we refer to [55, 108]. A method for finding consistent initial values for DAEs with index $\nu = 1$ is described in [12], with index $\nu \leq 2$ in [58] and for general problems in [79].

Without loss of generality, let the DAE be given in the form of (1.3), *i.e.*

$$\begin{aligned} f(t, y, y', z) &= 0, \\ g(t, y, z) &= 0 \end{aligned} \tag{2.1}$$

where $y, y' \in \mathbb{R}^{n_d}$ and $z \in \mathbb{R}^{n_a}$, $f(t, y, y', z) : \mathbb{R}^{2n_d+n_a+1} \to \mathbb{R}^p$ and $g(t, y, z) : \mathbb{R}^{n_d+n_a+1} \to \mathbb{R}^q$ with $n_d + n_a = n$ and $p + q = n$. Our aim is to obtain the initial values (y_0, y_0', z_0) by solving

$$\begin{aligned} f(t_0, y_0, y_0', z_0) &= 0, \\ g(t_0, y_0, z_0) &= 0. \end{aligned} \tag{2.2}$$

The above system is referred to as *initialization system.*

2.1. Consistent Initialization

Given initial values satisfying the initialization system does not imply that the corresponding IVP has a solution. If so, the initial values are not consistent, see definition 1.3 and example 2.2 below. A necessary condition for consistent initial values is given in the following statement.

Theorem 2.1 (Necessary Condition) *If a vector (t_0, y_0', y_0, z_0) is a consistent initial value, then it fulfills the initialization system.*

Proof. This is an immediate consequence of definition 1.3. □

The initialization system consists of $2n_d + n_a$ variables and $n_d + n_a$ equations since $t = t_0$ is fixed. Hence, the system has n_d degrees of freedom and therefore n_d variables have to be *initialized*, *i.e.* provided as given values. For ODEs initializing is trivial since an arbitrary vector $y_0 \in \mathbb{R}^n$ will be consistent but for DAEs finding consistent initial values can be a challenging task. In contrast to the literature, we only allow differential variables y_0 or their derivatives y_0' to be initialized. This restriction reduces the set of possible initializations and a sharper distinction between differential and algebraic variables is achieved. The initialization system in (2.2) together with n_d initializations of either the differential variables or their derivatives is then solved by Newton's method. The iteration matrix J is given by

$$J(t, y, y', z) = \begin{pmatrix} \frac{\partial f}{\partial y'} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ 0 & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \\ \tilde{I}_1 & \tilde{I}_2 & 0 \end{pmatrix} (t, y, y', z) \tag{2.3}$$

where $\tilde{I}_1 + \tilde{I}_2 = I$, the identity matrix in $\mathbb{R}^{n_d \times n_d}$. Obviously, $\tilde{I}_1$ and $\tilde{I}_2$ have to be chosen such that J is regular at least in a neighbourhood of the solution. The convergence of Newton's method applied to a continuously differential function is guaranteed in a neighbourhood of $(t_0,\ y_0,\ y_0',\ z_0)$. Hence starting values for Newton's method, referred to as *estimates*, sufficiently close to the solution have to be provided. Though, the example below demonstrates that a solution of the initialization system is not necessarily consistent.

Example 2.2 Let the DAE system with index 2 be

$$\begin{aligned} x' &= y, \\ x &= g(t). \end{aligned} \tag{2.4}$$

The differential variable x is determined by the function $g(t) : \mathbb{R} \to \mathbb{R}$ for all time and if x' is initialized with an arbitrary value $c \in \mathbb{R}$, the initialization system can be solved uniquely. Still, only the value $c = g'(t_0)$ is consistent for x'_0. $\diamondsuit$

The above example demonstrates that *hidden constraints* can appear which have to be satisfied by the initial values such that they are consistent. Hidden constraints emerge from existing equations by differentiation with respect to time. We follow [79] in which the Pantelides algorithm is proposed to identify subsets of equations which have to be differentiated and added to the initialization system in order to obtain consistent initial values. Let $\xi = (y', z)$ and $\varphi(t, y, \xi)$ be a subset of equations of (2.2) consisting of $1 \le k \le n$ equations. If

$$\operatorname{rank}\left(\frac{\partial \varphi}{\partial \xi}\right) < k, \tag{2.5}$$

these equation have to be differentiated with respect to time yielding new variables ζ which are derivatives of ξ with respect to time. The differentiated equations are hidden constraints and added to the set $\mathcal{H}$. After identifying all subsets $\varphi(t, y, \xi)$ for which (2.5) holds for minimal k, the equations $\varphi(t, y, \xi)$ are replaced with their time derivatives and the emerging system is rewritten such that only first order derivatives occur. Then the procedure is repeated for the transformed system until no further singular subsystems are found.

After the above algorithm terminated, all differentiated equations in $\mathcal{H}$ are added to (2.2) and the *augmented initialization system* is given by

$$\begin{aligned} f(t_0, y_0, y'_0, z_0) &= 0, \\ g(t_0, y_0, z_0) &= 0, \\ h(t_0, y_0, y'_0, z_0, \zeta_0) &= 0. \end{aligned} \tag{2.6}$$

Let the hidden constraints from $\mathcal{H}$ be collected in $h = (h_1, ..., h_m) : \mathbb{R}^{2n_d+n_a+1+d} \to \mathbb{R}^m$ and all new variables be gathered in $\zeta_0 \in \mathbb{R}^d$. Then $n_d + d - m$ variables have to be initialized such that the above system becomes regular. Solving (2.6) yields consistent initial values, see [79], and we have

Theorem 2.3 (Sufficient Condition) *If a vector $(t_0, y'_0, y_0, z_0, \zeta_0)$ fulfills the augmented initialization system then (t_0, y'_0, y_0, z_0) is a consistent initial value.*

Corollary 2.4 *Let the implicit DAE be of the form* (2.1). *If for arbitrary given $y_0 \in \mathbb{R}^{n_d}$ the initialization system* (2.2) *is solvable with respect to (y'_0, z_0) and the Jacobian given in* (2.3) *is regular, the system has at most index* 1. *Furthermore, the initial values (y_0, y'_0, z_0) are consistent.*

Proof. If $y_0 \in \mathbb{R}^{n_d}$ is initialized, the Jacobian of (2.3) is given by

$$J = \begin{pmatrix} \frac{\partial f}{\partial y'} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ 0 & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \\ 0 & I & 0 \end{pmatrix}. \tag{2.7}$$

Hence y can be removed from the system. This yields

$$\hat{J} = \begin{pmatrix} \frac{\partial f}{\partial y'} & \frac{\partial f}{\partial z} \\ 0 & \frac{\partial g}{\partial z} \end{pmatrix}. \tag{2.8}$$

Since J is assumed to be regular, also $\partial f/\partial y'$ and $\partial g/\partial z$ are regular and which guarantees an index-1 DAE systems, see theorem 1.12. For the second statement, let $\xi = (y', z)$ and $\varphi(t, x, \xi) = (f, g)(t, x, \xi)$, *i.e.* the whole system consisting of n equations. The regularity of $\hat{J}$ is equivalent to $\operatorname{rank}(\partial\varphi/\partial\xi) = n$ and hence no hidden constraints exist. □

In process models the initialization system is often sparse. In [23], a method is described to decompose sparse systems of equations into smaller ones which are then solved sequentially. Hence, the size of each subsystem is smaller from which further robustness and the reduction of computational costs is expected. The method is described for linear systems but can be applied to nonlinear systems as well. Following [23] we explain the method below.

2.2. Block Decomposition

It is assumed that a sparse nonlinear system of the form

$$F(x) = 0 \tag{2.9}$$

has to be solved with $F : \mathbb{R}^n \to \mathbb{R}^n$ and $x \in \mathbb{R}^n$. The goal is to find permutation matrices P and Q such that a system in *lower block triangular form* is gained, *i.e.*

$$PFQ = \begin{pmatrix} B_{11} & & & \\ B_{21} & B_{22} & & \\ \vdots & & \ddots & \\ B_{m1} & B_{m2} & \cdots & B_{mm} \end{pmatrix}. \tag{2.10}$$

If a system can not be decomposed into a block triangular form, it is said to be *irreducible*, otherwise *reducible*. If the form above is the final result, each block B_{ii} is square and irreducible. Given the lower block triangular form, instead of computing the solution of (2.9) we solve (2.10) sequentially beginning with B_{11} and substituting the results into the "lower" blocks. This procedure brings several benefits. Solving (2.10) sequentially is expected to need less Jacobian evaluations since variables are already substituted into lower blocks. The Jacobian for each block is smaller which reduces the effort of its factorization. Furthermore, the error bounds have to be satisfied by less variables which implies that the portion of each variable becomes more significant.

For some classes of problems the block decomposition is not applicable, *e.g.* for systems gained from the method of lines. There, the spatial derivatives of PDEs are discretized utilizing finite differences. The resulting systems are irreducible. Also if the system is rather dense, there may be hardly any benefits. We follow the approach of [23] and divide the computation of the blocks into two steps. First, a row permutation is performed to obtain a maximal transversal. Here, the *transversal* is defined as the set of non-zero elements on the main diagonal. The transversal is referred to as *maximal* if it contains the maximal number of non-zero main diagonal elements which can be obtained by (row) permutation. Hence, in the first step the goal is to find a row permutation P_1 such that the number of non-zero main diagonal elements in F is maximal. Second, a symmetric permutation Q is applied to P_1F to achieve the actual block structure. Together, this yields $PFQ := Q^\top P_1 F Q$. Both algorithms operate on a symbolic level, meaning that only the position of the variables in the system is of importance. Therefore the ideas can be applied to linear and nonlinear systems. Only a symbolic portrayal of the system in (2.9) is required. If the system is nonlinear, the symbolic portrayals of the system and the Jacobian are equivalent. We only present the basic ideas of the two steps and refer to the literature for further details.

2.2.1. Maximal Transversal Algorithm

The algorithm was originally proposed in [56] where it is applied to the assignment problem. This problem occurs *e.g.* in economics but the same ideas can be applied to problems in graph and games theory or finding a maximal transversal. The algorithm can be implemented in a breath-first or depth-first search and is described in a column orientation. Therefore, row permutations are used to attain the desired from. The two main requirements the algorithm has to fulfill are:

1. at each step one additional non-zero element is placed onto the diagonal and
2. all prior diagonal elements are preserved.

The algorithm starts from the first diagonal element in the first column and sequentially fills the main diagonal. The second point from above does not prohibit the exchange of rows in the already established region as long as the diagonal structure is preserved. Placing an element onto the diagonal is called an *assignment*. If a step fails and no assignment is made, the matrix is *structurally (symbolically) singular*. Nevertheless, the algorithm performs n steps but can leave empty spaces on the diagonal indicating structural singularities. If all non-zero diagonal elements are filled, the matrix is said to be *structurally regular*. The number of non-zero diagonal elements is referred to as *structural rank*. A detailed presentation of a general assignment algorithm would go beyond the scope of this thesis and we refer to [56] or [23], *p.* 109ff.

Example 2.5 To demonstrate the principles of the maximal transversal algorithm, a simple example is given. In equation (2.11) below, the steps of the algorithms are illustrated.

$$\begin{aligned}
\begin{pmatrix} & \times & & \\ \times & & \times & \\ \times & & & \times \\ & & & \end{pmatrix}
&\xrightarrow{(1)}
\begin{pmatrix} \boxed{\times} & & \times & \\ & \times & & \\ \times & & & \times \\ & & & \end{pmatrix}
\xrightarrow{(2)}
\begin{pmatrix} \boxed{\times} & & \times & \\ & \boxed{\times} & & \\ \times & & & \times \\ & & & \end{pmatrix} \\
&\xrightarrow{(3)}
\begin{pmatrix} \boxed{\times} & & & \times \\ & \boxed{\times} & & \\ \times & & \boxed{\times} & \\ & & & \end{pmatrix}
\xrightarrow{(4)}
\begin{pmatrix} \boxed{\times} & & & \times \\ & \boxed{\times} & & \\ \times & & \boxed{\times} & \\ & & & \boxed{} \end{pmatrix} .
\end{aligned} \tag{2.11}$$

In the first step (1), the rows 1 and 2 are interchanged to place a non-zero element on the diagonal entry of the first row. In the second step (2), a non-zero diagonal entry in the second row already exists and hence no permutation is performed. To assign a non-zero diagonal entry in the third row, in step (3) the rows 1 and 3 are exchanged. Finally in step (4), no permutation can be found to obtain a non-zero diagonal entry in the last row without violating the second requirement of the algorithm. Thus the system is structurally singular. In this example the permutated row sequence after the algorithm reads as $(3, 1, 2, 4)$. ♢

2.2.2. Block Triangular Algorithm

We assume that the maximal transversal algorithm was performed and the matrix is not structurally singular. Then a row permutation P_1 was found and we already computed P_1F having no zeros on the main diagonal. Next, a symmetric permutation Q is required to obtain the final form as desired in (2.10). Combining the permutations yields $P = Q^\top P_1$ yields the initial goal, *i.e.* $PFQ := Q^\top P_1FQ$. An advantage of symmetric permutations is that they preserve the maximal transversal, *i.e.* the main diagonal. In this case, it is convenient to regard P_1F as the adjacency matrix of a directed graph.

Then the symmetric permutations correspond to relabelling the nodes of the graph. This approach seems to be more illustrative than thinking of permuting certain equations and variables at the same time.

To find Q, first, Tarjan's strongly connected components Algorithm, presented in [101], is applied. This algorithm finds the strongly connected components in a directed graph and these components C_i correspond to the blocks B_{ii} from equation (2.10). To do so, the algorithm tries to find maximal closed paths through the nodes. A *closed path* is a sequence of connected nodes starting and ending at the same node. The smallest set containing a closed path is a single node. Given the strongly connected components, the nodes have to be relabelled such that the final form as desired in equation (2.10) is achieved. Therefore, first, the nodes corresponding to one component are gathered such that the indices of the nodes are consecutive and second, a block B_{ii} must precede B_{jj} if component C_j is connected to C_i. For a detailed description of the whole algorithm we refer to [23] *p.* 114ff and [101].

Example 2.6 The block triangular algorithm is illustrated by means of equation (2.12). Therein the system on the left hand side corresponds to P_1F and the strongly connected components are $\{1\}$, $\{2,3,5\}$ and $\{4\}$. In step (1) the nodes are relabelled such that the indices of the nodes in a component are consecutive. The nodes 4 and 5 are interchanged yielding the (sorted) components $C_1 = \{1\}$, $C_2 = \{2,3,4\}$ and $C_3 = \{5\}$ which are framed in the second matrix of (2.12). In step (2) the nodes are reordered such that the lower block triangular form is achieved. Since C_2 is not connected to any other component, it becomes first. Component C_3 is only connected to C_2 yielding that C_3 has to follow C_2. Finally, C_1 is connected to C_2 and C_3 implying that it has to be last. Consequently, the relabelling of the nodes reads as $\{2,3,4\} \to \{1,2,3\}$, $\{5\} \to \{4\}$ and $\{1\} \to \{5\}$ and on the right hand side of (2.12) the final form $Q^\top P_1FQ$ is given.

$$\begin{pmatrix} \times & \times & & \times & \\ & \times & \times & & \\ & \times & \times & & \times \\ & & \times & \times & \\ & & \times & & \times \end{pmatrix} \xrightarrow{(1)} \begin{pmatrix} \boxed{\times} & \times & & & \times \\ & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \\ & & \times & & \boxed{\times} \end{pmatrix} \xrightarrow{(2)} \begin{pmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & & \\ & \times & & \boxed{\times} & \\ \times & & & \times & \boxed{\times} \end{pmatrix} \tag{2.12}$$

◇

3. Regularization of the Initialization System

Consistent initial values are obtained from the augmented initialization system. If an automatic differentiation software is not available to compute the hidden constraints in $\mathcal{H}$, at least the initialization system has to be solved. Corollary 2.4 guarantees consistency if all states can be initialized. The conditions of this corollary hold *e.g.* for the models developed in chapter III. However, the initialization system for general DAEs is underdetermined and has n_d degrees of freedom. Therefore, n_d variables have to be initialized and there are

$$\binom{n}{n_d} = \frac{n!}{(n-n_d)! \cdot n_d!} \tag{3.1}$$

combinations of initialized variables if we neglect the restriction that we only allow differential variables to be initialized. Most certainly not all combinations will result in a solvable, *i.e. regular*, initialization system. The regularity of the Jacobian, at least in a neighbourhood of a solution, is essential when the initialization system (2.2) has to be solved using Newton's method. Therefore, we developed a method, the Echelon Analysis from below, to reduce the set of combinations to those which yield a non-singular Jacobian and hence a solvable system. Since the Echelon Analysis is independent of the initialization system, we formulate it for arbitrary underdetermined systems.

We distinguish between three types of singularities. The first are those which occur only in a specific region of the domain. *E.g.* if the temperature T has to be calculated from the enthalpy H and a given pressure p. If the point (p, H) lies within the two-phase region,

$$\frac{\partial T(p,H)}{\partial H} = 0 \tag{3.2}$$

holds. Thus the Jacobian becomes singular only in this region. This kind of singularity is denoted as *local singularity*. To deal with such problems, further knowledge of the investigated system is required and is not handled here.

The second type of singularities remains valid over the whole domain, referred to as *global singularity*. Regard the equations

$$\begin{aligned} -m_1 + m_2 + m_3 &= 0, \\ m_2 + m_3 &= 0. \end{aligned} \tag{3.3}$$

This system is underdetermined and has one degree of freedom. Hence, one variable has to be prescribed. Prescribing the value of a variable is referred to as *setting* the variable. Setting m_2 or m_3 leads to a regular Jacobian. If m_1 is set, the Jacobian of this system becomes singular for any $(m_2, m_3) \in \mathbb{R}^2$ since the rows are linearly dependent.

The third type are *structural singularities* which have been defined in the last section. They are completely independent of numerical values of the coefficients. Redundant settings can be made and the system is overdetermined in some subsystem leading to a structural singularity, *e.g.* for a system of the form

$$\begin{pmatrix} \times & & & \\ \times & \times & & \\ & & \times & \times \\ & \times & & \end{pmatrix}. \tag{3.4}$$

In section 3.1, we assume that the system is underdetermined and free of structural and global singularities. A method is described which determines a subset of the variables from which one can be set such that the degree of freedom is reduced by one and the resulting system is not structurally or globally singular. In section 3.2 and 3.3, systems containing a structural or global singularity, respectively, are investigated and methods for resolving the singularities are presented. We derive all methods for general systems.

3.1. Echelon Analysis

The first case considered occurs when the system is underdetermined and further settings are required. With equation (3.3) we demonstrated that the settings can not be chosen arbitrarily. We present a general approach and algorithms, referred to as *Echelon Analysis*, to determine in an underdetermined system the set $\mathcal{S}$ of variables from which *one* can be set such that the degree of freedom is reduced by one and the Jacobian does not become globally or structurally singular. Settings which lead to a local singularity can not be identified since this is process-dependent. If a variable from $\mathcal{S}$ is set, the set $\mathcal{S}$ is not necessarily valid anymore and the Echelon Analysis has to be repeated for the system endowed with one more setting.

We assume that $A : \mathbb{R}^n \to \mathbb{R}^m$ with $n > m$ is given and has maximal structural rank m in the sense that the maximal transversal algorithm finds m diagonal elements. Hence the system is underdetermined, has $d = n - m$ degrees of freedom and d variables have to be set to obtain a square system. Then applying the Echelon Analysis and setting one variable at a time yields an iterative procedure leading to a system which is not globally or structurally singular in d steps, see algorithm 3.9 on page 33.

The Echelon Analysis, *i.e.* finding the set $\mathcal{S}$ of variables from which one can be set, is split into two steps. First, we identify settings which do not lead to a structurally singular system. This yields the set of *structurally settable* variables $\mathcal{S}_s$. Second, we analyse which variables in $\mathcal{S}_s$ can not be set since the remaining system would become globally singular, *e.g.* m_1 in (3.3). The variables in the final set $\mathcal{S} \subseteq \mathcal{S}_s$ are called (numerically) *settable* or *possible settings.*

3.1.1. Structurally settable Variables

First we extend A by d artificial zero rows to obtain a square system $\tilde{A}$. Then the maximal transversal algorithm described in section 2.2.1 is applied to $\tilde{A}$ and we obtain $P_1\tilde{A}$. Since we assumed that A has maximal structural rank exactly d diagonal entries are zero, *i.e* void, and the corresponding rows are the artificial zero rows added before. If the zero rows are replaced by rows with only one element on the main diagonal, the system becomes structurally regular. On a symbolic level this act is equivalent to setting these variables. Consequently, already d structurally settable variables are found.

If row k is a zero row, variable k is structurally settable. If row i has an entry in column k, these two rows could be interchanged to fill the diagonal entry in row k. This preserves the length of the transversal but row i is now a zero row. Hence, variable i is structurally settable as well. This considerations yield the following algorithm which operates on $P_1\tilde{A}$.

Algorithm 3.1 *Let $\mathcal{S}_s$ be the set of structurally settable variables, $\mathcal{N}$ the set of not explored variables and $\mathcal{B}$ the set of visited variables.*

```
1. Set $\mathcal{S}_s = \emptyset$, $\mathcal{N} = \emptyset$ and $\mathcal{B} = \emptyset$.
2. Find all zero diagonal entries and add the corresponding variables to $\mathcal{S}_s$ and $\mathcal{N}$.
3. do
4.     Select $v \in \mathcal{N}$ and add $v$ to $\mathcal{B}$.
5.     Find all rows $i$ in which $v$ occurs.
6.     If $i \notin \mathcal{B}$, add the variables $i$ to $\mathcal{S}_s$ and $\mathcal{N}$.
7.     Remove $v$ from $\mathcal{N}$.
8. while $\mathcal{N} \neq \emptyset$.
```

Theorem 3.2 *The set $\mathcal{S}_s$ gained with algorithm 3.1 is maximal in the sense that $\mathcal{S}_s$ contains all structurally settable variables.*

Proof. The maximal transversal algorithm is based on row permutations P_1. The length of the transversal is unique but the ordering of the rows not necessarily. To preserve the length of the transversal, two nonzero rows i and j can only be interchanged if row i has an element in column j and vice versa. Assume that row i is a zero row and j nonzero, which implies that in row j a diagonal element exists. Then row i and j can be interchanged if row j has an element in column i. Thus, algorithm 3.1 records all row indices in which the zero rows could occur such that the transversal remains maximal. These indices correspond to the structurally settable variables. □

THEOREM 3.3 *Let $\mathcal{T}$ be the set of all variables which do not appear in $\mathcal{S}_s$. Structurally, all variables in $\mathcal{T}$ can be calculated.*

Proof. Assume that the system A has d degrees of freedom and $\mathcal{S}_s$ contains k variables. The proof of theorem 3.2 yields that algorithm 3.1 detects k rows from which $l = k - d$ are non zero rows. In the remaining $n - k$ rows no variables from $\mathcal{S}_s$ occur since otherwise these rows would have been recorded by the algorithm. Hence, by removing these l rows and k columns from $P_1 A$ a $(n-k) \times (n-k)$ system remains, all rows have a diagonal entry and thus the system has structural rank $(n-k)$. □

An implementation of the block triangular algorithm described in [22] and [24] is capable of identifying the variables in $\mathcal{T}$. The block triangular algorithm can be applied to $P_1 A$ even if the system is underdetermined. In the implementation a node i is labeled as invalid if the transversal is empty in the i-th entry. In matrix terms this means that in column i the diagonal element is zero. Furthermore, nodes which depend on an invalid node are treated equally, *i.e.* rows in which an invalid column occurs are removed from the system. Hence, for a structurally singular, underdetermined system this implementation yields the block decomposition of the structurally regular subsystem and numerical methods can be applied to calculate the valid variables.

Similar considerations for the determination of $\mathcal{S}_s$ can be found in [17]. There the problem is investigated from a graph point of view. The maximal transversal algorithm is applied and then the paths from variables which do not occur on the main diagonal are followed. This yields the same result as algorithm 3.1. In [17] only a structural analysis is presented to determine the set of variables from which one can be set such that the degree of freedom is reduced by one. An equivalent example to (3.3) is used to demonstrate the limitations of a pure structural approach, *i.e.* variables can occur in $\mathcal{S}_s$ yielding a structurally regular but numerically (globally) singular system if they are set. In the following, we present the second step of the Echelon Analysis which overcomes the limitations of a pure structural approach.

3.1.2. Numerically settable Variables

Let the set $\mathcal{S}_s$ of structurally settable variables be given. We recall the system from (3.3), *i.e.*

$$\begin{aligned} -m_1 + m_2 + m_3 &= 0, \\ m_2 + m_3 &= 0. \end{aligned} \tag{3.5}$$

In this example the variable m_1 is structurally settable but still a global singularity occurs if m_1 is set, independent of the prescribed value. Therefore we are looking for a subset $\mathcal{S} \subseteq \mathcal{S}_s$ whose elements do not lead to a global singularity.

First, we start with the case where all equations in the system A are linear. Assume that $A \in \mathbb{R}^{m \times n}$ with $n > m$ is linear and has rank m. From the proof of theorem 3.2 we obtain that algorithm 3.1 finds the rows and columns of a structurally singular $k \times k$ subsystem of $P_1 \tilde{A}$ where k is the number of structurally settable variables. Only this smaller system is of further interest since the other variables can already be calculated from the remaining equations. We define the subsystem $A_s \in \mathbb{R}^{l \times k}$ of $P_1 \tilde{A}$. Here, $l = k - d$ is the number of equations when neglecting the zero rows. A_s is gained by deleting the zero rows and all rows and columns of $P_1 \tilde{A}$ which are not recorded by algorithm 3.1. Applying the Gauß-Jordan elimination yields the unique reduced row echelon form A_r equivalent to A_s. A proof for the uniqueness of the reduced row echelon form can be found in [15]. If in A_r there is a row j in the system having only one entry in column i, then the corresponding variable v_i can not be set. If we set

v_i, the variable can be substituted by its prescribed value. Thus, the i-th column is removed from the system and row j becomes a zero row. This leads to a singular system for any prescribed values and hence a globally singular system.

Theorem 3.4 *Assume that the reduced row echelon form A_r of $A_s \in \mathbb{R}^{l\times k}$ with $l < k$ is given. All variables in $\mathcal{S}_s$ which occur as a single entry in a row of A_r lead to a global singularity when they are set. We collect them in the set $\mathcal{G}$. Then the set of possible settings is $\mathcal{S} = \mathcal{S}_s \backslash \mathcal{G}$.*

Proof. The statement follows directly from the considerations above. □

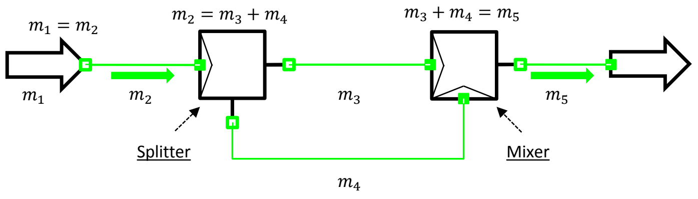


Figure II.1.: Flow sheet of a Splitter-Mixer system.

Example 3.5 (Splitter-Mixer) Assume that the steady-state mass flow has to be calculated in a splitter-mixer-system. The flowsheet of the system is illustrated in figure II.1. The mass balances reduce to algebraic equations. The system can be written as

$$\begin{aligned} m_1 &= m_2, \\ m_3 + m_4 &= m_2, \\ m_3 + m_4 &= m_5 \end{aligned} \tag{3.6}$$

where m_3 and m_4 are mass flows in the split region, m_1, m_2 before the splitter and m_5 after the mixer. Hence the system has two degrees of freedom. The Jacobian is given by

$$J = \begin{pmatrix} 1 & -1 & & & \\ & -1 & 1 & 1 & \\ & & 1 & 1 & -1 \end{pmatrix}. \tag{3.7}$$

All variables are structurally settable and the Gauß-Jordan Algorithm transforms the system into the reduced row echelon form

$$\tilde{J} = \begin{pmatrix} 1 & & & & -1 \\ & 1 & & & -1 \\ & & 1 & 1 & -1 \end{pmatrix}. \tag{3.8}$$

Since no rows with only one entry occur, all variables in $\mathcal{S}_s$ are possible settings, *i.e.* $\mathcal{S} = \mathcal{S}_s$. Setting m_5 leads to

$$J = \begin{pmatrix} 1 & -1 & & \\ & -1 & 1 & 1 \\ & & 1 & 1 \end{pmatrix} \quad \text{and} \quad \tilde{J} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & 1 \end{pmatrix}. \tag{3.9}$$

Again all variables are structurally settable, *i.e.* $\mathcal{S}_s = \{m_1, m_2, m_3, m_4\}$ but now the reduced row echelon form $\tilde{J}$ reveals that m_1 and m_2 can not be set and we obtain $\mathcal{S} = \{m_3, m_4\} \subsetneq \mathcal{S}_s$. ♢

Next we extend this results to nonlinear systems. When solving the initialization system the regularity of the Jacobian is of importance. The main difference to the linear case is that the entries in the

Jacobian can depend on the point in which they are evaluated. Since we are interested in settings which do not cause a global singularity the problem can we handled by applying a symbolic Gauß-Jordan algorithm. We only have to take care of identities which the following example shows. Identities occur frequently in flow sheet models.

Example 3.6 Let the following system of equations with three degrees of freedom be given as

$$\begin{aligned} -x_1 + y_1 x_2 + y_2 x_3 &= 0, \\ y_3 x_2 + y_4 x_3 &= 0, \\ y_1 - y_3 &= 0, \\ y_2 - y_4 &= 0. \end{aligned} \tag{3.10}$$

Assume that we set $y_1 = a$ and $y_2 = b$. Then one degree of freedom is left and with the ordering $(x_1, x_2, x_3, y_3, y_4)$ we obtain

$$J = \begin{pmatrix} -1 & a & b & & \\ & y_3 & y_4 & & \\ & & & -1 & \\ & & & & -1 \end{pmatrix} \tag{3.11}$$

and the symbolic Gauß-Jordan algorithm yields

$$\tilde{J} = \begin{pmatrix} 1 & & \frac{a y_4}{y_3} - b & & \\ & 1 & \frac{-y_4}{y_3} & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}. \tag{3.12}$$

The considerations so far would yield that x_1, x_2 and x_3 can be set. Unfortunately, if x_1 is set, a global singularity emerges. Using the identities, row 3 and 4 in equation (3.10), we obtain $\frac{a y_4}{y_3} - b = \frac{ab}{a} - b = 0$. Thus, the first row in the above system has only one entry which yields that x_1 can not be set. If the identities are erased from (3.10) in advance, we obtain

$$\begin{aligned} -x_1 + y_1 x_2 + y_2 x_3 &= 0, \\ y_1 x_2 + y_2 x_3 &= 0. \end{aligned} \tag{3.13}$$

Choosing the same settings, the transformed Jacobian reads as

$$\tilde{J} = \begin{pmatrix} 1 & & \\ & 1 & \frac{-b}{a} \end{pmatrix}. \tag{3.14}$$

Thus, removing the identities from the system yields the correct results. ♢

The above example demonstrates that a symbolic Gauß-Jordan elimination alone does not necessarily achieve the desired result on its own but some information from the equations has to be incorporated in advance. Furthermore, the symbolic algorithm can be written in the way that equations do not have to be given as explicit as above. Let

$$f(z_1, z_2, z_3, z_4) = z_3 z_1 + z_4 z_2. \tag{3.15}$$

Then (3.13) can be written as

$$\begin{aligned} -x_1 + f(x_2, x_3, y_1, y_2) &= 0, \\ f(x_2, x_3, y_1, y_2) &= 0. \end{aligned} \tag{3.16}$$

If y_1 and y_2 are set, the Jacobian is

$$J = \begin{pmatrix} 1 & \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \\ & \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{pmatrix} \tag{3.17}$$

and the reduced row echelon form has the same structure as in (3.14). Hence for implicitly given equations even an explicit derivative is in some cases not required.

The disadvantage of the symbolic Gauß-Jordan elimination is the difficulty of implementation and, as stated in [29], the potentially high costs of dealing with symbolic entries. Normalization of quotients and cancelations have to be performed and entries have be checked if they are zero. Besides the computational cost, implementing a symbolic Gauß-Jordan elimination is a nontrivial task. Therefore, we present some ideas to extend the usual Gauß-Jordan algorithm for linear systems to Jacobians arising from systems with linear and nonlinear parts.

First, only the linear equations can be regarded. Recall theorem 3.3, there is a subset $\mathcal{T}$ of variables which can be calculated although the system is structurally singular. We exploit this fact, calculate these variables and substitute the results into the singular part of the system. Obviously, the less degrees of freedom the more variables are found in $\mathcal{T}$. Hence symbolic, *i.e.* state dependent, coefficients in the remaining Jacobian can become constant for every evaluation point. In (3.11) this is made visible by replacing y_1, y_2 by a, b, respectively. Consequently, the first row no longer depends on the evaluation point. This implies that by calculation and substitution of variables in $\mathcal{T}$ the amount of linear equations in the singular subsystem can be increased and thus the result of the nonsymbolic Gauß-Jordan elimination can be improved.

Second, partially also symbolic entries can be handled by the elimination. Assume that a row with only one entry is given. Then this row can be used to erase the respective column entry in all other rows independent of their values. Regard

$$J = \begin{pmatrix} \times & \times & & \\ & a & & \\ \times & & b & c \end{pmatrix} \tag{3.18}$$

with given $a, b, c \in \mathbb{R}$. The $\times$-entries denote a non constant entry in the Jacobian. Whatever value is obtained in $J_{1,2}$ we can multiply the second row with the same value and subtract it from the first one providing a zero entry in $J_{1,2}$. Thus another row with a single entry is found and the procedure can be repeated. Although not all entries in the Jacobian are constant this procedure yields

$$\tilde{J} = \begin{pmatrix} \times & & & \\ & a & & \\ & & b & c \end{pmatrix} \tag{3.19}$$

which implies that again the first two variables can not be set.

3.1.3. Further Considerations

The results gained from the algorithms above are valid until a new variable is set. Then the procedure has to be repeated to ensure correct results. Let the system have more than one degree of freedom. If the singular subsystem can be split into two independent, *i.e.* decoupled, parts, one setting can be selected from each part simultaneously and no recalculation of $\mathcal{S}$ is required. To obtain the decoupled parts, methods from graph theory may be applied, *e.g.* an algorithm for finding connected components may be used. Furthermore, for each decoupled part the degree of freedom can be determined. Algorithm 3.1 first searches for zero diagonal entries in $P_1\tilde{A}$. We gather the corresponding variables in the set $\mathcal{I}$, the starting point for the set $\mathcal{S}_s$ and determine in which connected component the variables from $\mathcal{I}$ occur. For each appearance the degree of freedom of the component is increased. This procedure may lead to a better understanding of the singular system.

Up to now general systems have been investigated. We neglected the distinction between differential and algebraic variables in the initialization system. As a final point of this section, special considerations for the initialization system are made. We assume that the underlying DAE of (2.1) can also be underdetermined which implies that algebraic variables have to be set and differential variables have to be initialized. In section 2.1 it was mentioned that we demand that each differential variable or its derivative must be initialized but not both. This makes space for further restraints of possible settings which reduces the set $\mathcal{S}$. For an underdetermined system the degree of freedom and the number of missing initializations is known. If all differential variables are initialized properly, only algebraic settings are missing. Thus all differential variables can be removed from $\mathcal{S}$. Vice versa, if the degree of freedom is equal to the missing initializations no algebraic setting will lead to a correctly initialized system and these variables have to be removed from $\mathcal{S}$. Furthermore, a differential variable x must be initialized if its derivative x' can be calculated, *i.e.* $x' \in \mathcal{T}$, and vice versa. If a variable x has to be initialized, all variables y have to be removed from $\mathcal{S}$ which are connected to x via a single variable function

$$x = f(y). \tag{3.20}$$

Example 3.7 In [11], *p.* 138, the system

$$\begin{aligned} -y_1' + y_2' + y_1 &= g_1(t), \\ y_2 &= g_2(t) \end{aligned} \tag{3.21}$$

is presented as a counter example in which the developed simple algorithm therein for finding consistent initial values fails. The second equation yields $y_2(t_0)$ and the hidden constraint $y_2'(t_0) = g_2'(t_0)$. Thus, only y_1 or y_1' can to be initialized arbitrarily to obtain consistent initial values but $y_2'(t_0)$ can not. The algorithm in [11] and also the Echelon Analysis do not detect this hidden constraint. The Echelon Analysis only guides to a regular initialization system which does not imply consistent initial values.

Still, if we start without any initializations and run the Echelon Analysis, we obtain $y_2 \in \mathcal{T}$ and $\mathcal{S} = \{y_1,\ y_1',\ y_2'\}$. By the restriction that each differential variable or its derivative have to be initialized, y_2' and one of $\{y_1,\ y_1'\}$ have to be initialized leading to a regular system for any given values. Still, only $y_2' = g_2'(t_0)$ yields a solution $(x, y)(t) \in \mathcal{C}^1(I, \mathbb{R}^2)$ of the IVP, see definition 1.2, and hence consistency. Although there is no solution of the IVP for $y_2'(t_0) \neq g_2'(t_0)$, if $y_2'(t_0) \in U_\varepsilon(g_2'(t_0))$, DASSL is able to solve the system and $(x, y)(t)$ equals a solution $(\hat{x}, \hat{y})(t)$ for $t > t_0$. The solution $(\hat{x}, \hat{y})(t)$ corresponds to the consistent initial value $y_2'(t_0) = g_2'(t_0)$. Therefore, whenever derivatives of a differential variables have to be initialized there is only one consistent value which can be determined from hidden constraints. ◊

Example 3.8 (Mass-Spring-Damper) Regard

$$\begin{aligned} \Delta x &= x - x_0, \\ F &= d\Delta x' + c\Delta x, \\ F &= m(g - v'), \\ v &= x'. \end{aligned} \tag{3.22}$$

Here x_0 represents the position of the suspension, x the position of the mass, Δx the deflection, v the velocity, m the weight of the mass, c the stiffness of the spring, d the damping constant, F the force and g the gravitational constant. This system describes the motion of a damped spring and we assume that m, c, g and d are given. If the suspension x_0 is set constant, x and Δx are only shifted by the constant x_0 and we obtain the hidden constraint $\Delta x' = x'$. Four equations and four variables $\{F, x, \Delta x, v\}$ remain. Three variables are differential variables. No more algebraic settings are required but all differential variables have to be initialized but it is not possible to prescribe the state for each differential variable. The very first initialization can be chosen randomly. Assume that x was chosen. Thus, Δx can be calculated and $\Delta x'$ has to be initialized. This implies that algebraic variable F is prohibited and so is v'. Therefore v must be initialized which prohibits x'. The latter one would also be removed from $\mathcal{S}$ since the corresponding x is already initialized. Consequently the first initialization determines what choice has to be made for all others. Assume that the initializations are $x,\ v,\ \Delta x'$. The value for x can be chosen arbitrarily. Since x_0 constant, we obtain $\Delta x' = v$. These two variables have to be initialized with the same value to obtain consistent initial values. Hence, if the Echelon Analysis returns that a derivative has to be initialized, there is only one value which is consistent and yields a solution $y(t) \in \mathcal{C}^1(I, \mathbb{R}^n)$ of the IVP. $\diamondsuit$

We summarize the procedure presented in section 3.1 in the following theorem.

Algorithm and Theorem 3.9 *Let $A : \mathbb{R}^n \to \mathbb{R}^m$ with $n > m$ have structural rank m, be globally non-singular and the degrees of freedom be $d = n - m$. Let $\mathcal{S}$ be gained by the Echelon Analysis. Then the procedure*

```
1. for $i$ = 1, ..., $d$
2.     Find the set $\mathcal{S}$ for the system $A$.
3.     Set one variable from $\mathcal{S}$ and update $A$.
4. end
```

yields in d steps a structurally and globally non-singular system $\tilde{A} : \mathbb{R}^m \to \mathbb{R}^m$ consisting of the system A endowed with d additional settings. In particular, for $i > 1$ the set $\mathcal{S}_i$ found in the i-iteration is a subset of the set $\mathcal{S}_{i-1}$ from the previous iteration.

Proof. This is an immediate consequence from theorem 3.2 and 3.4. $\square$

3.2. Adjoint Echelon Analysis

In the following we consider overdetermined systems. At first we do not distinguish between differential and algebraic variables. Moreover, the overdetermination does not have to occur for the whole system but can also be caused by a subsystem as presented in (3.4) on page 26. This system is structurally singular and bearing an over- and underdetermined part. If the system contains an overdetermined part, we call this a *conflict*. An overdetermined subsystem $A : \mathbb{R}^n \to \mathbb{R}^m$ has more equations than variables, *i.e.* $m > n$ holds. We assume that the equations are set up in a proper way such that they are free of conflicts. Then conflicts can only occur by too many settings. Symbolic rows with only one variable are equivalent to settings since there are just two ways to generate such a row. These are

$$x = a \quad \text{and} \quad x = g(t) \tag{3.23}$$

with $a \in \mathbb{R}$ or $g : \mathbb{R} \to \mathbb{R}$ given. The first one corresponds to a setting the second one is a forcing term which can be treat equally for a fixed time. From now on a setting is interpreted as an equation added to the system. The goal is to find a set $\mathcal{R}$ of equations from which *one* can be removed from the system such that one conflict is solved. The set $\mathcal{R}$ can contain other equations than those corresponding to a setting but only the latter ones are of importance.

We subdivide the procedure of finding $\mathcal{R}$, referred to as *Adjoint Echelon Analysis*, into two parts. First the structural part is performed in which a set $\mathcal{R}_s$ is found consisting of equations which can be removed such that the structural rank of the system remains equal. An equation $e \in \mathcal{R}_s$ is referred to as *structurally removable*. The second part determines $\mathcal{R} \subseteq \mathcal{R}_s$ in which no equation causes a global singularity when it is removed. We call these equations (numerically) *removable*. For the sake of presentation, we assume that a system $A \in \mathbb{R}^{m\times n}$ with $m > n$ is given, the equations are set up properly and no underdetermined part exists. Hence the system contains $d = m - n$ conflicts.

3.2.1. Structurally removable Equations

We extend the system by d artificial zero-columns and perform the maximal transversal algorithm first. The system is not structurally regular and d rows occur having zero main diagonal entries. In particular, after the algorithm the first n rows have non-zero main diagonal entries and the rows from $n + 1$ to $n + d = m$ do not. Contrary to the underdetermined case no zero rows occur. Here, each row which does not have a diagonal entry has entries in other columns. The matrix P_1A has a transversal with length n but the arrangement of the rows is again not unique. Obviously, by removing the last d rows having no diagonal entry, a structurally regular system emerges. Assume that row i is one of these rows and has entries in columns j and k. Then row i can be interchanged with either row j of k and the length of the transversal is preserved. Thus, removing one of the other two rows preserves the structural rank. These considerations yield the following algorithm.

Algorithm 3.10 *Let $\mathcal{R}_s$ be the set of structurally removable equations, $\mathcal{N}$ the set of not explored equations and $\mathcal{B}$ the set of visited equations.*

```
1. Set $\mathcal{R} = \emptyset$, $\mathcal{N} = \emptyset$ and $\mathcal{B} = \emptyset$.
2. Find all zero diagonal entries and add the corresponding rows to $\mathcal{R}_s$ and $\mathcal{N}$.
3. do
4.     Select row $e \in \mathcal{N}$ and add it to $\mathcal{B}$.
5.     Add all column indices $j$ occurring in $e$ to $\mathcal{R}_s$ and $\mathcal{N}$ if $j \notin \mathcal{B}$.
6.     Remove $e$ from $\mathcal{N}$.
7. while $\mathcal{N} \neq \emptyset$.
```

Theorem 3.11 *The set $\mathcal{R}_s$ gained with algorithm 3.10 is maximal in the sense that $\mathcal{R}_s$ contains all structurally removable equations.*

Proof. Assume that P_1A is structurally singular. Let r_i be a structurally removable row which was not detected by Algorithm 3.10. Then r_i has an entry on the diagonal due to point 2 of Algorithm 3.10. A row r_i is structurally removable if the length of the transversal is preserved when it is removed. Hence, there must be another row r_j having an entry in column i such that r_i can replace r_j. If not, r_i is not structurally removable. The length of the transversal is preserved if row r_j has no diagonal entry in P_1A. If r_j has a diagonal entry, a row r_k has to exist which can replace r_j which has no diagonal entry. In general, a sequence of permutations has to be performed but only a row in P_1A without a diagonal entry can fill the zero diagonal entry. Due to point 2 of Algorithm 3.10 this row was detected and thus all above mentioned rows including r_i. This is a contradiction to the assumption. □

Although the system is overdetermined it is still possible to find variables which already can be calculated correctly. This means that the results for these variables remain equal after solving the conflicts. The block triangular algorithm from [24] does not yield the same results. As mentioned in section 3.1.1 the implementation labels a node as invalid if a diagonal element is zero for an equation in a block. The overdetermination yields n rows with transversal entries for n variables. Hence blocks can be found as if the d rows without diagonal element would have been removed from the system and they are labeled invalid. Therefore the result of the algorithm is not reliable in the overdetermined case. Nevertheless, by collecting all variables which occur in the equations of $\mathcal{R}_s$ in the set $\mathcal{I}$ and staring algorithm 3.1 with $\mathcal{S}_s = \mathcal{I}$ instead of the empty set yields all variables which can not be calculated reliably. Consequently, also the variables are known which can be calculated including the required equations. We gather these variables in the set $\mathcal{T}$.

3.2.2. Numerically removable Equations

The next step is to identify equations which would lead to a globally singular Jacobian if they are removed from the system. As mentioned above the set $\mathcal{R}_s$ also contains equations which are not settings. We are only interested in settings which can be removed but the following identification is done for the whole set. Let the system have d conflicts, $\mathcal{R}_s$ consist of k equations with $l = k - d$ variables. This yields a system $A_s \in \mathbb{R}^{k \times l}$. The adjoint (transposed) system $A_s^\top \in \mathbb{R}^{l \times k}$ with $l < k$ is underdetermined. As in section 3.1.2, we ask which columns can be removed from the system such that no global singularity emerges. Hence this problem can be solved again by transforming the system to the reduced echelon form with the Gauß-Jordan elimination and subsequent search for rows with a single value.

This can be interpreted as follows. The rows and columns of $A_s^\top$ correspond to the variables and the equations, respectively. A value a_{ij} in column j of row i means that variable v_i has coefficient a_{ij} in equation e_j. If this is a single entry in row i then only e_j is left in which v_i can be calculated. Therefore e_j must not be removed from the system. We present these considerations with the aid of the following example.

Example 3.12 (Splitter-Mixer) We regard the Splitter-Mixer presented in example 3.5. Recall

$$\begin{aligned} m_1 - m_2 &= 0, \\ m_3 + m_4 - m_2 &= 0, \\ m_3 + m_4 - m_5 &= 0 \end{aligned} \tag{3.24}$$

and assume that m_1, m_3 and m_5 are set which makes it overdetermined. The Jacobian and its adjoint are

$$J = \begin{pmatrix} 1 & -1 & & & \\ & -1 & 1 & 1 & \\ & & 1 & 1 & -1 \\ 1 & & & & \\ & & 1 & & \\ & & & & 1 \end{pmatrix} \quad \text{and} \quad J^\top = \begin{pmatrix} 1 & & & 1 & & \\ -1 & -1 & & & & \\ & 1 & 1 & & 1 & \\ & 1 & 1 & & & \\ & & -1 & & & 1 \end{pmatrix}. \tag{3.25}$$

Applying, the Gauß-Jordan elimination to $J^\top$ yields

$$\tilde{J}^\top = \begin{pmatrix} 1 & & & & & 1 \\ & 1 & & & & -1 \\ & & 1 & & & -1 \\ & & & 1 & & -1 \\ & & & & 1 & \end{pmatrix}. \tag{3.26}$$

Thus equation 5 can not be removed from the system which corresponds to the setting of m_3. ◇

3.2.3. Further Considerations

The set $\mathcal{R}$ contains all equations which can be removed. This also includes equations not corresponding to a setting which of course is not applicable unless used during the model development phase. We only suggest settings which can be removed. By nature flow sheet models are underdetermined after being set up. When the advice of the Echelon Analysis is followed no conflicts emerge. If the Echelon Analysis is not available, then in large models one might loose the overview of which variables already can be calculated. Setting one of the variables in $\mathcal{T}$ yields a conflict. A benefit of the Adjoint Echelon Analysis is that it can be applied to underdetermined system as well. Furthermore, for determined systems the Adjoint Echelon Analysis turned out to be a useful tool when a change in the setting structure is requested. In cases where good values for a variable in $\mathcal{T}$ exist, we can set it and obtain information which settings are in conflict with the desired one. Hence by removing another setting the structure was changed.

