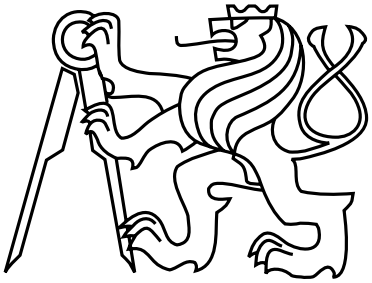
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Control Engineering



Diploma Thesis

Vapor Compression Cycle Solving for Optimal Controller Design

Bc. Bronislav Robenek

Diploma Thesis Supervisor: Ing. Radek Beňo

Study program: Cybernetics and Robotics

Specialization: Systems and Control

January 2016

Aknowledgements

I would like to thank to everybody who helped me. A special thanks to my supervisor Ing. Radek Beňo for his valuable time, support and advice.

To my friends and family, thanks for being there especially during that long days and long nights. Your encouragement and belief in me is what kept me going.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used in compliance with ethical principles in the preparation of university theses.

In Prague 10th January 2016 .....

Signature

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Bronislav Robenek**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Diploma Thesis: **Vapor Compression Cycle Solving for Optimal Controller Design**

Guidelines:

An operation of a Heat Pump (HP) is based on a Vapor Compression Cycle (VCC). The workflow of an optimal heat pump controller design includes VCC solving, where VCC refrigerant states are computed for interconnected models of HP components. This important step can be formulated as a problem of solving a set of nonlinear equations including functions with limited domains. A suitable approach for this problem can be based on a Particle Swarm Optimizaiton (PSO) as one of heuristic methods for non-convex functions.
Guidelines:
1. Get familiar with the HP modeling and constrained PSO techniques.
2. Implement a model of HP in a suitable form for VCC solving.
3. Implement algorithm(s) based on PSO for VCC solving.
4. Solve a VCC on a grid of system operating points and discuss results.
5. Optional: Implement another algorithm (e.g. using Genetic algorithms) for VCC solving and compare results.

Bibliography/Sources:

[1] Parsopoulos K. E. and Vrahatis M. N., Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems, Advances in Natural Computation, Lecture Notes in Computer Science, vol. 3612, pp. 582-591, 2005.
[2] Winkler J. M., Development of a Component Based Simulation Tool for the Steady State and Transient Analysis of Vapor Compression Systems, PhD thesis, University of Maryland, 2009.
[3] ZhiHong Shuang, "The Application of Particle Swarm Optimization to Solving Nonlinear Equations," 2010 International Conference on Computational Aspects of Social Networks (CASoN), pp.733-736, 26-28 Sept. 2010.
[4] Brits, R.; Engelbrecht, A.P.; van den Bergh, F., "Solving systems of unconstrained equations using particle swarm optimization," 2002 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 1-6, 6-9 Oct. 2002.

Diploma Thesis Supervisor: Ing. Radek Beňo

Valid until the summer semester 2015/2016

L.S.

Prof. Ing. Michael Šebek, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, October 13, 2014

Abstract

In this thesis, a particle swarm optimization algorithm is used to solve vapor compression cycle equations. The goal is to verify feasibility of this approach and to show practical application. This problem is further expanded to solve control optimization.

Keywords: Vapor Compression Cycle, Particle Swarm Optimization, Heat pump

Abstrakt

Algoritmus založený na optimalizaci rojem částit byl použit k řešení soustavy rovnic modelu parního kompresního cyklu. Cílem bylo ověřit použitelnost tohoto přístupu a ukázat praktickiou aplikaci. Tento problém byl dále rozšířen o současné řešení optimalizace řízení.

Klíčová slova: Parní kompresní cyklus, Optimalizace rojem částic, Tepelné čerpadlo

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Heat pump is one of inventions that its users barely understand even though it is heavily used in everyday life. Heat Pump transfers heat energy from a source of heat to a heat sink. This transfer is in opposite to spontaneous heat flow, so from colder body to warmer body. To achieve this work is required.

Typical applications are HVAC systems, air conditioners, freezers, fridges and coolers. Heat pumps are mostly based on thermodynamic cycle or thermoelectric Effect. Several thermodynamic cycles are used in modern devices however the Vapor Compression Cycle is widely used because of its efficiency and refrigerant options. This thesis is focused on VCC its modeling, control and optimization.

Heat pumps as well as HVAC systems gained some public interest when they became available for residential housing. However they are commonly used in industrial applications.

Since heat pump consists of several components they have to be controlled to achieve desired behaviour - e.g. heat transfer. Internal pressures and enthalpies determine heat pump operation. Power consumption of operating heat pump is considerable and as such should be minimized. This adds additional criteria to finding control variables. Heat pump should transfer desired heat with minimal power consumption. To asses how efficient heat pump is Coefficient of Performance (COP) is used:

$$COP = \frac{Q}{W} = \frac{\dot{Q}}{P} \tag{1.1}$$

where Q [J] is heat supplied or removed (in condenser or evaporator), W [J] is work consumed and P [W] is power consumed. $COP > 1$ means more energy is transferred then consumed and should be thus maximized.

In large maximum achievable COP is defined by temperature difference between hot and cold bodies. However in practice refrigerant and heat losses in electric motors drastically reduce achievable COP.

Only steady state behaviour of VCC is considered in this thesis so it is much more focused on practical optimization and equation solving rather than control theory of dynamical system. Problem further described is to find optimal steady state.

1.1 Heat Pump applications

Typical application consists of two units. One placed near the heat source which consist of heat exchanger and second unit which consists of heat exchanger, valve and compressor. These units are interconnected using a metal piping where the refrigerant is transported. Basic operation is always the same however applications have different requirements:

- HVAC Systems - Used to heat, cool or condition interior air according to requirements. Typically in office buildings, apartments, residential housing, factories and in vehicles. VCC based. HVAC system has to be capable of reverse operation so heat exchangers can be used either as heat source or heat sink.
- Residential heat pumps - Used to provide hot water for central heating and warm water production. Several sources of heat are considered - Water (Rivers, Lakes), Ground (Geothermal energy), Outside air. VCC based.
- Freezers - Used in industrial food production and material storage as well as in household food conservation. VCC based.
- Precision temperature control - Metrology and other specialized industries require precision temperature control of chambers or bodies which is achieved using combination of environment conditioning based on VCC and direct body heat transfers using thermoelectric effect.

1.2 Thesis Motivation

As discussed in further detail in chapter 2 working VCC usually requires at least 4 components - compressor, condenser, expansion valve and evaporator. All these components are interconnected and refrigerant circulates in closed loop.

From design perspective all components have to be sized appropriately for designed heat output. From control perspective operating set point has to be chosen and maintained. This results in transition state (e.g. heat pump power up) and steady state behaviour (once set point is reached).

Inputs to this tasks are typically ambient conditions (inside/outside temperature, humidity) and desired heat output (e.g. amount and temperature of air delivered).

Controlled inputs of heat pump are typically compressor frequency, valve opening and evaporator/condenser fan frequencies. It is obvious that some combinations of ambient conditions and control will lead to unstable refrigerant heating/cooling and some will result in stable steady state closed loop operation.

Motivation of this thesis is to find control of VCC components for set ambient conditions and desired heat output/input such that resulting VCC state:

- is steady state (VCC loop closes)
- has optimal power efficiency

It is assumed that once desired state (set point) is calculated low level controller will reach and maintain this state. Once ambient conditions change or desired heat output is reset new optimal state will be calculated.

According to the supervisor current practice is to use offline optimization on grid of ambient conditions and desired heat output. Once calculated optimal set points for various situations are preloaded in heat pump controller. One of possible approaches is shown in Figure 1.1.

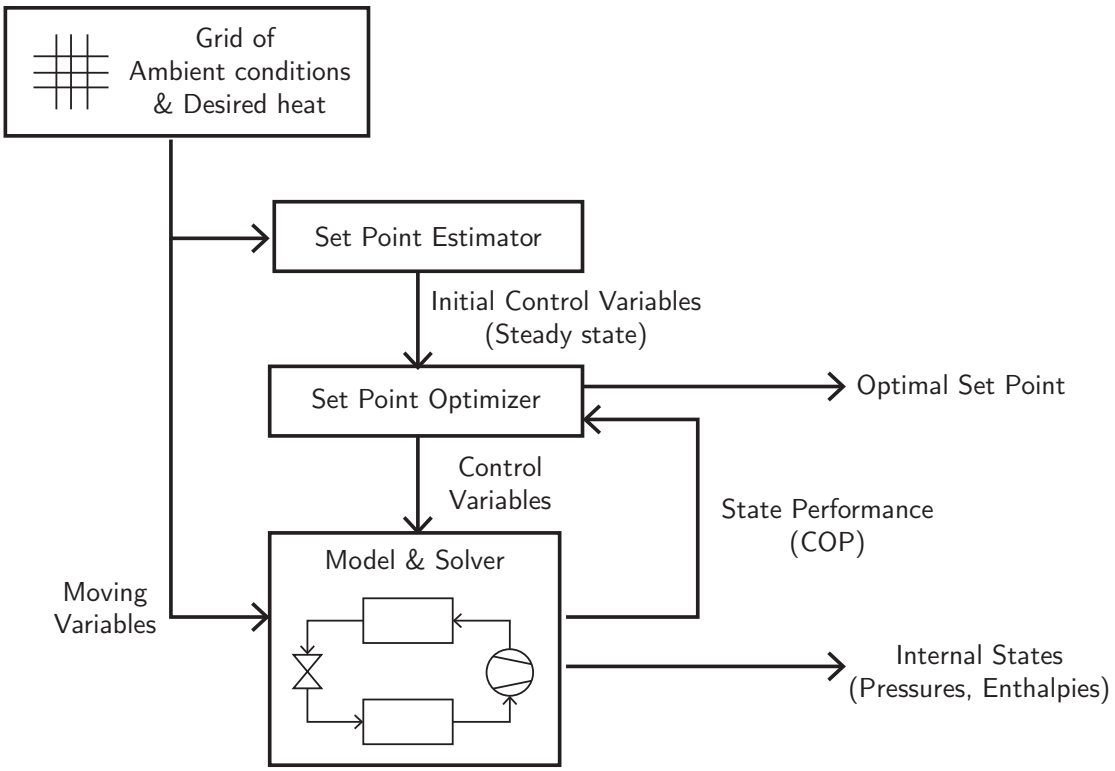


Figure 1.1: Block scheme of current VCC optimization.

It is important to note that:

- Performance optimization and VCC loop is solved separately.
- Initial control variables are estimated to ensure feasible initial state
- Model solver is used to calculate VCC solution and provide coefficient of performance (COP) and internal states (pressures, enthalpies, ...)
- Step based optimizer tries to move control variables to maximize COP
- Optimal control variables are stored in grid for offline use

Usage of model equation solver is necessary since expressing closed loop behaviour is not practical. VCC models often use material libraries and other tabular data rather than straightforward mathematical equations. Some VCC components such as evaporator might be even described using more sophisticated models thus expressing overall closed loop behavior equations might be impossible and only numerical methods are applicable.

Current approach known to thesis supervisor have problem with VCC Model Solver which sometimes fails to find global minimum (e.g. solution). In this thesis Particle Swarm Optimization (PSO) algorithm is used to solve this problem.

1.3 Problems Addressed

During this thesis several problems and tasks were addressed. First computational model of VCC was put together as documented in chapter 2. Aim of this part is to prepare suitable model for further optimization and evaluation.

Custom PSO Solver was developed. This is documented in chapter 3. Custom implementation allowed further tweaking and customization.

Problems for implemented solver are defined and solved in chapter 4. Standalone VCC solving usable for grid optimization is first addressed problem. Second problem addresses simultaneous VCC solving with relaxed control variables and COP optimization.

Unless specified otherwise all computations were executed in Matlab®2014b on Mac OS X 10.11.2, 2GHz Intel Core 2 Duo, 4GB RAM.

Chapter 2

Vapor Compression Cycle Model

Physical model of Vapor Compression Cycle used in this thesis is based on report Z-582/2013 of project "Program Alfa TAČR Pokročilé řízení a optimalitace provozu tepelných čerpadel"[1], which consists of heat pump equations using R410a refrigerant.

For desired use only steady state behaviour is required. This is characterized by constant refrigerant mass flow. As presented in introduction several configurations of heat pumps are deployed in practice. However from modeling perspective they are interchangeable up to exact evaporator and condenser design. Only air to air heat pump in heating mode is considered in this thesis.

Aim of this part is to construct computational model of VCC suitable for following optimization. This model have to represent physical behaviour of refrigerant, determine its pressures and enthalpies. It should be modular to allow further extensions.

Model is programmed in MATLAB®R2014b without usage of any specialized toolboxes. Only dependency is library CoolProp[2] used to model physical behaviour of refrigerant and humid air properties.

This chapter describes four models resembling required VCC components. Each component provides outputs which are in turn used as inputs to following component. To create cycle they have to be interconnected and solver has to be used. This is further discussed in Chapter 4.

2.1 Vapor Compression Cycle

Vapor Compression Cycle is thermodynamic cycle where refrigerant undergoes phase and pressure changes. This is constructively used to transfer heat from evaporator to condenser where outside environment is cooled or heated.

Heat transfer from colder body to warmer body is achieved using different evaporation and condensation temperatures of refrigerant which are altered as needed utilizing pressure changes that happen in compressor and expansion valve.

Ideal VCC consists of four thermodynamic processes assembled in thermodynamic cycle which is shown in Figure 2.1. Each of these processes can be modeled separately resulting in four components shown in Figure 2.2.

- 1. → 2. - Isentropic Compression (Gas → Gas, Compressor Component)
- 2. → 3. - Isobaric Condensation (Gas → Liquid, Condenser Component)
- 3. → 4. - Isenthalpic Expansion (Liquid → Liquid + Gas, Valve Component)
- 4. → 1. - Isobaric Evaporation (Liquid + Gas → Gas, Evaporator Component)

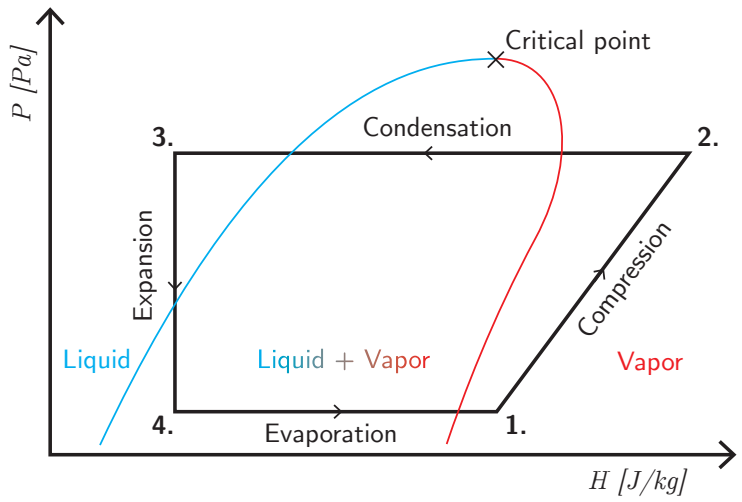


Figure 2.1: VCC P-H diagram. Blue line represents saturated liquid line. Red line represents saturated vapor line.

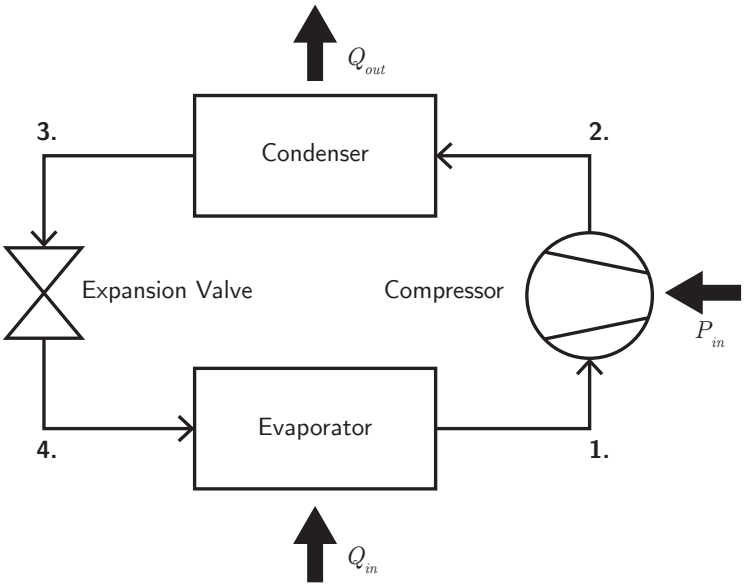


Figure 2.2: VCC block diagram.

Heat pump components are piped together to form a cycle. During the steady state refrigerant mass flow is constant and it is equal in all components. However refrigerant in each component can reach different Enthalpy and Pressure. Components have pressure/enthalpy input and output. Refrigerant mass flow is defined by compressor for all other components.

Numbers 1,2,3 and 4 used in corners of Figures 2.1 and 2.2 are used in this thesis to identify refrigerant pressures and enthalpies. For example p_1 is refrigerant pressure at point 1, however this pressure is same in point 4 thus p_{41} can be used as well. $p_{41,lb}$ is used to identify lower bound for pressure at point 4 and 1. $p_{41,ub}$ is upper bound. H_1 is refrigerant enthalpy at compressor input.

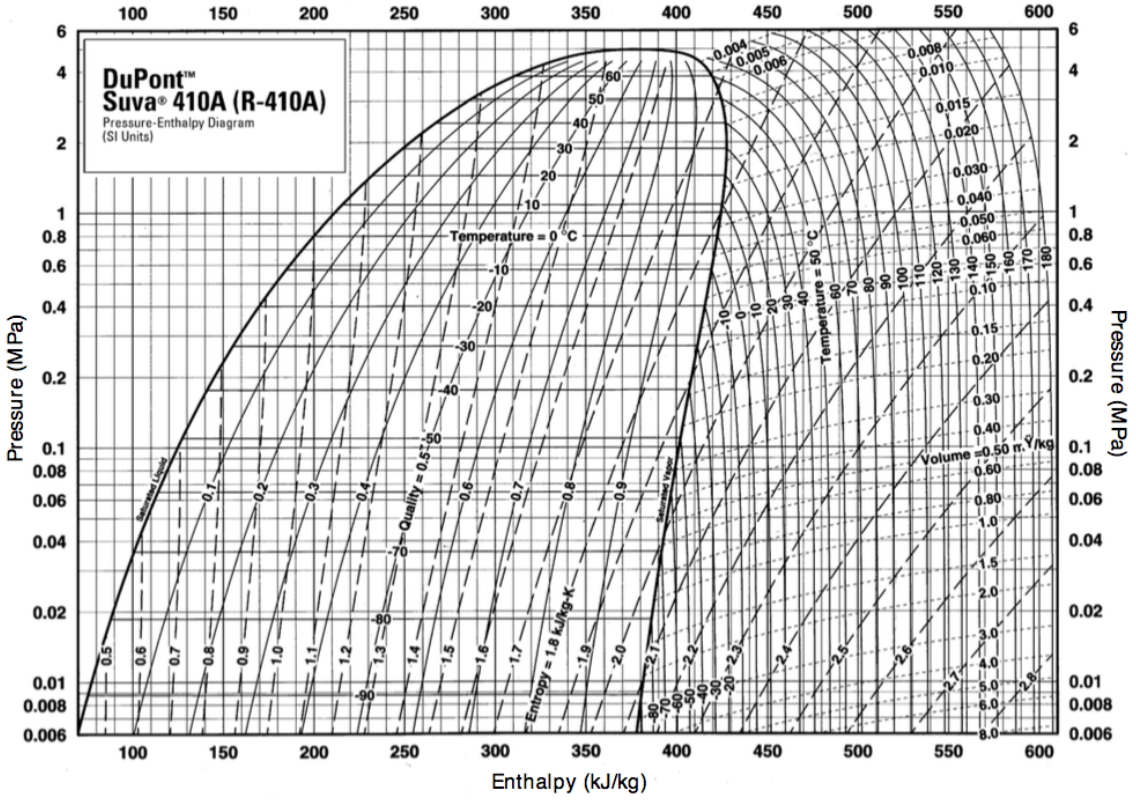


Figure 2.3: DuPont™Suva®410A P-H Diagram as presented in the product data sheet[3].

2.2 Gas Compressor

2.2.1 Description

Generally gas compressors are mechanical devices used to increase pressure of a gas by reducing its volume. To achieve this they consume mechanical energy for example provided using electric motor. Different types of gas compressors exists (centrifugal, reciprocating, rotary screw, scroll, ...). Used model is based on compressor model presented in [1]. Scroll compressor with variable rotation speed - Mitsubishi ANB33FBDMT.

Model consists of 3 output variables Medium Mass Flow $\dot{m}$, Output Enthalpy H_{out} and Power Consumption P, 4 input variables Input Pressure p_{in}, Output Pressure p_{out}, Input Enthalpy H_{in}, Frequency n. Model is based on equations show in Table 2.1.

Description	Unit	Definition
Displacement Volume	[m^3]	$V_d = f_1(n)$
Efficiency	[%]	$\eta = f_2(n, \phi, p_{out})(1 - \mu_k)$
Transport Coefficient	[−]	$\lambda = f_3(n, \sigma)$
Compression Ratio	[−]	$\sigma = \frac{p_{out}}{p_{in}}$
	[−]	$\phi = \sigma^{1/n}$
Medium Mass Flow	[kg/s]	$\dot{m} = \rho_{in} V_d n \lambda$
Isentropic Power	[W]	$P_{ie} = \dot{m}(H_{ie} - H_{in})$
Power	[W]	$P = \frac{P_{ie}}{\eta}$
Output Enthalpy	[J/kg]	$H_{out} = H_{in} + \frac{H_{ie} - H_{in}}{\eta}$

Table 2.1: Compressor model variables and equations.

As described in [1] while compressor manufacturers usually specify displacement volume V_d as static parameter or as a parameter rated at nominal frequency (RPM) effective displacement volume changes considerably with frequency and its dependency is non-linear. Generic polynomial function $f_1(n)$ is used, polynomial coefficients are identified using measurement. Displacement volume is further used to determine $\dot{m}$. Medium Mass Flow is further discriminated using Transport Coefficient λ to account for losses.

Consumed Power P and Output Enthalpy H_{out} is modeled using ideal isentropic compressor process discriminated for compressor efficiency. Efficiency is modeled as polynomial function $f_2(n, \phi, p_{out})$ with coefficients identified using measurements. Efficiency is further discriminated using μ_k coefficient which is typically 5% and accounts for thermal losses. Function f_1, f_2 and f_3 are identified and further discussed in [1].

To calculate enthalpy after isentropic compression CoolProp is used. First input entropy of coolant is calculated $S_{in} = f_S(p_{in}, H_{in})$. Since entropy is constant during isentropic compression and output pressure is known isentropic enthalpy can be calculated $H_{ie} = f_H(p_{out}, S_{in})$.

2.2.2 Definition

By putting all presented equations and constraints together final gas compressor valve model consists of following:

Model inputs:

- p_{in} - Refrigerant Input Pressure [Pa]
- p_{out} - Refrigerant Output Pressure [Pa]
- H_{in} - Refrigerant Input Enthalpy [J/kg]
- n - Frequency [Hz]

Model outputs:

- $\dot{m}$ - Refrigerant Mass Flow [kg/s]
- H_{out} - Refrigerant Output Enthalpy [J/kg]
- P - Power Consumed [W]

Model equations:

$$
\begin{aligned}
\sigma &= \frac{p_{out}}{p_{in}} \\
V_d &= C_1 n^3 + C_2 n^2 + C_3 n + C_4 \\
\lambda &= 1 - L(\sigma - 1) \\
\rho_{in} &= f_D(p_{in}, H_{in}) = \text{PropsSI}(D, P, p_{in}, H, H_{in}, medium) \\
\dot{m} &= \rho_{in} V_d n \lambda \\
S_{in} &= f_S(p_{in}, H_{in}) = \text{PropsSI}(S, P, p_{in}, H, H_{in}, medium) \\
H_{ie} &= f_H(p_{out}, S_{in}) = \text{PropsSI}(H, P, p_{out}, S, S_{in}, medium) \\
\phi &= \sigma^{1/n} \\
\eta_1 &= D_1 + D_2 n + D_3 n^2 + D_4 \phi + D_5 \phi^2 + D_6 n \phi + D_7 p_{out} \\
\eta &= \eta_1 * (1 - M) \\
P_{ie} &= \dot{m}(H_{ie} - H_{in}) \\
P &= \frac{P_{ie}}{\eta} \\
H_{out} &= H_{in} + \frac{H_{ie} - H_{in}}{\eta}
\end{aligned}
$$

Model parameters (constants):

Parameter	Description	Unit	Value
C_1	Displacement Volume Coefficient	[−]	1.02×10^{-11}
C_2	Displacement Volume Coefficient	[−]	-2.68×10^{-9}
C_3	Displacement Volume Coefficient	[−]	2.31×10^{-7}
C_4	Displacement Volume Coefficient	[−]	2.74×10^{-5}
D_1	Efficiency Coefficient	[−]	$8.292\,45 \times 10^{-1}$
D_2	Efficiency Coefficient	[−]	2.273×10^{-3}
D_3	Efficiency Coefficient	[−]	-2×10^{-5}
D_4	Efficiency Coefficient	[−]	-2.65×10^{-2}
D_5	Efficiency Coefficient	[−]	-3.86×10^{-3}
D_6	Efficiency Coefficient	[−]	5.58×10^{-5}
D_7	Efficiency Coefficient	[−]	-2.6×10^{-8}
L	Transport Coefficient	[−]	3.45×10^{-2}
M	Thermal Losses Coefficient	[%]	5×10^{-2}

Table 2.2: Compressor model parameters.