Finally we discuss aspects when distinguishing between algebraic and differential variables. Obviously, if a differential variable and its derivative are initialized at the same time, one of them has to be released since this situation is not allowed. This has highest priority. Furthermore, if the differential variable is initialized and the corresponding equation occurs in $\mathcal{R}$ but the derivative lies in $\mathcal{T}$, then this variable has to remain initialized and hence can be removed from $\mathcal{R}$, and vice versa. This may be done iteratively since after fixing an initialization the set $\mathcal{T}$ can become larger and thus further variables which have to remain initialized may be found.

We summarize the procedure presented in section 3.2 in the following theorem. The proof follows directly from theorem 3.11 and the considerations from section 3.2.2.

Algorithm and Theorem 3.13 *Let $A : \mathbb{R}^n \to \mathbb{R}^m$ with $m > n$ have structural rank n. Let the equations be set up properly and the system have $d = n - m$ conflicts emerging from settings. Further let $\mathcal{R}$ be gained by the Adjoint Echelon Analysis. Then the procedure*

```
1. for $i = 1, ..., d$
2.     Find the set $\mathcal{R}$ for the system $A$.
3.     Remove one setting from $\mathcal{S}$ and update the system $A$.
4. end
```

yields in d steps a structurally and globally non-singular system $\tilde{A} : \mathbb{R}^n \to \mathbb{R}^n$. In particular, for $i > 1$ the set $\mathcal{R}_i$ found in the i-iteration is a subset of the set $\mathcal{R}_{i-1}$ from the previous iteration.

3.3. A priori Model Analysis

So far we assumed that the equations are set up in a proper way. We drop this assumption and present a method to identify sets of dependent equations. Let $A \in \mathbb{R}^{m \times n}$ with $m \leq n$ and let the rows be linearly dependent. Hence, A does not have maximal row rank, *i.e.* $\operatorname{rank}(A) = l < m$ holds. The goal is to find maximal, decoupled subsystems $\mathcal{R}_i = \{r_1, ..., r_k\}$ of rows in A such that there exists an $\alpha \in \mathbb{R}^k \setminus \{0\}$ satisfying

$$\sum_{j=1}^{k} \alpha_j r_j = 0. \tag{3.27}$$

This implies that the rows in $\mathcal{R}_i$ are linearly dependent and no setting structure can lead to a regular system. In this case, the equations have to be revised. If, on the one hand, all equations in $\mathcal{R}_i$ occur within the same component then the corresponding component model contains an error. If, on the other hand, they are distributed over the flow sheet model then the arrangement of the components has to be investigated. The maximality of the subsystem $\mathcal{R}_i$ is of importance if

$$\dim \operatorname{span}(\mathcal{R}_i) < k - 1. \tag{3.28}$$

In this case also a subset $\mathcal{R}_s \subset \mathcal{R}_i$ can be found which satisfies (3.27) with $\alpha \neq 0$ as well.

Applying the Gauß-Jordan elimination to A yields $d = m - l$ zero rows. Hence, we have d degrees of freedom. The zero rows imply that there exist at most d dependent subsystems and the rows transformed into zero rows are part of these subsystems. Which rows of $\mathcal{R}_i$ are transformed into zero rows depends on the ordering of the columns.

To find a maximal, dependent subsystem, the adjoint system $A^\top$ is investigated. We perform the Gauß-Jordan algorithm which yields an equivalent system $\tilde{A}^\top$. Since

$$\operatorname{rank}(A) = \operatorname{rank}(A^\top) = \operatorname{rank}(\tilde{A}^\top) = l < m \tag{3.29}$$

holds we obtain $n - l = n - m + d$ zero rows in $\tilde{A}^\top$. Therefore in l rows m different column entries are found. In other words m equations exist to calculate l variables. This corresponds to a (partial) overdetermination. In sparse systems usually this overdetermination does not involve all l nonzero rows. With the following algorithm it is possible to determine the minimal set of rows containing more column than row indices. This information is provided by the sets $\mathcal{V}_i$ and $\mathcal{E}_i$ corresponding to the row and column indices, respectively.

Algorithm 3.14 *The algorithm operates on $\tilde{A}^\top$.*

```
1.  Add all rows with more than one column entry to $\mathcal{L}$.
2.  do
3.      Select $r \in \mathcal{L}$ and add it to a new $\mathcal{V}_i$
4.      Add all occurring column indices $c$ in $r$ to a new $\mathcal{E}_i$ and $\mathcal{N}$.
5.      do
6.          Select $c \in \mathcal{N}$.
7.          Add all rows from $\mathcal{L}$ with an entry in $c$ to $\mathcal{V}_i$.
8.          Add all new column indices within these rows to $\mathcal{E}_i$ and $\mathcal{N}$.
9.          Remove $c$ from $\mathcal{N}$.
10.     while $\mathcal{N} \neq \emptyset$.
11.     Remove all rows in $\mathcal{V}_i$ from $\mathcal{L}$.
12. while $\mathcal{L} \neq \emptyset$.
```

Lemma 3.15 *Let $(\mathcal{V}_i, \mathcal{E}_i)$ be a pair gained from algorithm 3.14. Then $|\mathcal{V}_i| < |\mathcal{E}_i|$ holds.*

Proof. The algorithm only notes rows with multiple entries. Each subsystem represented by the pair $(\mathcal{V}_i, \mathcal{E}_i)$ is fully decoupled from the remaining system since there are no connections via column entries. $|\mathcal{V}_i| > |\mathcal{E}_i|$ can not hold since the Gauß-Jordan elimination is performed in row orientation. Assume that $|\mathcal{V}_i| = |\mathcal{E}_i|$. On the one hand, if this subsystem is singular then zero rows would have been produced by the Gauß-Jordan elimination and hence $|\mathcal{V}_i|$ is reduced. On the other hand, if the subsystem is regular, it is equivalent to the identity. Then the elimination would have provided this and these entries would not have been recorded by the algorithm. $\square$

Theorem 3.16 *Let $A \in \mathbb{R}^{m \times n}$ have rank $l < m$ and let $\mathcal{E}_i$ for $i = 1, ..., \tau$ be provided by algorithm 3.14. Further let $\mathcal{R}_i$ be the set of rows corresponding to $\mathcal{E}_i$, i.e. $\mathcal{R}_i = \{r_j \in A : j \in \mathcal{E}_i\}$. The sets $\mathcal{R}_i$ are linearly dependent and maximal in the sense that there is no other row r in A such that $r \in \operatorname{span}(\mathcal{R}_i)$. In particular, these sets are decoupled and can be treated independently.*

Proof. The Gauß-Jordan elimination can be written as a sequence of elementary row transformations M_ι, *i.e.*

$$\tilde{A}^\top = \left(\prod_{\iota=1}^{\kappa} M_\iota\right) A^\top = MA^\top.$$

We assumed that $\operatorname{rank}(A) = l < m$ which implies that $A^\top$ has a nontrivial kernel. Thus $\beta \in \mathbb{R}^m \setminus \{0\}$ exists such that

$$0 = M(A^\top\beta) = (MA^\top)\beta.$$

Algorithm 3.14 yields pairs $(\mathcal{V}_i, \mathcal{E}_i)$ for $i = 1, ..., \tau$ and hence a decomposition of $MA^\top$ into independent parts. With suitable row and column permutations P_r and P_c based on the results of the algorithm, $MA^\top$ can be written as

$$P_r M\tilde{A}^\top P_c = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & B_1 & & \\ & & & & \ddots & \\ & & & & & B_\tau \\ & & & & & 0 \end{pmatrix}.$$

Here B_i denotes the i-th block corresponding to $(\mathcal{V}_i, \mathcal{E}_i)$. Obviously, the entries of a $\beta \in \ker(A^\top)$ which correspond to columns not occurring in a block are zero. Furthermore the kernel of $A^\top$ can be written as the direct sum, *i.e.*

$$\ker(A^\top) = \bigoplus_{i=1}^{\tau} U_i$$

and the elements of each subspace $U_i \subset \mathbb{R}^m$ have only nonzero entries in those corresponding to $\mathcal{E}_i$. Lemma 3.15 yields $|\mathcal{V}_i| < |\mathcal{E}_i|$ for each block B_i which implies that $d_i = |\mathcal{E}_i| - |\mathcal{V}_i|$ vectors can be found spanning U_i. Hence any linear combination of these vectors yields a $\beta \in \ker(A^\top)$. We choose a $\beta \in U_i$ with maximal nonzero entries and finally obtain

$$0 = (MA^\top)\beta = M(A^\top\beta) = (\beta^\top A)M^\top.$$

Since $M^\top$ is regular $\beta^\top A = 0$ holds. This corresponds to a linear combination of the rows of A and yields that the rows corresponding to $\mathcal{E}_i$ are linearly dependent. The maximality follows directly from the decomposition achieved with algorithm 3.14. □

Example 3.17 Let the following system and reduced echelon form be given as

$$A = \begin{pmatrix} & 1 & & & \\ 1 & 1 & & & \\ & 1 & 1 & 1 & \\ 1 & & 1 & & 1 \\ 1 & & & & \end{pmatrix} \quad \text{and} \quad \tilde{A} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & -1 \\ & & & 1 & 1 \end{pmatrix} \tag{3.30}$$

Investigation of the adjoint system yields

$$A^\top = \begin{pmatrix} & 1 & & 1 & 1 \\ 1 & 1 & 1 & & \\ & & 1 & 1 & \\ & & 1 & & \\ & & & 1 & \end{pmatrix} \quad \text{and} \quad \tilde{A}^\top = \begin{pmatrix} 1 & & & & -1 \\ & 1 & & & 1 \\ & & 1 & & \\ & & & 1 & \\ & & & & \end{pmatrix}. \tag{3.31}$$

Algorithm 3.14 provides $\mathcal{E}_1 = \{1, 2, 5\}$ which are the dependent rows. ♢

Example 3.18 Assume that we have a stationary closed cycle model and each component contains the mass conservation law, *i.e*

$$0 = m_{in} - m_{out}. \tag{3.32}$$

Given four components we obtain the following system matrix

$$A = \begin{pmatrix} 1 & -1 & & \\ & 1 & -1 & \\ & & 1 & -1 \\ -1 & & & 1 \end{pmatrix}. \tag{3.33}$$

Applying the Gauß-Jordan elimination yields

$$\tilde{A} = \begin{pmatrix} 1 & & & -1 \\ & 1 & & -1 \\ & & 1 & -1 \\ & & & \end{pmatrix}. \tag{3.34}$$

The zero row reveals that one equation is dependent. From the adjoint system we obtain

$$A^\top = \begin{pmatrix} 1 & & & -1 \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{pmatrix} \quad \text{and} \quad \tilde{A}^\top = \begin{pmatrix} 1 & & & -1 \\ & 1 & & -1 \\ & & 1 & -1 \\ & & & \end{pmatrix}. \tag{3.35}$$

Algorithm 3.14 yields $\mathcal{V}_1 = \{1, 2, 3\}$ and $\mathcal{E}_1 = \{1, 2, 3, 4\}$ which implies that all four equations belong to the same dependent subset. $\diamondsuit$

Algorithm 3.14 operates on $\tilde{A}^\top$ which implies that only one Gauß-Jordan elimination for the adjoint system has to be performed. For testing of the implementation one might consider the comparison with $\tilde{A}$ to evaluate if the zero rows occur only in the sets $\mathcal{E}_i$. The above algorithm is formulated for arbitrary linear systems. For the regularization of the initialization system, the algorithm is applied to the Jacobian J of the system. If in J the coefficients are non-constant and a symbolic Gauß-Jordan elimination is not available, $P_r M \tilde{J}^\top P_c$ does not have exactly the form presented in the proof of theorem 3.16. Rather blocks occur which fulfill $|\mathcal{V}_i| = |\mathcal{E}_i|$ since a full elimination is not possible. Furthermore, the developed algorithm can be used if a local singularity emerges. In this case the Jacobian becomes singular in a point. Hence all coefficients are given and we can determine the equations which are responsible for the singularity. In either case, the algorithm does not provide information of how to solve the problem, *i.e.* singular subsystem, since this is usually process dependent and has to be treated individually. However, the algorithm indicates the equations causing the problem which is supposed to accelerate the task of resolving the problem.

4. Improvements in DASSL

Once consistent initial values are available, the integration is performed by numerical solver DASSL. Over the years three variants have been developed, namely DASSL, DASPK and DASKR. The differences mainly concern the methods for solving the linear system arising at each integration step during the Newton iteration. In DASSL a dense direct linear solver is implemented. In DASPK the preconditioned Krylov iterative GMRES (Generalized Minimal Residual) is used and in DASKR the user can choose between the algorithms. Furthermore, in the latter variant also a mechanism to calculate consistent initial values is provided. To do so, either x_0 or x_0' has to be known and the converse variable is determined. Despite the variants which have been released over the years, we use the original version DASSL and implemented own improvements which do not effect the BDF method or step size control. First of all, as described in section 2, an initialization system is set up and solved to achieve consistent initial values. This is done outside DASSL and could therefore also be used as a stand-alone program. As in DASPK and DASKR, the method for solving the linear system was changed which we explain below.

4.1. Block Decomposition

Let the fully implicit system of differential-algebraic equations be given by

$$F(t,x,x') = 0. \tag{4.1}$$

In order to minimize the memory allocation all systems are assumed to be sparse and represented as such. In each time step the (nonlinear) system from (1.45) on page 19 has to be solved and the Newton method is applied. As for the initialization system the block decomposition presented in section 2.2 can be applied to reduce the computational cost. Of course the decomposition does not have to be performed in each time step but can be calculated once before the integration is started. In contrast to the initialization system, this system is has less variables since there is no division of the differential variables into two parts. Thus, we expect the block size to be larger. Then in each time step, each block is solved sequentially using Newton's method.

In flow sheet models variables occur which are constant over time, *e.g.* the mixture of a working fluid or the geometry of a tank. Such variables can be calculated in the beginning and do not change over time. Calculating such variables at each time step is therefore not necessary. Furthermore, there are variables which are only calculated explicitly from other state variables, *e.g.* the entropy. One might be interested in its value but usually it is not used to calculate a state point. Hence we distinguish between three types of variables. *Constant*, if they only have to be calculated by the initialization system and remain constant over time, *dynamic*, if they have to be determined in each integration step and *post calculation* variables which are computed only when an output is required. Hence less variables are passed to DASSL which saves computational time and we except a gain in stability. The determination of the type of each variable is done blockwise as described in the following algorithm.

Algorithm 4.1 *Determination of the calculation type of each variable. Let m be the number of blocks found, $\mathcal{A}$ and $\mathcal{I}$ be the sets of active and inactive variables per block, respectively. Active variable are those occurring in B_{ii}, inactive those in any B_{ij} with $j < i$. Hence, inactive variables are already calculated when block B_{ii} has to be solved.*

```
1.  for $i = 1, ..., m$
2.      if a differential variable occurs in $\mathcal{A}_i$ then
3.          all $v \in \mathcal{A}_i$ are dynamic.
4.          if a post calculation variable occurs in $\mathcal{I}_i$ then
5.              all $v$ in the corresponding block are dynamic (iterative update)
6.      if no differential variable occurs in $\mathcal{A}_i$ but a dynamic or post calculation variable in $\mathcal{I}_i$ then
7.          all $v \in \mathcal{A}_i$ are post calculation.
8.      if no differential or post calculation variable occurs in $\mathcal{A}_i$ and $\mathcal{I}_i$ then
9.          all $v \in \mathcal{A}_i$ are constant.
10. end for
```

The only delicate part in the algorithm above is point 4. If a post calculation variable occurs in the inactive part of a dynamic block then this variable v_i and all others in the block B_{ii} in which v_i is active have to become dynamic as well. Furthermore, a reinvestigation of the block B_{ii} is necessary to make sure that no post calculation variables are in the inactive set $\mathcal{I}_i$. This is an iterative procedure.

The evaluation of the variable types not only has computational benefits - since the system solved by DASSL becomes smaller - but also the user may gain a better understanding of the model and the effects of the settings. It can happen that variables become post calculation which one would have expected to be dynamic and vice versa. Furthermore, if Newton's method does not converge, the block decomposition can ease the search for the origin of the problem. Usually, a single block can be identified which does not converge. Hence less equations have to be investigated in order to find the problem.

4.2. Integration Termination Criteria

When DASSL is started we need to provide the starting and ending point of the integration. For many applications this is sufficient but there are problems in which the integration has to be stopped during the simulation interval but the time is a priori not known, *e.g.* if the equations of the model have to be altered. Such needs can emerge if a valve closes and the model is split into two independent parts or a flow reversal takes place and a splitter becomes a mixer and vice versa. In this case, only the condition but not the time for termination of the integration is known.

This condition may be written as a zero-crossing function $c(t, x(t)) : \mathbb{R} \times \mathbb{R}^l \to \mathbb{R}$ and a root finding algorithm is used to locate the time at which the integration has to be stopped. We shall use the short notation $c(t)$ to emphasize the time dependence. The time derivative is gained by applying the chain rule, *i.e.*

$$c'(t) = c_t(t) + c_x(t)x'(t), \tag{4.2}$$

where $c_x(t) \in \mathbb{R}^{1\times l}$ and $x'(t) \in \mathbb{R}^{l\times 1}$. In DASSL a new step size h is predicted and a step from t_k to t_{k+1} is performed which corresponds to solving (1.45). If Newton's method converges and the step is accepted, a reevaluation of $c(t)$ is required. If the sign of $c(t)$ changes, a condition became active and the time at which it occurs has to be found. In this case we say that the corresponding condition becomes active. In the explanations below we follow [13].

An inexact Newton method is used if several conditions became active within the last integration step. The focus lies on finding the time at which the first condition becomes active. Let $c_i(t)$ for $i = 1, ..., d$ be the zero crossing function representing the i-th condition which became active. Newton's method for any $c_i(t)$ can be written as

$$t_i^* = t_{k+1} - \frac{c_i(t_{k+1})}{c_i'(t_{k+1})}. \tag{4.3}$$

Replacing the derivative by the backward difference yields

$$t_i^* = t_{k+1} - \frac{c_i(t_{k+1})(t_{k+1} - t_k)}{c_i(t_{k+1}) - c_i(t_k)} = \frac{c_i(t_{k+1})t_k - c_i(t_k)t_{k+1}}{c_i(t_{k+1}) - c_i(t_k)} \tag{4.4}$$

and we choose the minimum, *i.e.* $t^* = \min_i t_i^*$. This yields the new step size $h = t^* - t_k$ and we let DASSL repeat the step. If still more than one $c_i(t)$ crosses zero, we set $t_{k+1} = t^*$. Otherwise, if no condition becomes active, we set $t_k = t^*$. This procedure is performed until only one $c_i(t)$ remains which crosses zero in the interval $[t_k, t_{k+1}]$. From that point on, we can apply any root finding algorithm to this single function $c_i(t)$.

We present one algorithm which converges with cubic speed and convergence is guaranteed. Assume that $c(t)$ is the only function which crossed zero in the interval $[t_k, t_{k+1}]$. Given the values $c(t_k)$, $c(t_{k+1})$, $c'(t_k)$ and $c'(t_{k+1})$ we obtain enough information to fit a third order polynomial p. As a short notation we write $c_k = c(t_k)$ and $c'_k = c'(t_k)$. Since $\operatorname{sgn}(c(t_k)) \neq \operatorname{sgn}(c(t_{k+1}))$ holds and p is continuous at least one root can be found. In particular p has at most three roots and at least one is real. To circumvent the search for the roots of p a polynomial interpolation of the inverse of $c(t)$ is applicable. Let $t(c)$ be the inverse of $c(t)$. Then four values of this function are given by $t_k = t(c_k)$, $t_{k+1} = t(c_{k+1})$, $t'_k = 1/c'_k$ and $t'_{k+1} = 1/c'_{k+1}$. The inverse cubic polynomial has the form

$$t(p) = \tilde{a}p^3 + \tilde{b}p^2 + \tilde{c}p + \tilde{d} \tag{4.5}$$

and the coefficients are gained by solving

$$\begin{pmatrix} c_k^3 & c_k^2 & c_k & 1 \\ c_{k+1}^3 & c_{k+1}^2 & c_{k+1} & 1 \\ 3c_k^2 & 2c_k & 1 & 0 \\ 3c_{k+1}^2 & 2c_{k+1} & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{c} \\ \tilde{d} \end{pmatrix} = \begin{pmatrix} t_k \\ t_{k+1} \\ 1/c'_k \\ 1/c'_{k+1} \end{pmatrix}. \tag{4.6}$$

Finally we obtain the new time t^* by evaluating (4.5) at 0, *i.e.*

$$t^* = t(p = 0) = \tilde{d}, \tag{4.7}$$

the last coefficient. Given the new evaluation time the DASSL step is repeated and the interval is updated via $t_k = t^*$ if the condition is not active at t^* or $t_{k+1} = t^*$ otherwise. This procedure is applied until $t_{k+1} - t_k < \delta$ or $|c_{k+1}| < \varepsilon$ for given $\delta, \varepsilon > 0$.

Next we answer the question what can be done if a condition becomes active and inactive again within the same time interval $[t_k, t_{k+1}]$. This implies that DASSL would miss the activation of the condition since the step size was chosen too large. We assume $c(t) \in \mathcal{C}^1(\mathbb{R}, \mathbb{R})$, the space of continuously differentiable functions. In this case the corresponding function $c(t)$ crosses zero twice and thus has an extremum. Hence, $c'(t)$ changes sign within $[t_k, t_{k+1}]$. By adding the derivative to the set of zero crossing functions the problem can be solved. If, during the search for a root of $c'(t)$, it is revealed that $c(t)$ became active, we pursue only $c(t)$ anymore. If $c(t)$ has an extremum but does not become active, the integration is resumed by DASSL without any changes.

Finally, we investigate the case when a termination criterion $c(t)$ is used describing the boundary of the domain of a function in the system. *E.g.* let a function be given by

$$f(x, y) = \sqrt{x} - y \tag{4.8}$$

and the condition by

$$c(t) = x. \tag{4.9}$$

As long as $x \geq 0$ holds the integration is performed but has to be stopped if x becomes negative since the square root is not defined. If x becomes negative, Newton's method does not converge since an error in (4.8) arises which stops the iteration. Thus $c(t_{k+1})$ can not be evaluated either and we do not know that actually the condition is becoming active. In this case DASSL will reduce the step size h until no errors occur which implies $c(t) > 0$. The same behaviour will repeat until $x(t_k)$ is very close to 0 and the step size becomes less than the minimum and DASSL aborts with the corresponding error. What can be done is to check $|c_{k+1}| < \varepsilon$ for given $\varepsilon > 0$ which avoids missing that a condition becomes active. This implies that the step size control is left with DASSL. Otherwise, we can use the fact that $c'(t_k)$ is known and predict $c(t_{k+1})$, *i.e.*

$$c^p(t_{k+1}) = c(t_k) + c'(t_k)h. \tag{4.10}$$

If $\operatorname{sgn}(c(t_k)) \neq \operatorname{sgn}(c^p(t_{k+1}))$, we may assume that the error in the Newton iteration occurs due to exceeding the termination criterion $c(t)$. We choose

$$h = -\frac{c(t_k)}{c'(t_k)}, \tag{4.11}$$

set $t^* = t_k + h$ and repeat the DASSL step. If the Newton iteration converges we set $t_k = t^*$. Otherwise we obtain an upper bound for t_{k+1} and have to choose the next t^* via *e.g.* bisection or leave the choice with DASSL.

III. Modelling

Within this chapter we present the component models of the refrigeration appliance which were developed in MDK and the flow sheet model set up in PSE. During the ECO-COOL research project a new, independent model library, named *Eco-Cool-Lib*, was implemented. But first, the investigated domestic refrigeration appliance is presented and we give an introduction into IPSEpro and its model architecture.

1. The Domestic Refrigeration Appliance

We begin this chapter with the introduction of the investigated domestic refrigeration appliance. In table III.1 the key data is given. The appliance is an upright freezer and its class is SN-T (subnormal-tropical) which means that it is designed for ambient temperatures of 10 °C to 43 °C. The energy consumption is approximately 0.645 kWh per day and the energy label is *A*. The refrigerant is Isobutane (R600a), also known as Methylpropane.

Table III.1.: Key data of the refrigeration appliance.

Gross/net capacity	162/158 L
Energy consumption	235 kWh/a
Energy efficiency index	43.9
Climate class	SN-T (subnormal-tropical)
Freezing capacity	19 kg
Door heating	Available
Type	Upright freezer
Compressor cooling capacity (ASHRAE −23.3 °C / 55 °C)	167.0 W

In figure III.1 the setup of an appliance is illustrated and in the following the design of the investigated freezer is described. First, there is a *casing* (1) and a *door* (2). If the door is opened we see the *interior*, also called the *compartment* (3). This is the place to store food and drinks. Between the casing and the compartment, the *insulation* is situated and therein the evaporator. Different from figure III.1, where a plate evaporator is depicted, the *evaporator* (8) is wound vertically around the compartment in the investigated freezer. In the evaporator the temperature of the refrigerant is less than the desired compartment temperature such that heat is absorbed and the interior is cooled. In some appliances an *accumulator* is built in at the end of the evaporator. The accumulator collects the liquid share of the refrigerant. In the optimal case only superheated fluid leaves the accumulator which increases the cooling capacity and prevents that liquid refrigerant enters the compressor.

The counterpart of the evaporator is the *condenser* (5) which is installed free-hanging at the back of the appliance. The condenser gives off heat to the ambiance. In the investigated freezer, we have a black fin condenser whose tubes are made of steel. The fins increase the surface area and thus more heat is transmitted. In freezers, often the end part of the condenser tube is led around the frame of the door to prevent that it freezes on. The disadvantage of this design, named door heating, is that the condenser tube also heats the compartment. At the end of the condenser, the *filter/dryer* (6) is

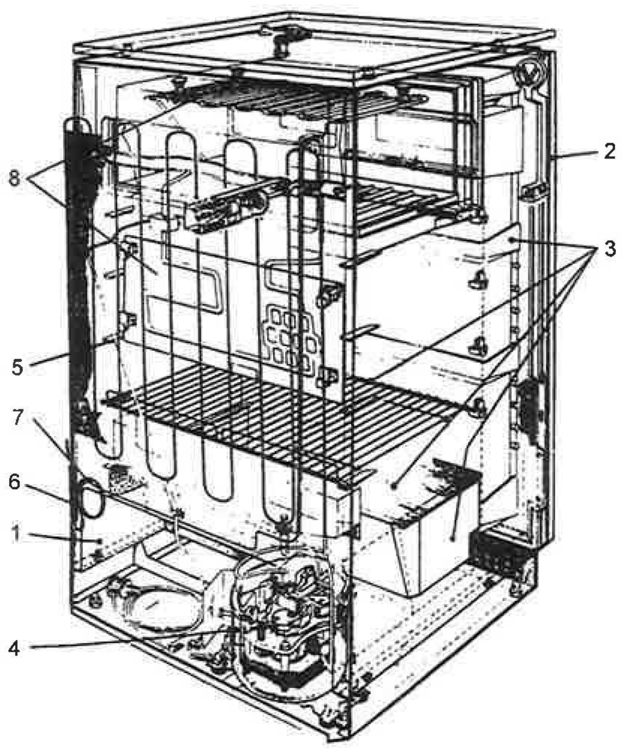


Figure III.1.: Setup of a domestic refrigeration appliance, [81]. 1 - Casing, 2 - Door, 3 - Interior/Compartment, 4 - Compressor, 5 - Condenser, 6 - Dryer, 7 - Capillary, 8 - Evaporator.

installed. Its purpose is to filter the possibly impure refrigerant and to remove water. Although water is not supposed to enter the refrigeration cycle during the operation, it can enter the cycle in the manufacture. If water flows into the capillary, it freezes, blocks the mass flow and can cause damage in the refrigeration cycle.

The *capillary* (7) itself, a very thin, long tube, is situated between the condenser and evaporator and expands the refrigerant. This means that the pressure of the refrigerant is decreased and thus likewise its temperature. The pressure drop depends on the length and the diameter of the tube. Additionally the capillary is connected to the *suction line*, the tube between the evaporator and the compressor, which enables heat exchange. This internal heat exchanger increases the cooling capacity of the appliance and shall prevent that non-superheated refrigerant enters the compressor.

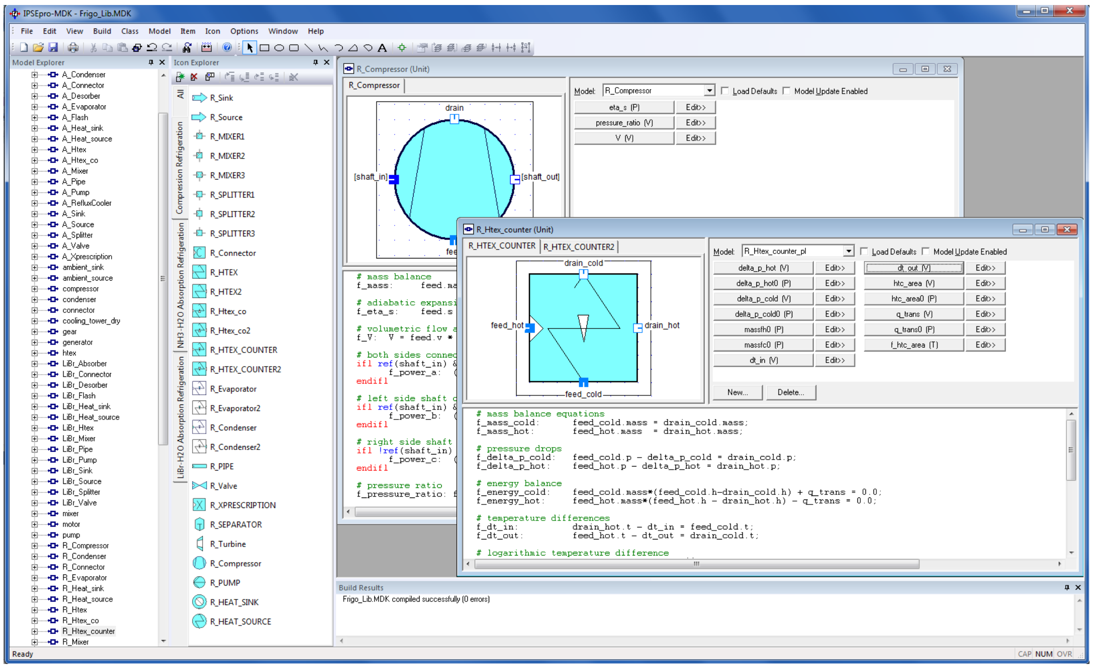
Finally, the *compressor* (4), including the engine, is typically located at the back of the appliance beneath the condenser. The compressor sucks vaporized refrigerant from the evaporator into its shell and from there into the cylinder where the refrigerant is compressed such that the temperature is above ambient temperature. If the engine is switched off, the oil used to lubricate and cool the compressor is collected at the bottom of the shell which is referred to as *oil sump*. On the one hand the oil can absorb and desorb refrigerant and on the other hand the oil may enter the refrigeration cycle if it is sucked in by the compressor. The sorption process depends among other parameters on the interface between oil and refrigerant. If the engine is switched on, the oil is distributed in the shell and hence the interface to the refrigerant is increased. If oil gets into the refrigeration cycle, the heat transfer is affected. Since the models of the effect of the oil on the heat transfer are uncertain, see [81], this influence is neglected and only the sorption process is modelled in this thesis.

2. Modules of IPSEpro

IPSEpro consists of several modules from which the two most important are the Model Development Kit (MDK) and Process Simulation Environment (PSE). Screenshots of these two modules are given in figure III.2 and III.3. First, the MDK consists of the model editor and the model compiler and is used to develop component model libraries. In the model editor the component models are described. An Icon is designed for each component model, the occurring variables are defined and the mathematical equations are formulated. The equations have to be written in the syntax of the Model Description Language (MDL) provided by IPSEpro. In the MDL each equation receives a label separated by a double dot and the end is indicated by a semicolon. An illustration is given below. It has to be emphasized that the lines written with the MDL are not assignments as in programm languages like C++ but represent actual equations. Also the sequence in which the equations are listed does not influence the solution. When the component models are fully described, the model compiler has to be invoked which translates the models into a binary format. This increases the performance when a model is solved in PSE.

```
#   Model Description Language Sample
f1: feed.mass = drain.mass;
f2: feed.mass*(feed.h - drain.h) = k*A*(t-t_out);
f3: feed.p - delta_p = drain.p;
f4: drain.t = t;

t1: test (mass >= 0.0) warning "mass flow negative";
```

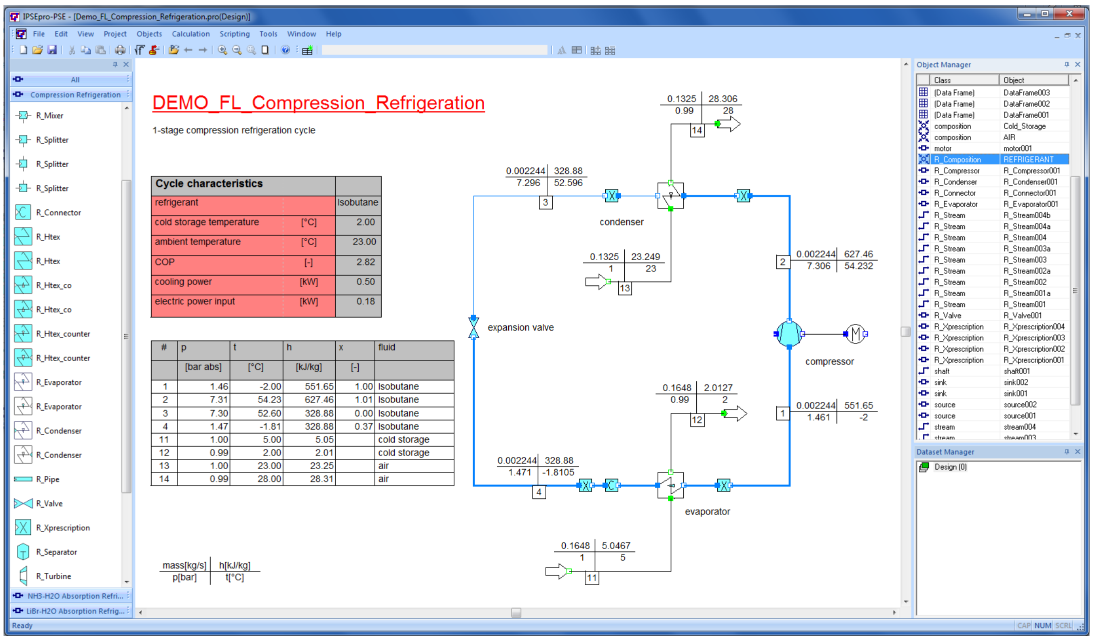
Second, in PSE the component models from a model library can be arranged as a flow sheet model. Per drag and drop the icons of the component models can be taken from a list, arranged and connected in the flow sheet editor. Then the boundary conditions and parameters have to be specified which is done directly in the flow sheet editor. After the simulation, results are displayed beside the component models.



```
# mass balance equations
f_mass_cold:      feed_cold.mass = drain_cold.mass;
f_mass_hot:       feed_hot.mass  = drain_hot.mass;

# pressure drops
f_delta_p_cold:   feed_cold.p - delta_p_cold = drain_cold.p;
f_delta_p_hot:    feed_hot.p - delta_p_hot = drain_hot.p;

# energy balance
f_energy_cold:    feed_cold.mass*(feed_cold.h-drain_cold.h) + q_trans = 0.0;
f_energy_hot:     feed_hot.mass*(feed_hot.h - drain_hot.h) - q_trans = 0.0;

# temperature differences
f_dt_in:          drain_hot.t - dt_in = feed_cold.t;
f_dt_out:         feed_hot.t - dt_out = drain_cold.t;

# logarithmic temperature difference
```

```
Frigo_Lib.MDK compiled successfully (0 errors)
```

Figure III.2.: Screenshot of the module MDK.


IPSEpro-PSE - [Demo_FL_Compression_Refrigeration.pro(Design)]
File Edit View Project Objects Calculation Scripting Tools Window Help
Compression Refrigeration
Object Manager
Dataset Manager
condenser
expansion valve
compressor
evaporator
Ready

DEMO_FL_Compression_Refrigeration

1-stage compression refrigeration cycle

Cycle characteristics		
refrigerant		Isobutane
cold storage temperature	[°C]	2.00
ambient temperature	[°C]	23.00
COP	[-]	2.82
cooling power	[kW]	0.50
electric power input	[kW]	0.18

#	p	t	h	x	fluid
	[bar abs]	[°C]	[kJ/kg]	[-]	
1	1.46	-2.00	551.65	1.00	Isobutane
2	7.31	54.23	627.46	1.01	Isobutane
3	7.30	52.60	328.88	0.00	Isobutane
4	1.47	-1.81	328.88	0.37	Isobutane
11	1.00	5.00	5.05		cold storage
12	0.99	2.00	2.01		cold storage
13	1.00	23.00	23.25		air
14	0.99	28.00	28.31		air

mass[kg/s]	h[kJ/kg]
p[bar]	t[°C]

Figure III.3.: Screenshot of the module PSE.

Third, having set up a valid flow sheet model, a mathematical solver kernel is called. Therein, the equations from MDK are gathered in a system of equations which is solved by appropriate numerical methods. The solution process is split into two stages. First, in the analysis phase the system is checked for errors and decomposed as described in section II.2.2. Second, in the numerical solution phase the system is solved. Next we introduce the model structure in IPSEpro. This concerns first and foremost the model development in MDK.

3. Model Structure in IPSEpro

In IPSEpro, there are three types of components: *Units*, *Connections* and *Globals*. First, the Units represent equipment such as heat exchangers, pipes or compressors. Second, the Connections indicate the link between Units and allow transfer between the connected Units. The transferred quantities can be *e.g.* mass, heat, or information. Third, the Globals enable to share information between Units or Connections on a higher level accessible to multiple Units and Connections. The structure which is common to all component types is presented first and then specific features are discussed. For a deeper insight we refer to [95].

A component consists of items and equations. The *items* are the basic elements which appear in the equations and comprise variables, parameters, switches, tables and external functions. These elements then can be arranged in equations which have to be written in the syntax of the MDL. Together, they form the mathematical model of the component. If the flow sheet model in PSE is set up and the numerical solver is called, the mathematical model is incorporated into the system of equations. The Units are the only components able to have more than one mathematical model. The number of items and equations can differ within each mathematical model. Thus, in PSE the flow sheet model can be conserved for several models. However, changing the mathematical model of a Unit may require an alternation of the settings, respectively boundary conditions.

The only items discussed in detail in this thesis are the *external functions*. Through a C-interface, functions of the form $f : \mathbb{R}^n \to \mathbb{R}$ can be called from a DLL. For the refrigeration appliance model, several calculations had to be outsourced which are too complex to be implemented in MDK itself. A detailed description of the developed *HTX.dll*, written in C++, and external functions used for the refrigeration appliance model is given in section 7. The advantages of this interface are, first, that models which are too complex to be implemented with the MDL can still be handled by IPSEpro, second, that variables are avoided which do not have to be given explicitly and, third, that the numerical stability is potentially increased since less variables are iterated.

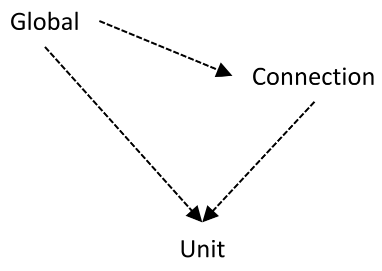
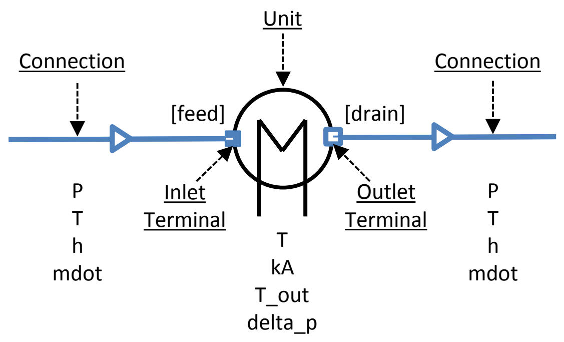


Figure III.4.: Component hierarchy in IPSEpro.

The interaction of components takes place through *referencing*. Referencing means that a component uses items of another component within its equations. There are three rules which have to be considered. First, Units can reference from Connections and Globals and, second, Connections can reference only from Globals, but not vice versa. Third, referencing is only possible from components which are attached. This hierarchy is illustrated in figure III.4. In MDK, a Global can be added to the list of items in Units and Connections which is sufficient to enable referencing from the Global. A Connection can be attached to a Unit in PSE if the Unit is equipped with *terminals*. There are two types of terminals, inlet and outlet, and plugging a Connection to two terminals of the same kind is prohibited. In the Unit, each terminal obtains a name, depicted between square brackets in figure III.5, and the type of the pluggable Connections is specified. Through the terminal the Unit is able to reference items of a Connection. Referencing an item in an equation is done just like accessing a member in object oriented programming. The name of the reference, Global or terminal, is followed by a dot and the desired item name. The syntax of the MDL reads as

$$ReferenceName.ItemName. \tag{3.1}$$



```
# Equations of Unit
f1: feed.mdot = drain.mdot;
f2: feed.mdot*(feed.h - drain.h)
              = kA*(T-T_out);
f3: feed.p - delta_p = drain.p;
f4: drain.T = T;
```

```
# Equations of Connection
f1: T = Composition.t_ph(p, h);
```

Figure III.5.: Simple model in IPSEpro.

In the remainder of this thesis, whenever equations of the components are given this syntax of the MDL is substituted by the expression

$$ItemName_{ReferenceName}. \tag{3.2}$$

Applying this to the MDL sample of figure III.5, the first equation reads as

$$\dot{m}_{feed} = \dot{m}_{drain}. \tag{3.3}$$

Example 3.1 The above general explanations shall be concretized with the steady-state model illustrated in figure III.5. In this model, there exists one Global, named *Composition*, which is referenced by the Connection and has one external function *t_ph*. The external function takes two arguments, pressure and enthalpy, and returns the temperature. Which working fluid is used is neglected in this simple example but the Global seems to be an appropriate location to specify it. The Connection itself has four variables which are the pressure p, temperature T, enthalpy h and the mass flow $\dot{m}$. The only equation of the Connection references the external function of the Global and expresses a thermodynamic equation of state.

The Unit represents an element through which a working fluid flows and heat is exchanged with an ambiance. The variables are the temperature of the working fluid T, the ambient temperature T_{out}, the product of the thermal transmittance and the surface area kA and the pressure drop Δp. The names of inlet and outlet terminals are *feed* and *drain*, respectively. Thus the describing equations read as

$$\begin{aligned} \dot{m}_{feed} &= \dot{m}_{drain}, & (3.4)\\ \dot{m}_{feed}(h_{feed} - h_{drain}) &= kA(T - T_{out}), & (3.5)\\ p_{feed} - \Delta p &= p_{drain}. & (3.6) \end{aligned}$$

The last equation

$$T_{drain} = T \tag{3.7}$$

emphasizes that the temperature in the interior of the element is assumed to equal the outlet temperature. This equation as well as the variable T can be omitted if T is substituted by T_{drain} in equation (3.5). If the model is set up as is in figure III.5, the system consists of 6 equations and 12 variables which leaves 6 degrees of freedom. Hence 6 variables have to be set to obtain a square system. Obviously, these settings can not be chosen arbitrarily. $\diamondsuit$

Within this chapter the units of all variables are assumed to be SI-units. In IPSEpro, different units were applied due to readability and numerical concerns. Correction factors can be seen in some equations in MDK to compensate the deviation from the SI-units or to scale the equations. Considering the numerical tolerances, see equation (1.50) in section II.1.2, a similar magnitude of each equation is of advantage. In the following the component models developed in MDK are described.

4. Globals

The first type of components which are presented are the Globals. These components are used if information has to be shared on a higher level or if the same information is used by several components. In the refrigeration cycle model, the working fluid does not change which makes its definition in one place ideal. Therefore, the Global *Composition* was introduced. The Composition has one switch to specify the name of the working fluid. Depending on the chosen name, the variable *FluidID* is determined. Several external functions are defined for calculating physical properties. For a full list of available functions we refer to the Eco-Cool-Lib library. The external functions are connected to a DLL which calls the programm Refprop, see [60]. To obtain the values for the correct fluid in Refprop, the variable *FluidID* is passed.

The Global *Wall* holds geometrical and physical properties, such as specific heat capacity c, density ρ and thermal conductivity λ, of the *Wall*-Units which are used for the insulation and compartment walls. Physical properties of the heat exchanger pipe walls are kept in the Global *Metal*. Finally, the Global *Geometry* manages the geometrical properties of the heat exchangers, such as inner and outer radius, volumes and various surface areas. Since a Global can have only one mathematical model and properties like area or volume depend on the geometry of the element, the *Geometry_*-Unit was introduced to calculate the variables in the *Geometry*-Global for various geometries.

5. Connections

The Connections are used to enable transfer between Units. In the flow sheet model they are represented by lines of different colour. The simplest transfer is the transport of information as implemented in the *Control*- and *Temperature*-Connection. First, the Control-Connection (dark blue) was designed to transfer one arbitrary value x to control the NOR of the compressor. There, the Temperature_Sensor-Unit passes the measured temperature to the I_Control-Unit which controls the Number of Revolutions (NOR) of the compressor. In general, the variable x in the Control-Connection changes its meaning depending on the Unit it is attached to.

Second, the Temperature-Connection (orange) has two variables T_feed and T_drain and is only used by the heat exchanger Units, Condenser and Evaporator. If the Temperature-Connection is attached, heat conduction in the pipe wall is enabled. To calculate the heat transfer, the Connection needs the wall temperature of the contiguous component. Hence each Unit has to pass its temperature to the Connection. The suffixes of the two variables indicate the terminal name of the Unit to which the Connection is attached to.