2.2.3 Usage and Behaviour

To verify model behaviour three charts are shown. If not stated otherwise $p_{in} = 1.3\,\mathrm{MPa}$, $H_{in} = 450\,\mathrm{kJ/kg}$. See Figures 2.4, 2.5 and 2.6. Example code usage is in listing 2.1.

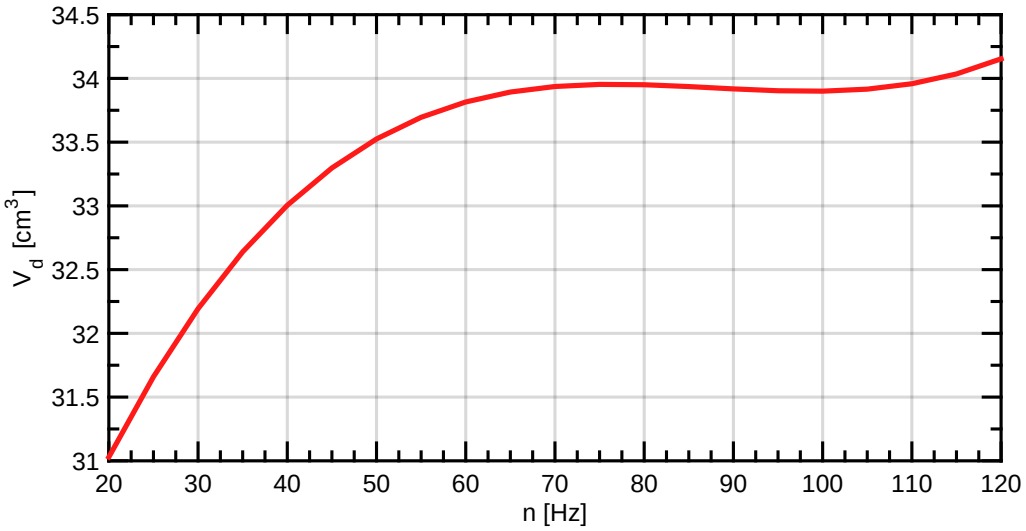


Figure 2.4: Compressor Displacement Volume.

Listing 2.1: Example usage of Compressor model

```
% Compressor object with default parameters
c = VCC.Compressor();

% Inputs are provided
% (p_in, p_out, H_in, n)
c.evaluate(1.3e6, 1.8e6, 4.5e5, 50);

% Outputs can be used
c.dotm
% = 0.0721
c.H_out
% = 4.6263e+05
c.P
% = 911.5110
```

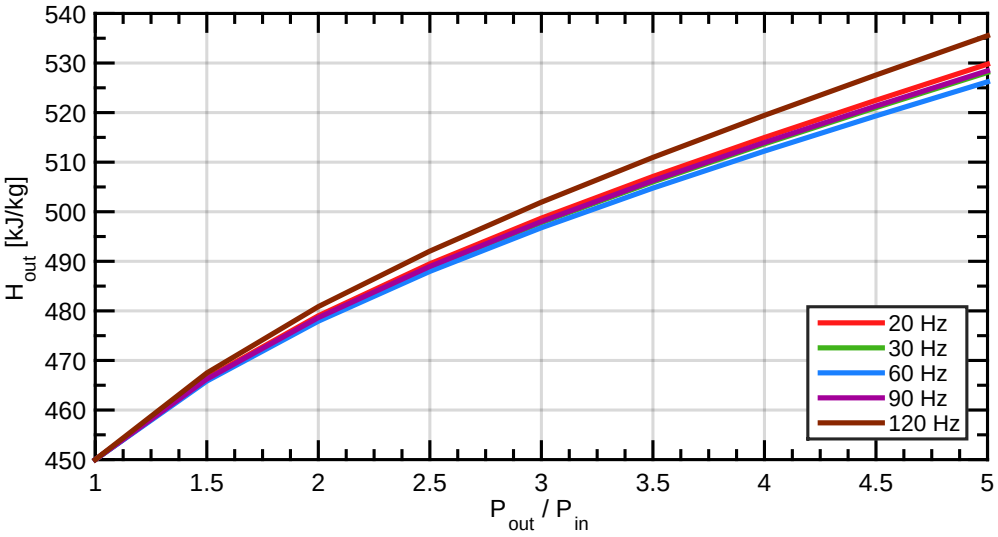


Figure 2.5: Compressor Output Enthalpy H_{out}.

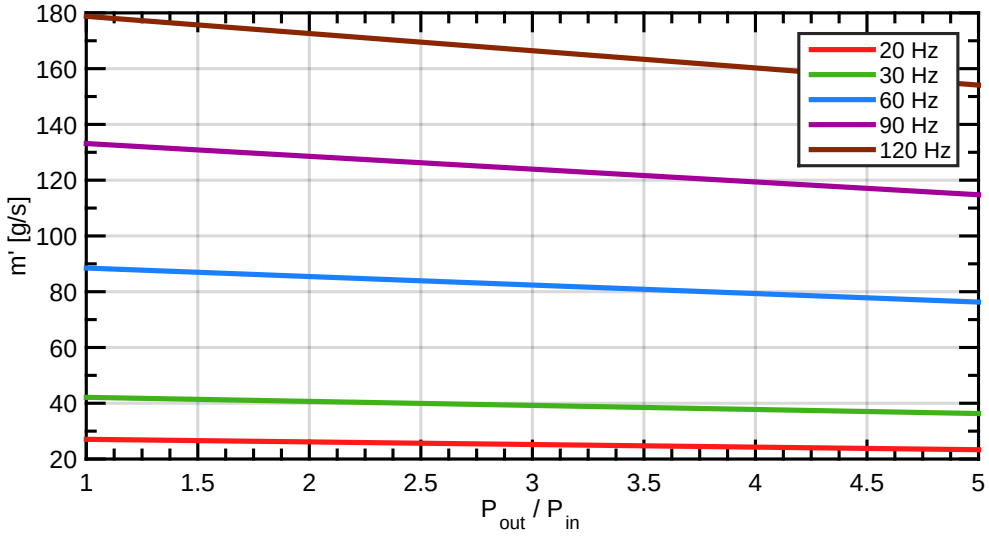


Figure 2.6: Compressor Medium Mass Flow $\dot{m}$.

2.3 Condenser

2.3.1 Description

In condenser the refrigerant in a gaseous state comes in. During isobaric condensation refrigerant condenses and exits condenser generally in a liquid state. Heat is transferred from the refrigerant to the outside environment thus the outside environment is heated. Pressure of the refrigerant in a condenser has to be such that the condensing temperature of refrigerant is above outside temperature.

Model consists of 5 output variables Refrigerant Output Enthalpy H_{out}, Refrigerant Output Pressure p_{out}, Heat Transfer Rate $\dot{Q}$, Air Output Temperature $T_{a,out}$ and Fan Power Consumption P_f, 7 input variables Refrigerant Input Pressure p_{in}, Refrigerant Input Enthalpy H_{in}, Refrigerant Mass Flow $\dot{m}$, Air Mass Flow $\dot{m}_a$, Air Input Temperature $T_{a,in}$, Air Input Pressure $p_{a,in}$ and Air Relative Humidity R. Model is based on equations show in Table 2.3.

Evaporator and Condenser Models used in this thesis are based on Evaporator model equations in Program Alfa Report [1]. Condenser in Program Alfa Report is using different equations than model described in this chapter ($T_{a,out}$ as input and $T_{a,in}$ as output).

Description	Unit	Definition
Evaporator parameter	[−]	$k = \frac{UA}{\dot{m}_a c_{ha}}$
Temperature Difference	[K]	$\Delta T = T_{a,in} - T_s$
Heat Transfer Rate	[J/s]	$\dot{Q} = (1 - e^{-k})\dot{m}_a c_{ha} \Delta T$
Output Pressure	[Pa]	$p_{out} = p_{in}$
Output Enthalpy	[J/kg]	$H_{out} = H_{in} + \frac{\dot{Q}}{\dot{m}}$
Air Output Temperature	[K]	$T_{a,out} = T_{a,in} - \frac{\dot{Q}}{\dot{m}_a c_{ha}}$

Table 2.3: Condenser model variables and equations.

Refrigerant Condensation Temperature T_s [K] (Saturated Liquid Temperature) is calculated using CoolProp as function of pressure and Vapor Mass Quality of 0.

$$T_s = f_T(p_{in}, 0) = \text{PropsSI}(T, P, p_{in}, Q, 0, medium)$$

Equations in Table 2.3 are further constrained to describe evaporator in desired physical state:

1. $T_{a,in} < T_s \leq T_{in}$ - Heat will flow only from refrigerant to air, achieved by:
   - Limiting $p_{in} > p_{in,lb}$ such that $p_{in,lb}$ is pressure for which saturated liquid temperature of refrigerant equals $T_{a,in}$.
   - Forcing $\Delta T = T_{a,in} - T_s$ always negative.
   - Assuming refrigerant is in gaseous state.
2. $T_{out} \geq T_{a,in}$ - Refrigerant can not be cooled below input air temperature, achieved by:
   - Limiting $\dot{Q} \leq \dot{Q_{max}}$, where Q_{max} is heat required to be removed so that refrigerant temperature reaches input air temperature.

$$\dot{Q_{max}} = (H_{out,max} - H_{in}) * \dot{m} \tag{2.1}$$
$$H_{out,max} = f_H(p_{in}, T_{a,in}) = \text{PropsSI}(H, P, p_{in}, T, T_{a,in}, medium) \tag{2.2}$$

2.3.2 Humid Air Mass Specific Heat

Mass specific heat c_{ha} [J/kg/K] of humid air is calculated as convex combination of water mass specific heat c_w and air mass specific heat c_a using humidity mass ratio $w[\text{kg}_{\text{water}}/\text{kg}_{\text{air}}]$. Properties of Humid Air are calculated using CoolProp. Chart of this function is shown in Figure 2.7.

$$\begin{aligned} w &= f_w(p_{a,in}, T_{a,in}, R) = \text{HAPropsSI}(W, P, p_{a,in}, T, T_{a,in}, R, R) \\ c_w &= f_{c,w}(p_{a,in}, T_{a,in}) = \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, water) \\ c_a &= f_{c,a}(p_{a,in}, T_{a,in}) = \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, air) \\ c_{ha} &= wc_w + (1-w)c_a \end{aligned} \tag{2.3}$$

Function to perform this computation is named `VCC.humid_air_cp(R, T, p)`.

2.3.3 Fan Power Consumption

Air Condenser is usually combined with fan which blows air around heat exchanger elements. This is essential for further optimization since requiring more air means higher fan power consumption and that is not always beneficial. Condenser model is thus further extended with equation to compute required power. This is based on assumption that product of pressure drop and volumetric flow results as power due to dimensional analysis and nature of variables as of effort and flow.

$$P_f = \frac{1}{\eta_f} \Delta p_f \dot{V_a} \tag{2.4}$$

To calculate air volumetric flow V_a air density is used and pressure drop Δp_f is considered fan parameter. Efficiency η_f is added as additional parameter to account other losses.

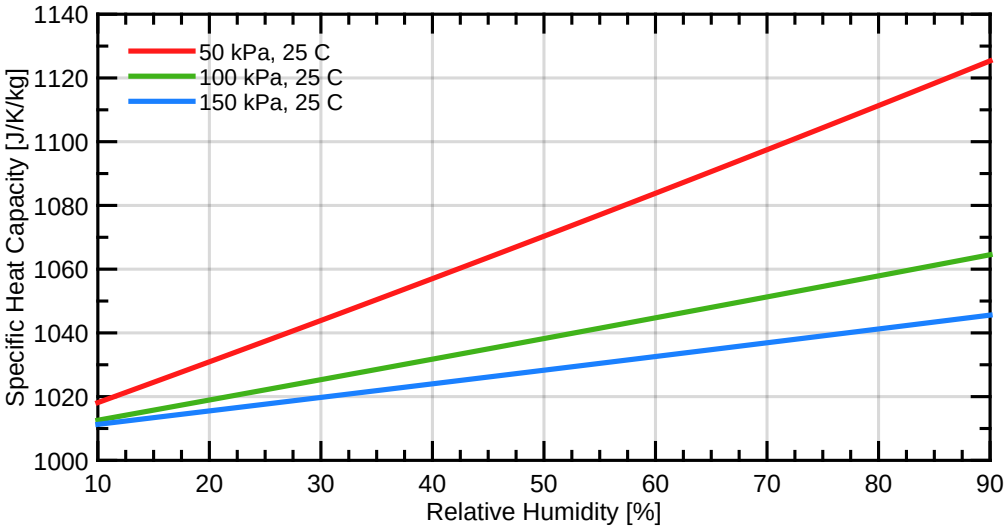


Figure 2.7: Humid Air Specific Heat Capacity as function of Temperature, Pressure and Relative Humidity.