5.1. Stream

The *Stream*-Connection (green) transports a working fluid which implies mass transfer. In the Stream, six variables are defined: pressure p, temperature T, enthalpy h, density ρ, entropy s and mass flow $\dot{m}$. The Stream references external functions from the Composition-Global to formulate three equations of state, *i.e.*

$$T = T_{Composition}(p,h), \tag{5.1}$$
$$\rho = \rho_{Composition}(p,h), \tag{5.2}$$
$$s = s_{Composition}(p,h). \tag{5.3}$$

In many flow sheet models the flow direction does not change. In these cases, the Stream is always interpreted as the outlet of the upstream Unit. Typically, in the Units the change of state is described by referencing items of the incoming and outgoing Stream and on the Stream itself the equations of state are formulated, see example 3.1. Due to readability, the reference of the Composition is neglected in the remainder. Thus, the equation of state connecting temperature, pressure and enthalpy is written as

$$T = T(p,h) \quad \big(= T_{Composition}(p,h)\big).$$

5.2. Stream_H

The heat exchangers Units, condenser and evaporator, are one-dimensional spatially discretized by the finite volume method and a first order upwind scheme is applied to determine the states on the outlet of the Units. This yields that the state of the volume, *i.e.* the Unit, equals the state on the outlet. It is assumed that the pressure p inside the Unit always equals the one at the outlet-terminal side but the enthalpy h on the surface, *i.e.* the Connection, has to depend on the flow direction. To guarantee that the correct surface enthalpy in the energy equations are used, the *Stream_H*-Connection (green) was introduced. It generalizes the Stream-Connection by a flow dependent enthalpy h. Since Connections can not reference to Units, two further variables h_feed and h_drain are available. As in the Temperature-Connection, the suffixes indicate the terminal name of the Unit to which the Connection is attached to and the Unit passes its enthalpy to the respective variable of the Stream_H. Since the Units represent discrete volumes, a jump discontinuity in the enthalpy h is expected at the flow turning point. To avoid problems in Newton's method, linear interpolation is applied in an ε-neighbourhood of the zero flow. Hence, the enthalpy on the Connection is defined as

$$h = \begin{cases} h\_drain, & \text{if } \dot{m} > \varepsilon, \\ h\_feed, & \text{if } \dot{m} < -\varepsilon, \\ h\_feed + (h\_drain - h\_feed)\frac{\dot{m}+\varepsilon}{2\varepsilon}, & \text{else.} \end{cases} \tag{5.4}$$

The equations of state equal the ones from the above Stream, *i.e.*

$$T = T(p,h), \tag{5.5}$$
$$\rho = \rho(p,h), \tag{5.6}$$
$$s = s(p,h). \tag{5.7}$$

5.3. HeatFlux

The last form of transport between Units modelled in the Eco-Cool-Lib library occurs through heat exchange. To ensure conservation in the energy equations of two contiguous Units, the heat flux $\dot{Q}$ will be calculated in a *HeatFlux*-Connection (dark red). Hence, $\dot{Q}$ is determined only once to make sure that the same value appears in the energy equations. The thermal transmittance k is determined by a combination of heat transfer coefficients α, thermal conductivities λ and distances d. For the *HeatFlux*-Connection, this yields

$$k = \frac{1}{\frac{d_1}{p_1} + \frac{d_2}{p_2}} \tag{5.8}$$

where $d_i = 1$ and $p_i = \alpha$, or $d_i = d$ and $p_i = \lambda$ for $i = 1, 2$ holds. These values have to be passed from the Units to which the Connection is attached. Then, the heat flux is given by

$$\dot{Q} = kA(T\_feed - T\_drain) \tag{5.9}$$

where the two temperatures are determined by the Units as for the Temperature- and Stream_H-Connection. Whenever $\dot{Q}$ is used in the energy balance of a Unit the correct sign has to be considered.

6. Units

The Units mostly describe pieces of equipment. For each Unit an icon has to be designed in MDK which is used as a representative for the mathematical models in PSE. A selection of Unit icons is given in figure III.6. The Units are the only components able to have more than one mathematical model. Most of the Units in the Eco-Cool-Lib library have a steady-state and a dynamic model. Since these models just differ whether time derivatives appear or not, only the dynamic models are presented.

As described in section 3, a Unit, which is supposed to be connected to others, has to be equipped with terminals. A name and the type of Connection which can be plugged have to be specified. Generally, inlet and outlet terminals are denoted as *feed* and *drain*, respectively. A naming convention is chosen to distinguish between the pluggable Connections. The type of Connection is incorporated in the terminal name as a suffix. The suffixes are equal for inlet and outlet terminals. For the outlet terminals the possibilities are *drain*, *drainS*, *drainH*, *drainT*, *drainC* which stand for the *Stream-*, *Stream H-*, *HeatFlux-*, *Temperature-* and *Control-*Connection, respectively. If a terminal name differs from the ones above, an explanation will be given, *e.g.* if terminals for multiple Connections of the same type are used in a Unit. In figure III.6, the terminal names are given in square brackets.

6.1. Heat Exchangers

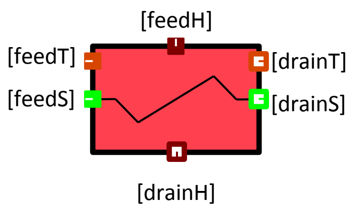
First, we start with one of the more challenging models. The heat exchangers, such as the condenser and evaporator of a refrigeration appliance have to cover a large range of physical parameters. This range includes operating points in the single and two-phase region. Several equations, such as the conservation laws, hold in both regions but the density ρ, the Heat Transfer Coefficient (HTC) α and the Frictional Pressure Drop (FPD) Δp are determined by different sets of equations for each region. For a more detailed description of the Condenser- and Evaporator-Unit, we refer to [111]. We present the equations independent of the operating point first.

The heat exchangers are assumed to be horizontal tubes. A distributed parameter approach is applied to the heat exchangers. The spatial discretization is handled by a one dimensional finite volume approach. Each volume is represented by one Unit. The Unit is subdivided into the working fluid flow region and the pipe wall which are treated independently. It is assumed that, first, the working fluid exchanges heat with the pipe wall and, second, the pipe wall exchanges heat with two contiguous Units. Furthermore, if the Temperature-Connections are attached, heat conduction is included in the pipe wall model. The variables associated with the pipe wall are indicated with a subscript p, *e.g.* for the pipe temperature we have T_p. For the working fluid flow, a first-order upwind scheme was applied. The upwind scheme implies that the state in the interior equals the state at the outlet. For a deeper insight into the theory of finite volume methods, we refer to [61, 94]. In figure III.7 typical flow sheet arrangements of evaporator and condenser are illustrated. In the refrigeration appliance the evaporator is situated inside the insulation which covers the compartment. The condenser is installed free-hanging at the back of the appliance which implies the condenser exchanges heat with the ambiance but also a heat input into the insulation is assumed.

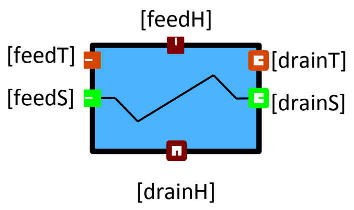
The ordinary differential equations resulting from the conservation of mass and energy for the working fluid read as

$$\frac{dm}{dt} = \dot{m}_{feedS} - \dot{m}_{drainS}, \tag{6.1}$$

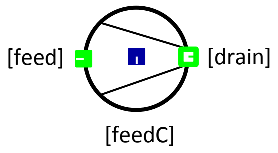
$$\frac{dU}{dt} = \alpha A(T_p - T) + \dot{m}_{feedS} h_{feedS} - \dot{m}_{drainS} h_{drainS} \tag{6.2}$$



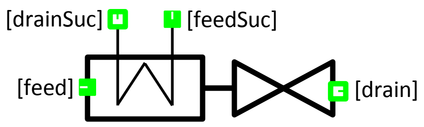
(a) Unit Condenser.



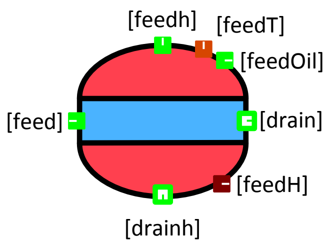
(b) Unit Evaporator.



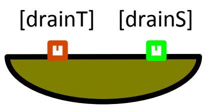
(c) Unit Compressor.



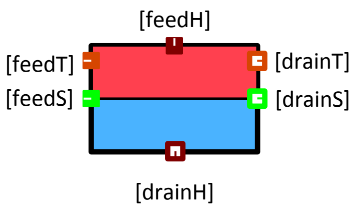
(d) Unit Capillary.



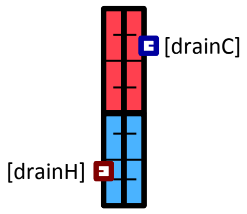
(e) Unit Shell.



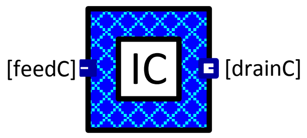
(f) Unit Oil Sump.



(g) Unit Accumulator.



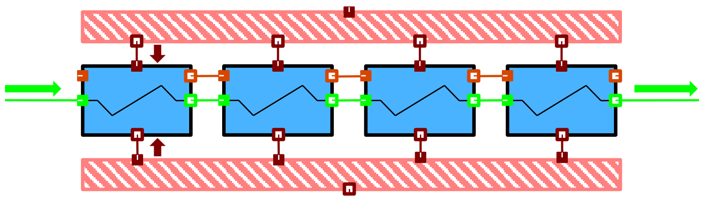
(h) Unit Sensor.



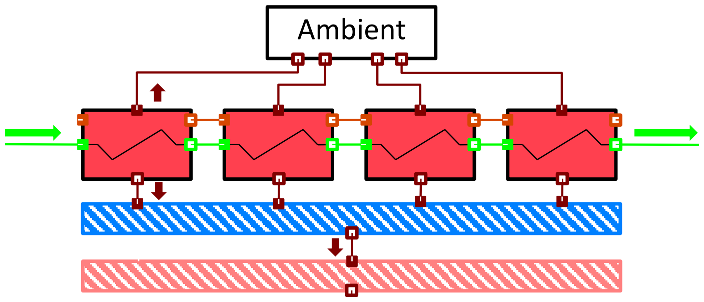
(i) Unit I_Control.

(j) Unit Geometry.

Figure III.6.: Selection of Unit icons.



(a) Evaporator-Units between the insulation (diagonally striped, pink Units) of the compartment.



(b) Condenser-Units connected on the one side to the ambiance and on the other side to the insulation (diagonally striped, pink) of the compartment with an air pad (diagonally striped, blue) in between.

Figure III.7.: Typical flow sheet arrangement of the heat exchangers in the refrigeration appliance. Appearing Connections are StreamH (green), HeatFlux (dark red) and Temperature (orange). Arrows indicate the expected direction of the flux.

with the mass m, mass flow $\dot{m}$, inner energy U, surface area A, temperature T, pipe temperature T_p, specific enthalpy h and HTC α. The inner engery can be expressed in terms of the specific inner energy u and the mass, *i.e.* $U = mu$. Substituting this in the energy equation and applying the product rule yields

$$m\frac{du}{dt} + u\frac{dm}{dt} \;=\; \alpha A(T_p - T) + \dot{m}_{feedS}h_{feedS} - \dot{m}_{drainS}h_{drainS}. \tag{6.3}$$

Since the velocity changes are small in the considered models, the momentum balance reduces to an algebraic equation describing the pressure drop Δp caused by friction, *i.e.*

$$p_{drainS} \;=\; p_{feedS} \;-\Delta p \tag{6.4}$$

where the FPD Δp is determined by empirical correlations, see the subsection 6.1.1 and 6.1.2. It is assumed that the pressure p in the interior equals the one at the outlet terminal, *i.e.*

$$p \;=\; p_{drainS}. \tag{6.5}$$

Loosely speaking, the pressure in the attached StreamH-Connections is determined by equation (6.4). To obtain a unique thermodynamic state a second state variable has to be given. In this case the enthalpy is chosen which also appears in the energy equations of the contiguous Units. Since further considerations are necessary, we postpone the determination the enthalpy in the StreamH-Connections and refer to equation (6.30). Next, the pipe wall is modelled. Neglecting heat conduction, the energy equation for the pipe wall is given by

$$\rho_p c_p V_p \frac{dT_p}{dt} \;=\; \dot{Q}_{drainH} - \dot{Q}_{feedH} - \alpha A(T_p - T). \tag{6.6}$$

The volume V_p, heat capacity c_p and density ρ_p of the pipe wall are fixed quantities independent of the temperature held by the Geometry- and Metal-Global. The heat fluxes, $\dot{Q}_{drainH}$ and $\dot{Q}_{feedH}$, are determined in HeatFlux-Connections. Concluding the part of the model which is independent of the operating point, for the working fluid we have the algebraic equations

$$h = u + \frac{p}{\rho}, \tag{6.7}$$

$$T = T(p, h), \tag{6.8}$$

$$m = \rho V. \tag{6.9}$$

Furthermore, there are several equations passing variables to Connections. We do not present these identities here and refer to the library. The models for the density, HTC and FPD depend on the operating point. We start with the single-phase region.

6.1.1. Single-phase

Let the state of the working fluid be either in the liquid or vapor region. Then the density is given by the equation of state which references an external function calling Refprop, *i.e.*

$$\rho = \rho(p, h). \tag{6.10}$$

The FPD Δp is determined by the Darcy–Weisbach equation

$$\frac{\Delta p}{l} = f \frac{\rho}{d_H} \frac{v^2}{2} \tag{6.11}$$

with the length of the volume l, the hydraulic diameter d_H, the friction factor f and the velocity v. The friction factor f is calculated differently for laminar and turbulent flows. The dimensionless Reynolds number Re is used to distinguish between the two types of flow. It is defined as

$$\mathrm{Re} = \frac{\dot{m} d_H}{A \eta} \tag{6.12}$$

with the mass flow $\dot{m}$, flow area A and dynamic viscosity η. For $\mathrm{Re} < 2300$ the flow in a pipe is laminar and for $\mathrm{Re} > 10^4$ fully turbulent, see [28]. The friction factors for each type of flow are

$$f_{laminar} = \frac{64}{\mathrm{Re}} \tag{6.13}$$

and

$$f_{turbulent} = \frac{0.3164}{\sqrt[4]{\mathrm{Re}}}. \tag{6.14}$$

If $2300 \leq \mathrm{Re} \leq 10^4$ holds, the flow type depends on further parameters, *e.g.* the geometry or the tube surface roughness, and is hard to determine properly. Therefore, linear interpolation between $f_{laminar}$ and $f_{turbulent}$ is applied.

The last variable which has to be determined is the HTC α between the working fluid and the pipe wall. We follow [28] and define the dimensionless Nusselt number

$$\mathrm{Nu} = \frac{\alpha d_H}{\lambda} \tag{6.15}$$

where λ is the heat conductivity. In the literature, correlations for the Nusselt number are given and α is obtained from the above equation. In general, it has to be distinguished between natural and forced convection. However, in the heat exchanger models always forced convection is assumed. Hence the correlations are of the form

$$\mathrm{Nu} = f(\mathrm{Re}, \mathrm{Pr}) \tag{6.16}$$

with the Reynolds number Re defined in equation (6.12) and the dimensionless Prandtl number

$$\mathrm{Pr} = \frac{\eta c_p}{\lambda} \tag{6.17}$$

where c_p is the isobaric heat capacity. As for the pressure drop, different correlations have been proposed for laminar and turbulent flow and linear interpolation is applied if $2300 \leq \mathrm{Re} \leq 10^4$ holds. For laminar flow the Nusselt number is approximated by

$$\mathrm{Nu}_{laminar} = 1.615 \left(\mathrm{Re}\,\mathrm{Pr}\,\frac{d}{l} \right)^{\frac{1}{3}}. \tag{6.18}$$

In [33], Gnielinski proposed for turbulent flow

$$\mathrm{Nu}_{turbulent} = \frac{(\xi/8)\mathrm{RePr}}{1 + 12.7\sqrt{\xi/8}\left(\mathrm{Pr}^{2/3} - 1\right)} \left[1 + \left(\frac{d_H}{l} \right)^{2/3} \right] \tag{6.19}$$

with the Darcy friction factor

$$\xi = (1.8 \log_{10} \mathrm{Re} - 1.5)^{-2}. \tag{6.20}$$

6.1.2. Two-phase

If the working fluid is two-phase, the models are much more complex. We start with the density. It is obtained by the interpolation

$$\rho = \varepsilon \rho_v + (1 - \varepsilon)\rho_l \tag{6.21}$$

between the densities ρ_l and ρ_v at the lower and upper phase boundary, respectively. Above, ε is the vapor void fraction which is defined as the ratio between the cross-sectional area filled with vapor A_v and the total area A, *i.e.*

$$\varepsilon = \frac{A_v}{A}. \tag{6.22}$$

The vapor void fraction is obtained from empirial correlations and calculated in an external function of the *HTX.dll*. The function call in MDK reads as

$$\varepsilon = \varepsilon(p, \rho_l, \rho_v, \dot{x}, G) \tag{6.23}$$

where G is the mass flow per (cross-sectional) area given by

$$G = \frac{\dot{m}_{feed}}{A} \tag{6.24}$$

and $\dot{x}$ is the flow quality defined as

$$\dot{x} = \frac{\dot{m}_v}{\dot{m}_l + \dot{m}_v}$$

with the mass flows $\dot{m}_l, \dot{m}_v$ of the liquid and vapour share. If the velocities of the vapor and liquid share are equal, the flow quality $\dot{x}$ equals the vapor quality x. We shall, however, drop this assumption and derive a relation between x and $\dot{x}$. To guarantee a continuous extension of the vapor quality to the single phase region, it is defined as

$$x = \frac{h - h_l}{h_v - h_l} \tag{6.25}$$

where h_l and h_v are the specific enthalpies at the lower and upper phase boundary, respectively. The original definition of the vapor quality holds in the two-phase region only, *i.e.*

$$x = \frac{m_v}{m_l + m_v}.$$

where m_v and m_l are the mass of the vapor and liquid share, respectively. Applying equations (6.9) and (6.22), the numerator in the above equation can be rewritten as

$$m_v = \rho_v V_v = \rho_v A_v l = \rho_v \varepsilon A l = \rho_v \varepsilon V.$$

Here, V_v denotes the volume filled with vapor, A the cross-sectional area and l the length of the tube. Applying the same transformation to the denominator and utilizing (6.21) yields

$$x = \frac{V}{V} \cdot \frac{\varepsilon \rho_v}{(1-\varepsilon)\rho_l + \varepsilon \rho_v} = \frac{\varepsilon \rho_v}{\rho}$$

where the vapor void fraction ε depends on $\dot{x}$, see (6.23). Outside the two-phase region we naturally set $\dot{x} = x$. Hence, the continuous relation between x and $\dot{x}$ is given by

$$x = \begin{cases} \frac{\varepsilon \rho_v}{\rho}, & \text{if } 0 \leq \dot{x} \leq 1, \\ \dot{x}, & \text{else} \end{cases} . \tag{6.26}$$

The choice to let $\dot{x}$ decide which equation is applied has a specific reason which results from Newton's method. In the implementation of equation (6.23), the variable $\dot{x}$ decides how to calculate ε. The vapor void fraction is either determined by correlations given below or set to 0 or 1 if $\dot{x} \leq 0$ or $\dot{x} \geq 1$, respectively. In either case, if the working fluid is single-phase, $\partial\varepsilon/\partial\dot{x} = 0$ holds. If the working fluid is two-phase, the partial derivative of equation (6.26) with respect to $\dot{x}$ equals 0 as well. Assume that x, and not $\dot{x}$, would be used in the conditional statement and the operating point is close to the two-phase boundary. Then, during the Newton iteration a situation can occur in which $\dot{x} > 1$ or $\dot{x} < 0$ but $0 < x < 1$ holds. This yields a singular iteration matrix and Newton's method fails to converge. To avoid this problem, $\dot{x}$ is chosen in all conditional statements.

The correlations to calculate the vapor void fraction are presented next. We follow [36] wherein a more detailed description can be found. If the liquid and vapor phase flow with the same velocity, the homogeneous vapor void fraction ε_{hom} is given by

$$\varepsilon_{hom} = \frac{1}{1 + \frac{1-\dot{x}}{\dot{x}} \frac{\rho_v}{\rho_l}}. \tag{6.27}$$

The Rouhani–Axelsson drift flux model for horizontal tubes, see [112], is applied for the nonhomogeneous vapor void fraction ε_{ra}, *i.e.*

$$\varepsilon_{ra} = \frac{\dot{x}}{\rho_v} \left((1 + 0.12(1-\dot{x}) \left[\frac{\dot{x}}{\rho_v} + \frac{1-\dot{x}}{\rho_l} \right] + \frac{1.18(1-\dot{x})(g\sigma(\rho_l - \rho_v))^{0.25}}{G\rho_l^{0.5}} \right)^{-1} . \tag{6.28}$$

The drift flux model is implemented in the Evaporator-Unit. The LMε model, a combination of the above correlations, reads as

$$\varepsilon_{lm} = \frac{\varepsilon_{hom} - \varepsilon_{ra}}{\ln(\varepsilon_{hom}) - \ln(\varepsilon_{ra})}. \tag{6.29}$$

In [36], the authors argue that the LMε model covers a larger pressure range. As proposed, we use the above combination in the Condenser-Unit. This concludes the discussion of the required variables and the calculations to obtain the density of the two-phase working fluid.

The models for the FPD Δp and HTC α are based on flow pattern maps, as explained in chapter I. Depending on physical properties, the geometry, the mass flow per area G and the flow quality $\dot{x}$, a flow pattern forms in the tubes which influences Δp and α. For each flow pattern different empirical correlations have been established in the literature which are not necessarily continuous across the pattern boundaries. We implement the flow pattern map developed in [36] for condensation and for evaporation the extended version proposed in [112]. In the latter the boundaries of the dryout regime were calculated using the approach of [72] due to stability reasons during Newton's method. Furthermore, the transition curves G_{dryout} and G_{mist} were not derived but we used the definitions of $\dot{x}_{di}$ and $\dot{x}_{de}$ (flow quality boundaries of the dryout pattern) directly to decide which flow pattern is apparent.

The FPD models were taken from [89] and [90]. These models are based on the extended flow pattern maps of [112] and hence cover the flow pattern map for condensation also - as aimed by the authors. Consequently, only one FPD model for both condensation and evaporation has to be implemented. The HTC is calculated as in [103] and [113] for condensation and evaporation, respectively.

Finally, we determine the enthalpies h_{drainS} and h_{feedS} in the StreamH-Connections. Assume that the refrigerant is two-phase. The transported energy through the mass transport from one Unit to the next reads as

$$\begin{aligned}\dot{m}_l h_l + \dot{m}_v h_v &= \dot{m}(1-\dot{x})h_l + \dot{m}\dot{x}h_v \\ &= \dot{m}(h_l + \dot{x}(h_v - h_l)).\end{aligned} \tag{6.30}$$

Since in the StreamH only the total mass flow $\dot{m}$ is defined, the enthalpies h_{drainS} and h_{feedS} have to be calculated accordingly. If the velocities of the vapour and liquid share are equal or the refrigerant is single phase, $x = \dot{x}$ holds and we obtain $h_{drainS} = h_{feedS} = h$, the enthalpy of the refrigerant in the interior. In the validated model of chapter IV and V the outlet enthalpy is, however, determined using the vapour quality. This has an impact on the local mass distribution in the system but *e.g.* the parameters fitted with steady-state measurements remain equal.

6.2. Accumulator

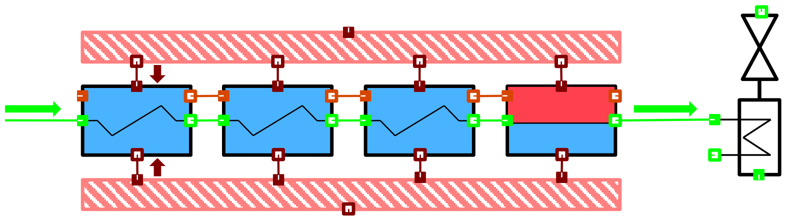


Figure III.8.: Typical flow sheet arrangement of the accumulator between the evaporator and the internal heat exchanger and situated inside the insulation of the compartment (diagonally striped, pink). Appearing Connections are StreamH (green), HeatFlux (dark red) and Temperature (orange). Arrows indicate the expected direction of the flux.

In principle, the Accumulator-Unit is modelled as the Evaporator described in the last section. They only differ in how the enthalpy at the outlet is determined. In the refrigeration cycle model, an Accumulator is added at the end of the evaporator, as illustrated in figure III.8, to guarantee that only saturated or superheated steam leaves. To ensure this, the Accumulator is designed and situated in such a way that the gravitational force holds the liquid share of the working fluid inside the component. This procedure improves the cooling capacity, since the liquid share can still absorb heat. The *Stream H*-Connection is attached to enable changing flow directions. Thus, the enthalpy at two possible outlets has to be determined.

We start with the enthalpy which is passed to the *drainH*-Connection. We assume that the accumulator may not be optimal and define an efficiency $\xi \in [0,1]$. If $\xi = 1$ holds, the accumulator is optimal and only saturated ($h = h_v$) or superheated steam leaves. Otherwise if $\xi < 1$, it is assumed that liquid also escapes. For the efficiency a linear ansatz was chosen, *i.e.*

$$\xi = a + b\frac{dp}{dt} \tag{6.31}$$

where $(a,b) \in [0,1] \times \mathbb{R}^+$ are parameters which have to be fit. The second term implies that the efficiency decreases if the pressure drops and vice versa. If the working fluid is in the two-phase region, the enthalpy at the outlet is given by

$$h_{drainS} = h_l + (h_v - h_l)\xi \tag{6.32}$$

where h_l and h_v denote the enthalpy on the lower and upper two-phase boundary, respectively. This implies that ξ can be interpreted as the flow quality at the outlet. Finally, some sort of interpolation has to be applied if the working fluid crosses the two-phase boundary to guarantee a continuous model.

If the flow direction changes, the enthalpy on the *feedH*-Connection is determined by the value passed from the Accumulator-Unit. We assume that for a vapor quality $x \in (0, 0.9)$ only boiling liquid ($h = h_l$) leaves. Applying linear interpolation for $x \in [0.9, 1]$, yields

$$h_{feedS} = \begin{cases} h, & \text{if } x \leq 0 \text{ of } x \geq 1, \\ h_l, & \text{if } 0 < x < 0.9, \\ h_l + (h_v - h_l)\frac{x-0.9}{0.1}, & \text{else.} \end{cases} \tag{6.33}$$

6.3. Compressor

The compressor component is situated between the evaporator and the condenser. Its purpose is to compress the fluid to a higher pressure such that the fluid temperature is above ambient temperature. Only then heat is transferred to the ambiance in the condenser. The whole compressor component consists of many individual parts from which not all are explained in detail here. In figure III.9 a schematic of a compressor is given and we explain shortly the path of the working fluid through this component.

The fluid leaving the evaporator enters the shell of the compressor through the suction tube. In the compressor of the investigated appliance the direct suction element is missing such that the fluid is not directed solely into the suction muffler. Thus the interior of the shell is filled with (superheated) fluid and oil which is used to cool and lubricate the compressor. From there the working fluid is sucked into the suction muffler. When the piston moves back, the suction valve opens and the fluid enters the cylinder. Near the bottom dead centre this valve closes. When the piston moves forward again, the fluid is compressed. Close to the top dead centre the discharge valve opens and the compressed fluid is directed into the discharge muffler, then into the internal discharge line and finally into the discharge tube which leaves the shell and is connected to the condenser.

In IPSEpro the component is split into three individual parts. First, the *Shell*-Unit comprises, the shell wall including all metal parts in the interior, the working fluid on the suction side, located inside the shell, and on the discharge side, hence in the discharge muffler and discharge line. Second, the *Compressor*-Unit handles the actual compression and the consumed electric power. Third, the *Oil_ Sump*-Unit concerns the oil in the shell used to cool and lubricate the compressor. It is crucial to add the oil since it absorbs and desorbs working fluid. In figure III.10 the typical arrangement of these Units is depicted.

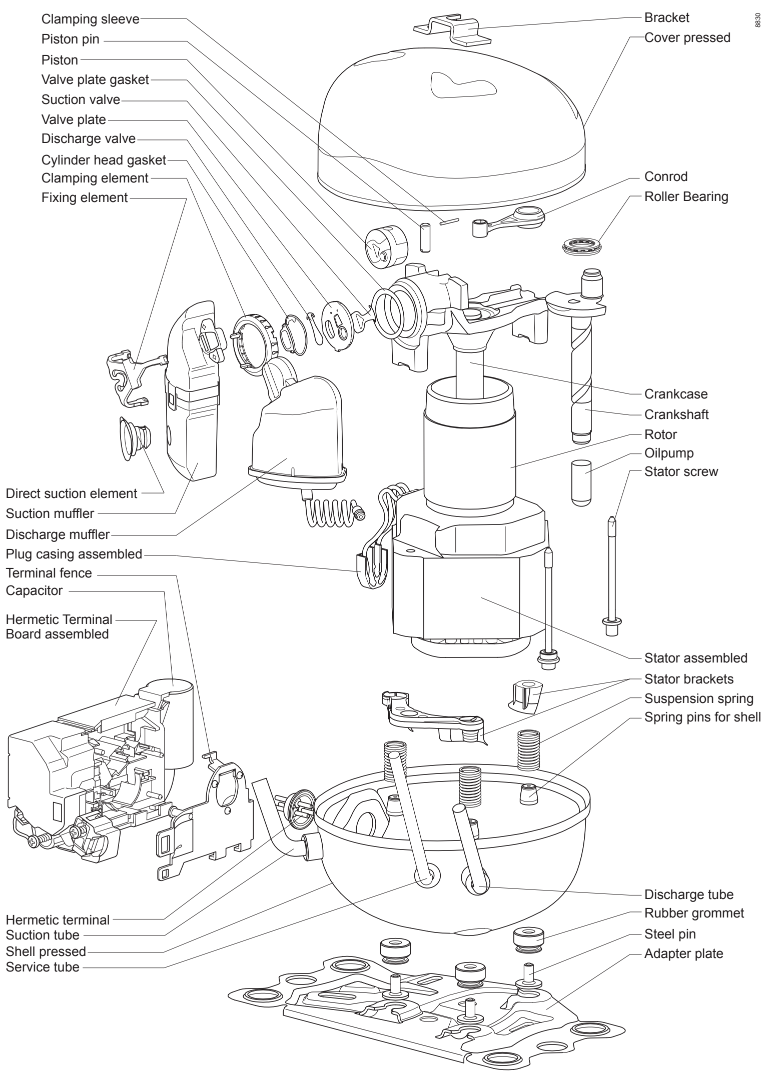


Figure III.9.: Schematic of a compressor, [93]

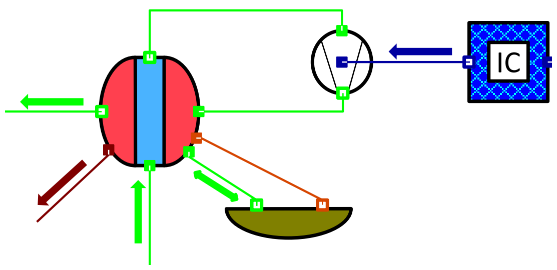


Figure III.10.: Typical flow sheet arrangement of the compressor components and the I_control Unit. Appearing Connections are Stream (green), HeatFlux (dark red), Temperature (orange) and Control (dark blue). Arrows indicate the expected direction of the flux.

6.3.1. Shell

Within this section, the Shell-Unit is presented. The Unit itself is subdivided into three parts. First, the working fluid on the suction side (inside the shell before compression), second, the shell wall including all metal parts in the interior and third, the working fluid on the discharge side (inside the discharge muffler and discharge tube after compression). Three heat transfers are assumed: first, $\dot{Q}_{sw}$ indicates the heat flux between the fluid suction side and the shell wall, second, $\dot{Q}_{dw}$ between the fluid discharge side and the shell wall and third, $\dot{Q}_{feedH}$ between the ambiance and the shell wall. Furthermore, mass transfer is modelled between the suction side fluid and the oil which is caused by absorption and desorption. Since multiple *Stream*-Connections are attached to this Unit the usual naming convention can not be applied anymore. The terminals for the suction side are denoted as $feed$ and $drain$, for the discharge side as $feedh$ and $drainh$, and for the oil side only one terminal $feedOil$ is available.

The differential equations for the fluid on the suction side read as

$$\frac{dm}{dt} = \dot{m}_{feed} - \dot{m}_{drain} + \dot{m}_{feedOil}, \tag{6.34}$$

$$\begin{aligned}\frac{dU}{dt} = {} & \dot{m}_{feed}h_{feed} - \dot{m}_{drain}h_{drain} \\ & + \dot{m}_{feedOil}h_{feedOil} + \dot{Q}_{sw}\end{aligned} \tag{6.35}$$

with the heat flux

$$\dot{Q}_{sw} = \alpha_{sw}A_{sw}(T_w - T). \tag{6.36}$$

The momentum balance reads as

$$p_{feed} - \Delta p = p_{drain} \tag{6.37}$$

where Δp has to be prescribed. Futhermore, the working fluid desorbed from the oil is supposed to have the same enthalpy as the fluid on the suction side immediately. This is not obvious from the energy equation above. The algebraic equations for the fluid on the suction side read as

$$h = u + \frac{p}{\rho}, \tag{6.38}$$

$$T = T(p,h), \tag{6.39}$$

$$\rho = \rho(p,h), \tag{6.40}$$

$$m = \rho V. \tag{6.41}$$

Next, the temperature evolution of the shell wall is described by

$$\rho_w c_w V_w \frac{T_w}{dt} = -\dot{Q}_{dw} - \dot{Q}_{sw} - \dot{Q}_{feedH} \tag{6.42}$$

with the volume V_w, heat capacity c_w and density ρ_w of the shell wall. The heat flux between the shell wall and the working fluid on the discharge side is defined as

$$\dot{Q}_{dw} = \alpha_{dw}A_{dw}(T_w - T_{drainh}). \tag{6.43}$$

The last term in equation (6.42) implies that the shell wall exchanges heat with another Unit. Usually this will be the ambiance. The heat flux $\dot{Q}_{feedH}$ is calculated in a *Heat Flux*-Connection. Finally, on the discharge side we assume quasi-stationary behavior and no pressure drop. Thus the describing equations reduce to

$$\dot{m}_{feedh} = \dot{m}_{drainh}, \tag{6.44}$$

$$0 = \dot{Q}_{dw} + \dot{m}_{feedh}(h_{feedh} - h_{drainh}), \tag{6.45}$$

$$p_{feedh} = p_{drainh}. \tag{6.46}$$

6.3.2. Compression

The second model takes care of the actual compression. We assume quasi-stationary behavior. Thus the mass balance reads as

$$\dot{m}_{feed} = \dot{m}_{drain}. \tag{6.47}$$

The volume flow $\dot{V}$ is defined as

$$\dot{V} = \frac{nV_h}{60} \tag{6.48}$$

where V_h is the displacement and n the NOR of the compressor. The NOR are determined by the I_Control-Unit and we have

$$n = x_{feedC}. \tag{6.49}$$

Considering a volumetric efficiency η_v, the mass flow is given by

$$\dot{m}_{drain} = \eta_v \dot{V} \rho. \tag{6.50}$$

Which density ρ is to be taken in the above equation is not addressed precisely in the literature. Since the density at the inlet of the Compressor-Unit is hard to measure, a density from a different position has to be chosen. In [85], the authors tested the model using densities at different positions and propose that the density at the inlet of the Shell-Unit achieves the best results. In figure III.11 the densities at these mentioned points are illustrated. The values are taken from the validated refrigeration appliance model presented in chapter IV. It can be seen that the density at the Compressor-Unit inlet is less than at the Shell-Unit inlet. Using the density at the Shell-Unit inlet yields a higher mass flow.

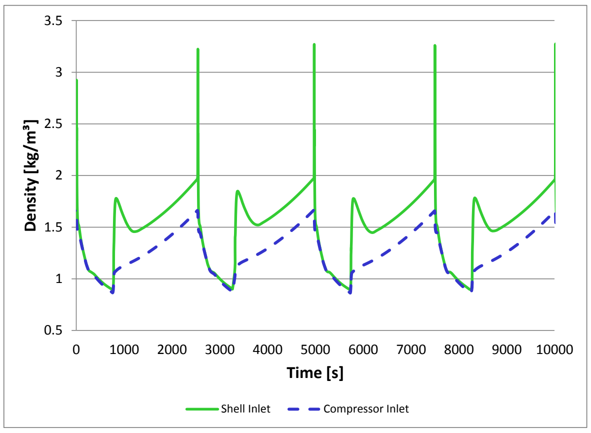


Figure III.11.: Density at Shell and Compressor inlet over four on-off-cycles.

Following [63], isentropic compression is assumed. The authors propose to determine the isentropic power P_{is} via

$$P_{is} = \dot{V} \frac{\kappa}{\kappa - 1} p_{feed} \left[\left(\frac{p_{drain}}{p_{feed}} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right] \tag{6.51}$$

with the isentropic exponent at the inlet

$$\kappa = \frac{c_p(p_{feed}, T_{feed})}{c_v(p_{feed}, T_{feed})} \tag{6.52}$$

and calculate the enthalpy h_{drain} at the outlet using the energy equation

$$P_{is} = -\dot{m}_{feed}(h_{feed} - h_{drain}). \tag{6.53}$$

The volumetric efficiency η_v, used in equation (6.50), is defined as

$$\eta_v = b_1 + b_2 \left(\frac{p_{drain}}{p_{feed}}\right)^{\frac{1}{\kappa}} \tag{6.54}$$

where $b_1, b_2 \in \mathbb{R}$ are fitting parameters which have to be adjusted for each compressor type. Finally, the compressor shaft power P is

$$P = \frac{1}{\eta_{comb}} P_{is} \tag{6.55}$$

where the combined efficiency reads as

$$\eta_{comb} = \frac{1}{a_1 + \frac{a_2}{p_{feed}} + \frac{a_3}{p_{drain}}}. \tag{6.56}$$

The parameters $a_1, a_2, a_3 \in \mathbb{R}$ have to be fit for each compressor type.

Since equation (6.51) holds for ideal gas only, a true isentropic compression is not achieved for real gas such as the refrigerant Isobutane (R600a) with this equation. This equation could be replaced by

$$s_{feed} = s_{drain}.$$

Loosely speaking, then h_{drain} is determined by the state equation of the Stream, see (5.3), and the isentropic power P_{is} is obtained from the energy equation. Since all parameters a_i, b_i have already been fitted for the model including equation (6.51), also in IPSEpro equation (6.51) is used.

6.3.3. Oil Sump

The final model developed for the compressor part describes the oil sump where we follow the work of [81]. We assume that the oil can absorb working fluid but no heat is exchanged and that desorbed working fluid has immediately the same enthalpy as the one on the suction side. Let m be the mass of the working fluid dissolved in oil which itself has constant mass m_{oil}. The ratio between these masses is denoted as μ, *i.e.*

$$\mu = \frac{m}{m_{oil}}. \tag{6.57}$$

The time evolution of μ is described by the ODE

$$\frac{d\mu}{dt} = \frac{1}{\tau}(\mu_{sat} - \mu) \tag{6.58}$$

with time constant τ. The mass ratio at saturation reads as

$$\mu_{sat} = \frac{m_{sat}}{m_{oil}} \tag{6.59}$$

where the mass of the working fluid at saturation is calculated using the relation

$$m_{sat} = \frac{p}{p_{sat}} m_{oil} \frac{M}{M_{oil}(1 - \frac{p}{p_{sat}})} \tag{6.60}$$

proposed in [77]. In the above equation, M and M_{oil} are the molar masses of the working fluid and the oil, respectively, p denotes the pressure of the working fluid on the suction side and $p_{sat} = p_{sat}(T_{oil})$ the saturation pressure of the working fluid at temperature of the oil which is assumed to equal the temperature of the shell wall. Finally the mass flow $\dot{m}_{drain}$ of working fluid between the suction side and the oil is given by the mass balance

$$\frac{dm}{dt} = -\dot{m}_{drain}. \tag{6.61}$$

It remains to provide reasonable values for the time constant τ. To achieve this goal we split the evaluation into three cases. First, we handle the case in which the compressor is turned off. Here the mass transfer at the surface of the oil is determined by diffusion which is described by Fick's law. We approximate it by equation (6.58) and the time constant is chosen in such a way that it fits the diffusion model on an interval which is relevant for a refrigeration appliance. This yields $\tau = 1.9 \times 10^5$ s. The other two cases appear when the compressor is switched on. We distinguish between absorption and desorption. In [81], the same relation for the saturation level of the oil is used to describe the effects in both cases. We deduce this relation in following lemma.

Lemma 6.1 *Let $\zeta = \mu/\mu_{sat}$ be the saturation level of the oil. Then*

$$\zeta(t) = 1 - \frac{\mu_{sat} - \mu_0}{\mu_{sat}} e^{-\frac{t}{\tau}} \tag{6.62}$$

is the analytic solution of (6.58) for time independent μ_{sat} and initial value $\mu_0 = \mu(0)$.

Proof. Let the differential equation be given by

$$\frac{d\mu}{dt} = \mu' = \frac{1}{\tau}(\mu_{sat} - \mu). \tag{†}$$

Rearranging the terms yields

$$-\frac{\mu'}{\mu_{sat} - \mu} = -\frac{1}{\tau}.$$

Integrating from 0 to t leads to

$$\ln((\mu_{sat} - \mu)(t)) - \ln((\mu_{sat} - \mu)(0)) = -\frac{t}{\tau}$$

which can be rewritten to

$$\frac{\mu_{sat} - \mu(t)}{\mu_{sat} - \mu_0} = e^{-\frac{t}{\tau}}$$

with initial value $\mu_0 = \mu(0)$. Thus the analytic solution of (†) reads as

$$\mu(t) = \mu_{sat} - (\mu_{sat} - \mu_0)e^{-\frac{t}{\tau}}.$$

Finally, division by μ_{sat} yields the result. □

Measurements were performed to fit the time constant τ in equation (6.62). If the compressor is switched on, $\tau = (42.8 \pm 7.0)$ s and $\tau = (14.1 \pm 1.5)$ s for absorption and desorption, respectively, are proposed in [81]. Consequently, the established model covers all cases but the time constant τ has to be adjusted whenever the operating mode changes.

6.4. Capillary

The capillary, counterpart of the compressor, is an expansion device. In a refrigeration appliance it is situated between the condenser and evaporator and its purpose is to decrease the pressure of the working fluid and thus the temperature of evaporation. Furthermore, the capillary can be used as an internal heat exchanger between working fluids F_c and F_e which come from the condenser and evaporator, respectively. This structure has two benefits. First, F_c gets pre-cooled before the expansion and hence increases the cooling capacity. Second, F_e obtains further energy which can avoid that two-phase working fluid flows into the compressor. The terminals for F_c are denoted as *feed* and *drain* and for F_e as *feedSuc* and *drainSuc*. In figure III.12 two designs of non-adiabatic capillary tubes are illustrated. Both designs have a countercurrent heat exchanger, but the capillary tube is placed into or onto the suction line in the A-A or B-B design, respectively.

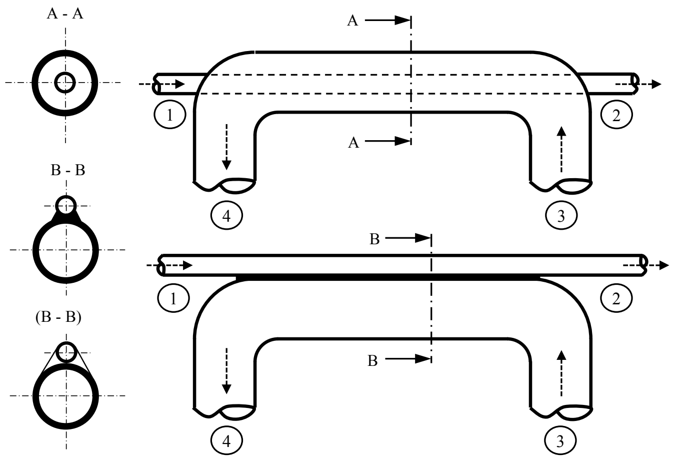


Figure III.12.: Capillary design, [38]. (1) $\rightarrow$ (2): capillary tube, (3) $\rightarrow$ (4): suction line. (1) from condenser (*feed*), (2) to evaporator (*drain*), (3) from evaporator/accumulator (*feedSuc*), (4) to compressor (*drainSuc*). A-A: non-adiabatic, coaxial, soldered, B-B: non-adiabatic, lateral, soldered, (B-B): non-adiabatic, lateral, fixed with foil.

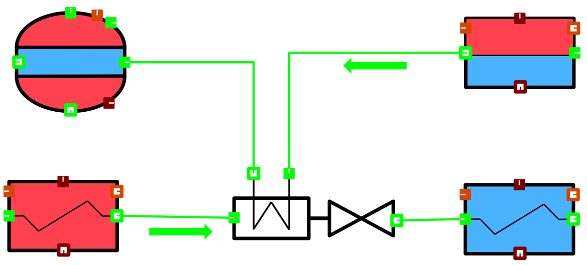


Figure III.13.: Typical flow sheet arrangement of the capillary in an refrigeration appliance. The only appearing Connections are Streams (green). Arrows indicate the expected direction of the flux.

We follow the work of [41] in which the non-adiabatic capillary is modelled by an Artificial Neuronal Network (ANN) which was trained with data from a complex one-dimensional homogeneous model validated with measurements. For a detailed description of the complex model we refer to [38]. The reason why the ANN is applied instead of the complex model lies in the high numerical costs of the complex model. The ANN calculates values for the mass flow $\dot{m}_{drain}$ through the capillary and the enthalpy $h_{drainSuc}$ at the outlet of the internal heat exchanger on the suction side. The latter was chosen instead of the heat flux $\dot{Q}$ because the ANN provided better results with this setting. Since IPSEpro is equation-based, it is still possible to prescribe the mass flow or enthalpy and obtain another value from the list of input parameters. This list contains the capillary inlet and outlet pressures p_{feed} and p_{drain} and enthalpies h_{feed} and h_{drain}, the radii r_{Cap}, r_{Suc} and lengths l_{Cap}, l_{Suc} of the capillary and heat exchanger, respectively, and the inlet pressure $p_{feedSuc}$, enthalpy $h_{feedSuc}$, outlet pressure $p_{drainSuc}$ and mass flow $\dot{m}_{drainSuc}$ of the suction line. Consequently, the first two equations read as

$$\dot{m}_{drain} = \dot{m}(\text{ANN}), \tag{6.63}$$
$$h_{drainSuc} = h(\text{ANN}). \tag{6.64}$$

In the capillary tube and suction line quasi-stationary behavior is assumed. Hence the mass balances are given by

$$\dot{m}_{feed} = \dot{m}_{drain}, \tag{6.65}$$
$$\dot{m}_{feedSuc} = \dot{m}_{drainSuc} \tag{6.66}$$

and the energy balances by

$$\dot{Q} = \dot{m}_{feed}(h_{feed} - h_{drain}), \tag{6.67}$$
$$\dot{Q} = \dot{m}_{feedSuc}(h_{feedSuc} - h_{drainSuc}). \tag{6.68}$$

The energy balances also imply that no heat is exchanged with the ambiance. This assumption is valid since most of the capillary and suction line are insulated by foam in the refrigeration appliance. In the suction line, a pressure loss independent of the length and the state of the fluid is assumed. Thus

$$p_{drainSuc} = p_{feedSuc} - \Delta p \tag{6.69}$$

where Δp is supposed to be prescribed.

One negative aspect of an ANN becomes apparent when the input parameters lie outside the trained region. Since the ANN does not ensure that physical laws are satisfied, non-physical results are possible especially outside the trained region. Therefore, precautions against such cases have to be taken.

The exceptional cases are dealt with as follows: First, we consider the mass flow calculation. If the compressor is turned off and the pressure difference $p_{feed} - p_{drain}$ becomes small, we calculate a lower bound $\dot{m}_{min}$ of the mass flow through the capillary tube. This is necessary because the ANN tends to predict to small mass flows in this case and thus the pressure equalizing does not take place. As a lower bound the simple relation

$$\delta = p_{feed} - p_{drain}, \tag{6.70}$$
$$\dot{m}_{min} = \frac{\text{sgn}(\delta)}{\sigma}\sqrt{|\delta|} \tag{6.71}$$

is used where $\sigma \in \mathbb{R}_+$ is a parameter and $\text{sgn}(\delta)$ denotes the usual "sign". If σ is increased, the mass flow decreases and the pressure equalizing takes longer. In all cycle simulations, $\sigma = 10^4$ is chosen.

Second, also the enthalpy at the suction line outlet of the internal heat exchanger has to be corrected in specific cases. On the one hand, if either the mass flow in the capillary tube or in the suction line becomes too small, the heat transfer has to be turned off smoothly since reasonable values are no longer achievable. On the other hand, also temperatures have to be observed on each side such that heat is only transferred from the side with higher to the side with lower temperature. This yields one condition on each end of the heat exchanger. The first condition at the suction line outlet is given by

$$T_{drainSuc} \leq T_{feed}. \tag{6.72}$$

Hence a maximal enthalpy $\hat{h}_{drainSuc}$ at the suction line outlet can be determined, *i.e.*

$$\hat{h}_{drainSuc} = h(p_{drainSuc},\ T_{feed}). \tag{6.73}$$

Since $T_{feed} > T_{sat}(p_{drainSuc})$ holds, a unique enthalpy is always guaranteed. The second condition concerns the enthalpy of the capillary tube outlet. Because expansion and heat exchange take place simultaneously we can not compare temperatures anymore. Therefore we consider the two extreme cases in which the heat is exchanged either before the isenthalpic expansion or afterwards. In the first case we obtain the minimal enthalpy

$$\check{h}_{drain} = h(p_{feed},\ T_{feedSuc}) \tag{6.74}$$

and in the second case

$$\check{h}_{drain} = h(p_{drain},\ T_{feedSuc}). \tag{6.75}$$

Given these two enthalpies, we take the minimum and determine a maximal enthalpy $\hat{h}_{drainSuc}$ at the suction line outlet by utilizing both energy equations.

6.5. Temperature Sensor

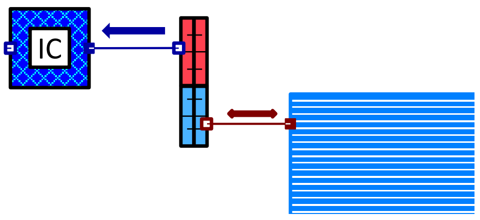


Figure III.14.: Typical flow sheet arrangement of the TemperatureSensor-Unit. It exchanges heat with the compartment (horizontally striped, light blue Unit) and passes information to the I_Control-Unit. Appearing Connections are Control (dark blue) and HeatFlux (dark red). Arrows indicate the expected direction of the flux.

Given the Units which describe the compressor, in the following we discuss how it is controlled. Usually, in a simple refrigeration appliance a sensor measures the temperature T_c in the compartment. In figure III.14 the typical flow sheet arrangement of the sensor is illustrated. This sensor has its own temperature T_s and the time evolution is described by the energy equation, *i.e.*

$$\begin{aligned} \rho_s c_s V_s \frac{dT_s}{dt} &= \dot{Q}_{drainH} \\ &= \alpha A (T_c - T_s) \end{aligned} \tag{6.76}$$

with the volume V_s, heat capacity c_s and density ρ_s of the sensor, the HTC α, and surface area A. Since the physical properties of such a sensor are poorly known and also the spatial position in the compartment has a big influence, an element of uncertainty is added to the system. If T_s reaches one of the two threshold values, T_{max} or T_{min}, the compressor has to be turned on or off, respectively. Therefore, the choice of the parameter ρ_s, c_s and V_s influences the switch-on and switch-off times of the compressor significantly. The actual controlling part is performed by the I_Control-Unit which is presented in a general form in the next section. Therefore the temperature of the sensor has to be passed to the Control-Connection, *i.e.*

$$x_{drainC} = T_s. \tag{6.77}$$

6.6. Integral Control

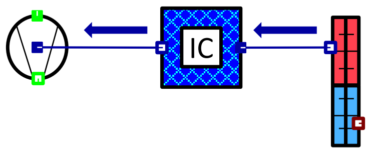


Figure III.15.: Typical flow sheet arrangement of the I_Control-Unit. The Temperature Sensor-Unit provides its temperature and the I_Control passes the appropriate NOR to the Compressor-Unit. Only Control-Connections appear (dark blue). Arrows indicate the expected direction of the flux.

The idea of the *I_Control*-Unit is the following: if the observed value x reaches a threshold, either minimum or maximum, the controlled value c will be adjusted. We have

$$x = x_{feedC}, \tag{6.78}$$
$$c = x_{drainC}. \tag{6.79}$$

The challenge lies in detecting the time t at which the observed value reaches a threshold, which is referred to as *event*, and a smooth transition between the possible values of c has to be guaranteed. In IPSEpro, termination criteria are provided as described in section II.4.2. Hence, the time t can be found at which an event occurs. Unfortunately, IPSEpro is not yet able to react automatically after the integration terminated. This implies that each time the integration stops the forcing term for the controlled value c has to be changed and the integration has to be restarted manually. To enable automatic switching, a new idea is developed which circumvents the use of termination criteria.

Let t_k be the last *valid* time at which the corrector equation, see equation (1.45) in chapter II, was solved and the integration step was accepted. Let t_{k+1} be the new suggested time at which the corrector equation has to be solved again. Let $x_k = x(t_k)$ and $x_{k+1} = x(t_{k+1})$ be the observed values at times t_k and t_{k+1}. If x_{k+1} lies on the same side of the given thresholds as x_k, nothing changes. Otherwise we assume that the observed value x changed linearly in time and determine the event time t^* at which x equals the threshold value θ, *i.e.*

$$t^* = t_k + (t_{k+1} - t_k)\frac{\theta - x_k}{x_{k+1} - x_k}. \tag{6.80}$$

We assume that the control element switched at t^* and calculate the controlled value c_{k+1} at time t_{k+1}. Let δ be the delay time it takes until the controlled value reaches its new operating value σ. If $t_{k+1} > (t^* + \delta)$ then $c_{k+1} = \sigma$ holds. Otherwise, a smooth transition of the controlled values is implemented. Let η be the controlled value before the event. Then

$$c_{k+1} = \eta + (\sigma - \eta)\left(3\tau^2 - 2\tau^3\right) \tag{6.81}$$

where

$$\tau = \frac{t_{k+1} - t^*}{\delta}. \tag{6.82}$$

If the difference between controlled values is significant, DASSL most certainly will decrease the step size which achieves a more accurate detection of the event time t^*.

The integral control Unit was originally designed to control the compressor of a refrigeration appliance. There, the compressor is turned to the maximal NOR if the temperature of the compartment reaches a maximum and vice versa. Therefore, the logic implemented in the *HTX.dll* uses the relation

$$x_{max} \rightarrow c_{max},$$
$$x_{min} \rightarrow c_{min}$$

and assumes that $c_{max} \geq c_{min}$. The other possibility of control reads as

$$x_{max} \rightarrow c_{min},$$
$$x_{min} \rightarrow c_{max}.$$

To use the same implementation as in the first case, we invert the control values. This is applied in the *I_ Control_ inv*-model which passes the inverse values $1/c_{max}$ and $1/c_{min}$ to the external function and inverts the return value as well.

6.7. Compartment, Wall and Air Pad

These Units all have a very similar model structure. We assume that no mass is transported. For the Wall-Units this is evident because the material is solid. For the Air Pad-Units, situated between the condenser and insulation of the compartment, only heat transfer is assumed as if the air would not flow there. The Compartment-Unit is modelled as one single volume. If the door is closed, no mass is transferred as well. Hence, in the energy balance only the heat transfer and the temperature evolution remain. This yields

$$\rho c V \frac{dT}{dt} = \sum_{i=1}^{n} \dot{Q}_i \tag{6.83}$$

where $\dot{Q}_i$ is the heat flux in the i-th *HeatFlux*-Connection. Furthermore, the surface area and the heat transfer coefficient are passed on to each Connection. In figure III.16 a schematic arrangement is depicted where the Compartment is surrounded by Wall-Units which represent either the Compartment wall or the insulation. The type of the Wall-Unit is specified by the physical properties prescribed in the corresponding Wall-Global.

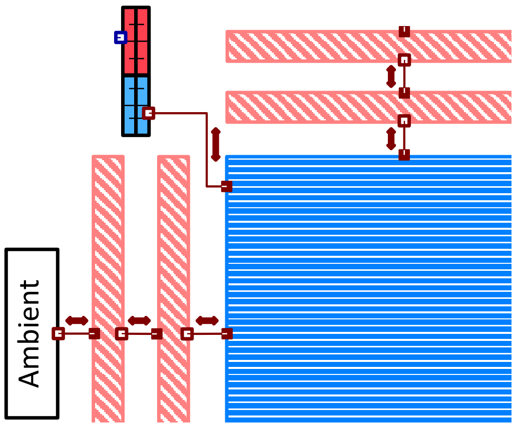


Figure III.16.: Schematic flow sheet arrangement of the Compartment (horizontally striped, light blue) and Wall (diagonally striped, pink) Units. Only HeatFlux-Connections (dark red) appear. Arrows indicate the expected direction of the flux.

6.8. Geometry

This Unit was added to the library to calculate geometrical properties. If a spatially discretized component is set up in a flow sheet, often multiple Units have the same geometry. Still, the finite volume approach allows different geometries for each Unit. In the presented models, a one-dimensional discretization is applied to the heat exchangers with volumes of the same size. A Global which holds the geometrical properties ensures that the same values are used by each Unit. Since in MDK a Global can not have multiple mathematical models, this Unit calculates the properties for various geometries and passes them to a Global.

One variable which has to be highlighted is the *radius_ tube* of the *Geometry*-Global. For numerous empirical correlations used in the heat exchanger Units a radius or diameter has to be given. If the geometry differs from a cylindric tube, the hydraulic diameter d_H is used instead. This generalization of the tube diameter is defined as

$$d_H = \frac{4A}{U} \tag{6.84}$$

where A is the cross-sectional area and U the perimeter. For turbulent flow the hydraulic diameter provides a good approximation to calculate various quantities, see [109]. For laminar flow the empirical correlations can differ from the measurements which is caused by the hydraulic diameter, see [120]. Since all formulas in the *HTX.dll* use a radius, the variable *radius_ tube* is half the hydraulic diameter. Note that this is not the hydraulic radius r_H which is defined as $r_H = d_H/4$.

7. Dynamic Link Library: HTX.dll

Within this section the *HTX.dll* is described. It was implemented to outsource critical calculations and is written in C++. A DLL can export functions which can be called outside. These external functions are incorporated in the models in MDK and executed during the simulation. We present the interface first and go into detail afterwards.

7.1. The Interface

We start with the *C*-interface which consists of the functions that can be accessed from outside the DLL. These external functions have to be declared within the `extern "C"` scope and provided with the `HTX_API` prefix which is defined in the *HTX.h*-file. This prefix simplifies the exporting function definition.

For MDK, external functions have to accept and return `double`-values. The input arguments are set in the external function declaration in MDK and have to coincide with the argument list of the DLL-interface. As last input parameter a message handler of type `HANDLERPROC` is passed which is a function pointer defined as `typedef void (CALLBACK* HANDLERPROC)(int, const char*)`. The message handler enables the DLL to write messages into the protocol of PSE. The first argument declares the type of the message and the second is the message itself. Possible values for the first argument are 0, 1 and 2 which stand for an error, warning or standard message, respectively. In the *DLL_ Basics.h*-file the enumeration declaration `EMsgType` is defined. These enumeration declarations shall be used whenever a message is written to the protocol.

An excerpt of the most important external functions used in MDK is given in table III.2. The prefixes `Co` and `Ev` stand for evaluations which are intended for the condenser and evaporator, respectively. Furthermore, for each input argument the partial derivative of the function with respect to the argument has to be provided as well. All partial derivative functions us the same naming convention as follows. Let *FuncName* be the function name and *VarName* the argument name. Then the partial derivative function is denoted as `d`***`FuncName`***`_d`***`VarName`***.

Each calculation is implemented in a separate class, see table III.2. Therefore, the external functions are brief and always follow the same syntax. The prototype of an external function reads as

```
1  double HTX_API ExFunc (...)
2  {
3      AFX_MANAGE_STATE(AfxGetStaticModuleState());
4
5      double RetVal = -1;
6
7      ClassName theClass(...);
8
9      if ( theClass.Continue() )
10         RetVal = theClass.PerformCalculation();
11
12     return RetVal;
13 }
```

Line 3 has to be added as a very first command to all external functions. This command guarantees that the correct resource handle is used. In line 7 an instance of a class is generated, which performs the desired calculation. The parameters of the constructor are all the variables which are necessary to obtain a result. Within every constructor the method `CheckInputData()` is called which verifies whether the input parameters are within the feasible range. If any parameter is outside the range, `Continue()` returns false and no calculation will take place. Furthermore, an error message will be printed to the protocol and Newton's method aborts. The actual calculation will be done in line 10. If the external function is a derivative, this line is substituted by

```
10         RetVal = theClass.NumericalDifferentiation(&theClass.VarName, EVarName);
```

where `&theClass.VarName` is a pointer to the public member variable assigned in the constructor with the corresponding input argument. The second argument is an enumeration declaration corresponding to the type of variable. Since the magnitude of the variables can differ significantly, different step sizes for the numerical differential formulas are chosen. The step size is selected according to the enumeration declaration.

7.2. Class Hierarchy

The actual calculations are wrapped in classes. An excerpt of the class hierarchy is illustrated in figure III.17. Since several similarities between the calculations become apparent when taking a closer look, an object-oriented programming approach was chosen. The aim is to reduce the number of duplicated code lines by a sophisticated structure of classes and polymorphism. The prototype code fragment for external functions and derivatives shows in line 9 and 10 already two methods which are common.

The `DllClass`-class constitutes the root for almost all classes used in the *HTX.dll* and there the two above mentioned methods are declared. The `PerformCalculation()`-method serves as a wrapper with a try-catch-block for the abstract `_PerformCalculation()`-method which is implemented by derived classes. The necessity of this structure emerges from exception handling since any exceptions thrown in a DLL have to be catched and handled there. Otherwise the DLL causes PSE to crash. From this class the `DllBaseClass`-, `Bdry`- and `I_Control`-class are derived. The last is described in section 7.3.

Table III.2.: Excerpt of the most important external function names and corresponding classes.

Function Name	Class	Return Value
`Co_EpsilonCal`	`Co_Epsilon`	Vapor void fraction ε by the LMε model
`Co_FlowPatternCal`	`Co_FlowPattern`	Integer of flow pattern by [36]
`Co_PressureDropCal`	`Co_PressureDrop`	FPD Δp by [90]
`Co_HTCCal`	`Co_HTC`	HTC α by [103]
`Ev_EpsilonCal`	`Ev_Epsilon`	Vapor void fraction ε by the Rouhani–Axelsson model
`Ev_FlowPatternCal`	`Ev_FlowPattern`	Integer of flow pattern by [112]
`Ev_PressureDropCal`	`Ev_PressureDrop`	FPD Δp by [90]
`Ev_HTCCal`	`Ev_HTC`	HTC α by [113]
`Sp_PressureDropCal`	`Sp_PressureDrop`	FPD Δp for a single phase fluid
`Sp_HTCCal`	`Sp_HTC`	HTC α for a single phase fluid
`CapMflowCal`	`Capillary`	Mass flow $\dot{m}_{drain}$ through the capillary
`CaphHx_Cal`	`Capillary`	Exit Enthalpy $h_{drainSuc}$ of internal heat exchanger
`IC_Cal`	`I_Control`	Controlled value c^{n+1} of I_Control Unit
`IC_EventTime`	`I_Control`	Time t^* of last (de)activation event of I_Control Unit

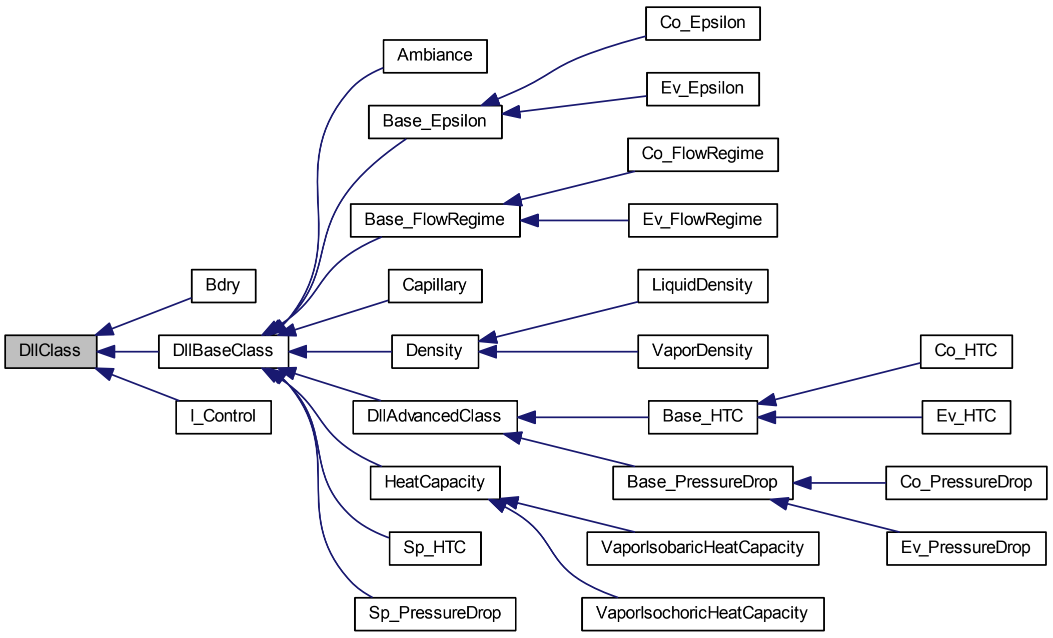


Figure III.17.: Excerpt of the class hierarchy in the *HTX.dll*. Only the most important classes are depicted. This illustration was automatically generated by the program Doxygen.

On the one hand, the `DllBaseClass`-class extends its base class by the framework for calculating physical properties. Therefore, further member variables are a pointer to an object of the `REFPROP`-class, described in section 7.3, and the fluid ID, which corresponds to working fluids available in PSE. The `m_bCalculatePP`-variable is used to decide whether physical properties have to be calculated or not. Usually, an object obtains pressure and temperature and calculates the appropriate properties in the

`CalculatePP()`-method but in some cases these properties are already available. To minimize the computational effort, in these cases the properties are passed directly to the constructor. Any other derived class is not mentioned in detail in this thesis.

On the other hand, the `DllBaseClass`-class can print messages to the protocol of PSE. The standard scheme to print a message is to first allocate a pointer to a `std::string` representing the message, then call the `AddMessage(...)`-method and finally the `SendMessage(...)`-method. In the last method the `std::string`-pointer is deallocated. All possible output strings are stored in the *HTX.rc*-file. Each string has a unique ID. The ID of the desired string has to be passed to `AddMessage(...)` which first uses the `LoadResourceString(...)`-function defined in *DLL_Basics.h* to access the correct string in the *HTX.rc*-file and second appends it to the `std::string` from above.

From the `DllBaseClass`-class several classes are derived. First, the `SpHTC`- and `SpPressureDrop`-class calculate the HTC and FPD for single-phase working fluids, given in section 6.1.1. Second, the `Capillary`-class serves as a wrapper class for the `ANN`-object, described in section 7.3. Third, the `DLLAdvancedClass`-class is the base class for the HTC and FPD calculation in the single and two-phase region. A more detailed description will be given below. And finally the `Base_Epsilon`- and `Base_FlowRegime`-class are the base classes for the vapor void fraction and flow pattern calculation, respectively.

Four major calculations are available for both, Condenser- and Evaporator-Units, which cover the whole physical range below the critical pressure. The four calculations comprise the vapor void fraction, the flow pattern, the HTC and the FPD. If the working fluid is single-phase, the calculations for both Units are equal and in the two-phase region at least the calculation scheme coincides. Therefore, for each calculation a base class with prefix `Base` and two derived classes with prefix `Co` and `Ev` corresponding to Condenser and Evaporator, respectively, are set up. The base classes are `Base_Epsilon`, `Base_FlowRegime`, `Base_HTC` and `Base_PressureDrop`.

Since the calculations for the condenser and evaporator equal in the single-phase region, the `Base`-class calculates the values there and performs the interpolation at the boundary of the two-phase region. Thus, `_PerformCalculation()` is always implemented in the `Base`-class. In the `Base`-class also common member variables are declared. A constructor is provided which assigns these members and `CheckInputData()` is implemented for the assigned variables. The `CalculatePP()`-method is implemented as well. For the two-phase region abstract methods are declared, *e.g.* the `EpsilonCal()`-method in the `Epsilon`-classes. There, the vapor void fraction as described in section 6.1.2 is calculated.

The `FlowRegime`-classes split the two-phase calculation into two parts. First, the transition curves are evaluated for the current state in `SetUpTransitionCurves()` and second, `DeterminePattern(...)` decides which pattern is apparent with respect to the mass flow per area G and flow quality $\dot{x}$. In the HTC and FPD calculations discontinuities can occur between the models used for each flow pattern. To avoid problems in Newton's method, linear interpolation is applied along the G-axis if the state is close to a pattern change. For this, the `FlowPatternCalculation(...)`-method calculates the current flow pattern and evaluates in `CheckInterpolationArea(...)` if an interpolation is necessary and which patterns are involved. The required information is stored in an `InterpolationInfo`-object. Since the flow pattern maps are different for the HTC and FPD calculation in the Condenser, the purpose `EInterpPurpose` is passed as well to gain appropriate results.

The `DLLAdvancedClass`-class serves as base class for the `HTC`- and `PressureDrop`-classes. The interpolation between flow patterns is implemented in the `DLLAdvancedClass`-class since it holds for any derived classes. The `PatternInterp()`-method uses the above mentioned `InterpolationInfo`-object to interpolate between values gained from the abstract `SinglePatternEval(...)`-method. In the latter method the correlations for the flow patterns are evaluated. This method is implemented in the `Co/Ev_HTC` and `Co/Ev_PressureDrop`-classes.

Each of these four classes uses a `FlowRegime`-object. For the sake of presentation we explain the following ideas for the `Co_HTC`-side only but they hold for all other cases as well. The `FlowRegime`-object is used in methods of the `Base_HTC`- and `Co_HTC`-class. Since in the `Base_HTC`-class it is not specified if the calculation is used for a Condenser or an Evaporator, a standard object-oriented technique is applied. In the `DLLAdvancedClass`-class a pointer to a `Base_FlowRegime`-object is declared, *i.e.* `Base_FlowRegime* m_ptheFlowPattern`. Additionally, the abstract `InitFlowPatternObj()`-method is declared but implemented in the `Co_HTC`-class. There the pointer is allocated with `new Co_FlowRegime(...)`. Hence, the appropriate `FlowRegime`-object is used always.

7.3. Long-living Objects

Principally, objects exist only during the external function call but in some situations it is convenient to have objects which live longer. When the *HTX.dll* is loaded by another programm the code written in the *HTX.cpp*-file will be executed. There, the global object `CHTXApp theApp` is allocated. This object is deallocated only when the *HTX.dll* is released. Although it is global, we access it always with `(CHTXApp*) AfxGetApp()`. All other objects which are supposed to live longer than an external function call are member variables of the `CHTXApp`-class and getter-methods are available to access them. The `ANN`-, `REFPROP`- and `WaterGlycol`-member objects are given as pointers and are initialized with `NULL` in the `CHTXApp`-constructor. Only if a function requires one of these objects, an instance will be allocated. Since these objects remain equal for every caller, this procedure is sufficient. If a model uses multiple I_Control-Units, a different approach has to be chosen. The `I_Control`-class handles the calculation for one I_Control-Unit. For every I_Control-Unit one object has to be allocated to avoid accidental overwriting of crucial data. Therefore, a `std::vector<I_Control*>`-member is added to the `CHTXApp`-class and every I_Control Unit has to be provided with a unique index in PSE. This index allows to access the corresponding entry in the vector which is resized individually to suit the largest index.

A brief overview of the above mentioned classes shall be given. First, the `ANN`-object is essential for the calculations of the capillary. The coefficients of the ANN are read from several text files. To save computational time, these files are read only once and stored in the `ANN`-object. Also the evaluation is handled by this object. The `Capillary`-object was introduced such that the external functions for the capillary fit the prototype and thus serves as a wrapper.

Second, the `REFPROP`- and `WaterGlycol` objects are necessary to calculate physical properties. The `WaterGlycol`-object is allocated only if the working fluid is water-glycol. This object reads tables from files once and interpolates (linearly) between the given points. There, the physical properties only depend on the temperature and not a second thermodynamical state value. Only enthalpy, density, heat conductivity, dynamic viscosity and isobaric heat capacity are provided. Any other working fluid is handled by the `REFPROP`-class. When the *HTX.dll* is loaded, the `CHTXApp::InitInstance()`-method is called. There, the *PPORCSys.dll* is loaded. The `DLLHandle`-object gives access to this DLL and has to be passed to the `REFPROP`-constructor where all necessary function pointers are allocated. The *PPORCSys.dll* is released in the `CHTXApp`-destructor which is called when the *HTX.dll* is released.

The last long living object is an instance of the `I_Control`-class. A crucial element in this object is the simulation time provided by DASSL. Since DASSL uses an adaptive time stepping scheme which can reduce the predicted time step and repeat the integration step, we have to compare the current simulation time t_c with the last time t_l the object was called. If $t_c > t_l$, DASSL accepted the last time step. We store the time t_l and other relevant variables which we refer to as valid variables since they will not change anymore. If $t_c < t_l$, the last predicted time step failed and DASSL reduced the step size. Hence the variables of the last valid time have to be restored. Afterwards the current values at time t_c are calculated. This strategy has to be applied to all models involving the simulation time.

8. Numerical Issues

The calculations performed in the *HTX.dll* are partially very complex and the devil is in the details. For each flow pattern a different set of equations and variables is used to calculate the HTC or FPD. Since the sets are not of the same size, it would have been hard to implement these models in MDK itself. Smooth extensions must have been found for the sets in regions where they are idle. Therefore, these complex calculations have been outsourced. In order to find a solution at each integration step, at least continuous functionals have to be given and numerous issues appeared during the development process.

8.1. Reasons for the Rejection of an Integration Step

In this section we discuss the two reasons why DASSL may not accept an integration step repeatedly and the simulation terminates prematurely. The reasons are that, first, Newton's method failed to converge or second, the integration error tolerance is not fulfilled. We start with the first reason. Let $F(t, x, x') : \mathbb{R}^{2n+1} \to \mathbb{R}^n$ be the dynamic system with $t \in \mathbb{R}$ and $x \in \mathbb{R}^n$. Then the derivatives with respect to time are replaced by a BDF-scheme at each time $t_{k+1} = t_k + h$, see section II.1.2. This yields a (nonlinear) $n \times n$-system which is solved by Newton's method. The iteration matrix has the special form

$$J_x(F)(t_{k+1}, x) = \left(\frac{\partial F}{\partial x} + \frac{\alpha_d}{h} \frac{\partial F}{\partial x'} \right) (t_{k+1}, x). \tag{8.1}$$

If in any equation of F no derivative appears, the second term in the Jacobian is singular. Consequently, for decreasing step sizes h the iteration matrix becomes ill-conditioned.

Theorem II.1.29 on the convergence of Newton's method requires that the system F has to be continuously differentiable and the starting point of the iteration has to lie in an ε-neighbourhood of the solution. Experience shows that convergence can mostly also be achieved if F is only continuous. The adaptive time stepping algorithm in DASSL reduces the step size h if Newton's method does not converge. Hence, it can be assumed that the second requirement for convergence is fulfilled at the latest after a finite number of step size reductions. In particular, for every step size we have a different starting point, iteration matrix and another root. Therefore, the adaptive time stepping increases the chance of convergence for models with so many non-differentiabilities and hidden discontinuities as in the developed refrigeration appliance model.

The convergence of Newton's method is also influenced by the accuracy of the partial derivatives. The partial derivatives for equations written with the MDL are given analytically. For external functions numerical derivatives have to be provided which are not exact in the *HTX.dll*. Therein, the derivatives are approximated by a central difference scheme. Since the magnitude of the variables is significantly different, an appropriate step size is chosen for each variable. In general, no inner derivatives have to be considered when numerical derivatives are evaluated. Since all models in the *HTX.dll* are established for positiv mass flow and heat flux, in the constructor the absolute values are assigned. Evaluating the derivative is a method of each class and is called after the constructor. Consequently, the inner derivative must be taken into account implying that the numerical derivative has to be multiplied with the signum of the respective variable.

The second reason why an integration step is rejected becomes apparent when Newton's method converges but the integration error tolerance is not fulfilled. In DASSL a predictor-corrector-strategy is implemented. In the predictor-stage, the new state is predicted by polynomial extrapolation with the same order as the chosen BDF-method. In the corrector-stage, the system of equations is solved where the predicted state serves as the starting point of Newton's method. Let $x_p, x_c \in \mathbb{R}^n$ be the predicted and corrected state, respectively. Then the integration error $\mathrm{E}(I)$ is bounded from above by

$$\mathrm{E}(I) \leq \|x_p - x_c\|_2 < \mathrm{tol}(I), \tag{8.2}$$

see [11], where $\mathrm{tol}(I)$ denotes the integration error tolerance provided by the user. This upper bound is used to decide whether an integration step is accepted or not. If the functions are continuous the distance decreases with the step size. Otherwise, $\|x_p - x_c\|_2$ may remain larger than the tolerance for any time step although Newton's method converges. To overcome this problem, the *Relax Tolerance Control*-option was added to IPSEpro. This enables DASSL to set $\mathrm{tol}(I) = \infty$ if the step size is less than a given threshold. Since any solution x_c is accepted then, the threshold has to be chosen carefully to guarantee reliable results.

Concluding, we expect the simulation to be successful if the flow sheet model, represented by F, is continuously differentiable. Since this is not the case in several refrigeration cycle Units, problems with the simulation and hence premature termination emerged. Since a model is only usable if the simulation can be performed successfully for a sufficiently large domain of parameters, these problems were investigated and the convergence behavior was improved significantly. Most difficulties appeared in the heat exchanger models, Condenser and Evaporator. In the following we discuss several discontinuities and non-differentiabilities existing in the Unit models.

8.2. Functional Discontinuities & Non-Differentiabilities

We start with three external functions, `Co_FlowPatternCal`, `Ev_FlowPatternCal` and `IC_EventTime`, which return discrete quantities by definition. The first two functions return the integer of the current flow pattern which is a natural number between 1 and 9. These functions are used in the heat exchanger Units such that the current flow pattern can be seen by the user. The last function yields the last time at which the respective `I_Control`-Unit switched. Since all these functions and the variables to which the return value is assigned appear in 1×1, post calculation blocks, see section II.4.1, the discontinuities in the variables do not influence the remaining model. Still, these functions cause a premature termination, if the Relax Tolerance Control-option is not activated.

A discontinuity which emerges from physical properties is not as obvious as the above ones. It occurs at the lower two-phase boundary and has its origin in the density of the working fluid. At this boundary the density is not continuously differentiable, see figure III.18.

Due to the finite volume approach in the heat exchanger Units, the volume V is fixed which yields that the mass m related to the density by

$$m = \rho V \tag{8.3}$$

is not continuously differentiable at the lower two-phase boundary as well. Thus, the derivative dm/dt which appears in the conservation law of mass, *i.e.*

$$\frac{dm}{dt} = \dot{m}_{feed} - \dot{m}_{drain}, \tag{8.4}$$

is not continuous. Therefore, a discontinuity in the mass flows $\dot{m}_{feed}$ and $\dot{m}_{drain}$ at the interface, *i.e.* the Connection, of the finite volume emerges. Since the mass flow appears in several equations such as the conservation of energy and the external functions of contiguous heat exchanger Units calculating ε, Δp and α, these equations are not continuous as well. Consequently, the non-differentiability of the density has an impact on the whole model. On the one hand, sufficiently small finite volumes have

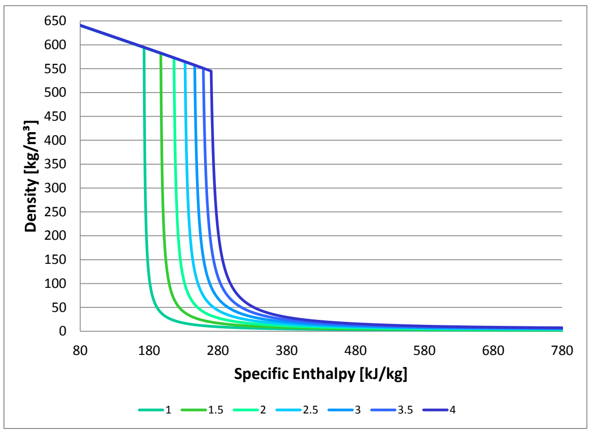


Figure III.18.: Density of Isobutane (R600a) for various pressures from 1 bar to 4 bar.

to be chosen to guarantee convergence. On the other hand, the *Relax Tolerance Control*-option has proved to be useful in this case if the integration error tolerance can not be satisfied.

Next we discuss the calculation of the density ρ_l on the lower two-phase boundary. This quantity is needed in equation (6.21) in which the density of the working fluid is determined. Since not all functions from Refprop are available in this MDK-library, the equation for density of the liquid share was written as

$$\rho_l = \begin{cases} \rho(p,h), & \text{if } \dot{x} \leq 0, \\ \rho(p,h_l), & \text{else} \end{cases} \tag{8.5}$$

where $h_l = h(p, x = 0)$ is the specific enthalpy at the lower two-phase boundary. Thus, the right hand side depends on four variables which are iterated as well. If $\dot{x}$ is close to zero, the above formulation leads to periodic iterations in Newton's method with a length of up to six iterations until the first state repeats. This convergence issue was resolved by implementing an external function which only depends on p and h, *i.e.*

$$\rho_l = \mathit{LiquidDensityCal}(p,h). \tag{8.6}$$

In this function, the specific enthalpy h_l is calculated from the pressure and the minimum of h and h_l is passed to the Refprop-function $\rho(p,h)$.

Next, the physical properties λ, c_p, c_v and η occur in the models for the HTC and FPD applied in the heat exchanger Units. These properties are defined in the single phase region only and are not continuous in the saturation temperature T_{sat} for any pressure less than the critical pressure. They are calculated from pressure and temperature T and even a slight deviation from T_{sat} yields either values for the liquid or vapor phase. Since the solution of Newton's method is not exact, $T < T_{sat}$ can hold although the state is in the vapor phase and vice versa. This would yield wrong values for these properties and hence wrong results of any further calculation. To prevent errors, the `CalculatePP()`-method in the *HTX.dll* also regards the flow quality $\dot{x}$ when these properties are calculated. *E.g.* if $\dot{x} > 1$ then the maximum of T and $T_{sat} + \delta$, where $\delta > 0$ is a small offset, is passed to the Refprop functions. Additionally, if $0 < \dot{x} < 1$ holds, the properties have to be calculated at the lower and upper two phase boundary. To achieve a continuous function, the properties are then evaluated at $T_{sat} \pm \delta$.

Further discontinuities occur in the HTC and the FPD calculation at the two-phase boundary and at almost every flow pattern change. To ensure continuous external functions, linear interpolation was applied wherever necessary. Here, one has to mind that the boundary curves of the flow patterns may intersect in certain cases. The non-differentiable points due to the interpolation do not have such a big impact on the convergence as those for the density calculation and experience shows that Newton's method still converges.

The last discontinuity mentioned emerged in the Compressor-Unit if the inlet is close to or inside the two-phase region. Such situations may occur in the simulation when the compressor is turning on. Of course, in reality this has to be prevented since the compressor can be damaged. Nonetheless, in equation (6.52) the physical properties c_p and c_v are calculated from pressure and temperature at the inlet. As mentioned above, these properties have a discontinuity in T_{sat}. No interpolation is applied, but the maximum of $T_{sat} + \delta$ and T is chosen always.

8.3. Limitation of the two-phase Heat Transfer Coefficient

Since at the two-phase boundary very steep gradients in the HTC occur, the HTC was bounded. As a consequence, we expect increased numerical robustness since the initial value x_p for Newton's method is closer to the solution x_c and the integration error tolerance will probably be satisfied with a larger time step (recall equation (8.2)). To find a good choice for an upper bound we consider the thermal transmittance k through the pipe wall of a heat exchanger, *i.e.*

$$k = \frac{1}{\frac{1}{\alpha_i} + \frac{d}{\lambda} + \frac{1}{\alpha_o}} \tag{8.7}$$

where α_i is the HTC between the working fluid and the pipe wall, d the diameter and λ the thermal conductivity of the pipe wall, and α_o the HTC between the pipe wall and the ambiance. Each term in the denominator is referred to as thermal resistance. In our case the pipe wall is usually very thin such that the middle term is negligible. Division of the above equation by α_o yields

$$\frac{k}{\alpha_o} = \frac{1}{1 + \frac{\alpha_o}{\alpha_i}}. \tag{8.8}$$

The limit for $\alpha_i \to \infty$ is given by

$$\lim_{\alpha_i \to \infty} \frac{1}{1 + \frac{\alpha_o}{\alpha_i}} = 1. \tag{8.9}$$

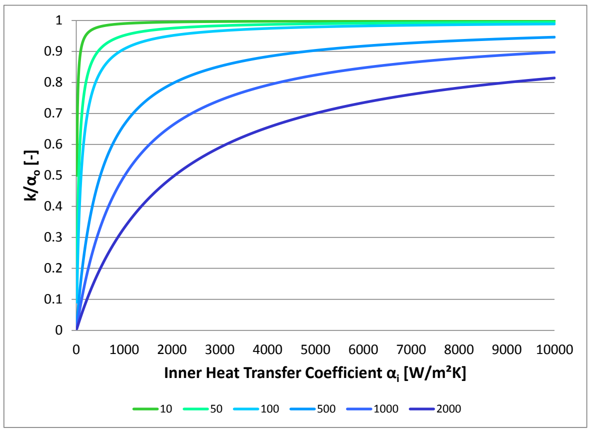
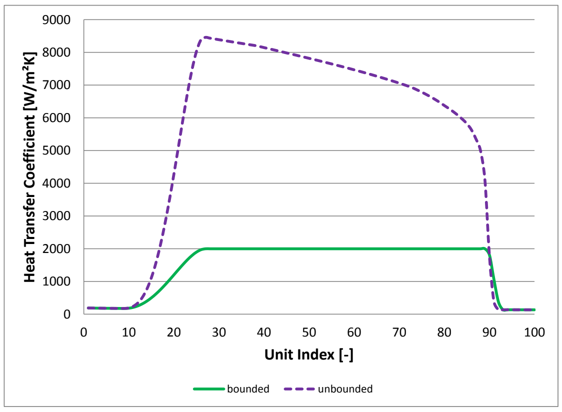


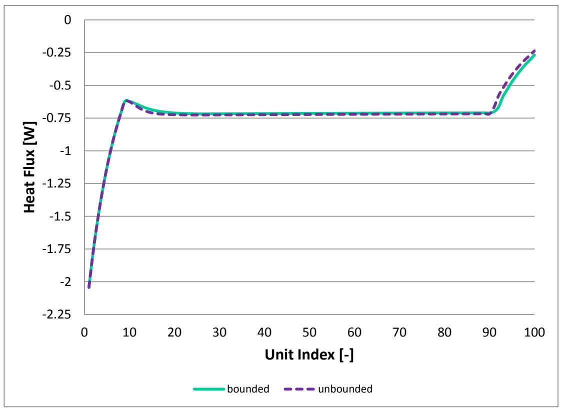
Figure III.19.: k/α_o over inner HTC α_i for various α_o from $10\,\mathrm{W/(m^2\,K)}$ to $2000\,\mathrm{W/(m^2\,K)}$.

Hence, the thermal transmittance is governed by the largest thermal resistance. In figure III.19, several graphs of the dimensionless number k/α_o are illustrated. In the refrigeration cycle, α_o is supposed to be less than $50\,\mathrm{W/(m^2\,K)}$ which implies that no perceptible changes take place in the thermal transmittance for $\alpha_i > 2000\,\mathrm{W/(m^2\,K)}$. Thus, α_i was bounded with $2000\,\mathrm{W/(m^2\,K)}$.

In figure III.20 the bounded and unbounded inner HTCs and the resulting heat fluxes $\dot{Q}$ in the condenser are illustrated for a steady-state calculation in which the compressor is turned on. The thermal resistance of the pipe wall is set constant with $d = 7 \times 10^{-4}\,\mathrm{m}$ and $\lambda = 50\,\mathrm{W/(m\,K)}$ and the ambient HTC has a constant value of $22.5\,\mathrm{W/(m^2\,K)}$. The bound of $2000\,\mathrm{W/(m^2\,K)}$ has a huge impact on the calculated HTCs but the heat flux $\dot{Q}$ remains almost equal. The summed up heat fluxes yield $-73.00\,\mathrm{W}$ and $-73.27\,\mathrm{W}$ for bounded and unbounded HTC, respectively. Figure V.27 shows that the HTC in the evaporator does not exceed $2000\,\mathrm{W/(m^2\,K)}$ during the simulation.



(a) HTC.



(b) $\dot{Q}$ for bounded and unbounded HTC.

Figure III.20.: HTC and heat flux $\dot{Q}$ over Condenser-Units in a steady-state point.

8.4. Spatial Discretization

Next we discuss the one dimensional spatial discretization of the heat exchangers. The aim is to find a good balance between accuracy, stability and numerical costs. On the one hand, the discretization error is reduced as the number of finite volumes for a fixed total length is increased. On the other hand, the increase of finite volumes leads to a larger system which has to be solved countless times and thus the numerical costs grow as well. Since the HTC and FPD depend nonlinearly on the state of the working fluid and change rapidly, an analytic derivation of error estimates seems to be impossible. Thus, numerical experiments are performed instead. Let m be the number of finite volumes and let x be a variable. The exact solution is not known, but from above considerations we deduce that

$$\lim_{k\to\infty} |x^{m+1} - x^m| = 0 \tag{8.10}$$

has to hold for all variables x. The different numbers of volumes yield different spatial positions of most of the variables which makes it impossible to compare them. Only the positions of inlet and outlet remain equal for all m. In the numerical experiments the Condenser-Unit is used. At the inlet the pressure and enthalpy are prescribed and at the outlet observed. From these two all other thermodynamical properties can be obtained. The mass flow is set constant and only steady-state calculations are performed. Furthermore, the HTC between the working fluid and the pipe wall is not bounded to avoid distortion of the results.

In figure III.21 the results are illustrated. The outlet pressure reaches a limit at $m = 50$ which corresponds to a single volume length of 0.4 m. The difference of the outlet enthalpy between 50 and 100 volumes, is 1.4 kJ/kg and between 70 and 100 volumes 0.6 kJ/kg. If the temperature of the condenser approaches the ambient temperature and hence no heat is exchanged anymore, the error of the outlet enthalpy diminishes.

Furthermore, we expect to have a larger error in the single phase region since the temperature of the working fluid changes. In the two-phase region where the temperature remains equal an error can only result from an inaccurate HTC. In the refrigeration appliance model the working fluid of the evaporator is expected to be almost entirely in the two-phase region. Therefore, longer volumes are chosen. At the condenser inlet single phase working fluid is assumed and a tighter spatial discretization is applied. Although in the refrigeration appliance model an equidistant discretization is chosen, higher accuracy is expected if the Condenser-Units in the beginning, where the working fluid is still single phase, are smaller. However, since the HTCs on the outside of the heat exchangers are fit with measurements, the discretization error can be compensated.

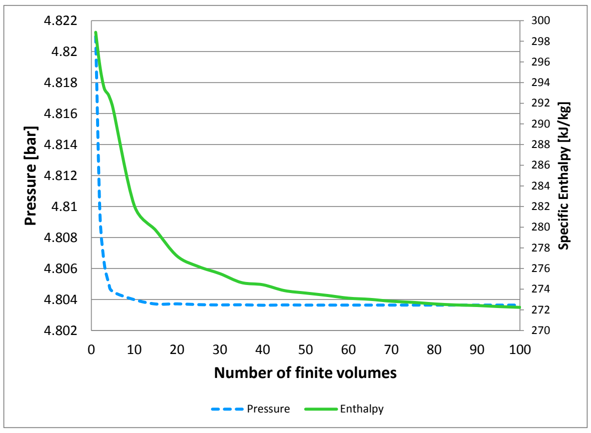


Figure III.21.: Enthalpy and pressure at condenser outlet over number of finite volumes. Steady-state calculation with mass flow 0.1877 g/s, enthalpy 661.33 kJ/kg and pressure 4.8249 bar at condenser inlet, constant total length of 20 m and ambient temperature 25.36 °C.

9. Flow Sheet Model in PSE

Concluding this chapter the flow sheet model set up in PSE is presented. The focus not only lies on the results but also on problems of the whole appliance and how to overcome them. The Echelon Analysis proved to be a useful tool even for experienced users. It helps when a new flow sheet model has to be established or an existing one has to be modified.

9.1. Setting up a Flow Sheet

In this thesis only a short guideline is given of how a flow sheet model is set up. For a more detailed explanation we refer to [82]. If a dynamic calculation is performed, at first the initialization system is solved, see section II.2.1. Solving this nonlinear system guarantees that the DAE solver does not have any problems at the initial time. If a dynamic model of a Unit is used, initial values have to be prescribed for the differential variables. In the heat exchanger Units these variables are the specific inner energy u, the mass m and the pipe temperature T_p. Since these values may be hard to guess, the steady-state models of these Units are used to find them. Then, the steady-state solver can be called. The boundary conditions remain equal and the initial values for u, m and T_p are gained from the steady-state model.

In IPSEpro, several *datasets* can be generated for the same flow sheet model. In each dataset the mathematical models of the Units and boundary conditions can be different. A hierarchy of the datasets, illustrated in the *Dataset Manager* in PSE, can be chosen by the user and the results of one dataset can be used for the start of the next. In the refrigeration appliance model, we use this framework, establish a steady-state model first and import the results into a dynamic dataset where the dynamic Unit-models are chosen. Thus, the derivatives of the differential variables can be prescribed with zero and the dynamic simulation starts from an equilibrium. Since periodic compressor on-off-cycles form after a few cycles, starting from an steady-state point can be recommended.

In principle, we built a steady-state flow sheet model as simple as possible and extend it step by step. Hence, dealing with the necessary settings of each Unit is straightforward and the chance of convergence of Newton's method is increased. In particular, often the Block Decomposition, presented in section II.2.2, decomposes the system of equations into smaller parts in simple flow sheet models. *E.g.* in a Source-Stream-Sink model, mostly the system can be decomposed into 1×1-blocks which usually have a very large neighbourhood of convergence. Such a simple model combined with an expected thermodynamic state is a good choice to start with since usually convergence is achieved easily. The PSE-command *Import Estimates* loads the results into the estimates of the variables which serve as the starting values for Newton's method. Consequently, if the calculation is repeated with the same model and same boundary conditions, the estimates equal the solution.

Next, a possible approach to obtain a steady-state flow sheet model of a refrigeration appliance is presented briefly. We start with the simplest model possible which consists of a Source, a Stream and a Sink. There, the composition of the working fluid has to be set in a Composition-Global and we choose the thermodynamical state such that it corresponds to the expected compressor outlet. To avoid problems with the two-phase region, the pressure p, enthalpy h and mass flow $\dot{m}$ are prescribed in the source. *Import Estimates* loads the solution into the estimates. Next the condenser is incorporated. The geometry and properties of the pipe wall have to be set and we extended the existing model by one Condenser-Unit. To have good estimates of the physical properties at the in- and outlet we duplicate the Connection instead of drawing a new one. This procedure increases the chance of convergence in the next calculations. If the system does not converge, rough estimates in the Condenser-Unit are often sufficient to obtain a solution. On the one hand, they can be gained from the Connection or on the other hand by experience. Furthermore, the length of the Unit can be decreased which reduces the difference between inlet and outlet state. Once the system converged, *Import Estimates* is applied again. Then the Unit and the Stream attached to the outlet are duplicated, incorporated between the first Condenser and the Sink and a new calculation is performed. This procedure is repeated until the desired number of Condenser-Units is obtained.

The whole compressor consists of three Units: the Compressor, the Shell and the Oil_Sump. The heat transfer coefficients, areas, the volume and the pressure drop have to be prescribed in the Shell, fit parameters, the NOR and the displacement in the Compressor and properties of the oil in the Oil_Sump. The Source is shifted from the Condenser to the Shell inlet, the settings have to be removed there and set, *e.g.* at the Stream between the Shell and the first Condenser. If all Stream-Connections have been duplicated, immediate convergence can be expected. Otherwise the new Units have to be provided with rough estimates since the default values may be poor.