2.3.4 Definition

By putting all presented equations and constraints together final condenser model consists of following:

Model inputs:

- p_{in} - Refrigerant Input Pressure [Pa]
- H_{in} - Refrigerant Input Enthalpy [J/kg]
- $\dot{m}$ - Refrigerant Mass Flow [kg/s]
- $\dot{m}_a$ - Air Mass Flow [kg/s]
- $T_{a,in}$ - Air Input Temperature [K]
- $p_{a,in}$ - Air Input Pressure [Pa]
- R - Air Relative Humidity [%]

Model outputs:

- p_{out} - Refrigerant output pressure [Pa]
- H_{out} - Refrigerant output enthalpy [J/kg]
- $\dot{Q}$ - Heat Transfer Rate [J/s]
- $T_{a,out}$ - Air Output Temperature [K]
- P_f - Fan Power Consumption [W]

Model equations:

$$
\begin{aligned}
w &= \text{HAPropsSI}(W, P, p_{a,in}, T, T_{a,in}, R, R) \\
c_w &= \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, water) \\
c_a &= \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, air) \\
c_{ha} &= wc_w + (1-w)c_a \\
T_s &= \text{PropsSI}(T, P, p_{in}, Q, 0, medium) \\
k &= \frac{UA}{\dot{m}_a c_{ha}} \\
\Delta T &= \min(0, T_{a,in} - T_s) \\
\dot{Q} &= (1 - e^{-k})\dot{m}_a c_{ha} \Delta T \\
H_{out,max} &= \text{PropsSI}(H, P, p_{in}, T, T_{a,in}, medium) \\
\dot{Q_{max}} &= (H_{out,max} - H_{in}) * \dot{m} \\
\dot{Q} &= \max(\dot{Q}, \dot{Q_{max}}) \\
H_{out} &= H_{in} + \frac{\dot{Q}}{\dot{m}} \\
T_{a,out} &= T_{a,in} - \frac{\dot{Q}}{\dot{m}_a c_{ha}} \\
p_{out} &= p_{in} \\
\rho_a &= \text{PropsSI}(D, P, p_{a,in}, T, T_{a,in}, air) \approx 1.2046 \\
\dot{V_a} &= \frac{\dot{m}_a}{\rho_a} \\
P_f &= \frac{1}{\eta_f} \Delta p_f \dot{V_a}
\end{aligned}
$$

Model parameters (constants):

Parameter	Description	Unit	Value
U	Heat Transfer Coefficient	[W/m^2/K]	2400
A	Heat Transfer Surface	[m^2]	3
η_f	Fan Efficiency	[−]	0.50
Δp_f	Fan Pressure Drop	[Pa]	50

Table 2.4: Condenser model parameters.

2.3.5 Usage and Behaviour

Model behaviour is based on Enthalpy-Temperature relation of refrigerant, this relation is show in Figure 2.8. This figure describes how boiling process happens. It is important to note that condenser model is not capable of reversed operations (evaporating). In Figure 2.9 temperature difference of air and refrigerant is shown. Once saturation

temperature is above input air temperature heat exchange occurs, refrigerant is cooled and air heated. Example code usage is in listing 2.2.

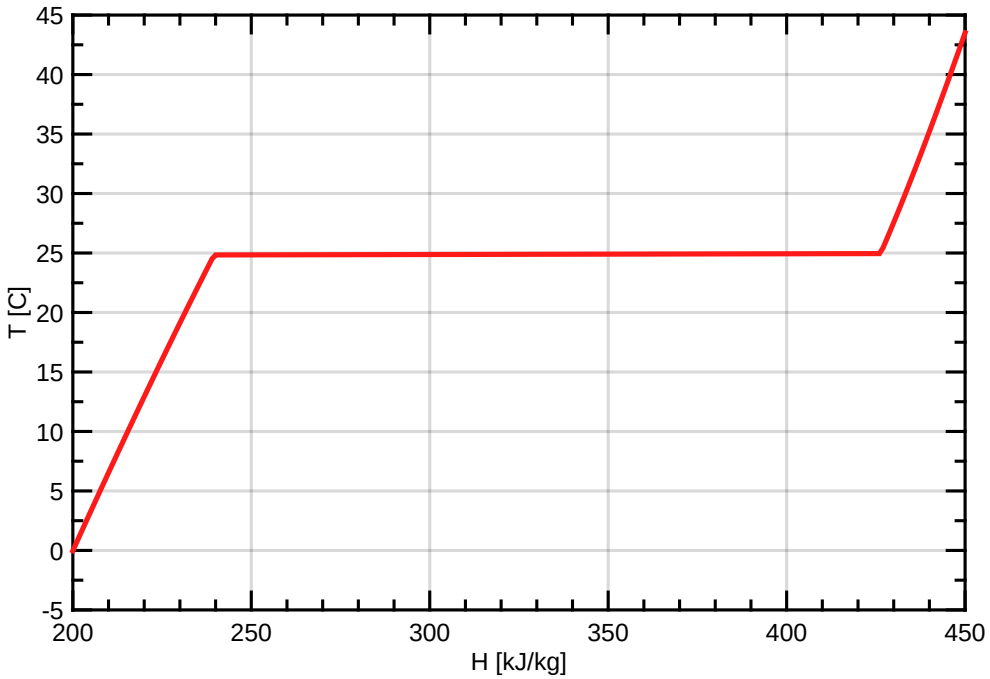


Figure 2.8: Refrigerant Temperature at 1.65 MPa as function of Enthalpy (As heat is added to the refrigerant it starts to boil)

Listing 2.2: Example usage of Condenser model

```
% Condenser object with default parameters
c = VCC.Condenser();

% Inputs are provided
% (p_in, H_in, dotm, dotma, Ta_in, p_a, R)
c.evaluate(1.8e6, 4.62e5, 7e-2, 1, 273.15+20, 101325, 0.4)

% Outputs can be used
c.H_out
% = 3.4302e+05
c.dotQ
% = -8.3288e+03
c.Ta_out
% = 301.2785
c.P
% = 83.0151
```

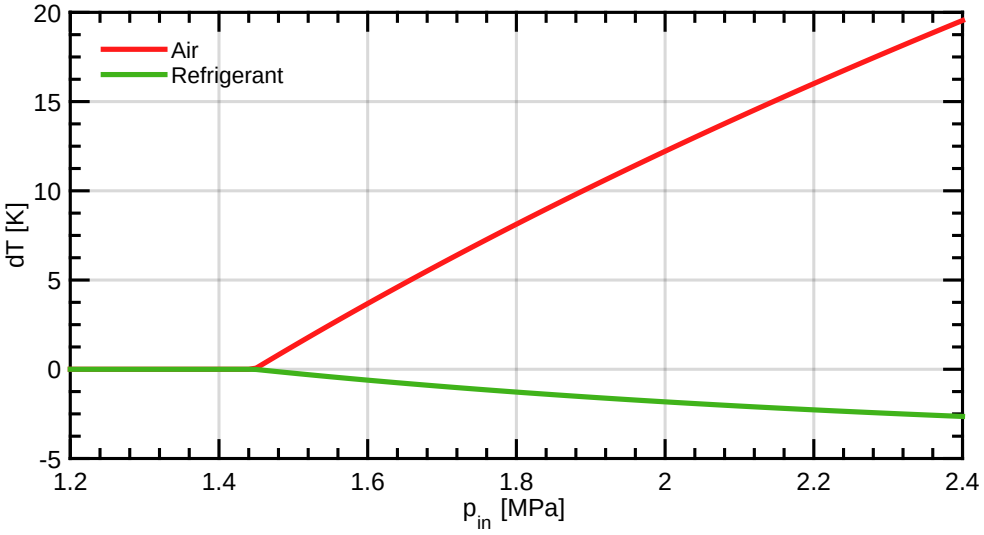


Figure 2.9: Condenser Refrigerant and Air Temperature changes as function of refrigerant pressure (thus saturation temperature) while other parameters are constant ($H_{in} = 440\,\mathrm{kJ/kg}$, $\dot{m} = 0.5\,\mathrm{kg/s}$, $\dot{m}_a = 1\,\mathrm{kg/s}$, $T_{a,in} = 20\,^{\circ}\mathrm{C}$, $p_{a,in} = 101.325\,\mathrm{kPa}$, $R = 40\,\%$) .

2.4 Expansion Valve

2.4.1 Description

Specific valve types are based on different principles and thus their characteristics may vary. Generally valves are used to regulate either volumetric flow rate (or mass flow rate) or downstream pressure (output pressure). Modeled valve is abstract instrument used to regulate output pressure. It is considered to represent variable orifice plate valve assuming incompressible flow (refrigerant is in liquid state).

Model consists of 1 output variable Output Pressure p_{out} and 5 input variables Input Pressure p_{in}, Input Enthalpy H_{in}, Refrigerant Mass Flow $\dot{m}$ and Valve Opening x. Model is based on these equations:

$$p_{out} = p_{in} - \Delta p \tag{2.5}$$

$$\dot{Q} = A_v \sqrt{\frac{\Delta p}{\rho}} \tag{2.6}$$

$$\dot{Q} = \frac{\dot{m}}{\rho} \tag{2.7}$$

$$H_{out} = H_{in} \tag{2.8}$$

Equation 2.5 is used to define Δp as pressure drop across valve openings. Equation 2.6 is taken from [4]. Parameter A_v has dimension of m^2 and thus represents effective orifice area. Equation 2.7 is simply derived from definition of density differentiated by time assuming time-invariant ρ. Equation 2.8 refers to assumption that valve has no impact on enthalpy of refrigerant.

By comparing equations 2.6 and 2.7 desired valve equation is found:

$$\frac{\dot{m}}{\rho} = A_v \sqrt{\frac{\Delta p}{\rho}} \tag{2.9}$$

$$\Delta p = \frac{1}{\rho} (\frac{\dot{m}}{A_v})^2 \tag{2.10}$$

Valve orifice is considered to be circular with variable diameter. However this relation between valve opening x and effective orifice area A_v is subject of model identification and might consist of other parameters. For used abstract valve A_v can be calculated using area of disk equation:

$$A_v = \pi (xr)^2 \tag{2.11}$$

where $x \in (0, 1]$ is valve opening and r is radius of orifice in fully opened state. Valve can not be fully closed because A_v is used as denominator in equation 2.10.

Last needed parameter is refrigerant density ρ measured in valve input opening. To get this density as a function of pressure and enthalpy CoolProp is used:

$$\rho = f_D(p_{in}, H_{in}) = \text{PropsSI}(D, P, p_{in}, H, H_{in}, medium) \tag{2.12}$$

Valve model consisting of equations 2.5, 2.8, 2.10 a 2.11 models physical properties of considered valve but it is affected with these practical shortcomings:

- For $\Delta p = p_{in}$ we get $p_{out} = 0$, which is inconsistent with defined $\dot{m}$.
- For $\Delta p > p_{in}$ we get $p_{out} < 0$, which is not possible by definition of pressure.

To deal with these problems following constraint is implement:

- Maximum Δp can be 99.5% of p_{in}

2.4.2 Definition

By putting all presented equations and constraints together final expansion valve model consists of following:

Model inputs:

- $x \in (0, 1]$ - Valve Opening [−]
- $\dot{m}$ - Refrigerant Mass Flow [kg/s]
- p_{in} - Refrigerant Input Pressure [Pa]
- H_{in} - Refrigerant Input Enthalpy [J/kg]

Model outputs:

- p_{out} - Refrigerant Output Pressure [Pa]
- H_{out} - Refrigerant Output Enthalpy [J/kg]

Model equations:

$$\begin{aligned} A_v &= \pi(xr)^2 \\ \rho &= f_D(p_{in}, H_{in}) \\ \Delta p' &= \frac{1}{\rho}(\frac{\dot{m}}{A_v})^2 \\ \Delta p &= \min(0.995 p_{in}, \Delta p') \\ p_{out} &= p_{in} - \Delta p \\ H_{out} &= H_{in} \end{aligned}$$

Model parameters (constants):

Parameter	Description	Unit	Value
r	Valve effective radius	[m]	2.5×10^{-3}

Table 2.5: Valve model parameters.

2.4.3 Usage and Behaviour

To verify model behaviour two simulations with varying parameters were performed. First simulation results of varying input enthalpy are presented in Figure 2.10 . Second simulation results of varying mass flow rate are presented in Figure 2.11. Example code usage is in listing 2.3.

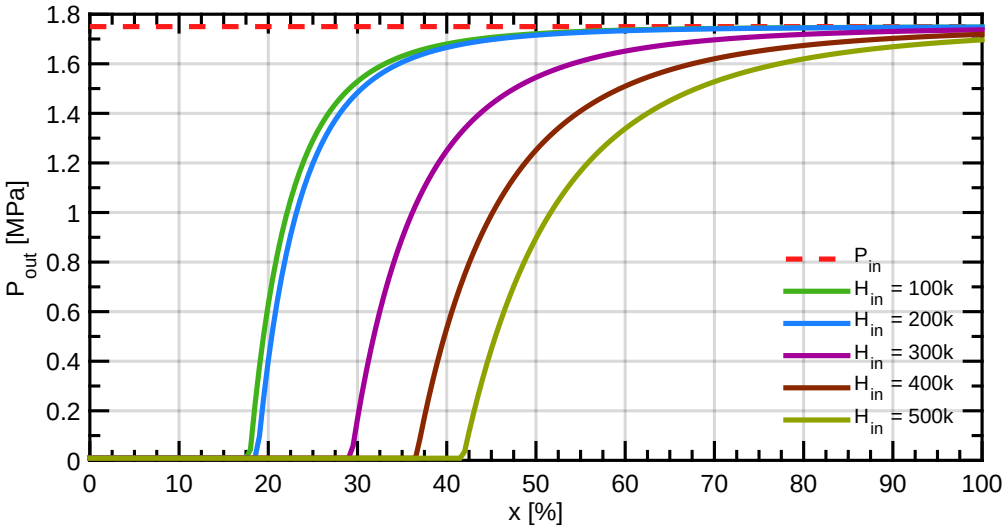


Figure 2.10: Valve Output Pressure with varying H_{in}, $r = 1\,\text{cm}$, $\dot{m} = 0.5\,\text{kg/s}$.

Listing 2.3: Example usage of Valve model

```
% Valve object with default parameters
v = VCC.Valve();

% Inputs are provided
% (p_in, H_in, dotm, x)
v.evaluate(1.8e6, 3.4e5, 7e-2, 0.66)

% Outputs can be used
v.p_out
% = 1.287e+06
```

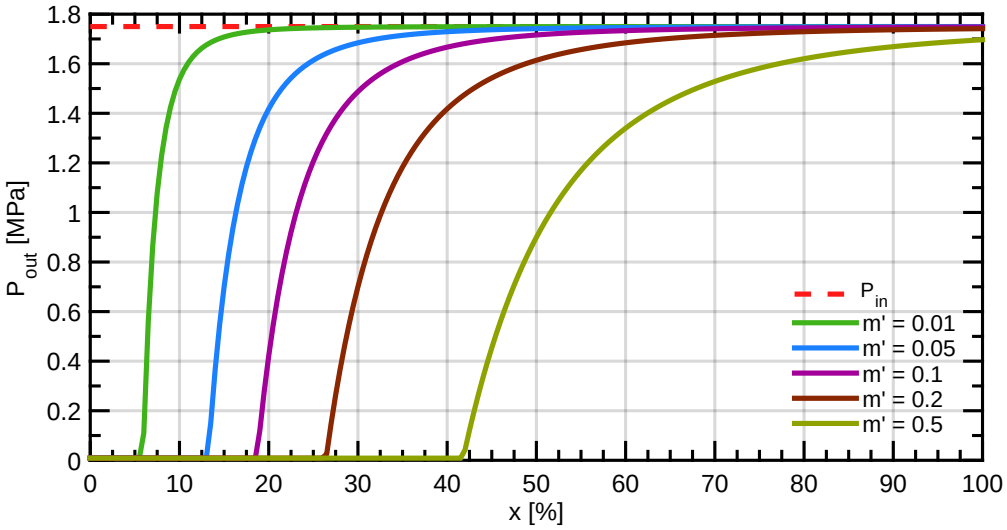


Figure 2.11: Valve Output Pressure with varying $\dot{m}$, $r = 1\,\mathrm{cm}$, $H_{in} = 499\,\mathrm{kJ/kg}$.

2.5 Evaporator

2.5.1 Description

In evaporator a mixture of refrigerant gas and liquid comes in. During isobaric evaporation refrigerant evaporates and exits evaporator in a gaseous state. Heat is transferred from the outside environment to the refrigerant thus the outside environment is cooled. Pressure of the refrigerant in a evaporator has to be such that the boiling temperature of refrigerant is below outside temperature.

Model consists of 5 output variables Refrigerant Output Enthalpy H_{out}, Refrigerant Output Pressure p_{out}, Heat Transfer Rate $\dot{Q}$, Air Output Temperature $T_{a,out}$ and Fan Power Consumption P_f, 7 input variables Refrigerant Input Pressure p_{in}, Refrigerant Input Enthalpy H_{in}, Refrigerant Mass Flow $\dot{m}$, Air Mass Flow $\dot{m}_a$, Air Input Temperature $T_{a,in}$, Air Input Pressure $p_{a,in}$ and Air Relative Humidity R. Model is based on equations show in Table 2.6.

Evaporator and Condenser models in this thesis are similar up to saturation/boiling temperature, parameters and direction in which heat is transferred ($\dot{Q} > 0$ for evaporator and $\dot{Q} < 0$ for condenser).

Description	Unit	Definition
Evaporator parameter	[−]	$k = \frac{UA}{\dot{m}_a c_{ha}}$
Temperature Difference	[K]	$\Delta T = T_{a,in} - T_b$
Heat Transfer Rate	[J/s]	$\dot{Q} = (1 - e^{-k})\dot{m}_a c_{ha} \Delta T$
Output Pressure	[Pa]	$p_{out} = p_{in}$
Output Enthalpy	[J/kg]	$H_{out} = H_{in} + \frac{\dot{Q}}{\dot{m}}$
Air Output Temperature	[K]	$T_{a,out} = T_{a,in} - \frac{\dot{Q}}{\dot{m}_a c_{ha}}$

Table 2.6: Evaporator model variables and equations.

Refrigerant Boiling Temperature T_b [K] (Saturated Vapor Temperature) is calculated using CoolProp as function of pressure and Vapor Mass Quality of 1.

$$T_b = f_T(p_{in}, 0) = \text{PropsSI}(T, P, p_{in}, Q, 1, medium)$$

Equations in Table 2.6 are further constrained to describe evaporator in desired physical state:

1. $T_{in} < T_b \leq T_{a,in}$ - Heat will flow only from air to refrigerant, achieved by:
   - Limiting $p_{in} < p_{in,ub}$ such that $p_{in,ub}$ is pressure for which saturated vapor temperature of refrigerant equals $T_{a,in}$.
   - Forcing $\Delta T = T_{a,in} - T_b$ always positive.
   - Assuming refrigerant is in liquid state.
2. $T_{out} \leq T_{a,in}$ - Refrigerant can not be heated above air temperature, achieved by:
   - Limiting $\dot{Q} \leq \dot{Q_{max}}$, where Q_{max} is heat required to be added so that refrigerant temperature reaches input air temperature.

$$\dot{Q_{max}} = (H_{out,max} - H_{in}) * \dot{m} \tag{2.13}$$

$$H_{out,max} = f_H(p_{in}, T_{a,in}) = \text{PropsSI}(H, P, p_{in}, T, T_{a,in}, medium) \tag{2.14}$$

Mass specific heat c_{ha} of humid air is calculated using same Equation 2.3 as in the Condenser Model.

Evaporator uses fan to generate air flow. Same Equation 2.4 is used as in the Condenser Model.

2.5.2 Definition

By putting all presented equations and constraints together final evaporator model consists of following:

Model inputs:

- p_{in} - Refrigerant Input Pressure [Pa]
- H_{in} - Refrigerant Input Enthalpy [J/kg]
- $\dot{m}$ - Refrigerant Mass Flow [kg/s]
- $\dot{m}_a$ - Air Mass Flow [kg/s]
- $T_{a,in}$ - Air Input Temperature [K]
- $p_{a,in}$ - Air Input Pressure [Pa]
- R - Air Relative Humidity [%]

Model outputs:

- p_{out} - Refrigerant output pressure [Pa]
- H_{out} - Refrigerant output enthalpy [J/kg]
- $\dot{Q}$ - Heat Transfer Rate [J/s]
- $T_{a,out}$ - Air Output Temperature [K]
- P_f - Fan Power Consumption [W]

Model equations:

$$
\begin{aligned}
w &= \text{HAPropsSI}(W, P, p_{a,in}, T, T_{a,in}, R, R) \\
c_w &= \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, water) \\
c_a &= \text{PropsSI}(C, P, p_{a,in}, T, T_{a,in}, air) \\
c_{ha} &= wc_w + (1-w)c_a \\
T_b &= \text{PropsSI}(T, P, p_{in}, Q, 1, medium) \\
k &= \frac{UA}{\dot{m}_a c_{ha}} \\
\Delta T &= \max(0, T_{a,in} - T_b) \\
\dot{Q} &= (1 - e^{-k})\dot{m}_a c_{ha} \Delta T \\
H_{out,max} &= \text{PropsSI}(H, P, p_{in}, T, T_{a,in}, medium) \\
\dot{Q_{max}} &= (H_{out,max} - H_{in}) * \dot{m} \\
\dot{Q} &= \min(\dot{Q}, \dot{Q_{max}}) \\
H_{out} &= H_{in} + \frac{\dot{Q}}{\dot{m}} \\
T_{a,out} &= T_{a,in} - \frac{\dot{Q}}{\dot{m}_a c_{ha}} \\
p_{out} &= p_{in} \\
\rho_a &= \text{PropsSI}(D, P, p_{a,in}, T, T_{a,in}, air) \approx 1.2046 \\
\dot{V}_a &= \frac{\dot{m}_a}{\rho_a} \\
P_f &= \frac{1}{\eta_f} \Delta p_f \dot{V}_a
\end{aligned}
$$

Model parameters (constants):

Parameter	Description	Unit	Value
U	Heat Transfer Coefficient	[W/m^2/K]	50.5
A	Heat Transfer Surface	[m^2]	69
η_f	Fan Efficiency	[−]	0.50
Δp_f	Fan Pressure Drop	[Pa]	50

Table 2.7: Evaporator model parameters.

2.5.3 Usage and Behaviour

Model behaviour is similar to the Condenser Model. However only evaporation is allowed as can be seen in Figure 2.12. Once boiling temperature is above input air temperature heat exchange stops. Example code usage is in listing 2.4.

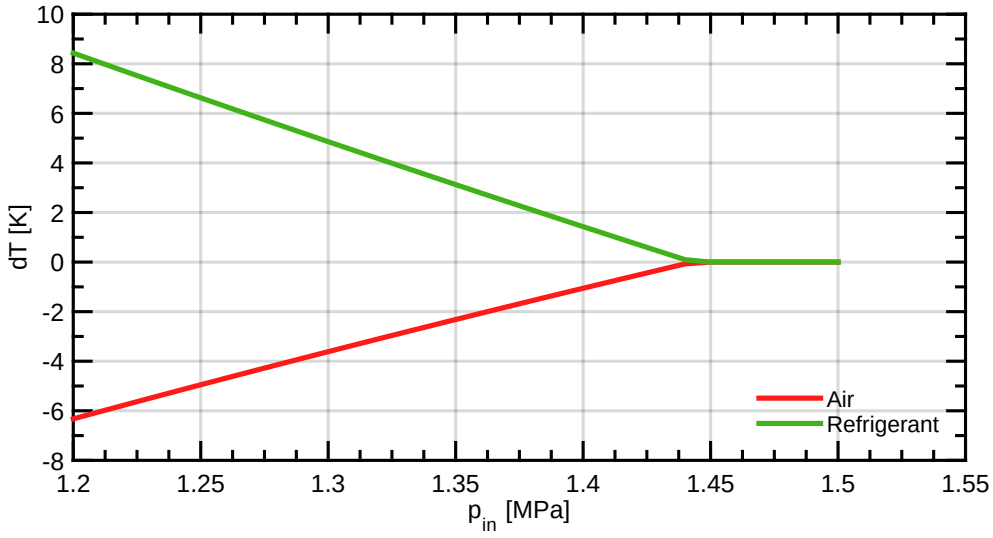


Figure 2.12: Evaporator Refrigerant and Air Temperature changes as function of refrigerant pressure (thus boiling temperature) while other parameters are constant ($H_{in} = 200\,\mathrm{kJ/kg}$, $\dot{m} = 0.5\,\mathrm{kg/s}$, $\dot{m}_a = 1\,\mathrm{kg/s}$, $T_{a,in} = 20\,^{\circ}\mathrm{C}$, $p_{a,in} = 101.325\,\mathrm{kPa}$, $R = 40\,\%$) .