Including the Capillary can be more demanding than adding the three Units of the compressor. In the first place, we recommend to build this Unit independently of the already established model and not to set the mass flow through the capillary or the enthalpy on the suction side outlet. Since these variables are output values of the ANN, invalid or physically impossible values could be prescribed. A possible choice of settings are the pressure and enthalpy at the inlet and the pressure at the outlet of the capillary as well as the pressure, enthalpy and mass flow at the inlet of the heat exchanger on the suction side. When a solution is obtained, the settings can be shifted in the desired direction such that the Capillary can be connected to the outlet of the last Condenser-Unit and Shell inlet. Connecting the Capillary reduces the amount of necessary settings. Hence, *e.g.* in an arbitrary Unit or Connection the mass flow and the pressures at the Capillary outlet and at the inlet of the heat exchanger on the suction side can be set.

The whole evaporator is incorporated as the condenser. If the pressure is prescribed at the capillary outlet as recommended above, the Evaporator-Units forms a separate block in the system of equations which is solved after the other Units. Thus, the established models are not influenced by the Evaporators. In the refrigeration appliance model, described in section 9.2, the evaporator is located in the insulation and exchanges heat with the compartment and ambiance. At this state of the model we connect the Evaporator to Ambient-Units. The temperature and HTC of the Ambient are chosen such that the cycle can be closed as easy as possible and afterwards are be replaced by values which allow the Ambient-Units to be exchanged with Wall- and Compartment-Units.

At this state of the flow sheet model, the working fluid streams from a Source arranged before the suction side of the Capillary through the internal heat exchanger and the Shell into the Compressor. Then it is discharged into the Condensers, expanded in the Capillary and streams through the Evaporators before it vanishes in a Sink. Closing the cycle can be a challenging task and often requires patience. The states of the "loose" ends have to be brought close such that Newton's method converges. This is achieved by changing any settings until these states almost equal. The changes in the settings can be reversed after the cycle is closed. In a steady-state cycle model, a Connector-Unit has to be used to connect the loose ends. In the Connector the equations

$$p_{feed} = p_{drain}, \tag{9.1}$$
$$h_{feed} = h_{drain} \tag{9.2}$$

are defined. Thus, the states are transferred unmodified but the conservation of mass is decoupled. As described in example II.3.18 on page 39, a cycle-system without a connector would yield a singular iteration matrix. Therefore, one mass balance is removed which yields one degree of freedom. *E.g.* the mass flow can be set instead or, as it was done in the refrigeration appliance model, the total mass of refrigerant in the system can be prescribed. In the dynamic dataset the connector has to be removed again to avoid a leakage of mass. Since the flow sheet can not be changed the Connector has another model which is extended by the conservation of mass. If all derivatives dm/dt of the mass are initialized with zero in the dynamic dataset, a singular system emerges which implies that at least one mass has be initialized instead. Finally, when the cycle is closed, the Compartment and Insulation is built around the Evaporators before the Temperature Sensor and the I_Control complete the flow-sheet model. Setting up a steady-state model has another benefit. Since numerous fitting parameters are validated using steady-state measurements as described in chapter IV, a steady-state model has to be set up anyway. Hence the steady-state dataset serves as the basis for both, dynamic simulation and validation.

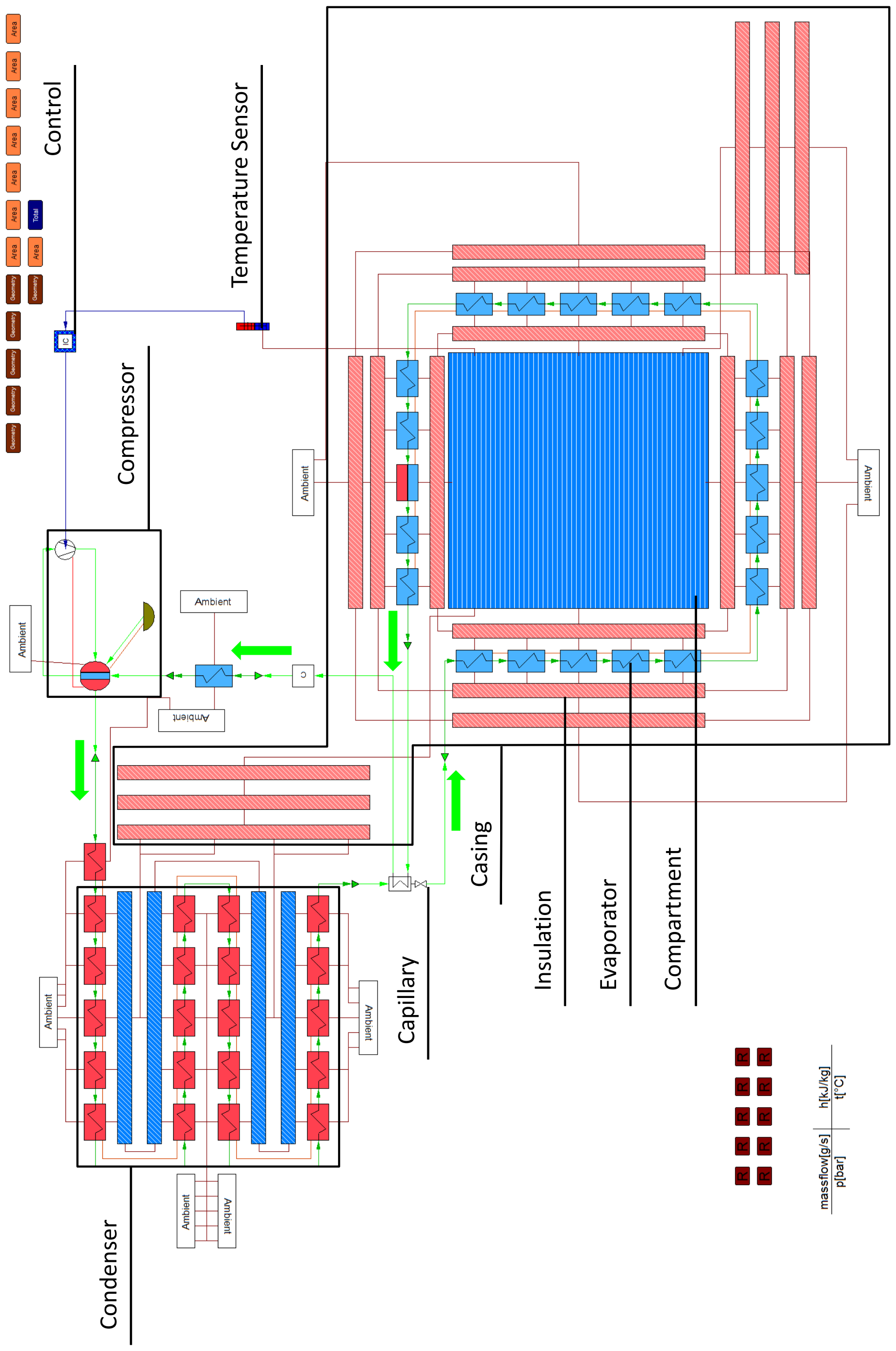


Figure III.22.: Flow sheet model of the refrigeration appliance.

9.2. Refrigeration Appliance Model

Within this section the flow sheet model of the refrigeration appliance is presented. The flow sheet can be seen in figure III.22. The four green arrows do not represent Units but illustrate the flow direction of the refrigerant. The three frames correspond to the compressor, condenser and casing. First, the compressor consists of Shell-, Compression- and Oil Sump-Unit. For the arrangement see also figure III.10 on page 61. Second, the condenser is connected on the one side to the ambiance and on the other side to Air Pads which themselves are connected to insulation at the back of the compartment. Between the second and the third row of Condenser-Units, the HeatFlux-Connections overlap such that only one dark red line is visible in the middle. Third, inside the casing we have the Compartment-Unit surrounded by six walls. In four walls the evaporator tubes are embedded. The remaining two walls correspond to the door and the rear. No door heating is modelled in IPSEpro but the condenser exchanges heat with the rear of the compartment which is not implemented in the VBA simulation tool. Differently from figure III.8 on page 59, the Accumulator-Unit is not situated at the very end of the evaporator but two Units before that. Always before and after one or more heat exchanger Units, green triangular Icons are visible. These triangular Units are referred to as Stream-Conversion and only used to convert a Stream- into a Stream_H-Connection and vice versa. The state does not change in these Units. Finally, the Units at the top right and bottom left are presented shortly. At the top right, the brown Icons correspond to the the Geometry-Units, the orange Icons are used to hold values of surface areas which appear multiple times and the blue Unit contains variables which are the sum of others, *e.g.* the total mass of refrigerant explained below. The dark red Units at the bottom left are needed to evaluate the penalty terms from section IV.1.1.1. Hence, they do not have any influence on the model but are necessary for the steady-state parameter fitting, explained in chapter IV

The flow sheet model was set up in a steady-state point. The degree of freedom emerging from the Connector-Unit was firstly compensated by setting the mass flow in an arbitrary Connection. Although this yielded convergence the desired total mass of refrigerant was by far not achieved. Consequently, the mass flow setting was replaced by a Free Equation prescribing the total mass m_{tot} in the system, *i.e.*

$$\sum_{i \in Components} m_i = m_{tot}. \tag{9.3}$$

The Free Equations are implemented in PSE with the syntax of the MDL. They are independent of any Component and all variables from the flow sheet can be referenced. Hence, the correct filling quantity of refrigerant was achieved.

To ensure a conservative model the mass and energy balance was validated for larger groups of Units. The conservation of energy was only verified in the initial, steady-state point and the residuum remained within the error tolerance. The total mass was prescribed by equation (9.3) for the initial state and monitored during the dynamic simulation. The difference between the maximal and minimal absolute deviation of the initially prescribed total mass is less than 0.012 %. In figure III.23, the relative deviation from the correct filling quantity is illustrated. The deviation shows a periodic behaviour. In comparison, the model developed in VBA gains 0.58 % of refrigerant in approximately 6.2 h simulation time.

It has to be mentioned that a real shutdown of the compressor could not be achieved since the system would become singular if the NOR are set to zero. Therefore, a residual NOR of 20 rpm was chosen. In this case the mass flow reduces to approximately 0.005 g/s and the masses in the condenser and evaporator remain almost equal over the whole period of 29 min in which the compressor is idle. Approximately 5 g of refrigerant are transferred from the Accumulator- to the Shell- and Oil Sump-Unit, see also section V.1. All values are taken from the validated model presented in chapter IV.

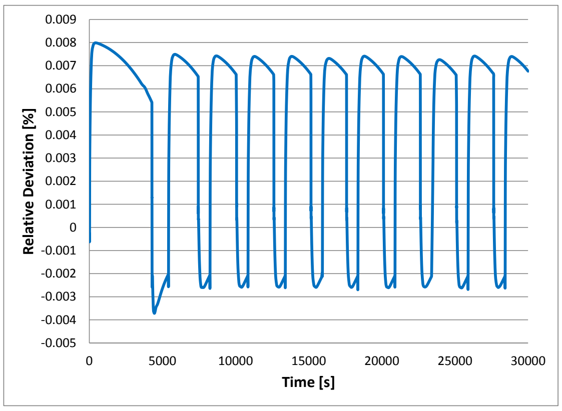


Figure III.23.: Relative deviation of the correct filling quantity of refrigerant.

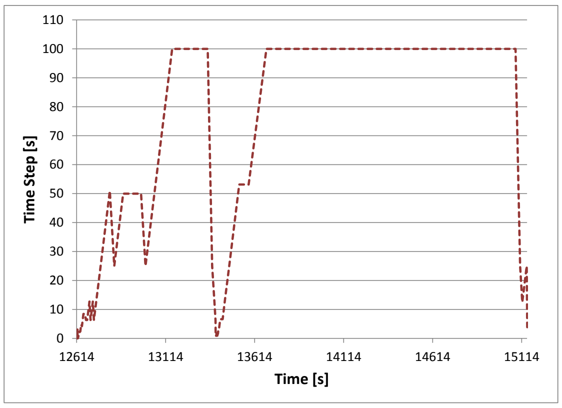


Figure III.24.: Step size selected by DASSL during one periodic on-off-cycle.

Next, shortly the chosen iteration parameters are explained. Experience shows that DASSL performs better if a tighter tolerance for Newton's method is chosen. The y-tolerance corresponding to ε in equation (1.50) on page 20 was set to 5×10^{-3}. The x-tolerance which is equivalent to δ in equation (1.50) on page 20 was set to 10^{-3}. The integration tolerance is calculated as the product of the x-tolerance and the integration tolerance correction factor which was set to 500. This factor can be reduced if a finer resolution in time is demanded. For simpler or problems with less variables we recommend to take a much smaller quantity, *e.g.* between 10 and 100. Generally, the smaller this factor is chosen the smaller the time steps will be. It is not recommended to choose a value below 10 but to decrease the x-tolerance if the time steps are too large. The initial step size was set to the default value, *i.e.* 0.1 s. The *Relax Tolerance Control*-option was enabled since, among others, the flow pattern number is a discrete integer quantity. This implies possible jumps which have to be allowed. Since flow pattern changes are expected to occur only while the compressor is turning on or off, smaller time steps can be assumed in advance anyway. Therefore we chose 0.1 s as the threshold value.

Finally, we give some information of the DAE system. The flow sheet model consists of 2706 algebraic and 180 differential variables, 197 algebraic variables are set, 27 differential and 153 derivatives of differential variables are initialized. This yields a system of size $n = 2689$. This number disregards the numerous variables and equations inside the *HTX.dll*. In the initialization system, the block decomposition found 1135 blocks and the largest one contains 1555 variables. In the dynamic system, we have 1114 blocks and the largest consists of 1576 variables and equations; 874 blocks are constant, 2 dynamic, and 238 of type post calculation. All constant and post calculation blocks have size 1×1. Since it is possible to initialize the states of all differential variables and the Jacobian is regular in the initial point, the system has index 1 and all initial values are consistent, see corollary 2.4 in chapter II.

The dynamic simulation is about 2 to 5 times faster than real time depending on the demanded accuracy. Here, especially the integration accuracy (integration tolerance correction factor) is crucial since it has a huge impact on the (time) step size. If the step size is chosen smaller, also the corrector equation has to be solved more often. Furthermore, the calculation takes longer if the compressor switches more often. When the compressor switches the step size, depicted in figure III.24, is reduced to a value below 0.5 s. When the NOR of the compressor do not change for a longer period, *i.e.* the simulation approaches a steady state, the step size is increased significantly by DASSL. In the simulation the step size is bounded by 100 s. If no limit is enforced, the step size would further increase when the system becomes steady state.

IV. Dynamic Validation

Once a model is set up and performs well for a sufficiently large domain of parameters, it has to be compared with measurements to obtain information of its quality. If the quality is not good enough, either the uncertain parameters or the model itself have to be reconsidered and potentially changed. This (possibly iterative) process is referred to as *model validation*, or simply validation. Since the compressor of the investigated domestic refrigeration appliance operates intermittently, the refrigeration cycle is never at steady-state. In this thesis, a case is considered in which the compartment door is never opened and the ambient temperature T_{amb} is constant. Thus, a (time) periodic behavior appears in the refrigeration appliance. This means that the on-off-duration of the compressor remains equal and the states repeat for each period of time. Therefore, we demand that the results of the dynamic model have to fit the measurements during this period which increases the challenge of the task. In this case, we speak of *dynamic validation* to emphasize the time dependence.

1. Nonlinear Regression

Assuming that the model is set up properly, the uncertain, time independent (constant) parameters have to be fitted such that the results approximate the time dependent (dynamic) measurements. This process is referred to as *nonlinear regression* or *parameter fitting*. Mathematical methods have been addressed to the problem of nonlinear regression and fall within the scope of optimization. For a comprehensive overview, we refer to [9, 97] and [65, 78] for unconstrained and constrained optimization, respectively.

Although there exist fully-automatic numerical methods to fit constant parameters to dynamic measurements, a different approach was chosen due to two reasons. First, no such method is available in IPSEpro so far and second, up to now an interface is missing which allows to extract the dynamic results automatically after a simulation has finished. Nevertheless, a fully-automatic method would be very time consuming. The simulation starts from a steady-state point, as described in the section III.9, and has to pass several compressor on-off-cycles until the periodic cycle is reached. In the measurements one cycle takes approximately 42 min and the simulation is 2 to 5 times faster than real time, depending on how often the compressor turns on or off and the demanded accuracy. This implies that the evaluation of the system for one single set of parameters may take up to several hours. Therefore, the nonlinear regression problem was split into a stationary and dynamic stage which are iterated.

A priori the parameters are partitioned into two sets. To the first set $\mathcal{P}_s$, parameters belong which can be fitted using steady-state measurements, *e.g.* heat transfer coefficients. In the second set $\mathcal{P}_d$ are those which have to be fitted with dynamic measurements, *e.g.* parameters having an impact on the thermal inertia. We assume that the model is asymptotically stable which means that if the boundary conditions are constant, the same states will occur in the system independent of the initial state after a sufficiently long time. Then the following strategy is pursued:

1. Fit all $p_s \in \mathcal{P}_s$ automatically and run the dynamic simulation.

2. Compare the dynamic results and measurements. If necessary, adjust $p_d \in \mathcal{P}_d$ manually and return to 1.

Before the automatic fit of $p_s \in \mathcal{P}_s$ is discussed, the measuring points and uncertain parameters of the refrigeration appliance model are presented. The measuring data are described in table IV.1. Steady-state and dynamic measurements were performed in a climatic chamber with two different ambient temperatures, 25 °C and 43 °C, and the thermostat was set to −18 °C. Only the measurements with 25 °C ambient temperature are taken to fit the parameters. For the steady-state data, the temperature sensor was switched off such that the compressor remains switched on. The NOR of the compressor are 3000 rpm. For a comprehensive description of the measurements and used instruments see [40].

The uncertain parameters are given in table IV.2. The condenser is modelled as a horizontal pipe. In reality the condenser has fins (cross-bracings) made of metal to increase the area and hence the heat flux. To compensate this, a factor for the heat flux η_{α_Co} is fitted. Due to the shape of the condenser and evaporator, we expect that the real pressure drop deviates from the one of the horizontal model and introduced the factors $\eta_{\Delta p\_Co}$ and $\eta_{\Delta p\_Ev}$. Since the heat fluxes inside the shell are poorly known, α_{Dis_Shell} is fitted. Further points of uncertainty are α_{Comp_in}, α_{Comp_out} and α_{Amb_Shell}. The above parameters can be fitted using steady-state measurements. Three dynamic parameters remain which are adjusted manually. All parameters can not be chosen arbitrarily but have to stay within physical limits. Hence, we have 10 parameters and 15 measurement values.

Table IV.1.: Measuring data.

Meas. data	Unit	Description
P	W	Electric power of compressor
t_{Shell}	°C	Surface temperature of shell
t_{Shell_out}	°C	Shell outlet pipe temperature
t_{Cond_in}	°C	Condenser inlet pipe temperature
t_{Cond_25}	°C	Condenser pipe temperature after 25% length
t_{Cond_50}	°C	Condenser pipe temperature after 50% length
t_{Cond_75}	°C	Condenser pipe temperature after 75% length
t_{Cond_out}	°C	Condenser outlet pipe temperature
t_{Cap_out}	°C	Capillary outlet pipe temperature
t_{Evap_out}	°C	Evaporator outlet pipe temperature
t_{Shell_in}	°C	Shell inlet pipe temperature
p_{Shell_in}	bar	Shell inlet pressure
p_{Shell_out}	bar	Shell outlet pressure
t_{Comp}	°C	Compartment temperature
$t_{Ambient}$	°C	Ambient temperature

Table IV.2.: Uncertain parameters.

	Parameter	Unit	Description
$\mathcal{P}_s$	α_{Iso_out}	$W/(m^2\,K)$	HTC between insulation and ambient
	α_{Comp_in}	$W/(m^2\,K)$	HTC between compartment and insulation
	α_{Dis_Shell}	$W/(m^2\,K)$	HTC between discharge side refrigerant and shell wall
	α_{Amb_Shell}	$W/(m^2\,K)$	HTC between ambient and shell wall
	$\eta_{\Delta p\_Ev}$	–	Factor for the FPD in the evaporator
	$\eta_{\Delta p\_Co}$	–	Factor for the FPD in the condenser
	η_{α_Co}	–	Factor for the heat flux between condenser pipe and ambient
$\mathcal{P}_d$	$(\rho c V)_s$	J/K	Thermal inertia of the temperature sensor
	a_{Acc}	–	Fit parameter in the accumulator model
	b_{Acc}	s/bar	Fit parameter in the accumulator model

1.1. Steady-State Stage

In the first stage a subset $\mathcal{P}_s$ of the parameters is fitted automatically using steady-state measurements. In IPSEpro, the steady-state calculation takes only seconds which guarantees a fast numerical method. In the set $\mathcal{P}_s$, parameters do not occur which have a big impact on the thermal inertia. Parameters, such as the product of density ρ, heat capacity c and volume V of the temperature sensor can obviously not be fitted in this stage. Since it only appears as a factor for the derivative of the temperature, it has no influence on the steady-state model. The nonlinear regression is done fully-automatic by the Gauß-Newton method which is described below. It is a method for unconstraint optimization. This implies that the bounds of the parameters have to be enforced by *e.g.* penalty or barrier functions. In this thesis, the penalty function approach was chosen.

1.1.1. The Gauß-Newton Method

Within this section we derive the Gauß Newton-Method and present results for the refrigeration appliance model. For further details we refer to [19]. First of all, we give the necessary definitions.

Definition 1.1 *Let $y \in \mathbb{R}^N$ be the vector of variables, $\tilde{y}, b \in \mathbb{R}^n$ the observed variables and measurement values, respectively, and $x \in \mathbb{R}^m$ the uncertain parameters. Let $n > m$. The steady-state model is described by the system $F : \mathbb{R}^{N+m} \to \mathbb{R}^N$ and let the solution satisfy $F(x, y) = 0$. Let $R = (r_1, r_2, ..., r_n) \in \mathbb{R}^n$ be the vector of residuals $r_j = w_j(\tilde{y}_j - b_j)$ with weights $w_j \in \mathbb{R}$. Furthermore let $D \in \mathbb{R}^{n\times N}$ with $Dy = \tilde{y}$ and the diagonal matrix $W \in \mathbb{R}^{n\times n}$ with $W = \operatorname{diag}(w_j)$.*

The steady-state nonlinear regression problem is given by

$$\min_{x\in\mathbb{R}^m} \Phi(x) = \min_{x\in\mathbb{R}^m} \frac{1}{2} R(x)^\top R(x) = \min_{x\in\mathbb{R}^m} \frac{1}{2}(Dy(x) - b)^\top W^2 (Dy(x) - b). \tag{1.1}$$

The cost functional Φ is a composition of three functions and we have $\Phi : \mathbb{R}^m \to \mathbb{R}^N \to \mathbb{R}^n \to \mathbb{R}$. Let $R'(x) \in \mathbb{R}^{n\times m}$ be the Jacobian matrix of $R(x)$. The optimality condition reads as

$$0 = \nabla\Phi(x) = R'(x)^\top R(x) \tag{1.2}$$

with $\nabla\Phi(x) \in \mathbb{R}^m$ and the Hessian matrix $\nabla^2\Phi(x) \in \mathbb{R}^{m\times m}$ is given by

$$\nabla^2\Phi(x) = R'(x)^\top R'(x) + \sum_{j=1}^{n} r_j(x)\nabla^2 r_j(x). \tag{1.3}$$

Since the minimizer has to satisfy the optimality condition, we search for a root of equation (1.2) using Newton's method. If we use the first term of the right hand side in the above equation as an approximation of the Hessian, the *Gauß-Newton iteration* reads as

$$x^{i+1} = x^i - \left(R'(x^i)^\top R(x^i)\right)^{-1} R'(x^i)^\top R(x^i) \tag{1.4}$$

where i denotes the iteration index. The Jacobian matrix in (1.2) is given by

$$R'(x) = WD\frac{\partial y}{\partial x}(x). \tag{1.5}$$

The matrix $\frac{\partial y}{\partial x}(x) \in \mathbb{R}^{N\times m}$ is also referred to as sensitivity matrix S. Since the solution y satisfies $F(x, y) = 0$, S can be determined analytically. Recall the implicit function theorem from chapter II.

COROLLARY 1.2 *Let the assumptions of the implicit function theorem hold and let $F(x, y) = 0$ with $(x, y) \in U' \times U''$. Then*

$$\frac{\partial y}{\partial x}(x) = -\left(\frac{\partial F}{\partial y}(x, y)\right)^{-1} \frac{\partial F}{\partial x}(x, y). \tag{1.6}$$

Proof. The statement is an immediate consequence of the multidimensional chain rule. The implicit function theorem constitutes that y can be expressed as a function of x. Let $y = g(x)$. Differentiating the equation $F(x, y) = 0$ with respect to x yields

$$\frac{\partial F}{\partial x}(x, g(x)) + \frac{\partial F}{\partial y}(x, g(x)) \cdot \frac{\partial g}{\partial x}(x) = 0. \tag{1.7}$$

Rewriting the equation leads to

$$\frac{\partial g}{\partial x}(x) = -\left(\frac{\partial F}{\partial y}(x, g(x))\right)^{-1} \frac{\partial F}{\partial x}(x, g(x)). \tag{1.8}$$

Substituting y again yields the result. □

From corollary 1.2 we obtain that the function $g : U' \to U''$ in the implicit function theorem has not to be given explicitly to evaluate the sensitivity matrix S. Furthermore, the system of equations from definition 1.1 fulfills the assumptions of the implicit function theorem. This implies that the results from the corollary can be used to calculate S analytically. Since in IPSEpro the partial derivatives $\frac{\partial F}{\partial y}$ are already calculated during Newton's method when the system is solved, only the partial derivatives with respect to the parameters have to be evaluated. A special version of the calculation kernel of IPSEpro was written to obtain $\frac{\partial F}{\partial x}$ and the mapping D. Hence by solving the system $F(x, y) = 0$ once, all necessary terms for the Gauß-Newton step are gained already.

To provide a sufficient descent, a line search method was implemented. Given the correction step

$$\Delta x = -\left(R'(x^i)^\top R(x^i)\right)^{-1} R'(x^i)^\top R(x^i), \tag{1.9}$$

Armijo's rule, see [78], reads as

$$\Phi(x + h\Delta x) \leq \Phi(x) + \sigma h \nabla\Phi^\top \Delta x \tag{1.10}$$

with the step size $h \in (h_{min}, 1)$ and parameter $\sigma \in (0, 1)$. The step size h is then chosen such that Newtons's method used for solving $F(x + h\Delta x, y) = 0$, where the previous solution serves as initial state, converges and Armijo's rule is not violated.

The Gauß-Newton method is a procedure for unconstrained optimization. Since the parameters have to satisfy certain bounds such as *e.g.* non-negativity, constraints emerge which can not be neglected by the numerical method. Therefore, we introduce penalty terms $p(x)$, see [65]. These terms add a positive quantity to the cost functional in (1.1) if a parameter approaches a bound. The quantity has to be chosen sufficiently large such that the minimizer remains within the given bounds. The *Residuum*-Unit (depicted on the bottom left in figure III.22) was added to the Eco-Cool-Lib model library such that the penalty terms are evaluated by IPSEpro directly. Then the form of the cost functional does not change. Let the function ψ map x to zero if no penalty has to be added to the cost functional, *i.e.*

$$\psi(x) = \begin{cases} \frac{x-(x_{max}-\varepsilon_{max})}{\varepsilon_{max}}, & \text{if } x > x_{max} - \varepsilon_{max}, \\ \frac{x-(x_{min}+\varepsilon_{min})}{\varepsilon_{min}}, & \text{if } x < x_{min} + \varepsilon_{min}, \\ 0, & \text{else.} \end{cases} \tag{1.11}$$

Here $x_{min}, x_{max} \in \mathbb{R}$ are the lower and upper bound, respectively, and $\varepsilon_{min}, \varepsilon_{max} > 0$ are the distances to the corresponding bound from which on a penalty is added. Finally the penalty term reads as

$$p(x) = a\psi(x)^4 \tag{1.12}$$

with the parameter $a > 0$ and the respective pseudo measurement is set to zero since we expect to have a parameter x within the given bounds. In PSE the parameters and the corresponding values in the Residuum-Units are connected via Free Equations.

All together the above considerations lead to a fast and robust method for steady-state parameter fitting. Still a global minimum is not guaranteed when the Gauß-Newton method terminates. To increase the probability of a global minimum the calculation can be repeated starting with different sets of parameters and check if the minimizer remains equal. Furthermore in section 1.1.2, a method is proposed to obtain a global minimum. The method was not yet implemented.

1.1.2. Global Minima

Numerical schemes such as the Gauß-Newton method can get stuck in local minima. Heuristical searches such as the *Genetic Algorithms* can be applied instead. There, from an initial population which consists of (randomly) distributed points, referred to as individuals, new generations are obtained from cross-over and mutations with the goal to filter the fitter individuals, *i.e.* parameters yielding a smaller value of the cost functional. Still, it is not guaranteed that heuristical searches of this kind find a global minimum.

In the set up as in IPSEpro such heuristics face the problem that often convergence can not be achieved for the whole population. The reason lies in Newton's method which is used to solve $F(x, y) = 0$ and that initial states are often outside the neighbourhood of convergence. This problem can be solved by ordering the population such that the distance between two consecutive individuals becomes minimal, eventually adding intermediate individuals and using the solution of the prior individual as the initial state for the following. We suggest to apply methods from graph theory to order the population. Let $\mathcal{X}$ be a set of individuals. Then the adjacency matrix A is formed from the Euclidian distances between the individuals, *i.e.*

$$a_{ij} = \|x_i - x_j\|_2 \quad \text{for } x_i, x_j \in \mathcal{X}. \tag{1.13}$$

The adjacency matrix describes a graph and we calculate the Minimum Spanning Tree or solve the Travelling Salesman Problem for a selected individual. It depends on the program's architecture which method suits better. For the Travelling Salesman Problem, the Nearest Neighbouring Algorithm is a very fast heuristic to gain an approximation of the exact solution. For randomly distributed vertices the Nearest Neighbouring Algorithm provides paths which exceed the Held-Karp lower bound, see [44], of the Travelling Salesman Problem by 25% on average, see [51]. Given a possible path, the individuals are evaluated in the corresponding order.

Above we mentioned that adding one ore more intermediate individuals x_I may be necessary to obtain convergence. Intuitively, a straight line connecting two consecutive individuals on which x_I is chosen seems appealing. The straight line is the shortest connection between two individuals. If the bounded domain $\mathcal{F} \subset \mathbb{R}^n$ of parameters is convex, $x_I \in \mathcal{F}$ holds for any x_I on the straight line. Otherwise, a function f has to be found whose trace lies within $\mathcal{F}$.

1.1.3. Fitted Parameters and Results

Within this section we present the fitted parameters and compare the results of the steady-state model with the measurements and the computed values of the VBA model. After setting up the refrigeration appliance model, the results may deviate fairly from the measurement data even if physically reasonable values for the uncertain parameters are chosen. The Gauß-Newton method proved to be still applicable. In table IV.3 the starting (Start) and final (Fit) parameter values, corresponding results and measurements are given. It took 54 iterations until the correction step satisfied the error tolerance. Within the iteration, the dynamic parameters, $(\rho c V)_s$, a_{Acc} and b_{Acc}, were already set to their final value.

Table IV.3.: Stationary fitted parameters by the Gauß-Newton method, measurement and calculated values for $T_{amb} = 25\,°\mathrm{C}$.

Parameter	Unit	Start	Fit	Meas. data	Unit	Start	Fit	Meas.
α_{Iso_out}	W/(m^2 K)	5	4.80900	P	W	142.10	58.60	57.87
α_{Comp_in}	W/(m^2 K)	5	15.1156	t_{Shell}	°C	104.37	62.20	62.25
α_{Dis_Shell}	W/(m^2 K)	10	12.5657	t_{Shell_out}	°C	156.02	67.57	67.28
α_{Amb_Shell}	W/(m^2 K)	6	10.1078	t_{Cond_in}	°C	150.54	53.11	58.36
$\eta_{\Delta p\_Ev}$	–	1	2.24868	t_{Cond_25}	°C	123.68	35.65	37.42
$\eta_{\Delta p\_Co}$	–	1	1.00482	t_{Cond_50}	°C	106.03	36.49	36.57
η_{α_Co}	–	1	4.77346	t_{Cond_75}	°C	102.60	36.44	35.99
				t_{Cond_out}	°C	102.50	36.42	35.59
$(\rho c V)_s$	J/K	-	12.65	t_{Cap_out}	°C	-10.80	-34.74	-35.73
a_{Acc}	–	-	0.75	t_{Evap_out}	°C	-13.08	-40.66	-40.78
b_{Acc}	s/bar	-	100	t_{Shell_in}	°C	64.03	26.54	27.31
				p_{Shell_in}	bar	0.961	0.276	0.238
				p_{Shell_out}	bar	20.989	4.878	4.879
				t_{Comp}	°C	-7.74	-34.95	-34.85

The temperature at the inlet of the condenser t_{Cond_in} shows a larger deviation to the measurement which can be explained by the spatial discretization. The length of one Condenser-Unit is approximately 0.5 m and the temperature of the refrigerant in the Unit equals the outlet temperature. Thus the true inlet temperature is supposed to be higher. To obtain more precise results, one possibility is to refine the condenser discretization in the beginning and repeat the calculation which was not done in this thesis.

The fact that the temperature of the Condenser pipe is lower at 25 % length than at 50 % length may not be obvious in the first place but this results from the non-constant HTC. In the steady-state result, at 25 % the refrigerant is barely in the vapour phase. There the temperature of the refrigerant is hardly higher than the saturation temperature but the HTC is much lower than in the two-phase region. Therefore the pipe temperature, which is measured, is lower at this point than in the region where the refrigerant is two-phase. This is illustrated in figure IV.1 where the same settings are taken as described in section III.8.3 and III.8.4, the HTC is unbounded and the condenser is discretized with 100 Units.

In figure IV.2 the Ts-diagram of the refrigeration cycle in steady-state is illustrated. We have to mention that this is an extreme state which is never reached in a household since the compressor does not operate continuously there. The Ts-diagram deviates from the ideal illustration in section I.1. First, the refrigerant is not subcooled in the condenser. If the heat flux between the condenser

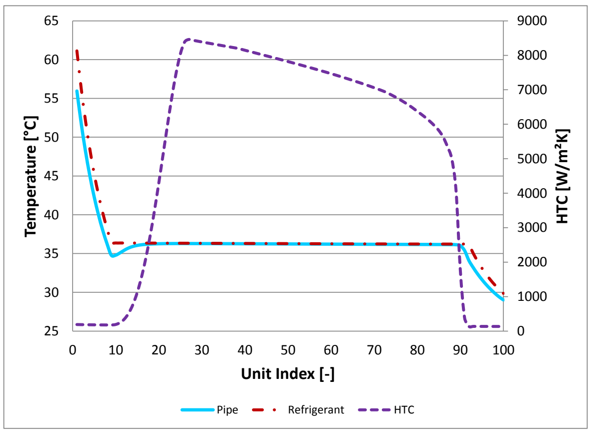


Figure IV.1.: Temperature of the condenser pipe wall and refrigerant and HTC over Units.

and the ambient is increased, the compartment temperature decreases. Second, the refrigerant in the evaporator is not superheated. Thus, cooling potential is wasted since the refrigerant exiting the evaporator could still absorb heat from the compartment. If the compressor operates intermittently, a typical strategy is to switch off the compressor if two-phase working fluid starts to exit the evaporator. In the internal heat exchanger, where the heat flux is 36.10 W, the refrigerant is fully vaporized which avoids that a liquid fraction enters the compressor. Finally, the compression is not isentropic as it is explained in section III.6.3.2.

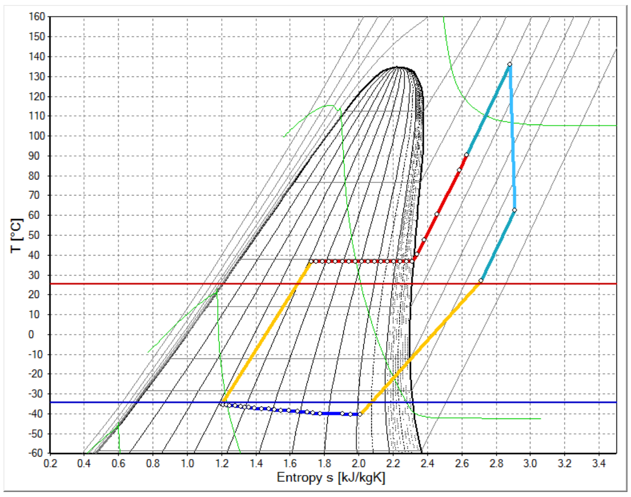


Figure IV.2.: Steady-state Ts-diagram. Condenser (red), evaporator (dark blue), compressor (light blue), shell (turquoise), capillary and internal heat exchanger (yellow). Horizontal lines represent the ambient (red) and compartment (blue) temperature.

1.2. Dynamic Stage

In this stage the three parameters in the subset $\mathcal{P}_d$ are adjusted manually. In principal the Gauß-Newton method is capable of fitting constant parameters to dynamic measurements automatically. Only the computational effort for a single evaluation increases significantly and the sensitivity matrix can only theoretically be calculated analytically anymore. Even if an implementation would already be possible in IPSEpro, the calculation times would be too long.

In the dynamic validation process, describing the dynamic behavior of the temperature sensor is a crucial part. First, the switching logic and the temperature measured by the sensor is not fully revealed by the manufacturer. In this refrigeration appliance an integral controller was used which implies that the two thresholds, lower and upper compartment temperature limit, have to be determined. The values are set to $-20\,°C$ and $-20.5\,°C$ for upper and lower threshold, respectively. Second, the product $(\rho c V)_s$ is uncertain and this thermal inertia of the temperature sensor has a huge impact on the operation times of the compressor. Since the compartment is modelled as a point mass the temperature stratification is neglected. Consequently, the actual temperature of the surrounding air differs between the model and the reality. Additionally the sensor is in contact with the compartment wall which leads to further heat fluxes. Therefore, $(\rho c V)_s$ is fitted such that the operation times of the model coincides with the reality. In the refrigeration appliance a temperature probe was positioned close to the temperature sensor. In figure IV.3e, the calculated and measured temperatures are compared.

The two other parameters concern the accumulator. In the investigated refrigeration appliance no typical accumulator is built in. Still, such a Unit had to be incorporated in the IPSEpro and VBA models. The reason lies in the design of the evaporator in the appliance. There the evaporator is wound vertically around the compartment. Hence some sort of accumulation of liquid refrigerant is expected in the "bottom" part. Since the heat exchangers are modelled as horizontal tubes this accumulation effect is lost and an additional component which compensates this insufficiency has to be built in. Its effect is not known and therefore has to be fitted. In IPSEpro a different approach was chosen than in the VBA model. See section III.6.2 and 2 for the approach applied in IPSEpro and VBA, respectively.

Table IV.4.: Gauß-Newton (G.N.) and manually fitted parameters and respective steady-state values for $T_{amb} = 25\,°C$.

Parameter	Unit	G.N.	Manual	Meas. data	Unit	G.N.	Manual
α_{Iso_out}	W/m^2K	4.80900	5	P_{Comp}	W	58.60	58.78
α_{Comp_in}	W/m^2K	15.1156	8	t_{Shell}	°C	62.20	63.99
α_{Dis_Shell}	W/m^2K	12.5657	15	t_{Shell_out}	°C	67.57	66.94
α_{Amb_Shell}	W/m^2K	10.1078	9.76	t_{Cond_in}	°C	53.11	52.90
$\eta_{\Delta p\_Ev}$	–	2.24868	2	t_{Cond_25}	°C	35.65	35.78
$\eta_{\Delta p\_Co}$	–	1.00482	1	t_{Cond_50}	°C	36.49	36.61
η_{α_Co}	–	4.77346	4.7	t_{Cond_75}	°C	36.44	36.56
$(\rho c V)_s$	J/K	12.65	6.38	t_{Cond_out}	°C	36.42	36.54
a_{Acc}	–	0.75	0.75	t_{Cap_out}	°C	-34.74	-35.18
b_{Acc}	s/bar	100	100	t_{Evap_out}	°C	-40.66	-40.59
				t_{Shell_in}	°C	26.54	26.31
				p_{Shell_in}	bar	0.276	0.278
				p_{Shell_out}	bar	4.878	4.894
				t_{Comp}	°C	-34.95	-34.26

First, the strategy recommended in the beginning of this chapter was pursued. The steady-state measurements with $T_{amb} = 25\,°\mathrm{C}$ were taken only to fit the parameters. In table IV.4 the fitted parameters and respective steady-state results can be seen in the Gauß-Newton (G.N.) columns. In figure IV.3, six dynamic values are compared with measurements. The graphs corresponding to parameters in the Gauß-Newton column are yellow-dashed and the measurements are red-dotted. Although the steady-state results are very good, see table IV.3, the temperature of the shell surface deviates by $2\,°\mathrm{C}$, see figure IV.3b. The reason lies in the assumption that the parameter α_{Amb_Shell} is constant. Considering the models for natural convection, a temperature dependence in the HTC may be expected. Because the shell surface temperature is higher in the steady-state case than in the dynamic case, the HTC is assumed to be higher as well. Consequently, a lower (constant) HTC has to be prescribed in the dynamic case if the temperature has to fit the measurements. To achieve better results in the dynamic case, subsequently α_{Amb_Shell} was also adjusted manually. Furthermore, α_{Dis_Shell} was increased as well such that the refrigerant discharged from the Compressor Unit passes more heat to the shell wall. Thus, in the condenser less heat has to be exchanged with the ambiance and η_{α_Co} was slightly decreased. The manually changed parameters and corresponding steady-state results are given in table IV.4, column *Manual*, and the dynamic results are illustrated as blue graph in figure IV.3. Noticeable is the big difference in α_{Comp_in} and $(\rho cV)_s$ between the automatic and manual fit. Both parameters are only about half as large now. The HTC α_{Comp_in} occurs in the energy equation of the Compartment-Unit and influences the heat flux between the insulation respectively temperature sensor and the compartment. Hence it also appears in the energy equation of the Temperature Sensor Unit, *i.e.*

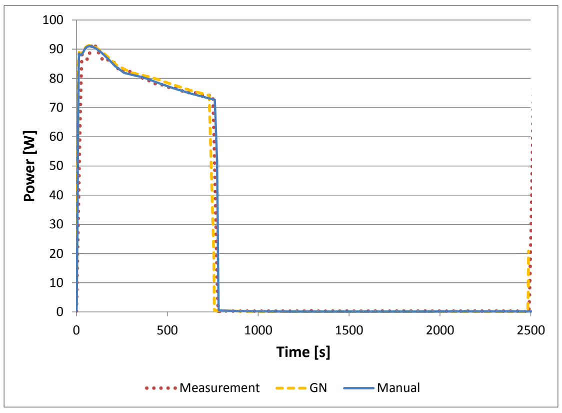
$$(\rho cV)_s \frac{dT_s}{dt} = \alpha_{Comp\_in} A_s (T_c - T_s) \tag{1.14}$$

where T_c is the temperature of the compartment and A_s the surface area of the sensor. Multiplying both parameters by 2 yields the same result. Still they have been adjusted in the final manual fit since the on-off-duration of the compressor was approximated better. The difference of the steady-state compartment temperatures is only $0.69\,°\mathrm{C}$ between the automatic and manual fit. In the periodic cycle, the compressor is switched on 13 (12.66, 13.04) min and off 29 (28.72, 29.15) min in the measurement (G.N., Manual). Hence the time period of one cycle is 42 (41.38, 42.19) min.

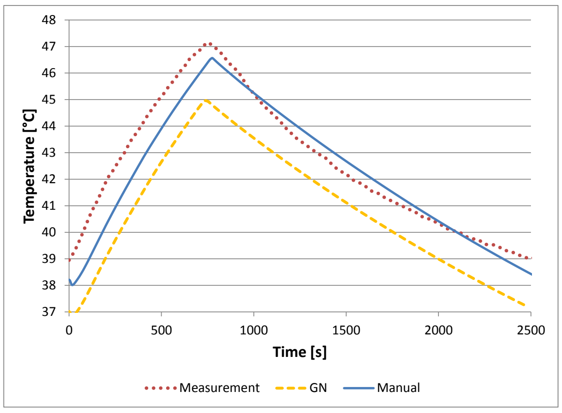
The dynamic results are illustrated in figure IV.3. In figure IV.3a, the blue graph coincides with the measurements well and the on-off durations of the compressor fits the measurements. In all three graphs shortly after the compressor switched on, the power consumption is not smooth. This results from the non-monotonically drop of the pressure at the inlet of the compressor, see figure IV.3d. Although not as distinct as in the measurements, the change in the monotonicity of the graph is seen also in the simulation. The calculated values of the shell surface temperature, see figure IV.3b, fall in the first seconds after the compressor switched on. The reason lies in the large amount of cold refrigerant entering the shell in this period. In figure IV.3c, the pressure at the outlet of the compressor is illustrated. If the compressor is idle, the simulated pressure deviates from the measurement since the compressor is not switched off completely in the simulation but a residual NOR of 20 rpm is chosen. Therefore also the pressure equalizing between condenser and evaporator does not take place. Finally, in figure IV.3f the condenser pipe temperature at 50 % length is depicted. The comparison suggests that the thermal inertia of the pipe wall may be higher in reality than in the simulation.

2. Comparison with the VBA model

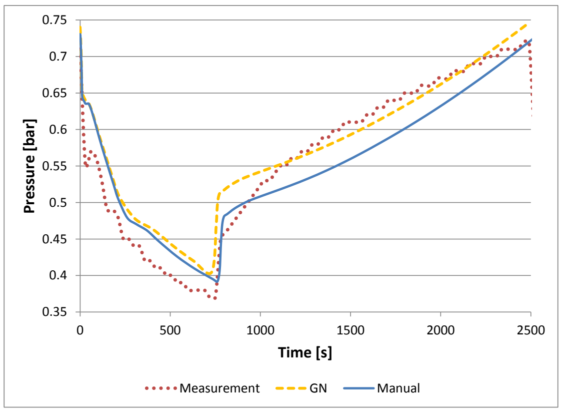
Within this section we compare simulation results gained with IPSEpro and the VBA-tool with the measurements. The refrigeration appliance was investigated for the ambient temperatures $25\,°\mathrm{C}$ and $43\,°\mathrm{C}$. The models in IPSEpro and VBA are not exactly the same. There are five major differences



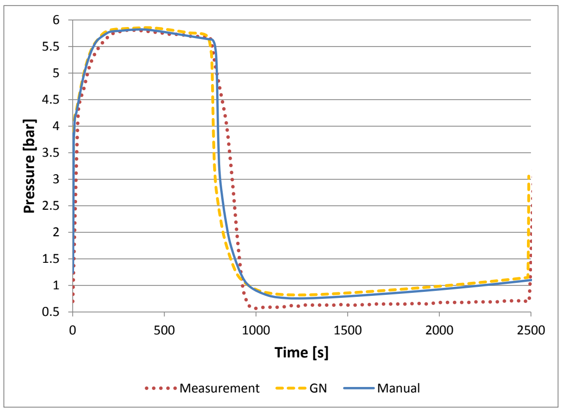
(a) Compressor electric power.



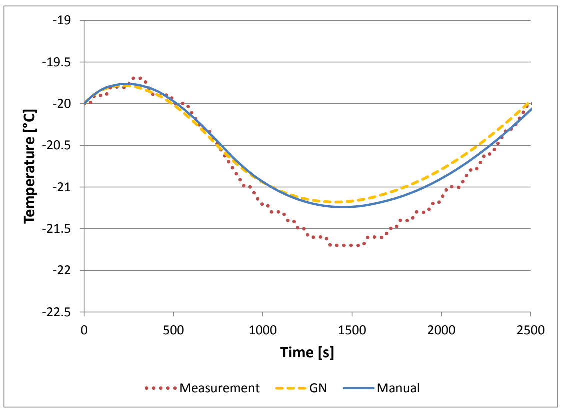
(b) Shell surface temperature.