Listing 2.4: Example usage of Evaporator model

```
% Evaporator object with default parameters
e = VCC.Evaporator();

% Inputs are provided
% (p_in, H_in, dotm, dotma, Ta_in, p_a, R)
e.evaluate(1.3e6, 3.4e5, 7e-2, 1, 273.15+20, 101325, 0.4)

% Outputs can be used
e.H_out
% = 3.9291e+05
e.dotQ
% = 3.7034e+03
e.Ta_out
% = 289.5356
e.P
% = 83.0151
```

2.6 Computational Performance

Since CoolProp is used to calculate physical properties model evaluation is resource intensive. To speedup up evaluation CoolProp calls were minimized and all models are capable of using CoolProp tabular interpolation.

However when using CoolProp Low Level API with tabular data some function calls result in different behaviour. So several CoolProp calls are using High Level Interface without interpolation anyway, in this cases tabular interpolation is not applicable.

Computation speedup using tabular interpolation is shown in Table 2.8. Code example in Listing 2.5.

Model	Native [ms]	Interpolation [ms]	Speedup
Compressor	3.4	1.9	44 %
Condenser	3.6	2.8	22 %
Valve	1.1	0.7	36 %
Evaporator	3.8	2.9	24 %

Table 2.8: Comparison of computation time using bicubic interpolation and native CoolProp. Average of 5000 evaluations is displayed (Matlab®2014b, Mac OS X 10.11.2, 2GHz Intel Core 2 Duo)

Listing 2.5: Using models with CoolProp tabular interpolation

```
% Single instance of CP is created using BICUBIC&HEOS
% First run takes several minutes
% Tabular data are stored in ~/.CoolProp/
CP = CoolProp.AbstractState ...
  .factory('BICUBIC&HEOS', 'R410a');

% Then model objects are created
% CP instance is passed to them using .setCP()
valve = VCC.Valve();
comp  = VCC.Compressor();
evap  = VCC.Evaporator();
cond  = VCC.Condenser();

valve.setCP(CP);
comp.setCP(CP);
evap.setCP(CP);
cond.setCP(CP);
```

Chapter 3

Solving Systems of Non-Linear Equations

Solving systems of non-linear equations is generic problem with variety of approaches and more or less promising algorithms. System of m non-linear equations with n variables is for purpose of this thesis defined as:

$$\begin{aligned} f_i(\mathbf{x}) &= 0, & \mathbf{x} &\in \mathbb{R}^n \\ f_i &: \mathbb{R}^n \mapsto \mathbb{R}, & i &= 1 \dots m, \end{aligned} \tag{3.1}$$

where f_i are particular functions of each equation and a $\mathbf{x}$ is system solution.

Analytical methods exists for specific cases which typically restrict possible f_i functions. For example Newton-Rapshon method requires continuous and practically differentiable f_i.

Stochastic methods which searches solution space represent quite opposite approach. However they can work with almost any f_i function. One of these methods is Particle Swarm Optimization which will be used in this thesis.

To solve system of equations using optimization method problem have to be formulated. In this thesis optimization problem of solving system of equations is defined as:

$$\begin{aligned} \min_{\hat{\mathbf{x}}} \quad & g(f_1(\hat{\mathbf{x}}), \dots, f_m(\hat{\mathbf{x}})) \\ \text{such that} \quad & \\ & \mathbf{lb} \leq \hat{\mathbf{x}} \leq \mathbf{ub} \\ & g \geq 0 \end{aligned} \tag{3.2}$$

where g is function such that $g \to 0 \Rightarrow \hat{\mathbf{x}} \to \mathbf{x}$. Bounds $\mathbf{lb}$ and $\mathbf{ub}$ limit searched space.

Various functions can be used for g. In this thesis L_1, L_2, L_∞ norms are used:

$$g_a(f_1(\hat{\mathbf{x}}), \ldots, f_m(\hat{\mathbf{x}})) = \|\{f_i(\hat{\mathbf{x}})\}\|_1 = \sum_{i=1}^{m} |f_i(\hat{\mathbf{x}})| \tag{3.3a}$$

$$g_b(f_1(\hat{\mathbf{x}}), \ldots, f_m(\hat{\mathbf{x}})) = \|\{f_i(\hat{\mathbf{x}})\}\|_2 = \sqrt{\sum_{i=1}^{m} f_i^2(\hat{\mathbf{x}})} \tag{3.3b}$$

$$g_c(f_1(\hat{\mathbf{x}}), \ldots, f_m(\hat{\mathbf{x}})) = \|\{f_i(\hat{\mathbf{x}})\}\|_\infty = \max\{|f_i(\hat{\mathbf{x}})|, \ldots, |f_m(\hat{\mathbf{x}})|\} \tag{3.3c}$$

For norm holds that $\|\{f_i(\hat{\mathbf{x}})\}\| \geq \|f_i(\hat{\mathbf{x}})\| \geq 0$. Due optimization $\|\{f_i(\hat{\mathbf{x}})\}\| \to 0$ which implies $\|f_i(\hat{\mathbf{x}})\| \to 0 \Rightarrow f_i(\hat{\mathbf{x}}) \to 0 \Rightarrow \hat{\mathbf{x}} \to \mathbf{x}$.

This is proof that $g_{a,b,c} \to 0 \Rightarrow \hat{\mathbf{x}} \to \mathbf{x}$ and that if objective function converges to 0 one of possible solutions of Equation 3.1 was found.

3.1 Particle Swarm Optimization

PSO is stochastic optimization method inspired by swarm intelligence typically perceived in realm of insect. Algorithm is based on finite set of particles which are iteratively updated and examined.

Algorithm first appeared in 1995 when Kennedy and Ebhart used PSO to simulate social behaviour[5]. Algorithm 1 explains their idea. Because of its simple extensibility it was often modified and improved. Even original authors published improvements during the same year[6].

This lead to large number of independent PSO clones which were nothing more than different implementation of typically same or similar enhancements. This is the reason why in 2007 Bratton and Kennedy proposed standardized PSO algorithm referenced as SPSO 2007 [7]. Together with SPSO authors proposed standardized set of problems and methodics to benchmark and compare different PSO versions.

This standard evolved with new ideas and contributions [8]. Last published standard is SPSO 2011 by M. Clerc [8]. New PSO variants should always overcome current SPSO before published or claimed revolutionary.

Algorithm 1 Particle swarm optimization principle

```
1:  function PrinciplePSO(f())                          ▷ f : ℝ^n ↦ ℝ
2:      S ← Number of particles
3:      for i ← 1, S do                                 ▷ Particle initialization
4:          X_i ← Initial position      ▷ Typically sampled from uniform distribution
5:          V_i ← Initial velocity      ▷ Typically sampled from uniform distribution
6:          P_i ← X_i                                   ▷ Initial local optimum
7:      end for
8:      G ← arg min_{P_i} f(P_i)                        ▷ Initial global optimum
9:      while not terminated do
10:         for i ← 1, S do                             ▷ Particle update
11:             V_i ← New velocity                      ▷ Stochastic function
12:             X_i ← X_i + V_i                         ▷ New position
13:             if f(X_i) < f(P_i) then     ▷ Update of local and global optimum
14:                 P_i ← X_i
15:                 if f(X_i) < f(G) then
16:                     G ← P_i
17:                 end if
18:             end if
19:         end for
20:     end while
21:     return G
22: end function
```

Algorithm 1 explains basic PSO idea. Implementation of generalized parts define PSO behaviour. Particular variants are typically unique in:

- Particle initialization
- Velocity $\mathbf{V}_i$ update
- Termination conditions

Function which updates velocity typically randomly combines velocity V_i from previous iteration, vector pointing towards local optimum (e.g. $(P_i - X_i)$) and vector pointing towards global optimum (e.g. $(G - X_i)$). Global optimum is best known position across all particles and after termination it is returned as result. Local optimum is best known position to either one particle or to some subset of particles. In case of subsets topology of swarm have to be considered.

Termination condition often consist of several criteria because PSO as well as other numerical algorithm is limited in its ability to detect reaching global optimum. Usually used conditions:

- Objective function stalling: $|f(\mathbf{G}(i-1)) - f(\mathbf{G}(i))| \leq f_{tol}$ for i_{tol} consecutive iterations,
- Reaching maximum number of iterations: $i > i_{max}$,
- Reaching objective function goal: $f(\mathbf{G}(i)) \leq f_{goal}$,
- Reaching maximum running time: $t_{run} > t_{max}$.

Algorithm 2 is simple PSO variant which terminates after reaching desired objective function value f_g. ω, ϕ_p, ϕ_g are parameters of PSO. S is number of particles in swarm. Topology of particles is global (one big set).

Social parameters significantly affect PSO behaviour and their selection is complex problem. Together with standardized set of problems used to benchmark various PSO modifications parameter selection was evaluated as well. One of these attempts is documented by [9] and another by [10]. Usually table of acceptable parameter ranges is presented. This evaluations however focus on overall best performance and convergence usually with large number of iterations.

Algorithm 2 Simple PSO

function SIMPLEPSO($f(\mathbf{x}), f_g, \mathbf{lb}, \mathbf{ub}, n, S, \omega, \phi_p, \phi_g$)
  $X_{i,j} \leftarrow U(lb_j, ub_j)$ ▷ Initial position is random from interval
  $V_{i,j} \leftarrow U(-|ub_j - lb_j|, |ub_j - lb_j|)$ ▷ Initial velocity
  $P_i \leftarrow X_i$ ▷ Best local optimum is its position
  $G \leftarrow arg\ min(f(P_i))$ ▷ Best global optimum
  while $f(G) > f_g$ **do** ▷ Termination condition
    for $i \leftarrow 1, n$ **do** ▷ Particle update
      $r_p \leftarrow U(0, 1)$
      $r_g \leftarrow U(0, 1)$
      $V_i \leftarrow \omega V_i + \phi_p r_p (P_i - X_i) + \phi_g r_g (G - X_i)$
      $X_i \leftarrow X_i + V_i$
      if $f(X_i) < f(P_i)$ **then**
        $P_i \leftarrow X_i$
        if $f(X_i) < f(G)$ **then**
          $G \leftarrow P_i$
        end if
      end if
    end for
  end while
  return G
end function

3.2 Solver Implementation

One of guidelines of this diploma thesis assignment was to implement PSO based solver. PSO solver was implemented in Matlab®2014b and is further referred as PSOS. Function to perform PSOS is named `psos(fun,lb,ub,opts,init_p)`, options structure is named `psooptions`. Both source files are well commented.Algorithm 2 was used and extended with following:

- **Scaling** - Lower and upper bounds are scaled so that searched space has range 1.0 in all dimension. This should improve numerical stability. If $lb = ub$ for particular dimension then this dimension is fixed.
- **Particle set seeding** - Set of initial particles can be provided as input parameter. If swarm size is larger then set of provided particles then the rest of particles is sampled from uniform distribution. This is usable for conducting pre-runs - evaluate first two iterations several times and then pass some of the best particles from all pre-runs to final run with normal termination conditions. Or any other solutions can be passed (e.g. guessed solutions).
- **Teams** - During initialization particles are randomly divided into separate teams, creating non flat swarm topology. Number of teams is one of parameters with default value 1 (Global topology).
- **Update equation** (r_p, r_g, r_t are sampled from $U(0,1)$, P_i is particle best known position, G is swarm best known position and T is particle's team best known position)

$$\begin{aligned} V_i &= \omega V_i + \phi_p r_p (P_i - X_i) + \phi_g r_g (G - X_i) + \phi_t r_t (T - X_i) \\ X_i &= X_i + V_i \end{aligned} \tag{3.4}$$

- **Position bounding** - Solver restricts possible particle positions using bounds provided. Once particle tries to escape its bounds position is clamped and speed is zeroed (per dimension) as recommended by Carlisle and Dozier [10].
- **Default parameters**
  - Swarm Size $S = 50 * n$, where n is dimension of objective function
  - Number of Teams $T = 1$
  - Damping $\omega = 0.05$ (Inertia parameter)
  - Particle Weight $\phi_p = 1.8$ (Cognitive parameter)
  - Swarm Weight $\phi_g = 0.1$ (Social parameter)
  - Team Weight $\phi_t = 0.9$ (Social parameter)
- **Termination conditions**
  1. Objective function stalling according to tolerance (default 10 iterations with 1×10^{-6} tolerance)
  2. Maximum number of iterations (default 50)
  3. Reaching objective function goal (default $-\infty$)

3.3 PSOS Usage

Example usage of PSOS for following problem is in Listing 3.1, following set of equations is solved using default parameters:

$$\begin{aligned} 3x_1^3 + 4x_2^2 - 145 &= 0 \\ 4x_1^2 - x_2^3 + 28 &= 0 \end{aligned} \tag{3.5}$$

Analytical solution of this problem yields $x = (3, 4)$, same solution is found using PSOS in 27 iterations with 100 particles.