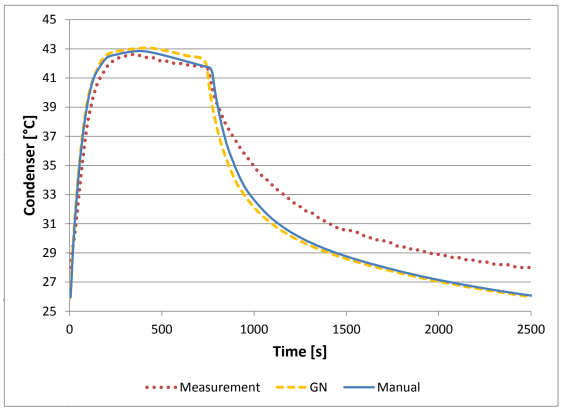
(c) Pressure at compressor inlet.



(d) Pressures at compressor outlet.



(e) Sensor temperature.



(f) Condenser Pipe temperature at 50 % length.

Figure IV.3.: Values calculated in IPSEpro with the Gauß-Newton (yellow-dashed) and manual (blue) fitted parameters and measured (red-dotted) with $T_{amb} = 25\,°C$.

between the IPSEpro and VBA model. First, the heat exchangers are spatially discretized with 100 volumes in VBA. Second, the door heating is modelled in VBA but not in IPSEpro. Conversely, a (heat flux) coupling between the condenser and the backside of the appliance is implemented in IPSEpro but not in VBA. Third, the whole compressor, involving Shell, Compressor and Oil_Sump Unit, is modelled differently. It is assumed that the refrigerant entering the shell has immediately the same temperature as the shell wall. The outlet temperature T_{out} of the shell is calculated by the correlation

$$T_{out} = eT_{in} + fT_{in}\left(\frac{p_{out}}{p_{in}}\right)^{\frac{\kappa-1}{\kappa}} \tag{2.1}$$

where the subscripts *in* and *out* indicate the states at the shell in- and outlet, respectively, and $e, f \in \mathbb{R}$ are parameters which have to be fitted with measurements for each compressor type. Fourth, a different ansatz for the efficiency $\xi \in [0.5, 1]$ of the accumulator is deployed, *i.e.*

$$\xi = 1 - a\dot{Q}. \tag{2.2}$$

Above, $a \in \mathbb{R}^+$ is a fit parameter and $\dot{Q}$ the heat flux between refrigerant and pipe wall in the accumulator. Since the above correlation yielded unsatisfying results in IPSEpro, a different approach was chosen there, see section III.6.2. Fifth and last, the internal heat exchanger on the suction side is discretized with $k = 15$ evaporator volumes in the VBA model. We explain the model by the notations from section III.6.4. Let $h_{drainSuc}$ be the outlet enthalpy of the internal heat exchanger on the suction side gained from the ANN. Then $h_{drainSuc}$ is only used to calculate the heat flux $\dot{Q}$, *i.e.*

$$\dot{Q} = \dot{m}_{feedSuc}(h_{feedSuc} - h_{drainSuc}). \tag{2.3}$$

The heat flux is limited if physical laws are violated as described in section III.6.4. Then an equal share $\dot{Q}/k$ is added to the pipe wall of each of the 15 volumes. Due to the thermal inertia of the pipe wall, less heat is transferred to the refrigerant on the suction side for the first period after the compressor. This implies that the outlet enthalpy of the internal heat exchanger on the suction side does not equal the return value from the ANN in the VBA model. However, the results approximate the measurements better. Finally, the whole dynamic validation including fitting all uncertain parameters was done manually and only dynamic measurements were used by the developers of the VBA model.

In table IV.5, the steady-state measured and calculated values in IPSEpro and VBA are given for ambient temperatures 25 °C and 43 °C. Of course the steady-state point in which the compressor is continuously switched on is an extreme point and the compressor models are evaluated at a point where their fit parameters are not validated themselves. Still, the shell surface temperatures fits the measurements well. The simple compressor models used in both simulation tools are not sufficient to predict the compressor outlet temperature satisfyingly. In the fitting point with an ambient temperature of 25 °C, IPSEpro performs better but an almost equally large deviation from the measurement emerges at 43 °C. We assume that some of the parameters depend on the temperature in reality. Since they are set constant in the model, a gap is expected if the temperature changes.

Next we discuss the dynamic results. We start with 25 °C ambient temperature. In table IV.6, the on-off-durations of the compressor are given. In VBA model the compressor is switched on 0.83 min longer than in the measurement, see row *Comp.-On*, and switched off 1.98 min shorter, see row *Comp.-Off*. Hence the whole on-off-duration deviates by 1.15 min. In comparison, in IPSEpro the deviation is 0.29 min. These deviations are also visible in figure IV.4 which is described in the following. We start with the power of the compressor, see figure IV.4a. The power in the VBA model has its maximum later than in IPSEpro. Again this can be explained with the single oscillation in the pressure at the compressor inlet. In figure IV.4c, it can be seen that the amplitude of this oscillation is the largest in VBA. The consumed electric power P is reduced during this oscillation due to efficiency η_{comb} defined in equation (6.56) on page 64. Next we compare the temperature sensor, see figure IV.4e. In VBA the lower and upper thresholds are set to -21 °C and -19.5 °C, respectively. Hence the

Table IV.5.: Steady-state measured and calculated values for ambient temperatures 25 °C and 43 °C.

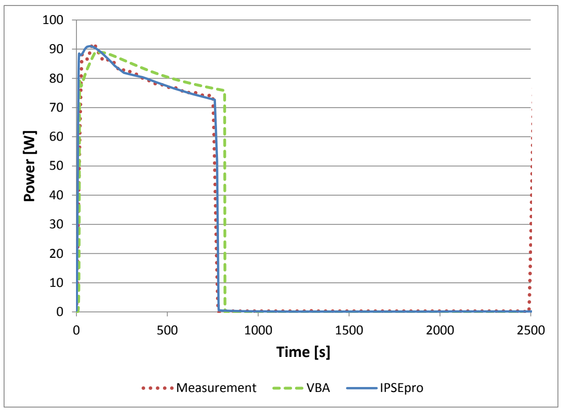
Meas. data	Unit	25 [$M.$]	25 [IPSEpro]	25 [VBA]	43 [$M.$]	43 [IPSEpro]	43 [VBA]
P_{Comp}	W	57.87	58.78	59.27	74.62	83.08	78.55
t_{Shell}	°C	62.25	63.99	60.33	83.99	83.57	84.23
t_{Shell_out}	°C	67.28	66.94	57.82	92.67	100.39	83.78
t_{Cond_in}	°C	58.36	52.90	52.47	85.41	85.39	78.24
t_{Cond_25}	°C	37.42	35.78	34.75	59.19	57.76	54.60
t_{Cond_50}	°C	36.57	36.61	34.71	56.68	57.57	54.57
t_{Cond_75}	°C	35.99	36.56	34.67	56.02	57.52	54.53
t_{Cond_out}	°C	35.59	36.54	34.78	55.06	57.50	54.68
t_{Cap_out}	°C	-35.73	-35.18	-37.61	-28.55	-24.54	-28.65
t_{Evap_out}	°C	-40.78	-40.59	-39.12	-32.50	-29.76	-30.44
t_{Shell_in}	°C	27.31	29.74	28.77	43.73	42.72	43.54
p_{Shell_in}	bar	0.238	0.278	0.276	0.375	0.470	0.428
p_{Shell_out}	bar	4.879	4.894	4.651	7.939	8.282	7.717
t_{Comp}	°C	-34.85	-34.26	-35.18	-24.60	-23.57	-26.02

Table IV.6.: Compressor operation times for ambient temperatures 25 °C and 43 °C.

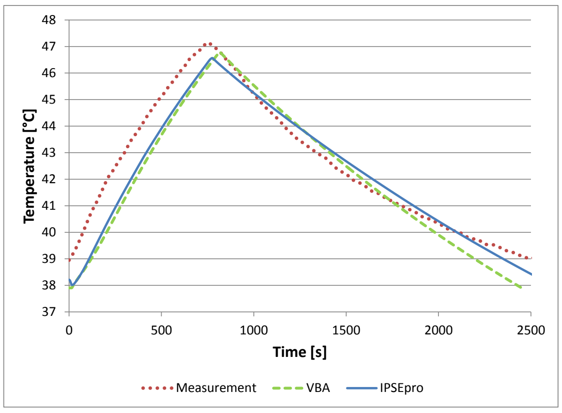
Meas. data	Unit	25 [$M.$]	25 [IPSEpro]	25 [VBA]	43 [$M.$]	43 [IPSEpro]	43 [VBA]
Comp.-On	min	13.00	13.04	13.83	28.00	30.56	27.83
Comp.-Off	min	29.00	29.15	27.02	13.50	12.61	15.53
Total	min	42.00	42.29	40.85	41.50	43.17	43.36

thermal inertia has to be reduced, compared to the value in IPSEpro, such that the sensor reaches the thresholds after the same time periods. Thus the graph shows a larger deflection in the VBA model. Depicted in figure IV.4b, the shell surface temperatures coincide with the measurements equally well. Finally the condenser pressure and pipe temperature at 50 % length are illustrated in IV.4d and figure IV.4f, respectively. It can be seen that in both figures the VBA graphs overshoot the IPSEpro results and the measurements. A reason may be the thermal inertia. If it would be increased in the VBA model, we expect that the green-dashed graph IV.4f shows less deflection. The more slowly increasing temperature enables the refrigerant to transfer more heat to the pipe and thus the maximal pressure of the refrigerant in the condenser, see figure IV.4d, is decreased.

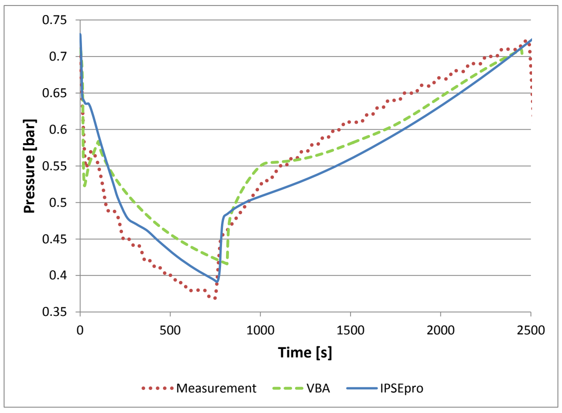
To conclude this chapter, we investigate the measured and calculated values for an ambient temperature of 43 °C which are illustrated in IV.5. The furthest deviation from the measurement emerges in the pressure at the compressor inlet, see figure IV.5c. Both simulation tools predict the pressure about 0.1 bar too high. Furthermore, the on-off-durations of the compressor do not fit as good as with 25 °C ambient temperature, see table IV.6. Although the accumulated times are almost equal in both simulations, IPSEpro predicts longer on- and shorter off durations than the VBA model and the measurements. Since the electric power consumed by the compressor nearly coincide in IPSEpro and VBA, see figure IV.5a, the total energy consumption per day is too large in IPSEpro compared with the other two values.



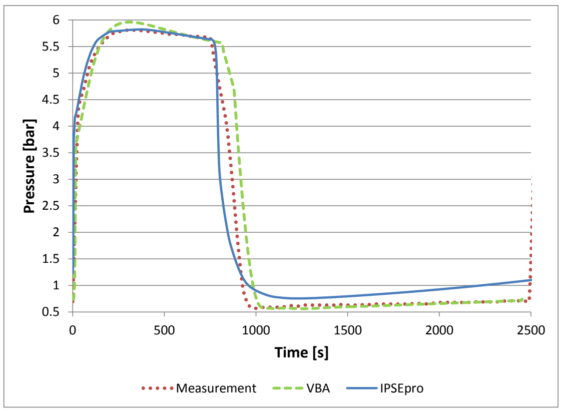
(a) Compressor electric power.



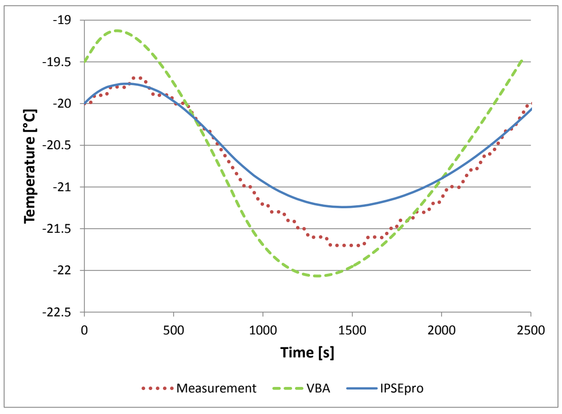
(b) Shell surface temperature.



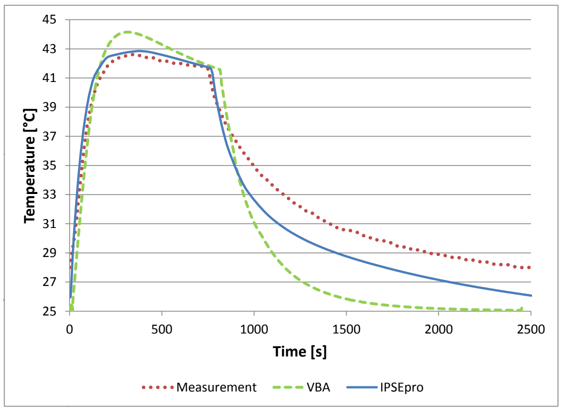
(c) Pressure at compressor inlet.



(d) Pressures at compressor outlet.

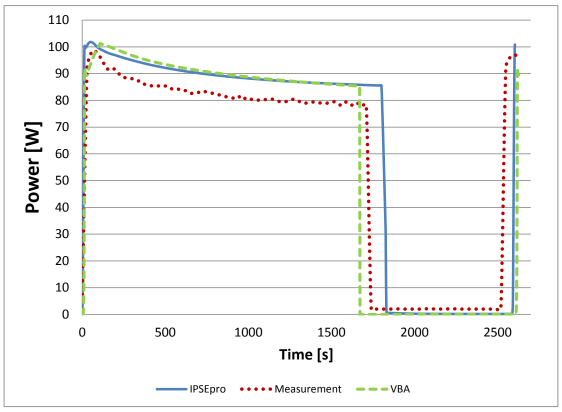


(e) Sensor temperature.

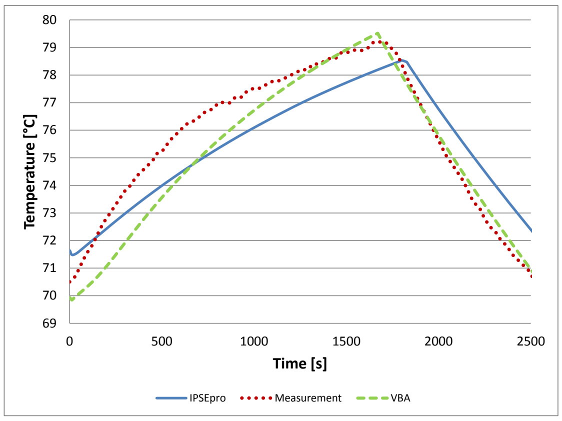


(f) Condenser pipe temperature at 50 % length.

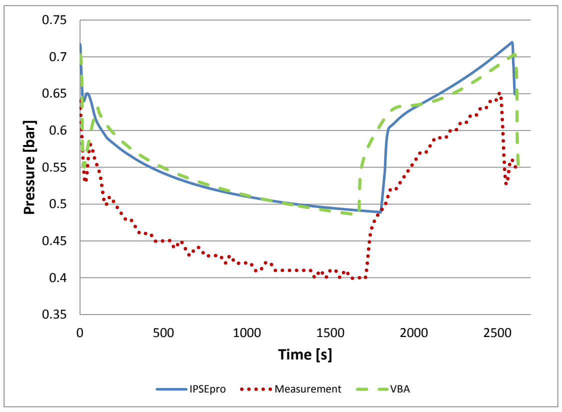
Figure IV.4.: Values calculated in IPSEpro (blue) and VBA (green-dashed) and measured (red-dotted) with $T_{amb} = 25\,°C$.



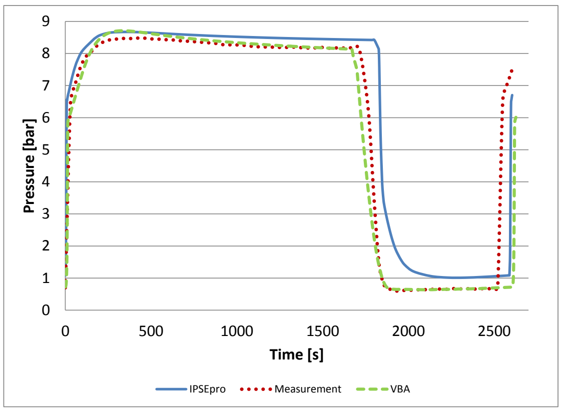
(a) Compressor electric power.



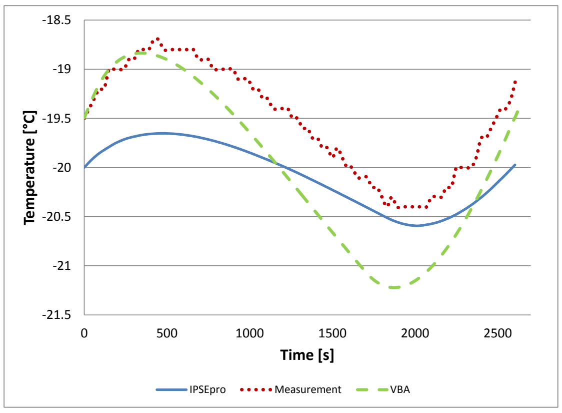
(b) Shell surface temperature.



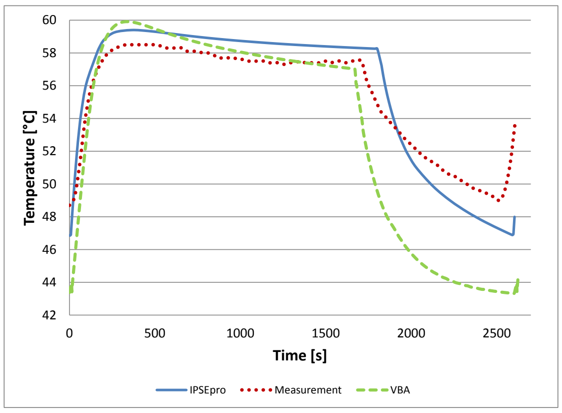
(c) Pressure at compressor inlet.



(d) Pressures at compressor outlet.



(e) Sensor temperature.



(f) Condenser pipe temperature at 50 % length.

Figure IV.5.: Values calculated in IPSEpro (blue) and VBA (green-dashed) and measured (red-dotted) with $T_{amb} = 43\,°C$.

V. Dynamic Simulation Results with IPSEpro

Within this chapter the results of the refrigeration appliance model gained with IPSEpro are presented. First, the validated model is discussed and a deeper insight into the dynamic behaviour than in chapter IV is given. The results are not compared with measurements. Afterwards, a parameter study is performed in which the NOR of the compressor is varied to locate an energy consumption minimum.

1. Refrigeration Appliance Model

Within this section we present the results of the refrigeration appliance model in detail. We focus as in chapter IV on the periodic cycle of the appliance. In the following figures always the periodic cycle is depicted. The compressor switches on at time $t = 0\,\mathrm{s}$ and off at $t = 768\,\mathrm{s}$ where a delay of $15\,\mathrm{s}$ is assumed until the new NOR is reached. It switches on again after $2531\,\mathrm{s}$. First, the actual compartment air temperature and the one measured by the sensor are investigated. Then the distribution of mass in the system and the heat transfers are discussed before the time evolution of the Ts-diagram is examined. Next, the behaviour of the refrigeration appliance model for different ambient temperatures is presented. Finally, the heat exchangers, condenser and evaporator, and the graphs of their variables are shown and explained in detail. We start with the compartment.

Compartment Temperature

The sensor used for the control of the compressor measures the temperature of the compartment. Since the probe has a considerable thermal inertia, the actual compartment air temperature is not monitored. In figure V.1, the actual and measured temperature of the compartment are illustrated and an obvious deviation is noticeable. This effect can be exploited to change the on-off-duration of the compressor. Either by adjusting the switching thresholds or the thermal inertia of the probe, the energy consumption of the appliance is influenced. In [43], a parameter study is presented which was performed with the VBA model.

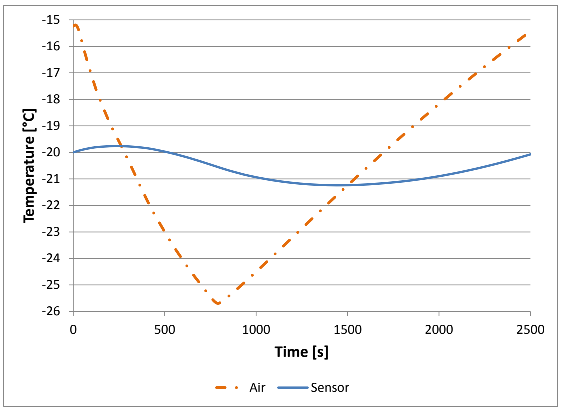
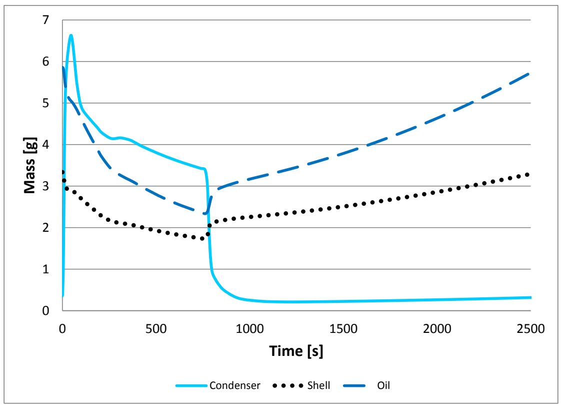
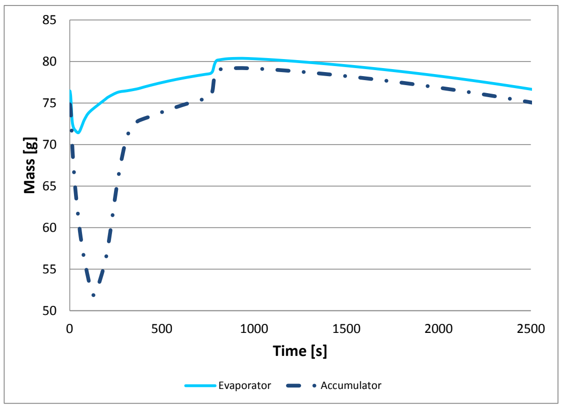


Figure V.1.: Compartment Air and Sensor Temperature.

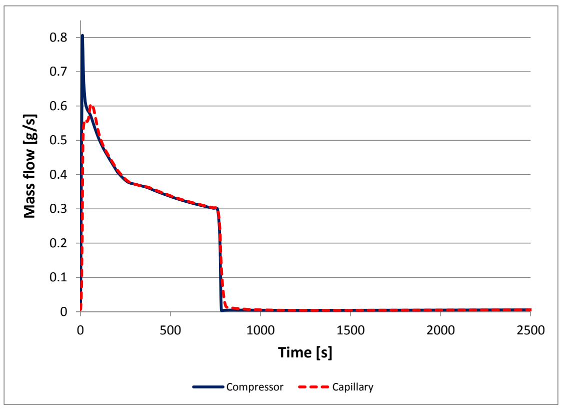


(a) In the condenser, shell and dissolved in the oil.

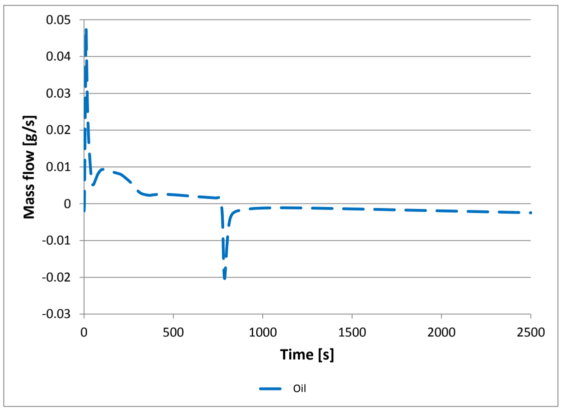


(b) In the evaporator and therein in the accumulator.

Figure V.2.: Stored Mass.



(a) Through the compressor and capillary.



(b) Between the oil and refrigerant inside the shell.

Figure V.3.: Mass flows.

Distribution of Mass

Next the local distribution of refrigerant in the appliance is investigated. The total filling quantity is 86 g and in figure V.2 the stored masses in the components are illustrated. Additionally, in figure V.3 the mass flows through the compressor and capillary and the mass flow between the suction side in the shell and the dissolved refrigerant in the oil are shown. The major part of refrigerant is stored in the evaporator and therein in the accumulator, see figure V.2b. It is assumed that the gravitational force holds the refrigerant in the vertically wound evaporator.

Shortly after the compressor switched on, a peak in the mass flow $\dot{m}_{Comp}$ through the compressor is reached, see figure V.3a. Since the mass flow $\dot{m}_{Cap}$ through the capillary increases more slowly, mass is stored in the condenser as shown in figure V.2a. The refrigerant stored in the condenser comes from the evaporator and the shell and is also desorbed from the oil. Almost 60 s after the compressor switched on, $\dot{m}_{Cap}$ surpasses $\dot{m}_{Comp}$ and mass is transferred from the condenser back into the evaporator. In figure V.3b, the mass flow $\dot{m}_{Oil}$ caused by sorption is given. Positive values imply desorption, negative values absorption. After a peak in the beginning, the mass flow is less than 0.01 g/s. In total, about 3 g of refrigerant are desorbed from the oil while the compressor operators.

When the compressor switches off, the condenser is almost emptied, see V.2a, since $\dot{m}_{Comp}$ is reduced to a minimum immediately but refrigerant flows still through the capillary due to the pressure difference between condenser and evaporator. In the first seconds after the shut down also a (negative) peak in $\dot{m}_{Oil}$ is visible, see figure V.3b. Since $\dot{m}_{Oil} < 0$ holds while the compressor is in idle mode, refrigerant is absorbed by the oil and before the compressor turns on again almost 6 g of refrigerant are dissolved in the oil. Finally, during the switch-off period, mass is transferred from the evaporator to the shell caused by the residual NOR which had to be chosen such that the simulation could be performed.

Heat Transfers

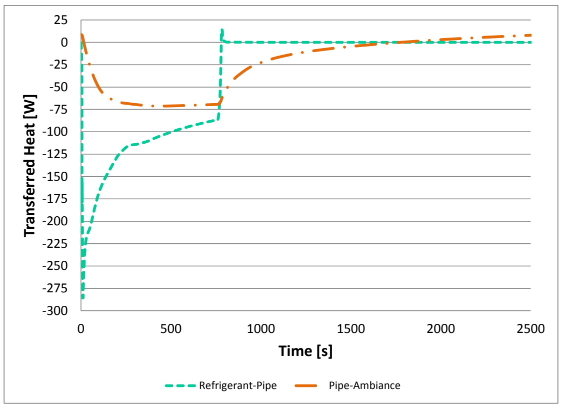
Next, the heat transfers illustrated in figure V.4 are discussed. Each graph is described by two names. These names are the components whose heat exchange is represented by the graph. The following convention is chosen for the sign of the transferred heat: if heat is transferred from the first to the second component, the flux is negative. Otherwise it is positive. In figure V.4a, the heat transfers in the condenser are shown. Right after the compressor switched on, the heat flux $\dot{Q}_{rp}$ between the refrigerant and the pipe wall becomes huge. This results from the high HTCs in the two-phase region and that the temperature T_p of the pipe wall increases more slowly than the one of the refrigerant due to the thermal inertia of the pipe wall. Because of the slower rise of T_p and the lower HTC, the gradient of the heat flux $\dot{Q}_{pa}$ between the pipe wall and the ambiance (orange graph) is lower compared to the one of $\dot{Q}_{rp}$. The thermal inertia of the pipe wall leads to a smaller $\dot{Q}_{pa}$ in the beginning, but this is made up by the fact that after the compressor switched off, heat is still transferred to the ambiance. Finally, $\dot{Q}_{rp}$ becomes positiv right after the compressor switches off. Thus the refrigerant absorbs heat from the pipe wall since the temperature of the refrigerant decreases, due to the pressure drop, faster than T_p. In [43], a parameter study, performed with the VBA model, is presented in which the condenser pipe mass and hence the thermal inertia is increased. The authors claim, that if the condenser pipe mass is multiplied by 5 or 20, the energy consumption is decreased by 5 % or 17 %, respectively.

In figure V.4b, the heat transfers concerning the evaporator and the compartment are depicted. The heat transfer $\dot{Q}_{ia}$ between the insulation and the ambiance rises while the compressor operates. Due to the coupling with the condenser, the air pad between the condenser and insulation is heated up and hence $\dot{Q}_{ia}$ is larger. Next, the refrigerant absorbs the most heat after the compressor switched on. This implies that the appliance operates at highest efficiency in this period since the current COP

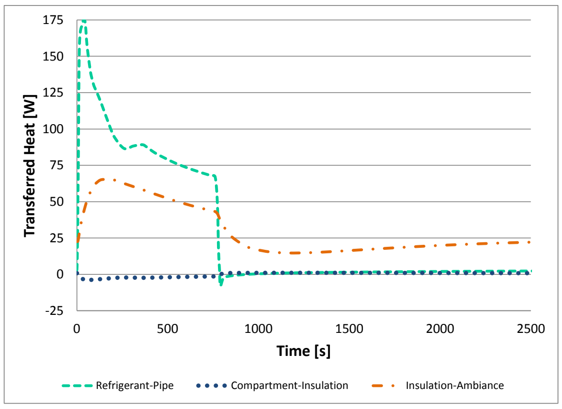
$$\varepsilon_c(t) = \frac{\dot{Q}_0(t)}{P_{el}(t)} \tag{1.1}$$

is the largest. As in the condenser, the sign of the heat flux for the refrigerant changes when the compressor is switched off. Thus the refrigerant in the evaporator heats the pipe wall. In figure V.4c, the heat transfer in the internal heat exchanger of the capillary is shown. It can be seen, that it is turned off when the compressor is idle since the ANN does not yield reasonable values in this case. The heat flux corresponds to the mass flow through the capillary illustrated in figure V.3a.

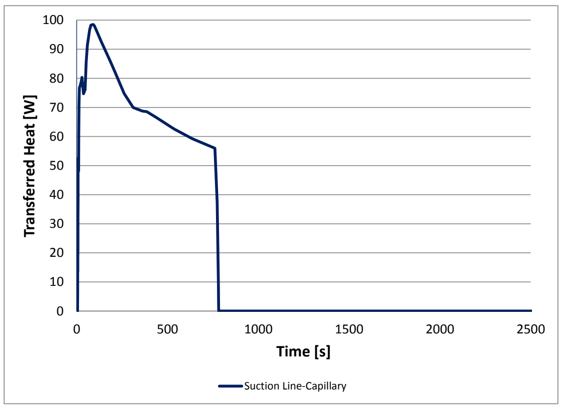
Finally in figure V.4d, the heat transfers inside the shell are shown. Since the areas and HTCs between the components are set constant, the heat transfers solely depend on the temperature differences. When the compressor is switched on, two-phase cold refrigerant is sucked from the evaporator. Partly because



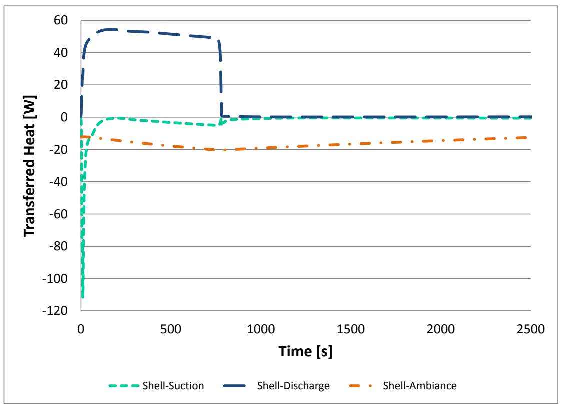
(a) In the condenser.



(b) Between the refrigerant and the pipe wall in the evaporator, compartment and insulation as well as between the insulation and the ambiance.



(c) In the internal heat exchanger.



(d) In the Shell.

Figure V.4.: Heat Transfers.

the heat transfer in the capillary heat exchanger is still off. Hence, the refrigerant entering the shell absorbs heat from the shell wall. Shortly after only superheated warm refrigerant leaves the capillary heat exchanger. Thus less heat is transferred from the shell wall to the suction side refrigerant. The graph of the heat flux between the shell wall and the ambient (orange) corresponds to the temperature of the shell wall since also the ambient temperature is set constant. Finally, the refrigerant on the discharge side has a higher temperature than the shell and heats the shell.

Evolution of the Ts-diagram

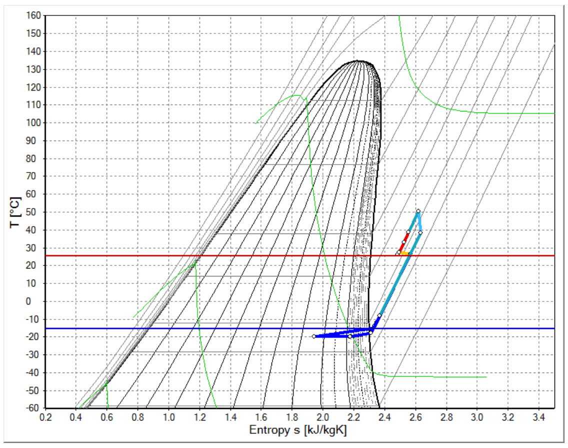
In the figures V.5 and V.6 the time evolution of the Ts-diagram for selected time instants is given. The depicted points (cycle markers) indicate the state in the connections, thus at the inlet and outlet of the Units and not in the interior. Hence, the pair (T, s) corresponds to the flow. The points are connected by straight lines and do not necessarily represent the physical path. We start to investigate the evolution of the Ts-diagram with figure V.5 corresponding to the switch-on period of the compressor.

In figure V.5a, the compressor is about to turn on. The major part of the refrigerant is stored in the evaporator, see figure V.2b. The compartment temperature T_c (blue horizontal line) is approximately $-15\,°\text{C}$, compare with figure V.1. The heat transfer $\dot{Q}_{Cap}$ in the internal heat exchanger is turned off. In figure V.5b, the compressor starts to turn on. Recall that in the simulation it takes 15 s until the new NOR is reached. The mass flow in the compressor increases and the condenser is being filled up. Also $\dot{Q}_{Cap}$ is turned on. This is visible by a second yellow line. Nevertheless, two-phase refrigerant streams into the shell. In figure V.5c the compressor reached the operational speed of 3000 rpm. The condenser is already filled. The inlet of the evaporator is subcooled for a short moment which results from the heat transfer $\dot{Q}_{Cap}$. The difference in the enthalpy and entropy between the evaporator inlet and outlet starts to increase. The inlet of the shell is barely superheated. In figure V.5d, the condenser reaches it maximal filling quantity, but the refrigerant at the outlet is not subcooled. The inlet of the evaporator is two-phase again. The refrigerant at the outlet of the internal heat exchanger on the suction line is now superheated and its temperature T_{Suc} is close to the ambient temperature T_{Amb}. Next we consider figure V.5e. There the saturation temperature inside the condenser is already above $40\,°\text{C}$ and the mass of stored refrigerant decreases again, compare with figure V.2a. The first four Units of the condenser are superheated now and the inlet of the evaporator is subcooled again. Both condenser and evaporator are fully spanned. The pressure drop is clearly visible in the evaporator. The refrigerant at the condenser outlet is not subcooled and at the outlet of the evaporator not superheated. Thus refrigerant leaves the evaporator which could still absorb heat from the compartment. Furthermore, T_c has already decreased and is about $-18\,°\text{C}$. At the suction line, $T_{Suc} > T_{Amb}$ holds and T_{Suc} is close to the temperature of the shell. This is possible due to the temperature at the condenser outlet which is above T_{Amb}. From then on the process cycle starts to move rightwards in the Ts-diagram as can be seen in figure V.5f. 386 s after the compressor startet, T_c dropped below $-20\,°\text{C}$ in the compartment.

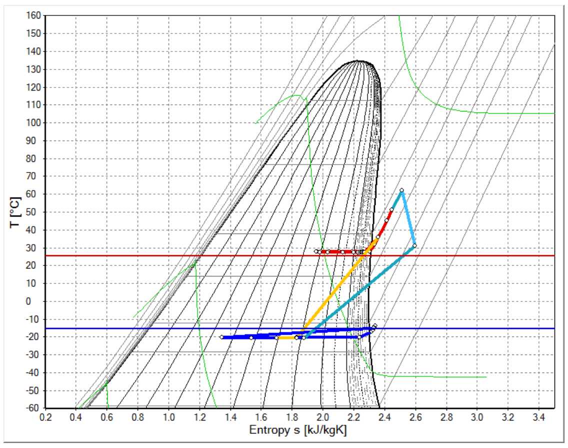
We proceed with figure V.6 where the process is illustrated when the compressor shuts down and is idle. In figure V.6a, the Ts-diagram just before the compressor starts to shut down is shown. The compartment temperature T_c has already decreased below $-25.5\,°\text{C}$. In figure V.6b the NOR of the compressor is decreasing to 20 rpm. The circulating mass flow in the refrigeration cycle drops. Hence, the refrigerant inside the internal discharge line of the compressor gives off more heat to the shell and the condenser inlet temperature is lower. Since the mass flow in the capillary is now higher than the one in the compressor, the condenser empties. Also $\dot{Q}_{Cap}$ drops and the inclination of the yellow line (capillary) changes from left to right. This implies that the enthalpy and entropy at the evaporator inlet rise. The whole cycle starts to collapse. Due to the small mass flows, $\dot{Q}_{Cap}$ is turned off (manually) shortly after, see figure V.6c. This is visible since the right yellow line now vanished. Therefore the expansion in the capillary is isenthalpic and the shell inlet temperature drops from over $30\,°\text{C}$ to about $-20\,°\text{C}$. The condenser further empties. In figure V.6d, only the last Unit of the condenser, the dryer, is two-phase. Also superheated refrigerant flows into the evaporator. This evolution proceeds and superheated refrigerant flows through most of the volumes in the heat exchangers, see figure V.6e. Recall that in the Ts-diagram the flow is illustrated. Finally, in figure V.6f the process cycle is collapsed as at time $t = 0\,\text{s}$ and the T_c starts to rise slowly until the compressor is switched on again.

Various Ambient Temperatures

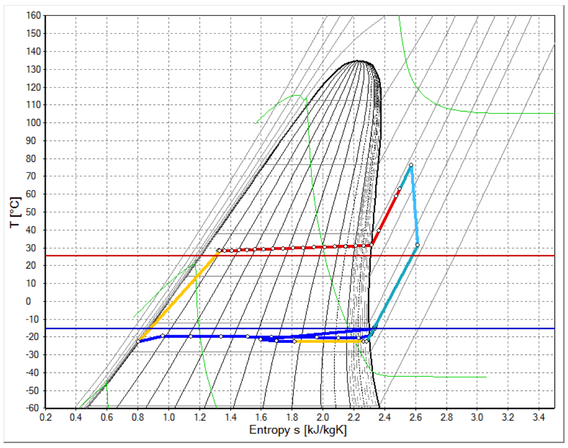
Next, the dynamic behaviour of the validated refrigeration appliance model for various ambient temperatures is discussed shortly. In chapter IV, the model is compared with measurements for the ambient temperatures $25\,°\text{C}$ and $43\,°\text{C}$ and results are given in figure IV.4 and IV.5, respectively. In figure V.7, the same variables are illustrated except that the temperature of the condenser pipe wall at 50 % length is substituted by the compartment air temperature. In figure V.7a the compressor power is



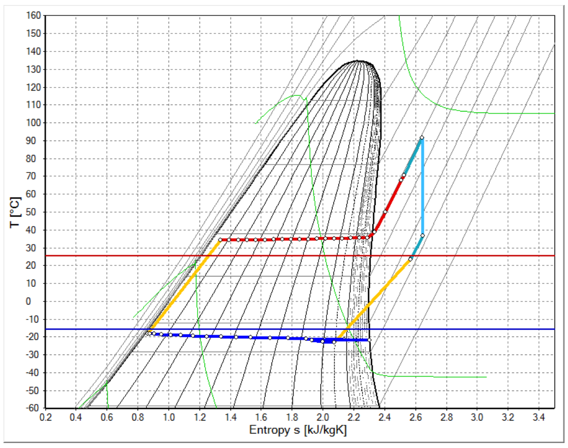
(a) The compressor turns on. $t = 0\,\mathrm{s}$.



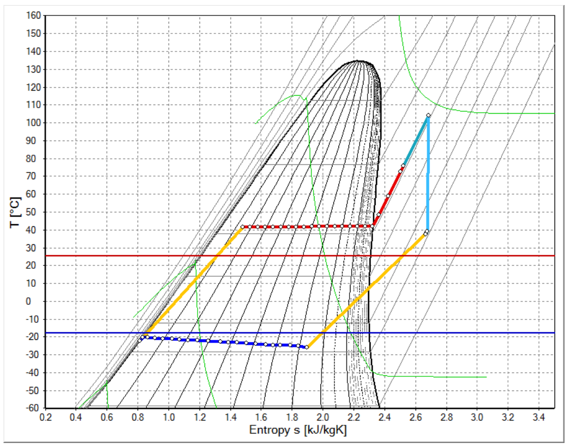
(b) $t = 5.76\,\mathrm{s}$.



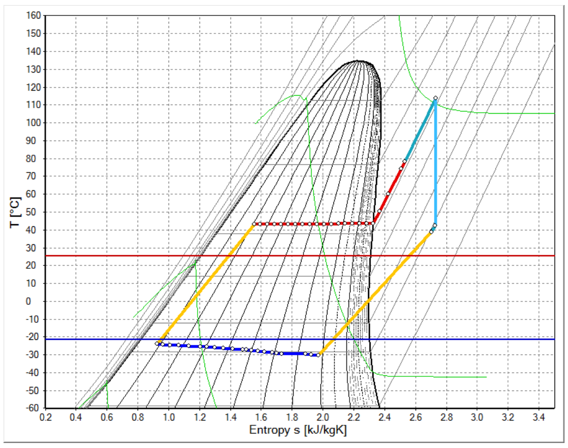
(c) $t = 15.14\,\mathrm{s}$.



(d) $t = 45.99\,\mathrm{s}$.

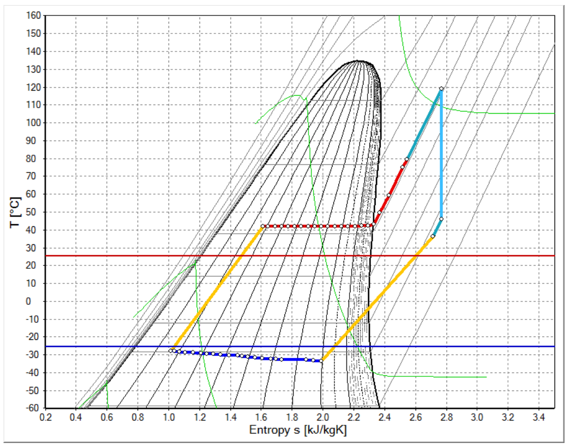


(e) $t = 135.19\,\mathrm{s}$.

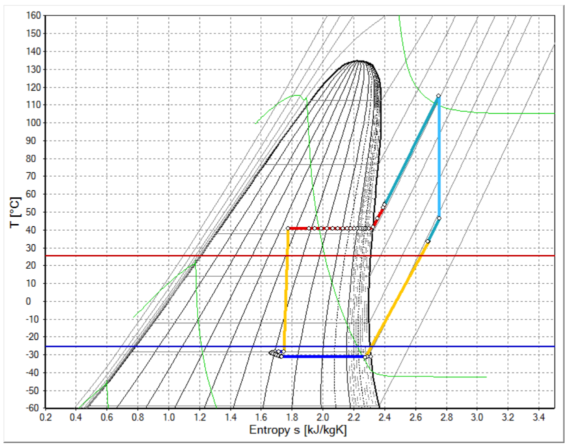


(f) $t = 386.16\,\mathrm{s}$.

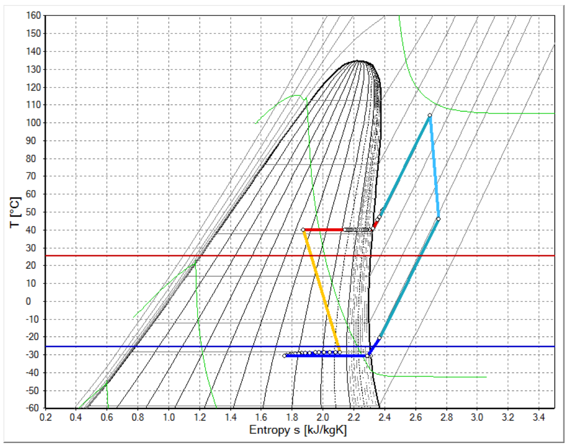
Figure V.5.: Ts-diagrams for one on-off-cycle. Part 1. Condenser (red), evaporator (dark blue), compressor (light blue), shell (turquoise), capillary and internal heat exchanger (yellow). Horizontal lines represent the ambient (red) and compartment (blue) temperature.



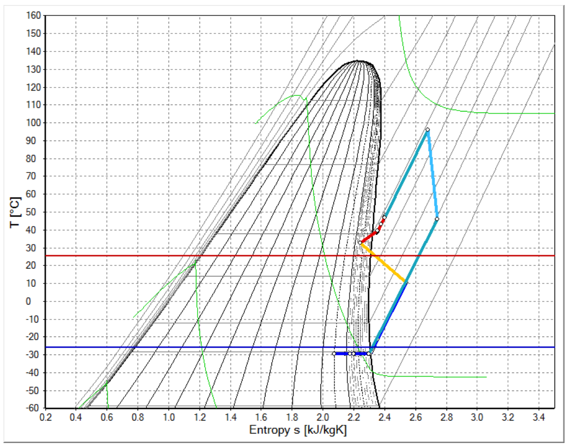
(a) $t = 761.16\,\mathrm{s}$.



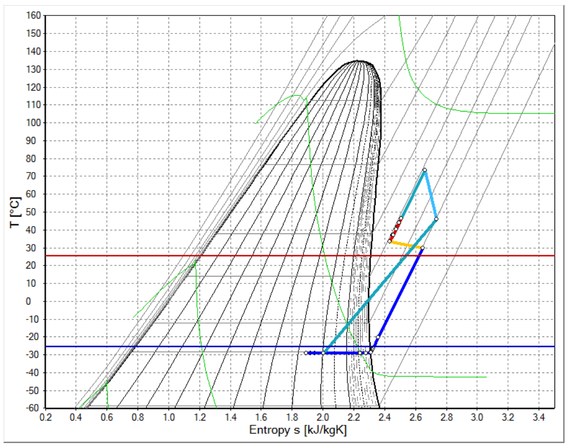
(b) $t = 779.91\,\mathrm{s}$. The compressor is turning off.



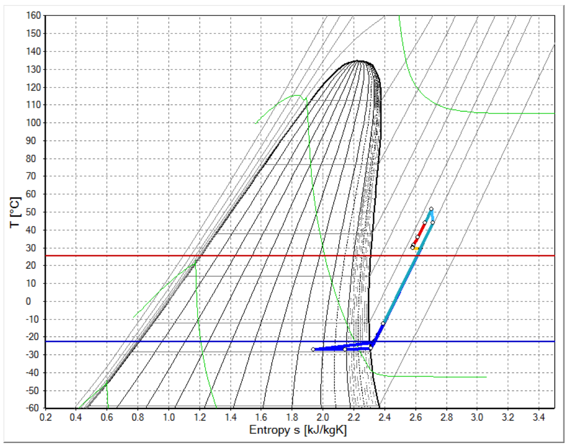
(c) $t = 782.54\,\mathrm{s}$. The heat transfer in the internal heat exchanger has been turned off.