Listing 3.1: Example usage of PSOS to solve set of polynomial equations

```
% Set of equations (PSOS minimizes L_1 Norm by default)
f = @(x) [ ...
  3*x(1)^3 + 4*x(2)^2 - 145; ...
  4*x(1)^2 - x(2)^3 + 28 ...
];

% Bounds
ub = [10 10];
lb = [-10 -10];

% Call solver with default options
rng default;
opts = psooptions;
res = psos(f,lb,ub,opts);

%% Expected Output
%Particle Team Sizes = [ 100 ]
%PSOS done with 27 iter., f(X) = [8.86e-11], X = [3  4]
%PSO was terminated by Stall/TolFun condition.
%Total duration 0.648 seconds.
```

Default parameters are more or less compatible with parameters recommended in various sources. Their fine tuning using large number of experiments and statistical evaluation for particular problem is reasonable continuation of this thesis. However it would be more applicable to use standard PSO implementation as results would be comparable with other peers.

Chapter 4

Vapor Compression Cycle Solving

4.1 Model Configuration

There are several ways how model presented in Chapter 2 can be put together to form set of equations. Used configuration of the model enable using output variables of one component as input variables for the next one. It is important to choose state variables and formulate loop closing conditions. Some variables have ambient/desired character and some are control variables.

Configuration used in this thesis is shown in Figure 4.1. Other configurations were tried as well. Especially using each component as standalone model and creating loop closing state variables between components for each variable. However this approach resulted in 12+ state variables and more loop closing conditions. This required using more particles for PSO solver and sometimes even did not converged. Used configuration mitigates this problem and has only 7 state variables and 2 loop closing conditions. More dimensions generally means more problems.

Valve does not change enthalpy $H_{in} = H_{out}$ and Evaporator neither Condenser do not change pressure $p_{in} = p_{out}$ this is used to minimize number of state variables.

State variables:

- p_{23} - Low pressure at Compressor input
- p_{41} - High pressure at Compressor output
- H_1 - Refrigerant enthalpy at Compressor input
- n - Compressor frequency (Control Variable)
- x - Valve opening (Control variable)
- $\dot{m}_{e,a}$ - Evaporator Air Mass Flow (Control variable)
- $\dot{m}_{c,a}$ - Condenser Air Mass Flow (Control variable)

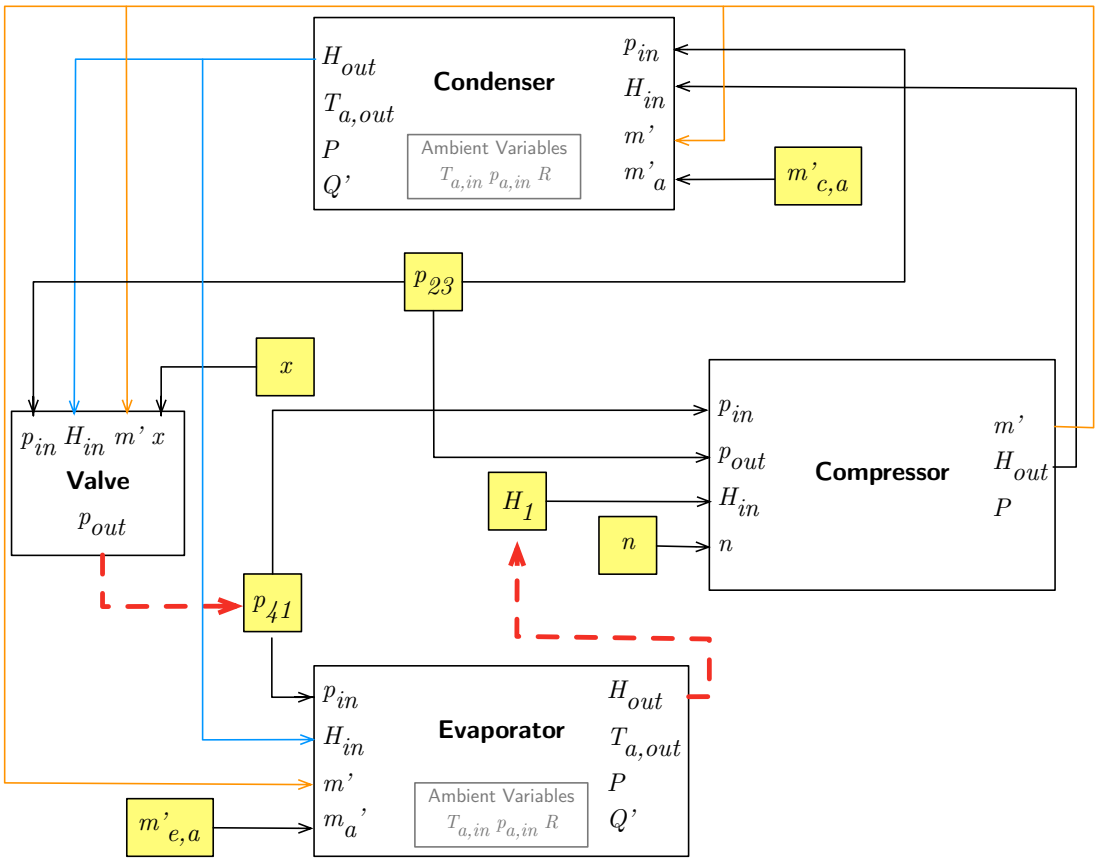


Figure 4.1: VCC Model configuration used in this thesis. State variables are in yellow boxes and red dashed arrows show where loops are. Evaluation happens in order of VCC cycle starting with compressor. Compressor defines $\dot{m}$ for all following components.

Ambient variables (Provided by sensors or estimate):

- Condenser $T_{a,in}$ - Input Air Temperature
- Condenser $p_{a,in}$ - Input Air Pressure
- Condenser R - Input Air Relative Humidity
- Evaporator $T_{a,in}$ - Input Air Temperature
- Evaporator $p_{a,in}$ - Input Air Pressure
- Evaporator R - Input Air Relative Humidity

Evaluation of model in this configuration starts with compressor and ends with evaporator as inputs are provided after each component is evaluated. To find steady state (close loop) following equations must hold for candidate solution:

$$\begin{aligned} p_{out} - p_{23} &= 0 \\ H_{out} - H_1 &= 0 \end{aligned} \tag{4.1}$$

where p_{out} is valve output pressure and H_{out} is evaporator output enthalpy. From configuration perspective valve output is not used anywhere else than in this condition so valve can be solved separately from rest of the model assuming solution always exists. This is used in Section 4.6.

Interesting output variables:

- Compressor P - Power Consumption
- Condenser P - Fan Power Consumption
- Evaporator P - Fan Power Consumption
- Condenser $\dot{Q}$ - Heat Transfer Rate
- Condenser $T_{a,out}$ - Output Air Temperature
- Evaporator $T_{a,out}$ - Output Air Temperature

From output variables COP can be calculated:

$$COP = \frac{\dot{Q}}{P_{comp} + P_{evap} + P_{cond}} \tag{4.2}$$

where $\dot{Q}$ is Heat Transfer Rate in condenser, and P_{comp}, P_{evap} and P_{comp} are Power Consumption of respective components.

At this point model and problem of finding steady state according to this configuration can be formulated as set of equations:

$$\begin{aligned} \mathbf{x} &= (p_{23}, p_{41}, H_1, n, x, \dot{m}_{e,a}, \dot{m}_{c,a}) \\ f_1(\mathbf{x}) &= p_{out} - p_{23} = 0 \\ f_2(\mathbf{x}) &= H_{out} - H_1 = 0 \end{aligned} \tag{4.3}$$

p_{out} and H_{out} are evaluated from $\mathbf{x}$ using VCC components interconnected as shown in Figure 4.1.

This of course does not work because further bounds and constraints have to be implemented. As well as two function f_1 and f_2 have to be combined into single real function as described in Equation 3.2 (using norm). This step is application specific and is further discussed in Sections 4.3 and 4.6.

4.2 Bounds and Constraints

To use PSOS all variables have to be box bounded. This bounds are used to sample initial particles as well as to limit particle movement. Using appropriate bounds enforce model behaviour. Used bounds are shown in Table 4.1.

State variable bounds define problem solved using PSOS. For example if used together with grid optimization control variables bounds would be set exactly to current set point.

Variable	Lower Bound	Upper Bound	Note
p_{23}	$p_{23,lb}$	3 MPa	
p_{41}	0.5 MPa	$p_{41,ub}$	
H_1	$H_{1,lb}$	500 kJ/kg	
n	5 Hz	120 Hz	Control variable
x	0.01 %	90 %	Control variable
$\dot{m}_{e,a}$	0.1 kg/s	10 kg/s	Control/Desired variable
$\dot{m}_{c,a}$	0.1 kg/s	10 kg/s	Control/Desired variable

Table 4.1: State variables bounds. This bounds are considered default and should be altered according to particular problem.

Following constraints are imposed:

1. Ensure Heat Exchange in Evaporator ($T_b < T_{a,in}$). This is achieved setting upper bound of low pressure such that refrigerant boiling temperature is below input air temperature at evaporator.

$$p_{41,ub} = \text{PropsSI}(P, T, T_{a,in}^{evap}, Q, 1) * 0.99999 \tag{4.4}$$

2. Ensure Heat Exchange in Condenser ($T_s > T_{a,in}$). This is achieved setting lower bound of high pressure such that refrigerant saturation temperature is above input air temperature at condenser.

$$p_{23,lb} = \text{PropsSI}(P, T, T_{a,in}^{cond}, Q, 0) * 1.00001 \tag{4.5}$$

3. Ensure refrigerant is in gaseous state reaching compressor. This is achieved setting lower bound on enthalpy at compressor input such that even for lowest possible pressure vapor saturation curve is crossed. Since saturated vapor curve is almost vertical in region from 0.5 MPa to 3 MPa as show in Figure 2.3, constant value of $p_{41,lb}$ can be used with minimal error.

$$H_{1,lb} = \text{PropsSI}(H, P, p_{41,lb}, Q, 1) \tag{4.6}$$

4. Ensure refrigerant is in liquid state reaching valve. This is achieved setting upper bound on enthalpy at valve input such that even for lowest possible pressure liquid saturation curve is crossed.

$$H_{34,ub} = \text{PropsSI}(H, P, p_{23,lb}, Q, 0) \tag{4.7}$$

All constraints but the last are realized using bounding box. H_{34} is not one of state variables thus meeting this constraint is achieved using additional Equation 4.8. So once saturation curve is crossed this equation holds.

$$f_3(\mathbf{x}) = \max(0, H_{out}^{cond} - H_{34,ub}) = 0 \tag{4.8}$$

4.3 Standalone VCC for Grid Optimization

First problem addressed in this chapter is solving VCC for grid optimization as described in Figure 1.1. In this situation initial solution is provided by solution estimator and all control variables are fixed according to step based optimizer. PSOS should solve VCC for steady state solution. COP should be calculated as well as all internal variables (pressures, enthalpies).

To formulate problem for PSOS cost function and variable bounds have to be provided. Variable bounds are based on defaults in Table 4.1.

Example set point is solved. This set point is chosen so heat is transferred from outside (evaporator) to inside (condenser). Air temperature outside is lower then inside so this is typical heat pump application.

Ambient conditions:

- Evaporator (Outside)
  - $T_{a,in} = 20\,°\mathrm{C} = 293.15\,\mathrm{K}$
  - $p_{a,in} = 101\,325\,\mathrm{Pa}$
  - $R = 40\,\%$
- Condenser (Inside)
  - $T_{a,in} = 24\,°\mathrm{C} = 297.15\,\mathrm{K}$
  - $p_{a,in} = 101\,325\,\mathrm{Pa}$
  - $R = 40\,\%$

Control variables are set (changing bounds of state variables to exact value):

- $n = 15\,\mathrm{Hz}$
- $\dot{m}_{e,a} = 0.5\,\mathrm{kg/s}$
- $\dot{m}_{c,a} = 0.8\,\mathrm{kg/s}$

VCC equations for steady state are assembled to form final set. Cost function F is defined:

$$\begin{aligned} f_1(\mathbf{x}) &= p_{out} - p_{23} = 0 \\ f_2(\mathbf{x}) &= H_{out} - H_1 = 0 \\ f_3(\mathbf{x}) &= \max(0, H_{out}^{cond} - H_{34,ub}) = 0 \\ F(\mathbf{x}) &= \left\| \begin{pmatrix} 10^{-5} \cdot f_1(\mathbf{x}) \\ 10^{-3} \cdot f_2(\mathbf{x}) \\ 10^{-3} \cdot f_3(\mathbf{x}) \end{pmatrix} \right\|_2 \end{aligned} \tag{4.9}$$

Variable	Unit	LB	UB
p41	MPa	0.5000	1.4429
p23	MPa	1.6136	3.0000
H1	kJ/kg	416.5936	500.0000
comp.n	Hz	15.0000	15.0000
valve.x	Hz	0.0100	0.9000
evap.dotma	kg/s	0.5000	0.5000
cond.dotma	kg/s	0.8000	0.8000

Figure 4.2: Standalone VCC Calculated Bounds. Notice $ub = lb$ for control variables.