(d) $t = 789.97\,\mathrm{s}$.

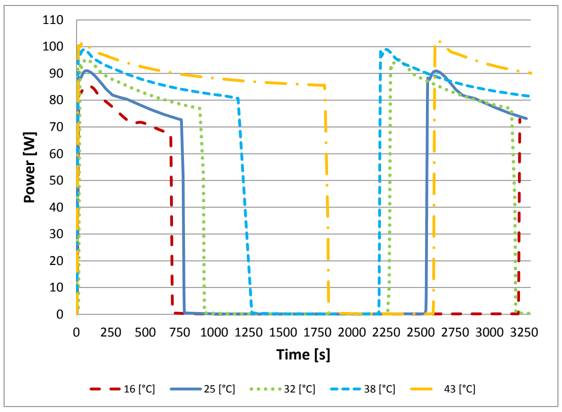


(e) $t = 833.16\,\mathrm{s}$.

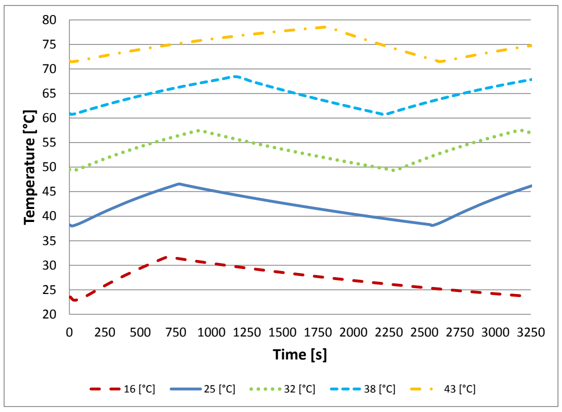


(f) $t = 1266.06\,\mathrm{s}$. The compressor will switch on again at $t = 2531.68\,\mathrm{s}$.

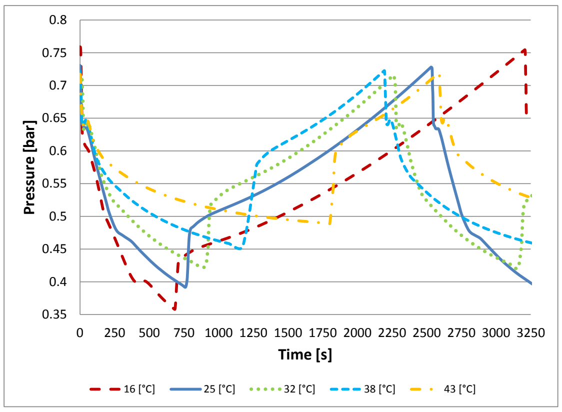
Figure V.6.: Ts-diagrams for one on-off-cycle. Part 2. Condenser (red), evaporator (dark blue), compressor (light blue), shell (turquoise), capillary and internal heat exchanger (yellow). Horizontal lines represent the ambient (red) and compartment (blue) temperature.



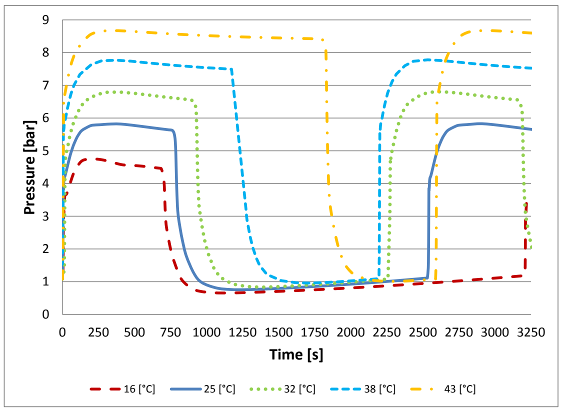
(a) Compressor electric power.



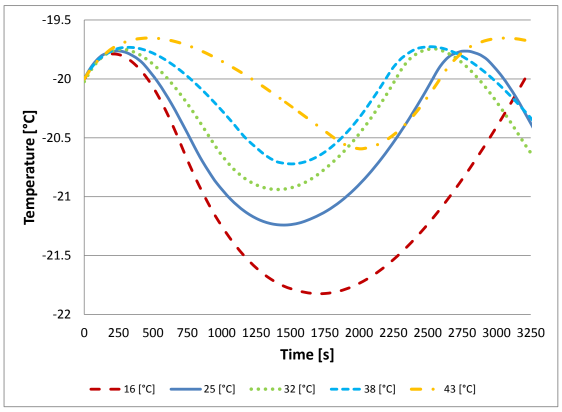
(b) Shell surface temperature.



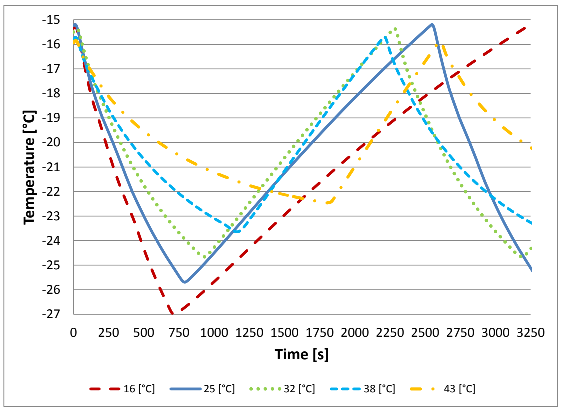
(c) Pressure at compressor inlet.



(d) Pressures at compressor outlet.



(e) Sensor temperature.



(f) Compartment temperature.

Figure V.7.: Values for various ambient temperatures.

shown. We see that for lower ambient temperature T_{Amb}, the maximum compressor power is smaller, the switch-on time is shorter and the switch-off time is longer. In figure V.7b, the surface temperature T_{Shell} of the compressor shell is given. The difference $|T_{Shell} - T_{Amb}|$ increases superlinearly with T_{Amb}. In figures V.7c and V.7d the suction and discharge pressure, respectively, are shown. Both pressures decrease monotonically with the ambient temperature. Finally, in figures V.7e and V.7f, the sensor T_{Sens} and compartment air T_{Air} temperature are illustrated. It can be seen that the difference between the maximal and minimal compartment air temperature increases as T_{Amb} decreases. Since the pressure in the evaporator is less for smaller T_{Amb}, the saturation temperature in the evaporator is smaller as well. Thus, the compartment is cooled faster which yields a lower T_{Air}. Due to the thermal inertia of the sensor, T_{Air} is less for smaller T_{Amb} when the compressor switches off. After the compressor switched off, $T_{Air} < T_{Sens}$ holds. Hence, T_{Sens} still decreases and the minimal temperature reached by the sensor is less for smaller T_{Amb}. Finally, this implies that it takes longer for smaller T_{Amb} until the upper temperature threshold is reached. This longer period allows T_{Air} to be higher for smaller T_{Amb} when the compressor switches on again.

Next, results of the one-dimensional spatially discretized heat exchangers are presented. In the following figures, the shown time interval corresponds to one compressor on-off-cycle as above and each finite volume, *i.e.* each Unit, is represented by one graph. The colour of the graphs range from blue (first volume) to red (last volume). The graphs correspond to the state inside the Unit. The condenser is discretized by 21 Units and the last condenser Unit ($C21$) corresponds to the dryer/filter which has a different geometry than the others. Hence, *e.g.* the mass differs in this Unit, see figure V.21, but does not imply a similar deviation in the density, see figure V.20. The evaporator is discretized by 20 Units and the third last Unit ($E18$) represents the accumulator. Therefore different states occur in this Unit and in the remaining two due to the outlet state of the accumulator. We discuss the condenser first.

Condenser

We start with the mass flow $\dot{m}_{feed}$ at the inlet of each Condenser-Unit illustrated in figure V.8. To this figure also the mass flow at the outlet of the dryer which equals the mass flow $\dot{m}_{Cap}$ through the capillary is added. Comparing with figure V.3a, in which the mass flow through the compressor $\dot{m}_{Comp}$ and $\dot{m}_{Cap}$ are given, the mass flow in each Unit hardly differs from $\dot{m}_{Comp}$. The dryer/filter has a larger volume than the other Units and acts in a compensatory way. When the compressor is idle, $\dot{m}_{feed}$ is reduced to a minimum in all Units.

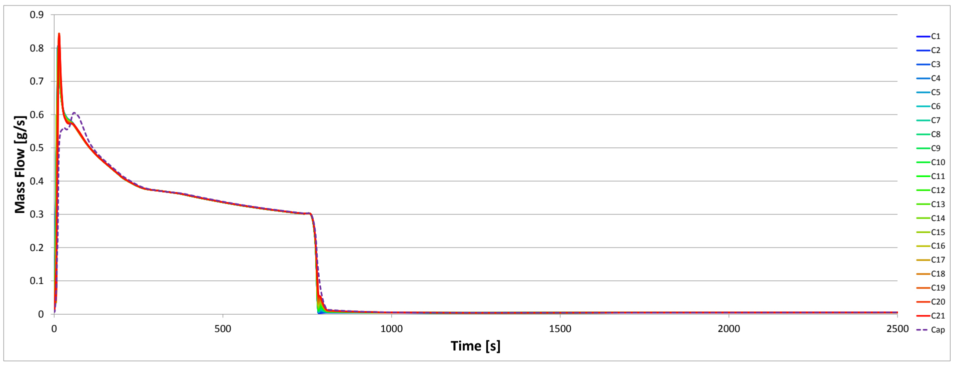


Figure V.8.: Mass flow $\dot{m}_{feedS}$ at the inlet of each Condenser-Unit and through the capillary.

The pressure in each Condenser-Unit is shown in figure V.9. When the compressor operates, the pressure rises to about 5.8 bar. The pressure drop, illustrated in figure V.10, is caused by friction. The mass flow has a large influence on the magnitude of the FPD. Furthermore, the FPD is higher if the vapour quality x is closer to 1.

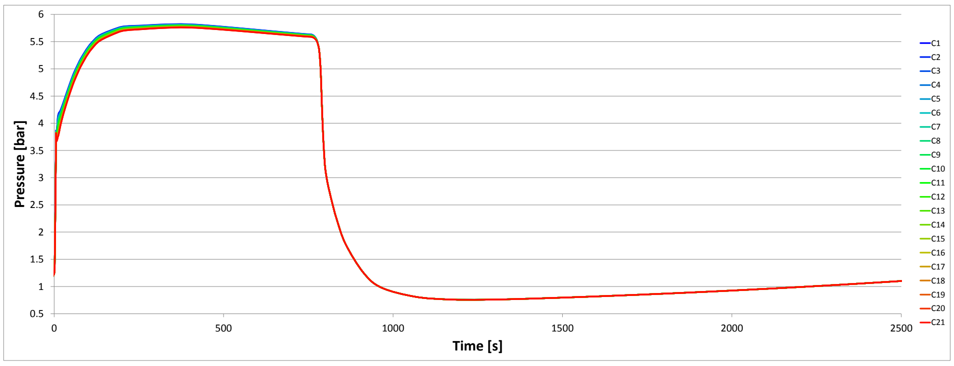


Figure V.9.: Pressure p in each Condenser-Unit.

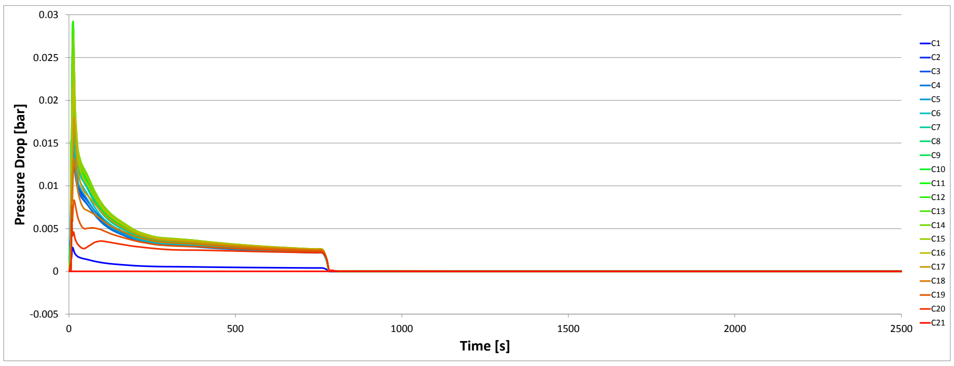


Figure V.10.: Pressure drop Δp in each Condenser-Unit.

Next, the temperature T of the refrigerant and temperature T_p of the pipe wall are discussed. In figure V.11, T for each Condenser-Unit is depicted. Right after the start of the compressor, the first three Units are superheated and the fourth one joins later. In the figure all other temperatures are fairly close. This results from the fact that in the whole two-phase region T only depends on the pressure. Due to the pressure drop, not all temperatures are equal in the two-phase Units. The temperatures are monotonically decreasing with the flow direction. This fact does not hold for T_p, see figure V.12. There are three Units having a remarkably higher and two a lower pipe temperature than the majority of the Units. This behavior comes from the non-constant HTC illustrated in figure V.13. The HTC is bounded by $2000\,\mathrm{W}/(\mathrm{m}^2\,\mathrm{K})$ as described in section III.8.3. In the single phase region, *i.e.* the first four Units, the HTC is relatively low. In the first three Units T is significantly higher than the saturation temperature T_{sat}. This together yields a higher T_p for the first three Units. In the next two Units, T is close to T_{sat}. Unit $C4$ is still superheated and $C5$ is in the interpolation interval between the single- and two-phase region. This yields a relative low HTC and hence a smaller T_p. The remaining Units are all two-phase and have a very high HTC which implies that $|T - T_p|$ is small. Therefore, T_p is higher there than in the Units $C4$ and $C5$.

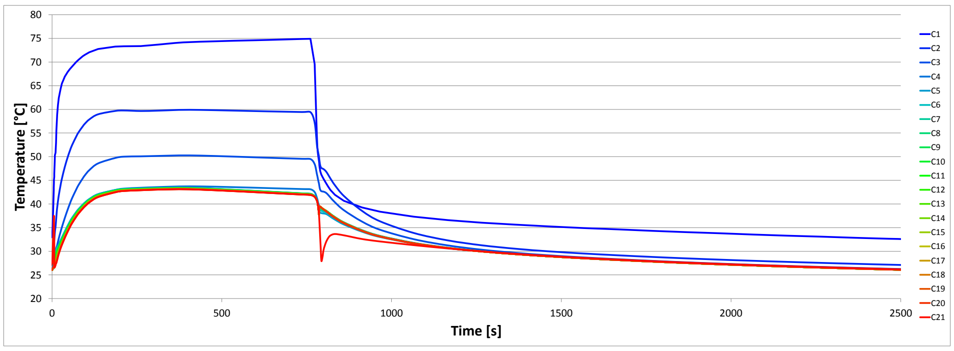


Figure V.11.: Temperature t in each Condenser-Unit.

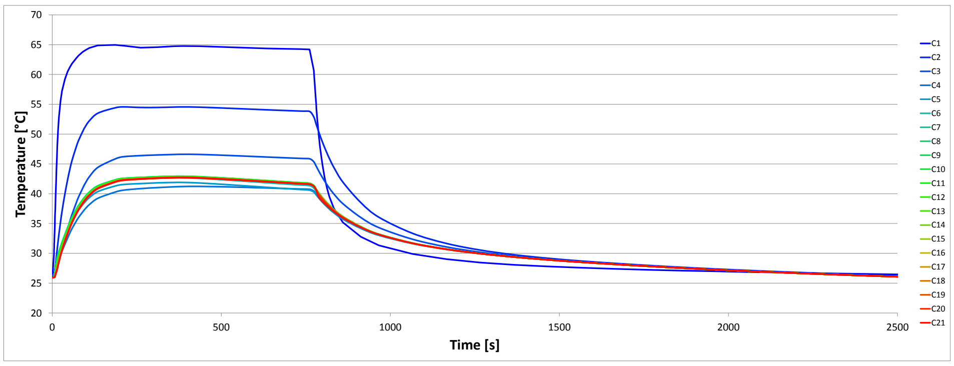


Figure V.12.: Pipe temperature t_p in each Condenser-Unit.

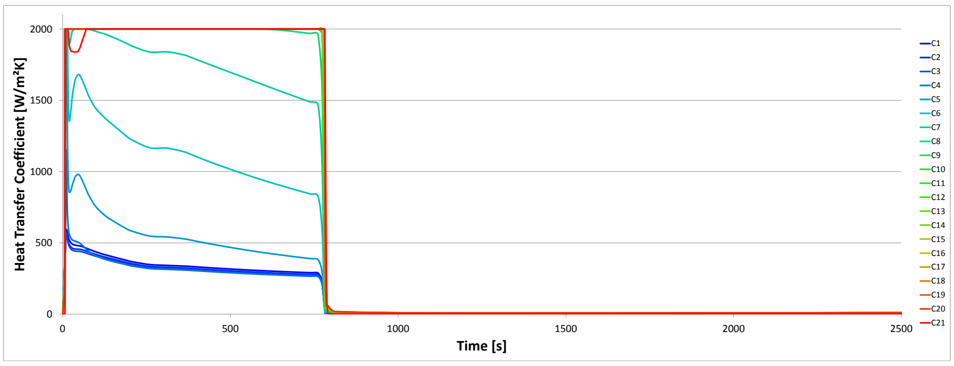


Figure V.13.: Heat transfer coefficient α in each Condenser-Unit.

The specific enthalpy h and specific inner energy u for the Condenser-Units are shown in figures V.14 and V.15, respectively. When the compressor operates, the difference between the values of h and u at the condenser inlet and outlet increases. The difference between the graphs is influenced by the heat transferred to the pipe wall. This heat flux $\dot{Q}$ is depicted in figure V.16. The negative values during the switch-on period of the compressor imply that heat is transferred from the refrigerant to the pipe wall. When the compressor shuts down, the pressure, and thus the temperature, drops in the Condenser-Units. Due to the thermal inertia of the pipe wall, T_p decreases more slowly. Thus the heat flux changes sign and the refrigerant absorbs heat from the pipe wall.

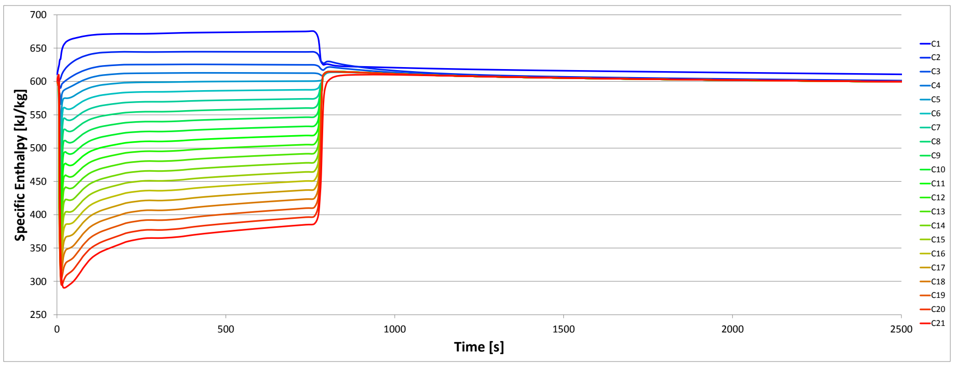


Figure V.14.: Specific enthalpy h in each Condenser-Unit.

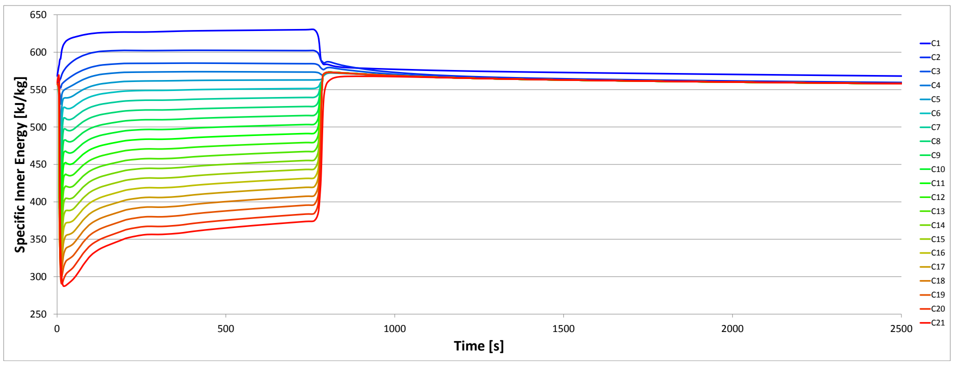


Figure V.15.: Specific inner energy u in each Condenser-Unit.

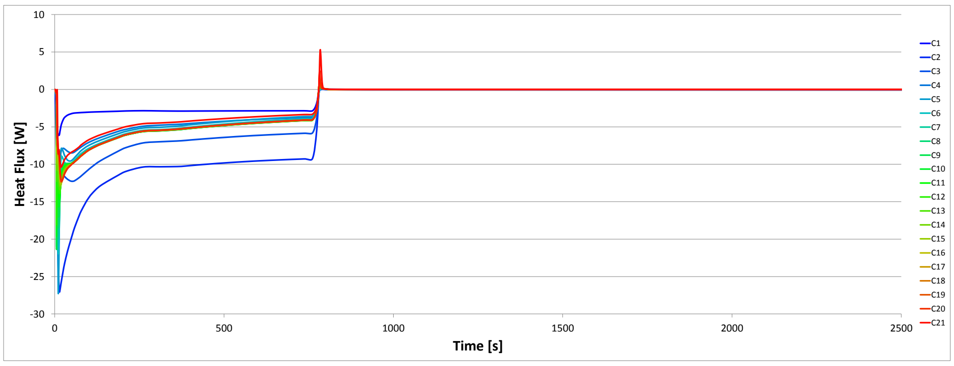


Figure V.16.: Heat flux $\dot{Q}$ in each Condenser-Unit.

The vapour quality x, shown in figure V.17, behaves as the specific enthalpy. This is not surprising when thinking of its definition in equation (6.25) on page 57. A vapour quality $x > 1$ and $x < 0$ indicates superheated and subcooled refrigerant, respectively. As claimed above the first three Condenser-Units are superheated (almost) right after the compressor switches on and a fourth Unit becomes fully vaporized a little bit later. Still, as can be also seen in the Ts-diagrams depicted in figure V.5, no Condenser-Unit becomes subcooled. If the compressor is switched off, the condenser empties and only superheated refrigerant remains. A similar behaviour as the vapour quality is observed in the flow quality $\dot{x}$, see figure V.18. In the two-phase region $\dot{x} \geq x$ holds due to the vapour void fraction model. In the single-phase region the two quantities are equal by definition.

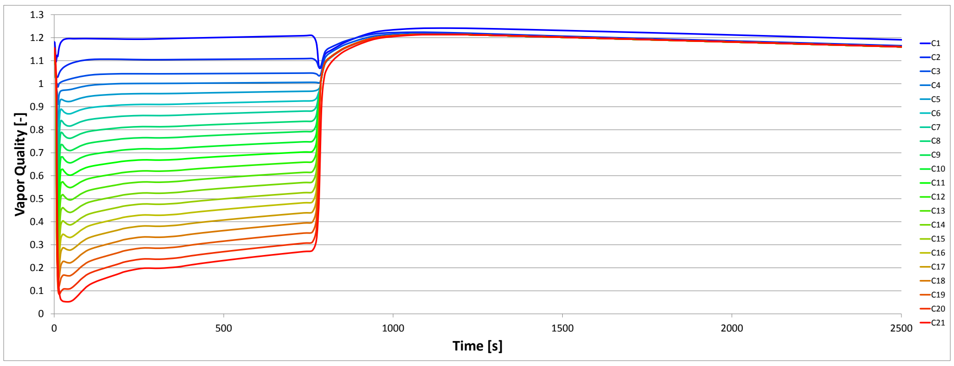


Figure V.17.: Vapor quality x in each Condenser-Unit.

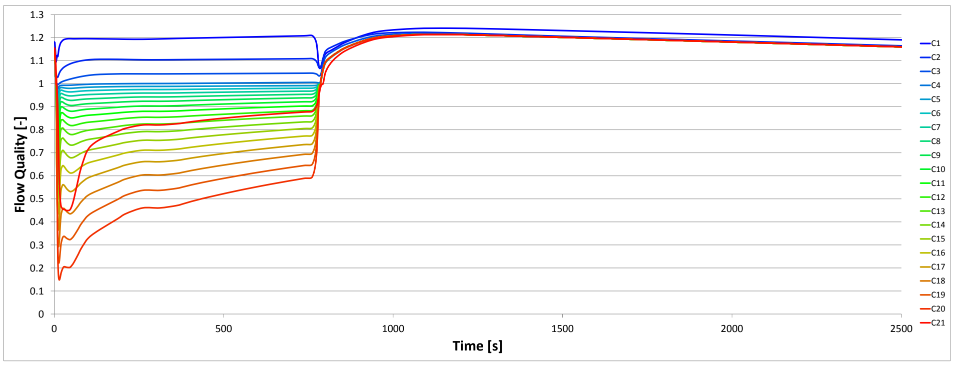


Figure V.18.: Flow quality $\dot{x}$ in each Condenser-Unit.

Concluding the condenser, the vapour void fraction, density and mass are presented. Recall section III.6.1.2, where the vapour void fraction ε is defined as the ratio between the cross-sectional area filled with vapor A_v and the total area A. Hence, $0 \leq \varepsilon \leq 1$ holds and the boundaries are only reached if the refrigerant becomes single-phase. In figure V.19 the graphs for ε are illustrated. In the two-phase region, the density ρ, depicted in figure V.20, is an interpolant between the densities ρ_l, ρ_v on the two-phase boundaries and ε determines the share of ρ_l and ρ_v. Since the pressure does not change significantly after the compressor reached 3000 rpm, ρ is strongly linked to ε. Finally, the stored mass of refrigerant in each Condenser-Unit is shown in figure V.21. As the volume is constant over time, the graphs correspond to the ones of the density. In the dryer/filter the mass is significantly higher than in the other Units since it has a different geometry and a larger volume V.

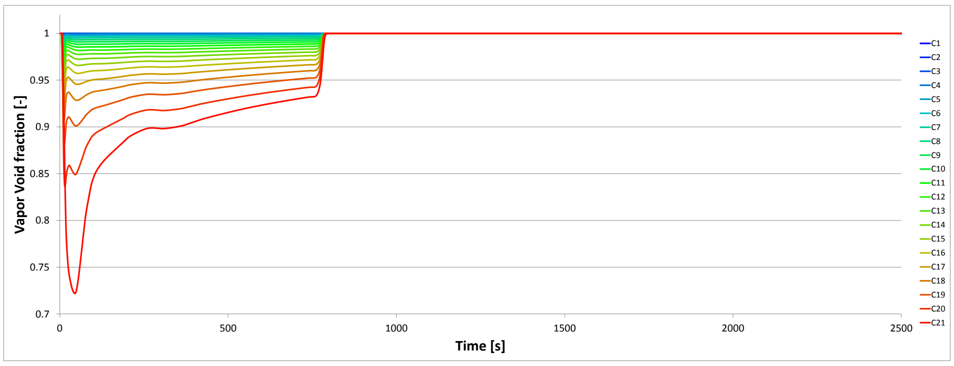


Figure V.19.: Vapor void fraction ε in each Condenser-Unit.

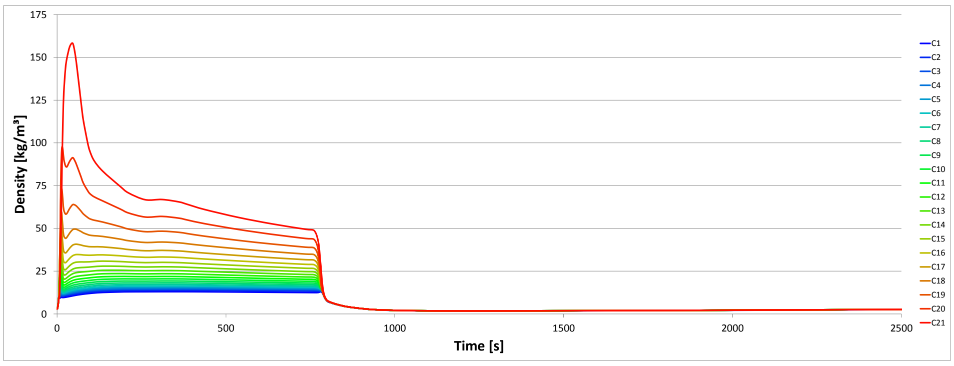


Figure V.20.: Density ρ in each Condenser-Unit.

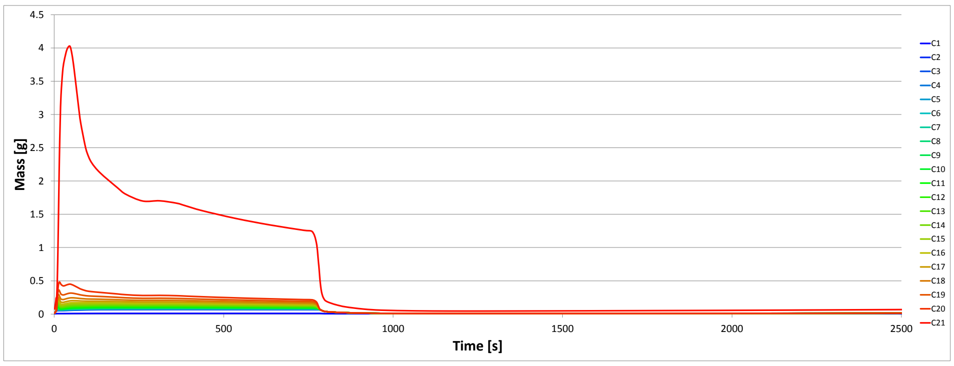


Figure V.21.: Mass m in each Condenser-Unit.

Evaporator

In the following the same order of the figures as for the condenser is preserved. Therefore, first the mass flow $\dot{m}$ at the inlet of each Evaporator-Unit is discussed. The inlet mass flow of the first Unit $E1$ (blue dotted) equals the mass flow $\dot{m}_{Cap}$ through the capillary. The mass flow of the last two Units corresponds to $\dot{m}_{Comp}$, see figure V.3a. The artificial accumulator, Unit $E18$, acts in a compensatory way. Its inlet mass flow (black dashed) does not show the same behaviour as $\dot{m}_{Comp}$ (red line). Since the outlet enthalpy of the accumulator is prescribed by equation (6.32) on page 60 and thus by the efficiency ξ gained from equation (6.31), the inlet mass flow has to adjust appropriately because the inlet enthalpy is more or less fixed by the downstream Unit. Consequently, the same behaviour of the mass flows may not appear in reality.

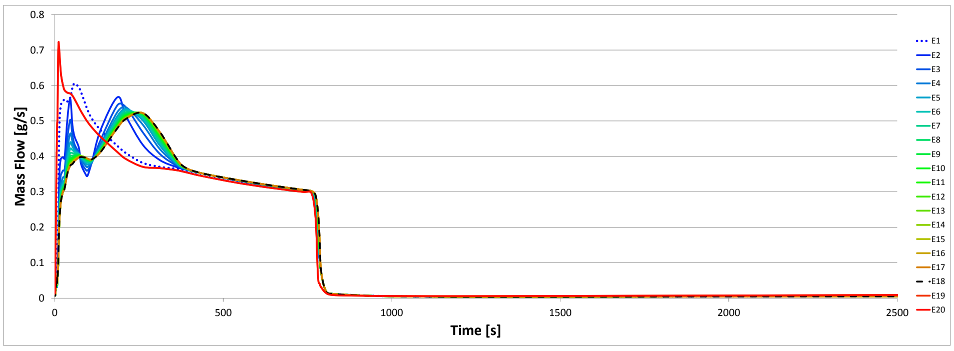


Figure V.22.: Mass flow $\dot{m}_{feedS}$ at the inlet of each Evaporator-Unit.

Next, the pressure in the Evaporator-Units illustrated in figure V.23 is presented. The pressure drop Δp in each Evaporator-Unit is higher than in the Condenser-Units. The reason lies particularly in the correction factor $\eta_{\Delta p\_Ev} = 4.7$ which was validated in chapter IV. The results are given in figure V.24. Thus, the model yields a total pressure drop of up to 0.2 bar for the whole evaporator. Also in reality a higher pressure drop is expected in the evaporator than in the condenser since it is vertically wound around the compartment and therefore several bends occur. The time evolution of Δp is linked to the mass flow. During the switch-off period of the compressor, the pressure in the evaporator slowly rises due to the heat input from the ambiance.

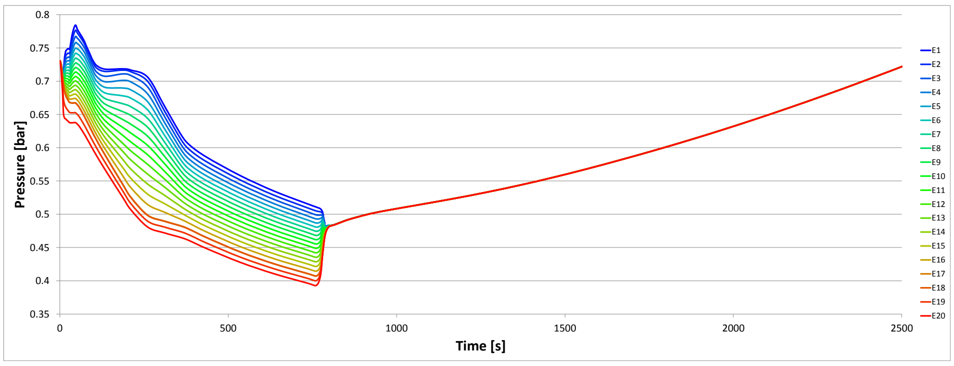


Figure V.23.: Pressure p in each Evaporator-Unit.

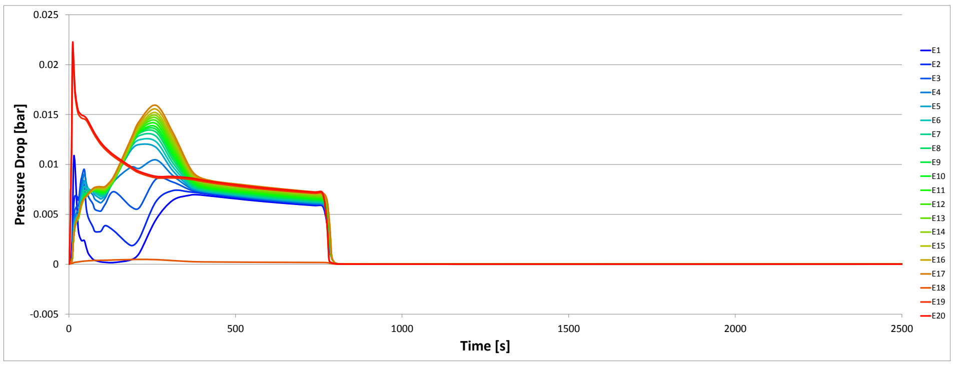


Figure V.24.: Pressure drop Δp in each Evaporator-Unit.

When the compressor operates, all Evaporator-Units are two-phase. Therefore, the difference in the temperatures depicted in figure V.25 is caused by the pressure drop from figure V.24. When the compressor is idle, the pressure drop is reduced to a minimum due to the small residual mass flow and the temperatures rise with the pressure. However, the Units become superheated and the temperatures differ in each Unit further on. The temperature T_p of the pipe wall is illustrated in figure V.26. Since always five Units are connected to one insulation Unit, see the flow sheet in figure III.22, the difference in T_p are explainable. Each insulation Unit has a uniform temperature. Since the refrigerant temperature drops along the evaporator due to the pressure drop, the discrete values in the insulation lead to a jump in the pipe wall temperature every five Units. The HTC between the refrigerant and the pipe wall are given in figure V.27. Compared to the condenser the values are much lower and are not affected from the upper bound. As for the pressure drop the graphs correspond to the mass flow. A higher HTC is achieved in Units with a lower flow quality $\dot{x}$. When the compressor is idle, the small mass flow and the fact that most Units are single-phase induce a small HTC.

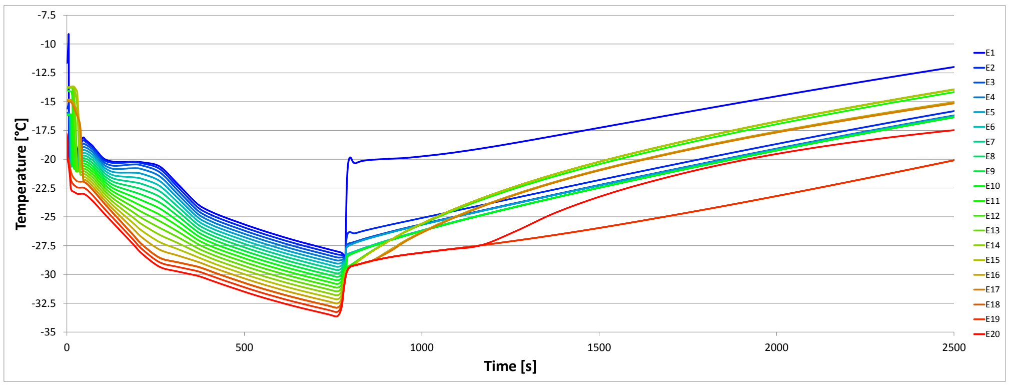


Figure V.25.: Temperature t in each Evaporator-Unit.

Next, the specific enthalpy h and specific inner energy u of the Evaporator-Units are discussed. The corresponding graphs are shown in figure V.28 and V.29, respectively. During the switch-off period of the compressor, the values of all Units except the Accumulator-Unit $E18$ are close to each other. In the accumulator a major part of the refrigerant is stored. Since the volume is fixed, the liquid share in these Units has to be much higher than in the other Units. This can be seen by a significantly lower h and u. Furthermore, the outlet enthalpy of the accumulator is calculated differently, see equation (6.32) on page 60, which yields the gap between the other Evaporator-Units in the graphs when the

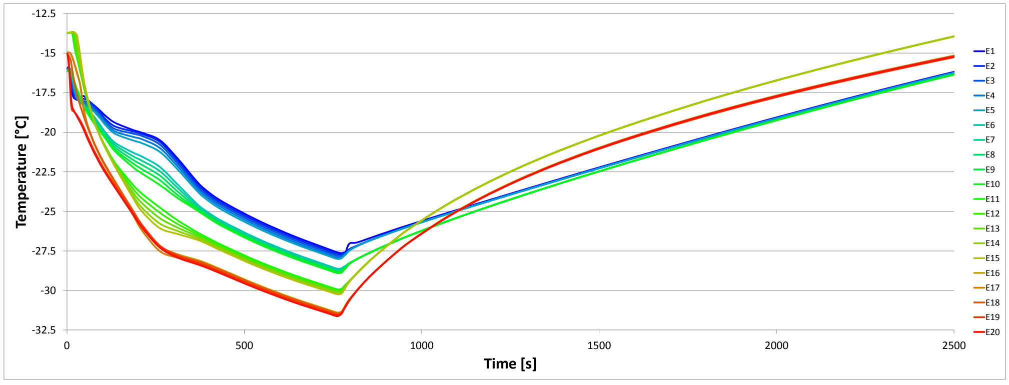


Figure V.26.: Pipe temperature t_p in each Evaporator-Unit.

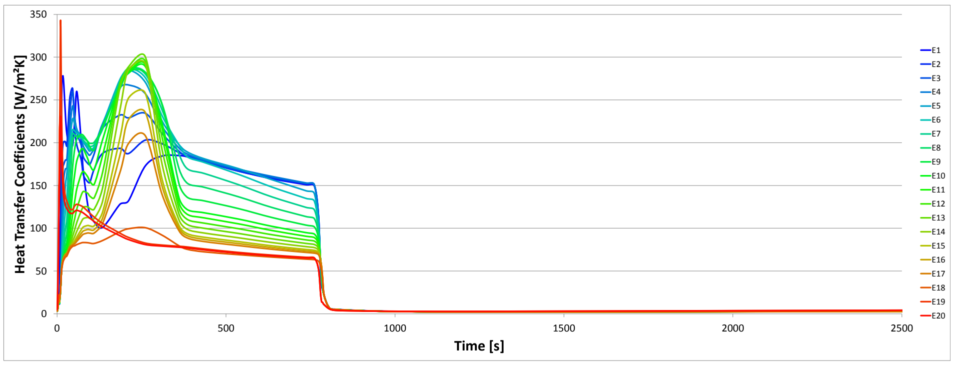


Figure V.27.: Heat transfer coefficient α in each Evaporator-Unit.

compressor operates. During the switch-on period of the compressor, refrigerant with a vapour quality x close to 0 leaves the capillary and fills the Evaporator-Units. Due to the heat absorbed from the insulation and compartment, the enthalpy and inner energy increase with each Unit. The bump in h and u results from the change in the mass flow. If the mass flow is higher, the enthalpy difference from one Unit to the next decreases. Thus, these two quantities are inversely proportional.

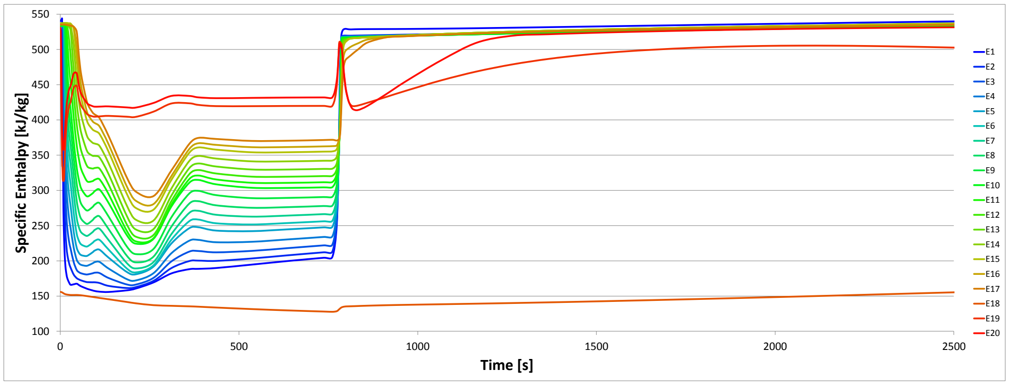


Figure V.28.: Specific enthalpy h in each Evaporator-Unit.

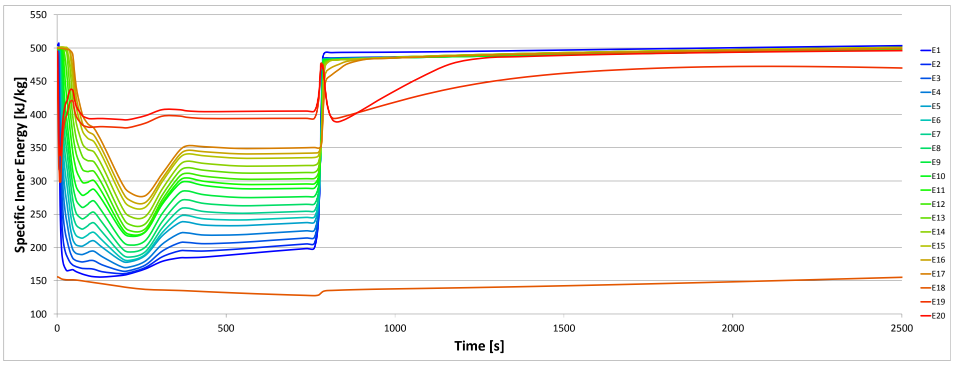


Figure V.29.: Specific inner energy u in each Evaporator-Unit.

The heat transfer $\dot{Q}$ between the refrigerant and the pipe wall is depicted in figure V.30. Positive values imply that the refrigerant absorbs heat from the pipe wall. When the compressor switches on, for a short period some Units heat the pipe wall which is visible due to the negative peak at the beginning of the time axis. The Accumulator-Unit $E18$ has a higher $\dot{Q}$ when the compressor operates since it has a different geometry and a larger surface area than the other Evaporator-Units. Hence, this does not imply that the heat transfer is higher in the Accumulator-Unit. More likely the opposite is the case when regarding the HTC in figure V.27. When the compressor switches off, the refrigerant heats the pipe wall, as shown by the second negative peak.

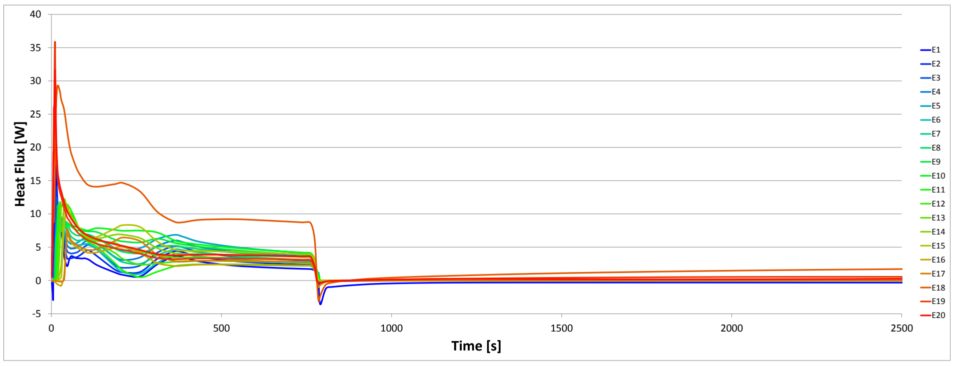


Figure V.30.: Heat flux $\dot{Q}$ in each Evaporator-Unit.

Next the vapour and flow quality are presented. Their values for each Unit in the evaporator are given in figure V.31 and V.32. As in the condenser the vapour quality x shows the same behaviour as the specific enthalpy h. Since the accumulator holds most of the refrigerant the vapour quality for this Unit is very close to 0. When the compressor operates, $0 \leq x \leq 1$ yields that the Units are two-phase. During the switch-off period most of the Units are superheated but still the Accumulator-Unit is filled with liquid refrigerant. As in the condenser the flow quality is higher than the vapour quality since it is assumed that the vapour share flows with a higher velocity than the liquid share. Since the flow quality does not make sense in the accumulator it is omitted in figure V.32.

Concluding the results of the refrigeration appliance model, the vapour void fraction, density and mass are discussed. The vapour void fraction ε is illustrated in figure V.33. As above, the Accumulator-Unit has a special position in these graphs. When the compressor is idle, more than 90 % of the

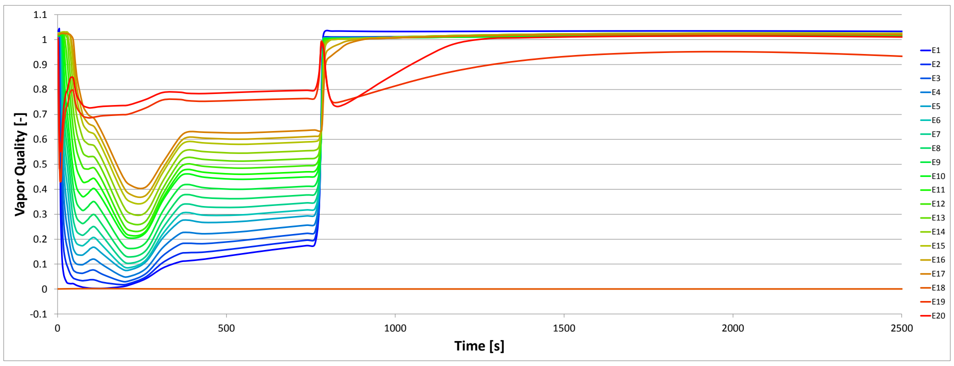


Figure V.31.: Vapor quality x in each Evaporator-Unit.

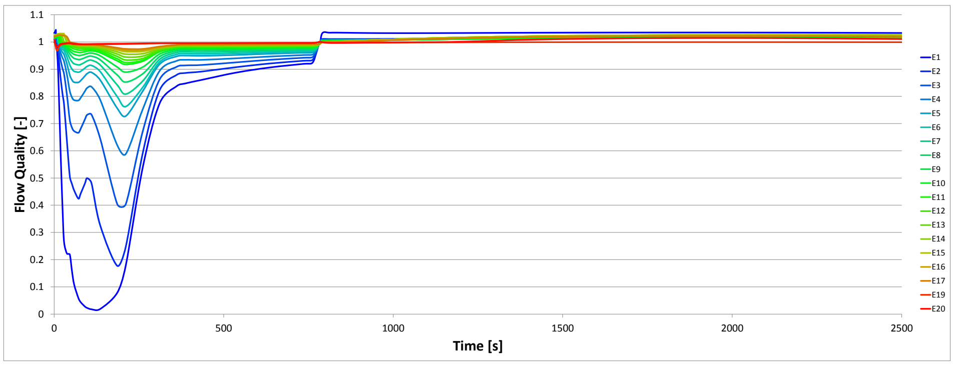


Figure V.32.: Flow quality $\dot{x}$ in each Evaporator-Unit.

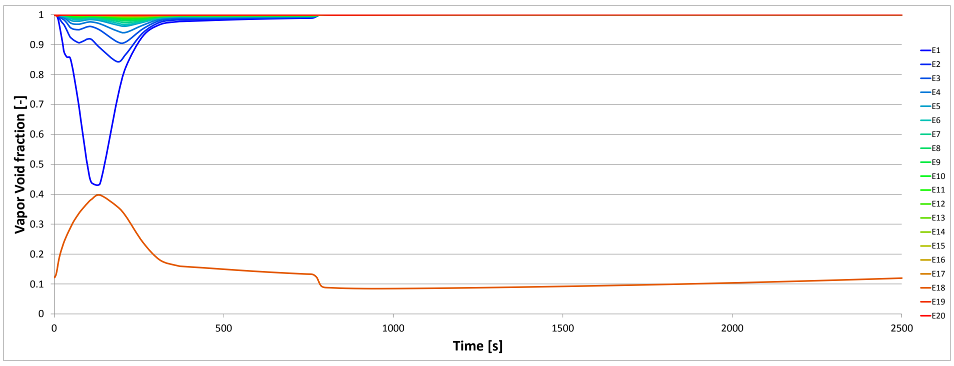


Figure V.33.: Vapor void fraction ε in each Evaporator-Unit.

cross-sectional area of the tube in the accumulator is filled with liquid whereas in the other Units only vaporized refrigerant is stored. After the compressor switched on, a bigger amount of refrigerant leaves the accumulator. Thus, ε increases and density and mass decrease. The latter two variables are illustrated in figure V.34 and V.35, respectively. Shortly after that refrigerant with a low vapour quality flows into the first Evaporator-Unit and the Units are filled with two-phase refrigerant. At the maximum filling level, less than 45 % of the cross-sectional area are filled with vapour in the first Unit. This corresponds to a filling quantity of about 15 g. At the same time, ε increases to about 0.4 in the accumulator before the mass is transferred back. Consequently during the whole simulation a major part of the refrigerant is located in the accumulator.

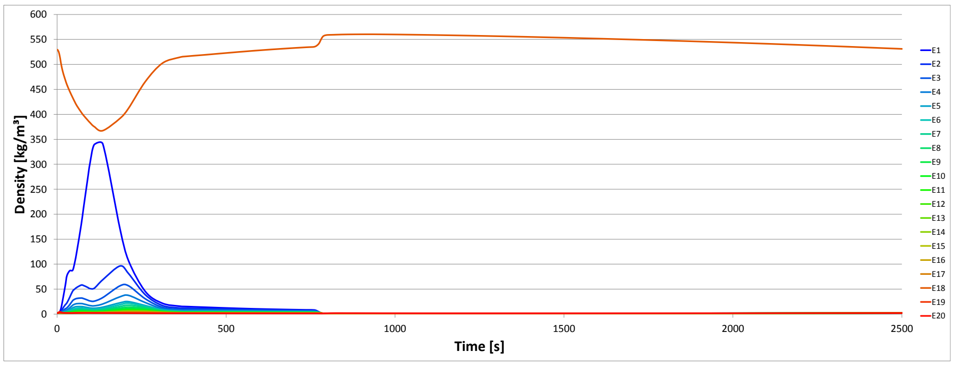


Figure V.34.: Density ρ in each Evaporator-Unit.

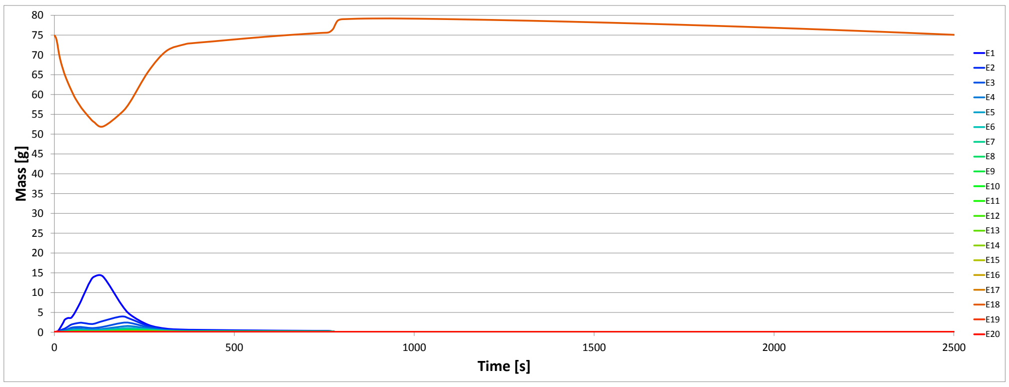
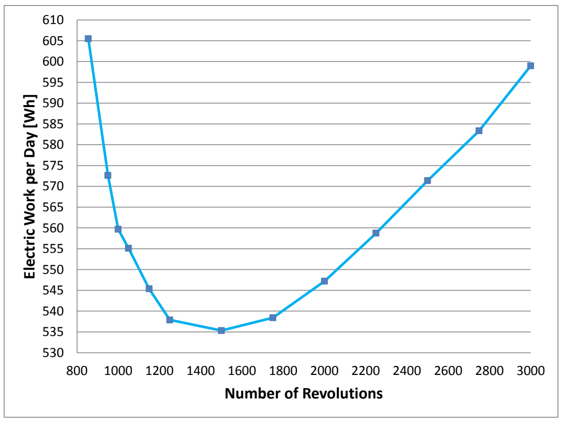


Figure V.35.: Mass m in each Evaporator-Unit.

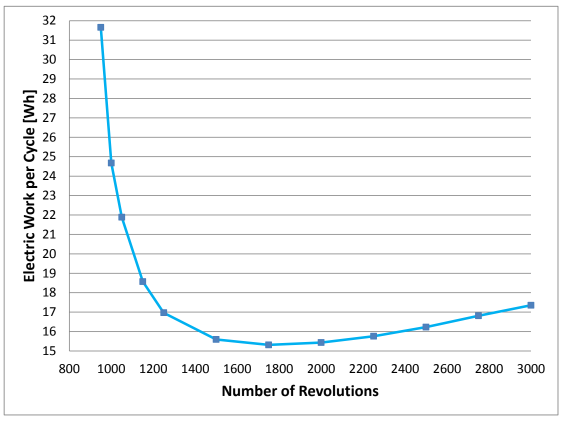
2. Optimal Number of Revolutions of the Compressor

Concluding this chapter a first optimization application of the developed and validated model in IPSEpro is presented. The impact of the prescribed Number of Revolutions (NOR) of the compressor on the energy consumption is investigated. As for the validation the compartment is empty, the door is never opened and the NOR is constant for each case. Furthermore, the fit parameters η_v and η_{comb} in the Compressor-Unit remain equal for each NOR. This assumption does not hold in reality. In this thesis no algorithm is applied to locate the optimal NOR, but 12 discrete values are investigated. If an optimization with respect to more than one variable is aimed for, we highly recommend to chose one of the many methods of mathematical optimization referred to in the beginning of chapter IV.

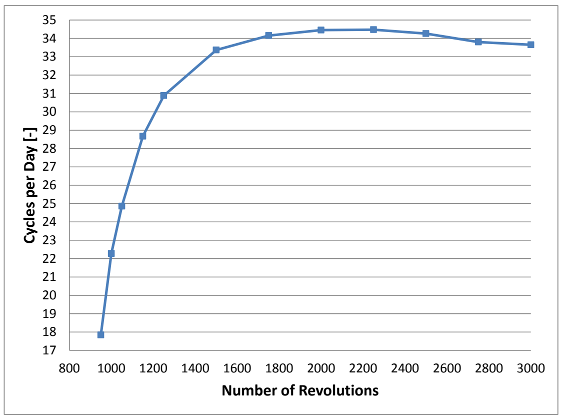
In figure V.36 the results are illustrated. The simulation for each NOR starts from a steady-state point and is performed until the periodic on-off-cycle is reached. To obtain the energy consumption per cycle, shown in figure V.36b, the time integral of the electric power P of the compressor was determined for



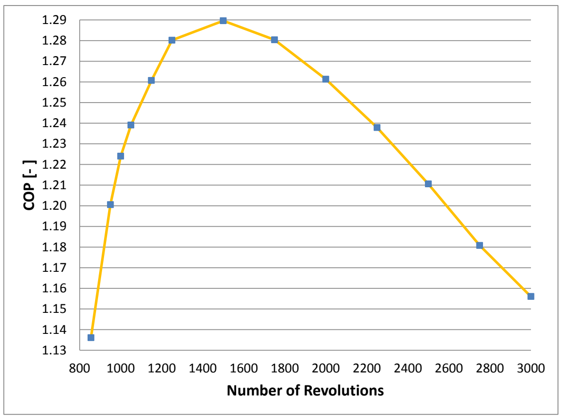
(a) Electric work per day.



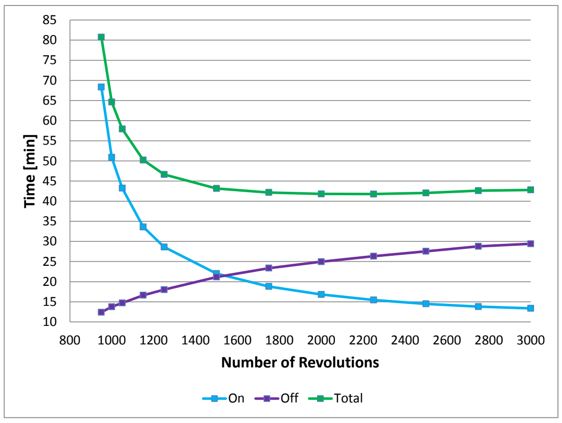
(b) Electric work per cycle.



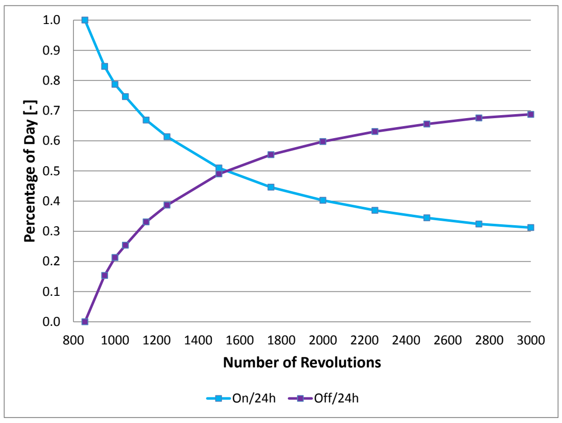
(c) Cycles per day.



(d) Coefficient of Performance ε_d.



(e) Time per cycle.



(f) On/Off-Percentage per day.

Figure V.36.: Results for the refrigeration appliance model using various NOR.

the on-period of the periodic on-off-cycle using the midpoint rule and a fixed start up loss of 0.45 Wh is added. The consumed energy per cycle is minimal if the NOR is 1700 rpm. The number of cycles per day, illustrated in figure V.36c, is the number of periodic on-off-cycles passed in 24 h. The longer one cycle takes the less cycles are passed per day. Thus, from figure V.36c we deduce that for NOR less than 1500 rpm the total time for one on-off-cycle increases significantly. The energy consumption per day, depicted in figure V.36a, is calculated by multiplying the consumed energy per cycle with the number of cycles per day. The start up loss, as long as it is not chosen extremely high, does not have a big impact on the optimum since the number of cycles per day remains almost equal for NOR between 1500 rpm and 3000 rpm. We obtain that for a NOR of 1500 rpm the model predicts the least energy consumption per day. The consumed energy per day is 10.63 % less for 1500 rpm than for the original 3000 rpm. However, in reality the same improvement may not be achieved due to the assumptions on the compressor model. In this optimal point also the COP ε_d defined in equation (1.4) on page 3 is maximal, see figure V.36d.

In figure V.36e, the on-duration (blue) and off-duration (purple) of the compressor and their sum (green), *i.e.* the time it takes until one full cycle is passed, are illustrated. Of course, the on-duration of the compressor decreases as the NOR increases but the total on-off duration for one cycle is nearly constant for NOR between 1500 rpm and 3000 rpm. The reason lies in the thermal inertia of the sensor. In figures V.37 and V.38 the temperature evolution of the compartment air T_c and sensor T_s, respectively, are illustrated. It can be seen that the time derivative dT_c/dt steepens during the switch-on period of the compressor as the NOR is increased. If the compressor is switched off, the derivative dT_c/dt, determined by the ambient temperature and the quality of the insulation, is almost equal for all NOR. Due to the thermal inertia of the sensor, T_c is lower for a higher NOR when the compressor switches off. Consequently, a shorter on- and longer off-duration is achieved for a higher NOR and the sum is similar for NORs between 1500 rpm and 3000 rpm. If the NOR is chosen less than 1500 rpm, the on-duration and thus total cycle period starts to increase significantly. Then the appliance becomes inefficient.

Finally, figure V.36f is discussed. This figure shows the percentage of the on- and off-duration per day. For a NOR of 1500 rpm the percentages are nearly equal which implies that the compressor operates about 12 h per day. If the NOR is chosen less, the percentage of the switch-on period of the compressor increases until the maximum of 1 is reached at a NOR of 878 rpm. Choosing this NOR implies that the compressor operates 24 h per day. Thus, this is a steady-state point. In this point the compartment temperature is −20.49 °C, just above the lower threshold of −20.5 °C specified in section IV.1.2. The threshold is not reached and therefore the compressor control never switches the compressor off. The NOR can be further reduced to 855 rpm where the corresponding compartment temperature is −20.07 °C, hence just below the upper threshold of −20 °C. If the NOR is further turned down, the consumed electric power decreases but the target temperature in the compartment is not achieved anymore. In figure V.36a the lowest depicted NOR is 855 rpm. The NOR of 878 rpm is not illustrated but the consumed energy per day would be slightly higher than for 855 rpm.

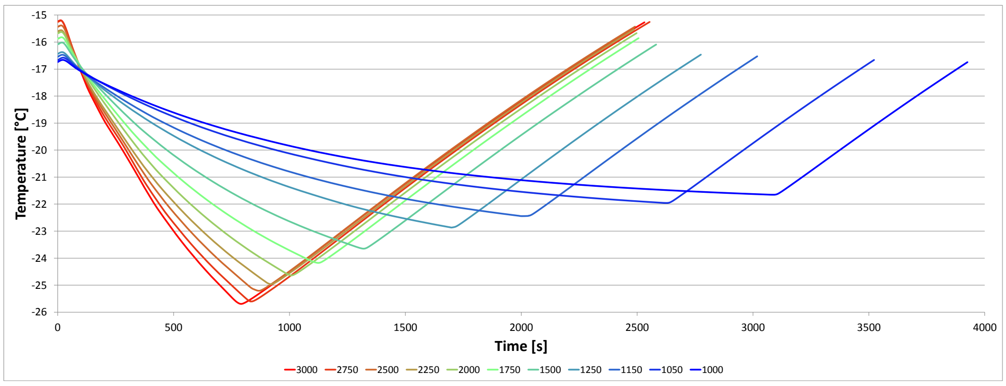


Figure V.37.: Compartment temperature for various NOR.

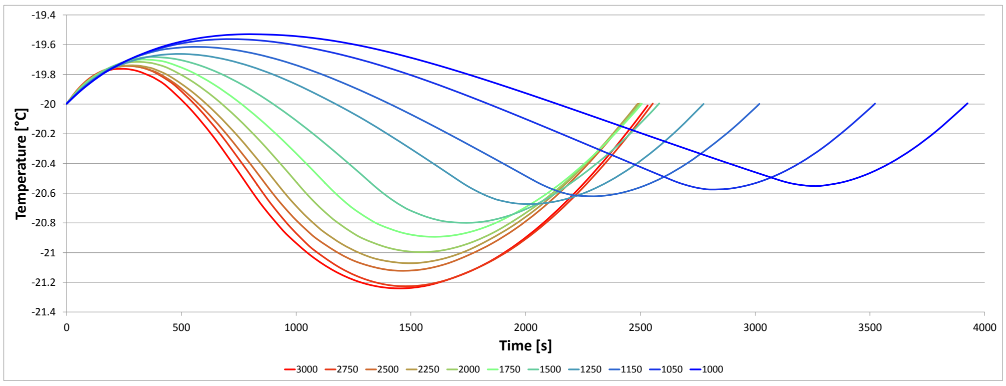


Figure V.38.: Sensor temperature for various NOR.

VI. Conclusion and Outlook

Within this final chapter the main parts of this thesis shall be summarized and once more investigated. Not only the positive aspects shall be presented but also the shortcomings. Furthermore, we give ideas and ask questions which emerged during the work for this thesis but are not yet pursued or elaborated. First, the numerical and then the model issues are discussed.

Numerical Issues

During the work for this thesis, a general purpose numerical solver for DAEs was incorporated into the commercial software IPSEpro. It is clear that for a commercial software the standards for the implementation are by far higher than for a typical research code, especially when it is general purpose. This includes error handling, efficient data management, performance and that the program must not crash in any case. Besides the incorporation of the numerical solver, the problem of the (consistent) initialization of the solver, which appears repeatedly in the literature, was handled. So far only the initialization but not the augmented initialization system is solved at the initial time before the dynamic solver is called. This procedure is sufficient for the models developed in this thesis since the emerging system of DAEs fulfills the conditions of corollary II.2.4. However, arbitrary systems with higher differentiation index can cause problems at the initial time although the initialization system was solved, see *e.g.* example II.2.2 or II.3.7. To set up the augmented initialization system, a method for automatic differentiation of the equations with respect to time and the Pantelides algorithm have to be implemented.

Further, regularization techniques for the initialization system were elaborated. The initialization system and flow sheet systems after the set up are underdetermined by nature which motivated the development of the Echelon Analysis. In large flow sheet models the overview of which variables can still be set is lost easily, even for experienced users. If a variable is set which already could be calculated from the equations, a (partial) overdetermination emerges in the system. To resolve this problem, the Adjoint Echelon Analysis was developed. However by following the Echelon Analysis, now a structurally and globally nonsingular initialization system is always achieved. Still, the Adjoint Echelon Analysis turned out to be a useful tool if the setting structure has to be changed. Since these two analyses are formulated in a general form, they are applicable for purely algebraic systems and DAEs. During the development of the Echelon Analysis the question arose if it is possible to identify the possible settings which lead to the minimal size of the blocks found by the block decomposition from section II.2.2. A smaller size of each block is expected to increase the chance of convergence. This question was not further investigated yet and is left for future work.

The integration termination criteria from section II.4.2 are implemented already in IPSEpro but are just the basis of a framework yet to be developed. The idea is that after the integration terminated due to a condition also an automatic restart shall be possible. At the "breakpoint" of the integration, the flow sheet model may be transformed arbitrarily. This is relevant if *e.g.* in a large model a valve closes and a part of the model becomes idle. Then this part may be excluded from the integration until the valve opens again. This is also of interest for the refrigeration appliance model since it is imaginable that a total shutdown of the compressor may be achieved by altering the models during the dynamic simulation.

Finally, the one-dimensional spatial discretization becomes tedious in IPSEpro if the number of finite volumes, *i.e.* Units, increases. At least in IPSEpro it is possible to duplicate an arbitrary number of Units, but in the heat exchanger models often convergence is not achieved if too many new Units are added before the next calculation is performed. Additionally, the appearance of the flow sheet is affected if a fine discretization is required. Therefore, further work may be invested to ease the handling of one-dimensional spatial discretization in flow sheet models.

Modelling Issues

Next the models and their implementation in IPSEpro are discussed. Within this thesis it was shown that also such complex models as for the heat exchangers, presented in section III.6.1, can be handled by IPSEpro. Implementation of most other component models was fairly straight forward. In the heat exchanger models, the calculation of the HTC, FPD and vapour void fraction was outsourced into an external DLL since the models are too complex to be handled by the MDL.

One major issue during the development process, in IPSEpro as well as in the VBA simulation tool, was the high number of discontinuities and non-differentiabilities emerging in the heat exchanger models. Some have a physical origin, some are caused by the switch between different empirical models for the HTC and FPD given for each flow pattern in the two-phase region, see section III.8.2. Interpolation at the flow pattern boundaries and the two-phase boundary is applied such that at least continuous external functions for the HTC and FPD are obtained. Still these models and the arising system can be handled by the numerical methods developed in section II.1.2. Since during Newton's method, used to solve the corrector equation at each integration step, all input arguments of the external functions from section III.7 are iterated and therefore do not necessarily fulfill the equations written in MDK, problems during the iteration emerged in IPSEpro which are not observed in the VBA simulation tool since in the VBA tool only one of the variables is iterated at a time. Physical properties such as λ, c_p, c_v or η, calculated from pressure and temperature, have a jump discontinuity in the saturation temperature and are not defined within the two-phase region. Therefore, the temperature passed to the external functions has to be checked such that correct physical properties are obtained and no convergence problems occur. However, the equation based approach of IPSEpro enabled to extend the heat exchanger models with the possibility of changing flow directions which is not available in the VBA simulation tool. In the validated refrigeration appliance model the flow direction does not change but it was the case for different sets of uncertain parameters. Furthermore, now also parallel heat exchangers involving splitting and mixing can be modelled and due to the graphical user interface of IPSEpro the connections of the component models are clearly visible. This, the fact that in PSE no model equations have to be considered when the flow sheet is set up and the Echelon Analysis guiding the user to a regular system are expected to ease modelling the refrigeration appliances; especially, for someone who did not take part in the component model development process.

In the future, the manufacturers who participate in the ECO-COOL research project aim to use IPSEpro and the models therein to design, investigate and improve their appliances. Besides that, the heat exchanger models are also applied within a different field of active research with a different working fluid. The models are utilized for a liquified natural gas tank system in trucks, see [86]. There, manufacturers already use IPSEpro to design the heat exchangers between the tank and the engine.

Potential of improvement lies in the shell and evaporator model. Considering the design of a compressor, the Shell-Unit model is very simple and a more sophisticated model should be elaborated. The implemented evaporator models are developed for horizontal tubes. Since here the evaporator is vertically wound around the compartment, an accumulator was incorporated to compensate the insufficiencies of the models. In the literature, *e.g.* in [3], also models are proposed for vertical tubes. To avoid the necessity of the accumulator, these models could be added to the MDK-library *Eco-Cool-Lib*.

Validation Issues

The component and flow sheet models developed in IPSEpro were dynamically validated with measurements, see chapter IV, and achieved satisfactory results. In comparison to the VBA models, where the validation was performed manually, the parameter fitting was semi-automatized. This means that for the results in this thesis the majority of the uncertain parameters is fitted using a numerical method and stationary measurements and the few remaining parameters are adjusted manually such that the model fits the dynamic, *i.e.* time dependent, measurements. This procedure accelerates the task of parameter fitting significantly. The numerical method is not implemented in IPSEpro directly but the scripting interface is used instead. It is written in JavaScript and calls IPSEpro whenever the evaluation of the steady-state model is required but the determination of the descent direction, performed in an executable written in C++, and the update of the uncertain parameters are done outside IPSEpro.

Nevertheless, the applied numerical method is absolutely capable of fitting all uncertain parameters automatically to time dependent measurements. Due to the expected long calculation times and the current lack of an interface of IPSEpro to extract dynamic results automatically, the method is not implemented yet. In the future also the number of uncertain parameters has to be reduced. Especially, the HTCs α_{Iso_out}, α_{Comp_in} and α_{Amb_Shell} may be determined by correlations and the factor $\eta_{\Delta p\_Co}$ for the FPD in the condenser can be omitted since its fit is $\eta_{\Delta p\_Co} = 1$. Consequently, if also vertical evaporator models are incorporated, the remaining uncertain parameters may be reduced to $(\rho c V)_s$, η_{α_Co}, $\eta_{\Delta p\_Ev}$ and α_{Dis_Shell} if we neglect that in the Compressor-Unit also fit parameters are included.

Optimization Issues

A complete and comprehensive optimization of the investigated appliance was not performed. Only one parameter study is presented in which the NOR of the compressor is changed, see section V.2. In this study it is assumed that the fit parameters of the compression model remain equal for all NOR which does not hold in reality. However, the results indicated that the energy consumption of the appliance is reduced by about 10 % if a NOR of 1500 rpm instead of 3000 rpm is chosen. Furthermore, with the VBA model several different parameter studies were performed from which a few are presented *e.g.* in [43]. Though a parameter study as investigated in these cases, *i.e.* varying only one parameter, shows only the "directional derivative" with resect to the varied variable. To verify the least achievable energy consumption of the investigated appliance, an optimization problem has to be formulated and minimized with respect to all chosen parameters.

A further point of interest starting to emerge in refrigeration appliances are variable-speed compressors. These compressors regulate the NOR depending on the demanded cooling power. Here the question is how the NOR shall be controlled such that the energy consumption of the appliance is minimized. This yields an optimal control problem and we refer to [104] for an introduction into the field.

Concluding this thesis, we claim that a sustainable simulation tool was elaborated which goes beyond the investigation of refrigeration appliances. Due to the general purpose numerical solver, basically any system which is represented by DAEs with appropriate differentiation index can be handled. Furthermore, both the flow sheet arrangement and the specified working fluid are not fixed and thus the component models may be used in different research fields as it has already been demonstrated. Consequently, the developed models and numerical methods of this thesis have a large field of application.

Bibliography

[1] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[2] BDEW Bundesverband der Energie- und Wasserwirtschaft e.V. Stromverbrauch im Haushalt. Berlin, 2014.

[3] K.J. Bell and A. C. Mueller. *Engineering Data Book II.* Wolverine Tube, Inc, 2001.

[4] J.M. Belman-Flores, J.M. Barroso-Maldonado, A.P. Rodríguez-Muñoz, and G. Camacho-Vázquez. Enhancements in domestic refrigeration, approaching a sustainable refrigerator – a review. *Renewable and Sustainable Energy Reviews*, 51:955 – 968, 2015.

[5] S. Bendapudi and J. E. Braun. A review of literature on dynamic models of vapor compression equipment. Technical report, ASHRAE, 2002.

[6] E. Berger, M. Heimel, R. Almbauer, and W. Lang. 1D heat exchanger simulation to capture the cycling transients of domestic refrigeration appliances working with R600a. In *Proc. Int. Refrigeration and Air Conditioning Conference at Purdue, West-Lafayette, USA*. Paper 2452, 2012, 2012.

[7] E. Berger, M. Heimel, S. Posch, R. Almbauer, and M. Eichinger. Transient 1D heat exchanger model for the simulation of domestic cooling cycles working with R600a. In *Proc. Int. Refrigeration and Air Conditioning Conference at Purdue, West-Lafayette, USA*. Paper 2294, 2014, 2014.

[8] E. Berger, S. Posch, M. Heimel, R. Almbauer, M. Eichinger, and A. Stupnik. Transient 1d heat exchanger model for the simulation of domestic cooling cycles working with R600a. *Science and Technology for the Built Environment*, 21(7):1010–1017, 2015.

[9] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific, Belmont, Mass., 1995.

[10] W. Blajer. Index of differential-algebraic equations governing the dynamics of constrained mechanical systems. *Applied Mathematical Modelling*, 16(2):70–77, 1992.

[11] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14 of *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Revised and corrected reprint of the 1989 original.

[12] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Consistent initial condition calculation for differential-algebraic systems. *SIAM J. Sci. Comput.*, 19(5):1495–1512 (electronic), 1998.

[13] F. E. Cellier and E. Kofman. *Continuous System Simulation.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[14] Z.J. Chen and W.-H. Lin. Dynamic simulation and optimal matching of a small-scale refrigeration system. *International Journal of Refrigeration*, 14:329–335, 1991.

[15] W. Cheney and D.R. Kincaid. *Linear Algebra: Theory and Applications.* Jones & Bartlett Learning International Series in Mathematic. Jones & Bartlett Learning, 2010.

[16] J. Chi and D. Didion. A first-principles simulation model for start-up and cycling transients of household refrigerators. *International Journal of Refrigeration*, 5(3):176–184, 1982.

[17] R. de Pelegrini Soares and A. R. Secchi. Structural analysis for static and dynamic models. *Mathematical and Computer Modelling*, 55(3-4):1051–1067, 2012.

[18] P. Deuflhard and F. Bornemann. *Numerische Mathematik 2.* De Gruyter Lehrbuch. De Gruyter, 3rd edition, 2008.

[19] P. Deuflhard and A. Hohmann. *Numerische Mathematik 1.* De Gruyter Lehrbuch. De Gruyter, 3rd edition, 2002.

[20] M. Dhar. *Transient Analysis of Refrigeration System.* PhD thesis, Purdue University, West Lafayette, IN, USA, 1978.

[21] G. L. Ding. Recent developments in simulation techniques for vapour-compression refrigeration systems. *International Journal of Refrigeration*, 30:1119–1133, 2007.

[22] I. S. Duff. On algorithms for obtaining a maximum transversal. *ACM Transactions on Mathematical Software*, 7(3):315–330, sep 1981.

[23] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices.* Monographs on Numerical Analysis. The Clarendon Press, Oxford University Press, New York, second edition, 1989. Oxford Science Publications.

[24] I. S. Duff and J. K. Reid. Algorithm 529: Permutations to block triangular form [f1]. *ACM Trans. Math. Softw.*, 4(2):189–192, 1978.

[25] Bruce Eckel. *Thinking in C++, Vol. 1.* Prentice Hall, 2 edition, 2000.

[26] Bruce Eckel, Chuck D. Allison, and Chuck Allison. *Thinking in C++, Vol. 2.* Pearson Education, 2 edition, 2003.

[27] European Union. Directive 2010/30/eu of the european parliament and of the council of 19 may 2010 on the indication by labelling and standard product information of the consumption of energy and other resources by energy-related products. *Official Journal of the European Union, L* 153, 53, 18 June 2010. http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=OJ%3AL%3A2010%3A153%3ATOC, Accessed: 2017-03-09.

[28] VDI e.V. *VDI-Wärmeatlas.* VDI-Wärmeatlas. Springer Berlin Heidelberg, 2013.

[29] B. Fuchssteiner. Modified Gauss algorithm for matrices with symbolic entries. *ACM Commun. Comput. Algebra*, 42(3-4):108–121, 2008.

[30] C. W. Gear. Differential-Algebraic Equation Index Transformations. *SIAM J. Sci. and Stat. Comput.*, 9(1):39–47, 1988.

[31] C. W. Gear. Differential-algebraic equations, indices, and integral algebraic equations. *SIAM J. Numer. Anal.*, 27(6):1527–1534, 1990.

[32] C. W. Gear and L. R. Petzold. ODE methods for the solution of differential/algebraic systems. *SIAM J. Numer. Anal.*, 21(4):716–728, 1984.

[33] V. Gnielinski. Neue Gleichungen für den Wärme- und Stoffübergang in turbulent durchströmten Rohren und Kanälen. *Forsch. Ing.-Wes.*, 41, No. 1:8–16, 1975.

[34] E. Hairer and G. Wanner. On the instability of the BDF formulas. *SIAM J. Numer. Anal.*, 20(6):1206–1209, 1983.

[35] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2010. Stiff and differential-algebraic problems, Second revised edition, paperback.

[36] J. El Hajal, J. R. Thome, and A. Cavallini. Condensation in horizontal tubes, part 1: two-phase flow pattern map. *International Journal of Heat and Mass Transfer*, 46(18):3349 – 3363, 2003.

[37] X.-D. He, S. Liu, and H. Asada. A moving-interface model of two-phase flow heat exchanger dynamics for control of vapor compression cycle. *Heat Pump and Refrigeration Systems Design, Analysis and Applications, AES*, 32:69–75, 1994.

[38] M. Heimel. *Simulation and experimental validation of adiabatic and non-adiabatic capillary tubes.* PhD thesis, Graz University of Technology, 2015.

[39] M. Heimel, E. Berger, S. Posch, J. Hopfgartner, S. Schlemmer, and R. Almbauer. Calibration Strategies And Limitations Of Cycle Simulations Representing Complex Domestic Cooling Devices. In *International Refrigeration and Air Conditioning Conference at Purdue, West-Lafayette, USA*, 2016.

[40] M. Heimel, E. Berger, S. Posch, A. Stupnik, J. Hopfgartner, and R. Almbauer. Transient cycle simulation of domestic appliances and experimental validation. *International Journal of Refrigeration*, 69:28–41, 2016.

[41] M. Heimel, W. Lang, and R. Almbauer. Performance predictions using Artificial Neural Network for isobutane flow in non-adiabatic capillary tubes. *International Journal of Refrigeration*, 38:281–289, 2014.

[42] M. Heimel, W. Lang, E. Berger, and Almbauer R. A homogeneous capillary tube model - comprehensive parameter studies using r600a. In *International Refrigeration and Air Conditioning Conference at Purdue, West Lafayette, USA*, 2012.

[43] M. Heimel, S. Posch, J. Hopfgartner, E. Berger, A. Stupnik, and R. Almbauer. Simulationsgestützte Optimierung eines Hauhaltsgefrierschrankes. In *DKV-Tagungsband, AA III.08, Dresden, Germany*, 2015.

[44] M. Held and R. M. Karp. The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18:1138–1162, 1970.

[45] C. J. Hermes, C. Melo, and F. T. Knabben. Algebraic solution of capillary tube flows Part I: Adiabatic capillary tubes. *Applied Thermal Engineering*, 30:449–457, 2010.

[46] C. J. Hermes, C. Melo, and F. T. Knabben. Algebraic solution of capillary tube flows. Part II: Capillary tube suction line heat exchangers. *Applied Thermal Engineering*, 30:770–775, 2010.

[47] C.J.L Hermes and C. Melo. A first-principles simulation model for start-up and cycling transients of household refrigerators. *International Journal of Refrigeration*, 31:1341–1357, 2008.

[48] IEA (International Energy Agency). Gadgets and Gigawatts – Policies for Energy Efficient Electronics. OECD-IEA Publishing, Paris, 2009.

[49] M.J.P Janssen, L.J.M Kuijpers, and J.A. de Witt. Theoretical and experimental investigation of a dynamic model for small refrigerating systems. In *International Refrigeration and Air Conditioning Conference at Purdue, West Lafayette, USA*, 1988.

[50] D.I. Jähnig, D.T. Reindl, and S.A. Klein. A semi-empirical method for representing domestic refrigerator/freezer compressor calorimeter test data. *ASHRAE Transactions*, 106:122–130, 2000.

[51] D. S. Johnson and L. A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. In E. H. L. Aarts and J. K. Local Lenstra, editors, *Search in Combinatorial Optimization*, page 215–310. John Wiley and Sons Ltd., London, 1997.

[52] D. Jung, K. Song, Y. Cho, and S. Kim. Flow condensation heat transfer coefficients of pure refrigerants. *International Journal of Refrigeration*, 26:4–11, 2003.

[53] K. Königsberger. *Analysis. 2*. Springer, Berlin, 1993.

[54] R.N.N. Koury, L. Machado, and K.A.R. Ismail. Numerical simulation of a variable speed refrigeration system. *International Journal of Refrigeration*, 24:192–200, 2001.

[55] A. Kröner, W. Marquardt, and E. D. Gilles. Getting around consistent initialization of DAE systems? *Computers & Chemical Engineering*, 21(14), 1997.

[56] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2:83–97, 1955.

[57] P. Kunkel and V. Mehrmann. Stability properties of differential-algebraic equations and spin-stabilized discretizations. *Electronic Transactions on Numerical Analysis*, 26:385–420, 2007.

[58] R. Lamour. Index Determination and Calculation of Consistent Initial values for DAEs. *Comp. Math. Appl.*, 50(7):1125–1140, 2005.

[59] B. Leimkuhler, L. R. Petzold, and C. W. Gear. Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations. *SIAM J. Numer. Anal.*, 28(1):205–226, 1991.

[60] E.W. Lemmon, M.L. Huber, and M.O. McLinden. *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties – REFPROP, Version 9.0.* National Institute of Standards and Technology, Gaithersburg, 2010.

[61] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[62] P. Li, Y. Li, and J. E. Seem. Consistent initialization of system of differential-algebraic equations for dynamic simulation of centrifugal chillers. *Journal of Building Performance Simulation*, 5(2):115–139, 2012.

[63] W. Li. Simplified steady-state modeling for hermetic compressors with focus on extrapolation. *International Journal of Refrigeration*, 35:1722–1733, 2012.

[64] J. Liu, W. Wei, G. Ding, C. Zhang, M. Fukaya, K. Wang, and T. Inagaki. A general steady state mathematical model for fin-and-tube heat exchanger based on graph theory. *International Journal of Refrigeration*, 27(8):965 – 973, 2004.

[65] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming.* Springer, New York, NY, 2008.

[66] V. Mehrmann. Index concepts for differential-algebraic equations. In *Encyclopedia Applied and Computational Mathematics, B. Engquist (Ed.)*, pages 676–681. Springer, Heidelberg, 2015.

[67] C. Melo, R. T. S. Ferreira, R. H. Pereira, and C. O. R. Negrao. Dynamic behavior of a vapour compression refrigerator: A theoretical and experimental analysis. In *International Refrigeration and Air Conditioning Conference at Purdue, West Lafayette, USA*, 1988.

[68] Claudio Melo, Luis Antonio Torquato Vieira, and Roberto Horn Pereira. Non-adiabatic capillary tube flow with isobutane. *Applied Thermal Engineering*, 22:1661–1672, 2002.

[69] R. N. Methekar, V. Ramadesigan, J. C. Pirkle Jr., and V. R. Subramanian. A perturbation approach for consistent initialization of index-1 explicit differential–algebraic equations arising from battery model simulations. *Computers & Chemical Engineering*, 35(11):2227 – 2234, 2011.

[70] Scott Meyers. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition).* Addison-Wesley Professional, 2005.

[71] P. K. Moore and L. R. Petzold. A stepsize control strategy for stiff systems of ordinary differential equations. *Applied Numerical Mathematics*, 15:449–463, 1994.

[72] H. Mori, S. Yoshida, K. Ohishi, and Y. Kokimoto. Dryout quality and post dryout heat transfer coefficient in horizontal evaporator tubes. *Proc of 3rd European Thermal Sciences Conference*, page 839–844, 2000.

[73] N. S. Nedialkov and J. D. Pryce. Solving differential-algebraic equations by Taylor series. I. Computing Taylor coefficients. *BIT*, 45(3):561–591, 2005.

[74] N. S. Nedialkov and J. D. Pryce. Solving differential-algebraic equations by Taylor series. II. Computing the system Jacobian. *BIT*, 47(1):121–135, 2007.

[75] N. S. Nedialkov and J. D. Pryce. Solving differential algebraic equations by Taylor series. III. The DAETS code. *JNAIAM J. Numer. Anal. Ind. Appl. Math.*, 3(1-2):61–80, 2008.

[76] Cezar O.R. Negrao, Raul H. Erthal, Diogo E.V. Andrade, and Luciana Wasnievski da Silva. A semi-empirical model for the unsteady-state simulation of reciprocating compressors for household refrigeration applications. *Applied Thermal Engineering*, 31:1114–1124, 2011.

[77] M. A. M. Neto and J. R. Barbosa Jr. Solubility, density and viscosity of mixtures of isobutane (r600a) and a linear alkylbenzene lubricant oil. *Fluid Phase Equilibria*, 292:7–12, 2010.

[78] J. Nocedal and S. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[79] C. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Statist. Comput.*, 9(2):213–231, 1988.

[80] L. R. Petzold. Differential/Algebraic Equations are not ODE's. *SIAM J. Sci. Stat. Comput.*, 3(3):367–384, 1982.

[81] J. Philipp. *Optimierung von Haushaltskühlgeräten mittels numerischer Modellierung.* PhD thesis, Fakultät Maschinenwesen, Technische Universität Dresden, Deutschland, 1966.

[82] P. Pöll. Modellierung von Haushaltskühlgeräten für den transienten Betriebsfall in IPSEpro. Master's thesis, Graz University of Technology, 2016.

[83] L. Ploug-Sørensen, J.P. Fredsted, and M. Willatzen. Improvements in the modelling and simulation of refrigeration systems: aerospace tools applied to a domestic refrigerator. *Journal of HVAC&R Research*, 3(4):387–403, 1997.

[84] S. Posch, E. Berger, M. Heimel, R. Almbauer, A. Strasser, and A. Stupnik. Semi-empirische Modelle für hermetische Kühlschrankkompressoren. *KI Kälte Luft Klimatechnik*, pages 51–55, 5 2016.

[85] S. Posch, E. Berger, M. Heimel, Almbauer R., A. Stupnik, and H. Schögler. Comparison and validation of semi-empirical compressor models for cycle simulation application. In *International Compressor Engineering Conference at Purdue, West Lafayette, USA*, 2014.

[86] S. Posch, M. Rohrhofer, J. Hopfgartner, E. Berger, R. Almbauer, and E. Perz. Numerische Untersuchung eines LNG-Tanksystems. In *DKV-Tagungsband, AA I.10, Kassel, Germany*, 2016.

[87] J. D. Pryce. Solving high-index DAEs by Taylor series. *Numer. Algorithms*, 19(1-4):195–211, 1998. Differential algebraic equations (Grenoble, 1997).

[88] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

[89] J. M. Quibén and J. R. Thome. Flow pattern based two-phase frictional pressure drop model for horizontal tubes. Part I: Diabatic and adiabatic experimental study. *International Journal of Heat and Fluid Flow*, 28(5):1049 – 1059, 2007.

[90] J. M. Quibén and J. R. Thome. Flow pattern based two-phase frictional pressure drop model for horizontal tubes, Part II: New phenomenological model. *International Journal of Heat and Fluid Flow*, 28(5):1060 – 1072, 2007.

[91] R. Radermacher, E. Gercek, and V.C. Aute. Transient simulation tool for refrigeration systems. In *IIR Conference on Commercial Refrigeration, Vicenza, Italy*, pages 349–355, 2005.

[92] R. Riaza. *Differential-Algebraic Systems: Analytical Aspects and Circuit Applications.* World Scientific, 2008.

[93] Secop GmbH, Fürstenfeld, Austria. Hermetic compressors for ac voltage (catalogue), 2015.

[94] H. Sigloch. *Technische Fluidmechanik.* Springer-Verlag Berlin Heidelberg, 5th edition, 2004.

[95] SimTech GmbH, Graz, Austria. *Model Development Kit, Documentation, Version 6.0.001.*

[96] R. D. Skeel. Construction of variable-stepsize multistep formulas. *Math. Comp.*, 47(176):503–510, S45–S52, 1986.

[97] P. Spellucci. *Numerische Verfahren der nichtlinearen Optimierung.* Birkhäuser, Basel, 1993.

[98] K. Stephan. *Wärmeübergang beim Kondensieren und beim Sieden.* Wärme- und Stoffübertragung. Springer Verlag Berlin, 1988.

[99] Bjarne Stroustrup. *Programming: Principles and Practice Using C++.* Addison-Wesley Professional, 1st edition, 2008.

[100] M. Takamatsu and S. Iwata. Index Determination and Calculation of Consistent Initial values for DAEs. *Linear Algebra and its Applications*, 429:2268–2277, 2008.

[101] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[102] S. A. Tassou and T. Q. Qureshi. Comparative performance evaluation of positive displacement compressors in variable-speed refrigeration applications. *International Journal of Refrigeration*, 21:29–41, 1998.

[103] J. R. Thome, Hajal J. El, and A. Cavallini. Condensation in horizontal tubes, part 2: new heat transfer model based on flow regimes. *International Journal of Heat and Mass Transfer*, 46(18):3365 – 3387, 2003.

[104] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen.* Vieweg Studium. Vieweg+Teubner Verlag, 2010.

[105] J. Unger, A. Kröner, and W. Marquardt. Structural analysis of differential-algebraic equation systems - theory and applications. *Computers & Chemical Engineering*, 19(8):867–882, 1995.

[106] J.V.C. Vargas and J.A.R. Parise. Simulation in transient regime of a heat pump with closed-loop and on–off control. *International Journal of Refrigeration*, 18(4):235–243, 1995.

[107] V. Vidmar and B. Gaspersic. Dynamic simulation of domestic refrigerators with refrigerants R12 and R134a. In *IIR International Congress of Refrigeration, Montreal, Canada*, page 1250–1254, 1991.

[108] R.C. Vieira and E.C. Biscaia Jr. Direct methods for consistent initialization of DAE systems. *Computers & Chemical Engineering*, 25(9–10):1299 – 1311, 2001.

[109] W. Wibel. *Untersuchungen zu laminarer, transitioneller und turbulenter Strömung in rechteckigen Mikrokanälen.* PhD thesis, Fakultät Bio- und Chemieingenieurwesen der Technischen Universität Dortmund, 2009.

[110] S. Wiggins. *Global Bifurcations and Chaos: Analytical Methods.* Applied Mathematical Sciences. Springer New York, 2013.

[111] C. Windhager. Dynamische Simulation von Kühlkreisläufen. Master's thesis, Graz University of Technology, 2015.

[112] L. Wojtan, T. Ursenbacher, and J. R. Thome. Investigation of flow boiling in horizontal tubes: Part I— A new diabatic two-phase flow pattern map. *International Journal of Heat and Mass Transfer*, 48(14):2955 – 2969, 2005.

[113] L. Wojtan, T. Ursenbacher, and J. R. Thome. Investigation of flow boiling in horizontal tubes: Part II— Development of a new heat transfer model for stratified-wavy, dryout and mist flow regimes. *International Journal of Heat and Mass Transfer*, 48(14):2970 – 2985, 2005.

[114] S. Wongwises, S. Disawas, J. Kaewon, and C. Onurai. Two- phase evaporative heat transfer coefficients of refrigerant hfc-134a under forced flow conditions in a small horizontal tube. *International Communications in Heat and Mass Transfer*, 27(1):35–48, 2000.

[115] B. Wu and R. C. White. An initialization subroutine for DAEs solvers: DAEIS. *Computers & Chemical Engineering*, 25(2-3):301–311, 2001.

[116] H. Yasuda, S. Touber, and C.H.M. Machielsen. Simulation model of a vapor compression refrigeration system. *ASHRAE Transactions*, 89 (Part 2):408–425, 1983.

[117] B.F. Yu, Z.G. Wang, B. Yue, B.Q. Han, H.S. Wang, and F.X. Chen. Simulation computation and experimental investigation for on–off procedure of refrigerator. In *IIR International Congress of Refrigeration, The Hague, The Netherlands, vol. 3*, page 546–553, 1995.

[118] X. Yuan and D.L. O'Neal. Development of a Transient Simulation Model of a Freezer Part I: Model Development. In *International Refrigeration and Air Conditioning Conference at Purdue, West Lafayette, USA*, 1994.

[119] C. Zhang and G. Ding. Approximate analytic solutions of adiabatic capillary tube. *International Journal of Refrigeration*, 27:17–24, 2004.

[120] J. Zierep and K. Bühler. *Strömungsmechanik*. Springer-Verlag Berlin Heidelberg, 1991.