Weights used in Equation 4.11 are set to compare precision of different physical units ($1\,\mathrm{kJ/kg} \sim 0.1\,^{\circ}\mathrm{C} \sim 100\,\mathrm{kPa}$).

State variable bounds together with F function which is to be minimized are passed to PSOS. See Listing 4.1 for code usage example, Figure 4.3 for found solution and Figure 4.4 for solution P-H Diagram. PSOS options were default but limited to 15 Iterations, 3 Teams and 250 Particles.

Listing 4.1: Example usage of Standalone VCC Solver

```
m=VCC.Model(); %(comp_n,
% evap_dotma, evap_Ta_in, evap_p_a, evap_R,
% cond_dotma, cond_Ta_in, cond_p_a, cond_R)
m.evaluate1(15, ...
    0.5, 273.15+20, 101325, 0.4,...
    0.8, 273.15+24, 101325, 0.4...
);

%Initiating PSOS to find solution
%Particle Team Sizes = [ 93  74  83 ]
%PSOS done with 15 iterations, f(X) = [4.5138e-07], X =
   [...]
%PSO was terminated by MaxIter condition.
%Total duration 50.877 seconds.

%m.print();
%m.plot_ph();
%m.print_bounds();
```

Variable	Unit	Value	Error
Solution			
p41	MPa	0.8907	0.0000
p23	MPa	2.2861	-
H1	kJ/kg	440.257	0.0000
comp.n	Hz	15.000	-
valve.x	%	13.185	-
evap.dotma	kg/s	0.500	-
cond.dotma	kg/s	0.800	-
Quality			
comp.P	W	515.29	-
evap.P	W	41.51	-
cond.P	W	66.41	-
Total P	W	623.21	-
COP	-	5.11	-
Internal			
comp.dotm	g/s	13.19	-
Air		**In**	**Out**
cond.Ta	C	24.00	27.87
evap.Ta	C	20.00	14.79
Error			
Total Error	-	4.5e-07	-

Figure 4.3: Standalone VCC PSOS solution and other interesting variables. Output generated using function $VCC.Model.print()$.

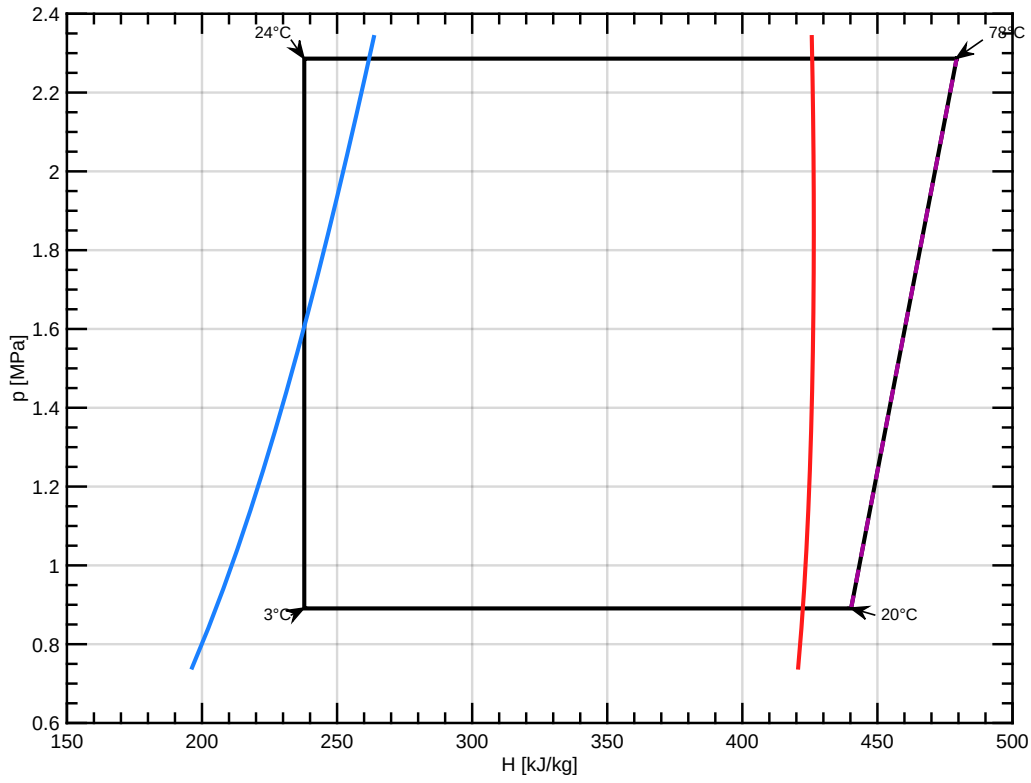


Figure 4.4: Standalone VCC PSOS solution P-H diagram. Blue line is saturated liquid line and red line is saturated vapor line. In each corner of the cycle is refrigerant temperature. This diagram was generated using function $VCC.Model.plot\_ph()$.

4.4 Initial Particle Sampling Sensitivity

PSOS is stochastic solver and requires random numbers. Single run of PSOS is only sample of solver behaviour. To asses this multiple runs were performed. Result of 250 runs is shown in Figure 4.5. Solutions were compared and analyzed see median and standard deviation in Equation 4.10. Single run takes 1 minute and any tuning of parameters is extremely time consuming since every parameter alteration have to be evaluated in considerable number of sample runs to assess wherever it is advantageous.

$$\begin{aligned} \mathrm{Med}(\hat{\mathbf{x}}) &= (930\,\mathrm{kPa}, 2.21\,\mathrm{MPa}, 439\,\mathrm{kJ/kg}, 15\,\mathrm{Hz}, 0.1379, 0.5\,\mathrm{kg/s}, 0.8\,\mathrm{kg/s}) \\ \mathrm{Std}(\hat{\mathbf{x}}) &= (130\,\mathrm{kPa}, 222\,\mathrm{kPa}, 3\,\mathrm{kJ/kg}, 0, 0.028, 0, 0) \end{aligned} \tag{4.10}$$

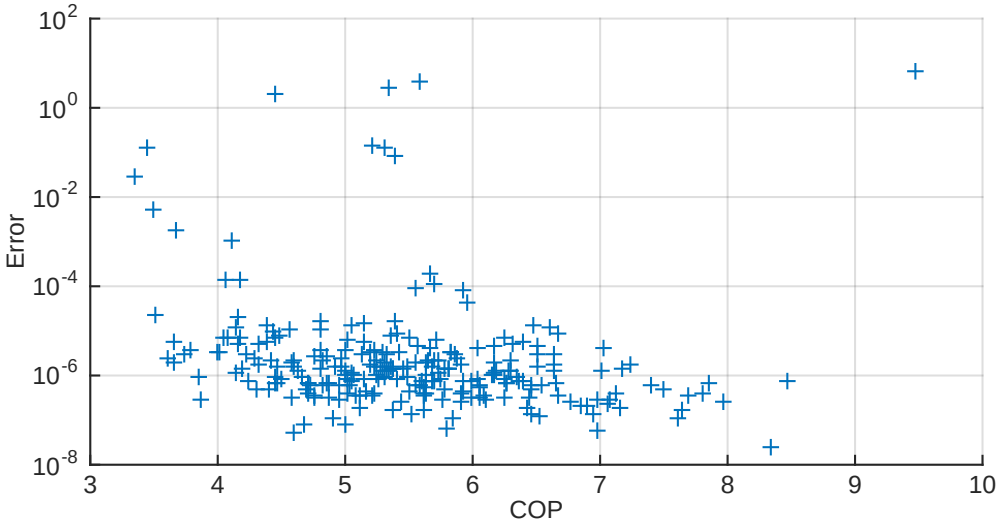


Figure 4.5: Standalone VCC PSOS 250 solutions according to Listing 4.1. Error is plotted in logarithmic scale. All solutions above 10^{-4} are unacceptable. Varying COP is witnessing how solutions differ.

Following ideas were evaluated to mitigate observed behaviour:

1. **Random numbers** - Quality of random numbers was examined. Random numbers are generated using Matlab®function `rand` which uses Mersenne twister algorithm to generate pseudo-random numbers. This generator is suitable for this kind of application and random numbers should not be causing this problem.
2. **Objective function** - Idea of changing objective function to find some enhancements naturally arised. When using L_1 instead of L_2 in Equation 4.9 solutions tend to be less dispersed however no great enhancement was found. Altering weights did not show any benefit.

3. **PSO parameters** - PSO parameters are extremely complex to assess and thus only several experiments were performed. One notable example of parameters tuning - increasing $\phi_t = 0.9 \rightarrow 1.5$ speeds up convergence (Maybe even 10 iterations would suffice), but still solutions did not converged around single one. Several articles about PSO parameters were studied however since custom PSO implementation is used their recommendation were less applicable.
4. **Initial particle sampling** - Solver was very sensitive to changing initial particle sampling. More particles generated during initial phase increases probability of hitting some sweet spots in state space from which PSO converges. However adding particles results in longer computation time and thus this is not practical.
    - One of possible enhancements consists of running several runs (e.g. 10) of PSO with lower number of particles (e.g. 100) and choosing best found solution - this decreases probability of getting stuck with the worst one. However most of computational time is wasted.
    - Another approach which was found usable is to run several (5) quick pre-runs with only 2-3 iterations and then take best 50 particles of each pre-run and use then to seed final run with standard options, swarm size and iteration count. Pre-runs appeared as decent approach to suppress this sensitivity at reasonable cost. Solutions were still not converging consistently.
5. **Swarm divided into Teams** - Increasing number of particles helps PSO with initial sampling however it is happens that after several iterations PSO will get stuck in some local minimum. To mitigate this behaviour swarm topology was changed into several teams (e.g. 3) which have weak coupling but in the end converge to single solution. Team enhancement helps mitigate the problem and does not require any additional particles or PSO runs. This was implemented in PSOS and is used by default for VCC solving. More then 3 teams did not proven any additional benefit.
6. **Narrow bounds** - More tightly set bounds benefits PSOS since search space is considerably smaller (particle density increases). One approach to get narrow bounds would be to use solution estimator and set bounds around it. Another idea was to get solution from first PSO run and have second run with narrow bounds (around first solution). This always helped at cost of additional PSO run. However this approach is not generally applicable.

Root cause of this behaviour was not identified. None of approaches was promising enough to be more studied. In-depth analysis should be done while using standard PSO implementation so results could be compared with other peers. VCC Model should be verified and compared with real VCC behaviour.

4.5 Comparison to fmincon

Same problem of Standalone VCC Solving was solved using `fmincon` Matlab®solver. With default options this solver uses interior-point algorithm and requires initial solution guess. Initial solution was set to $\mathbf{x}_0 = \frac{\mathbf{ub}-\mathbf{lb}}{2} + \mathbf{lb}$. Evaluation took 7 seconds. Found solution is shown in Figures 4.6 and 4.6.

Different solution is found if using $\mathbf{x}_0 = \mathbf{lb}$ thus fmincon is strongly sensitive to provided solution and converges to local optimum. This is somehow similar to how PSO solves this problem and is sensitive to initial particle sampling.

Both algorithms are successful with finding feasible solution. PSOS has potential to find better solution. fmincon is faster. Providing better initial solution to fmincon will render PSOS inferior.

Variable	Unit	Value	Error
Solution			
p41	MPa	0.9709	0.0000
p23	MPa	2.3068	-
H1	kJ/kg	438.420	-0.0000
comp.n	Hz	15.000	-
valve.x	%	14.079	-
evap.dotma	kg/s	0.500	-
cond.dotma	kg/s	0.800	-
Quality			
comp.P	W	516.68	-
evap.P	W	41.51	-
cond.P	W	66.41	-
Total P	W	624.60	-
COP	-	5.55	-
Internal			
comp.dotm	g/s	14.72	-
Air		**In**	**Out**
cond.Ta	C	24.00	28.21
evap.Ta	C	20.00	14.24
Error			
Total Error	-	1.5e-06	-

Figure 4.6: Standalone VCC fmincon solution and other interesting variables. Output generated using function $VCC.Model.print()$.

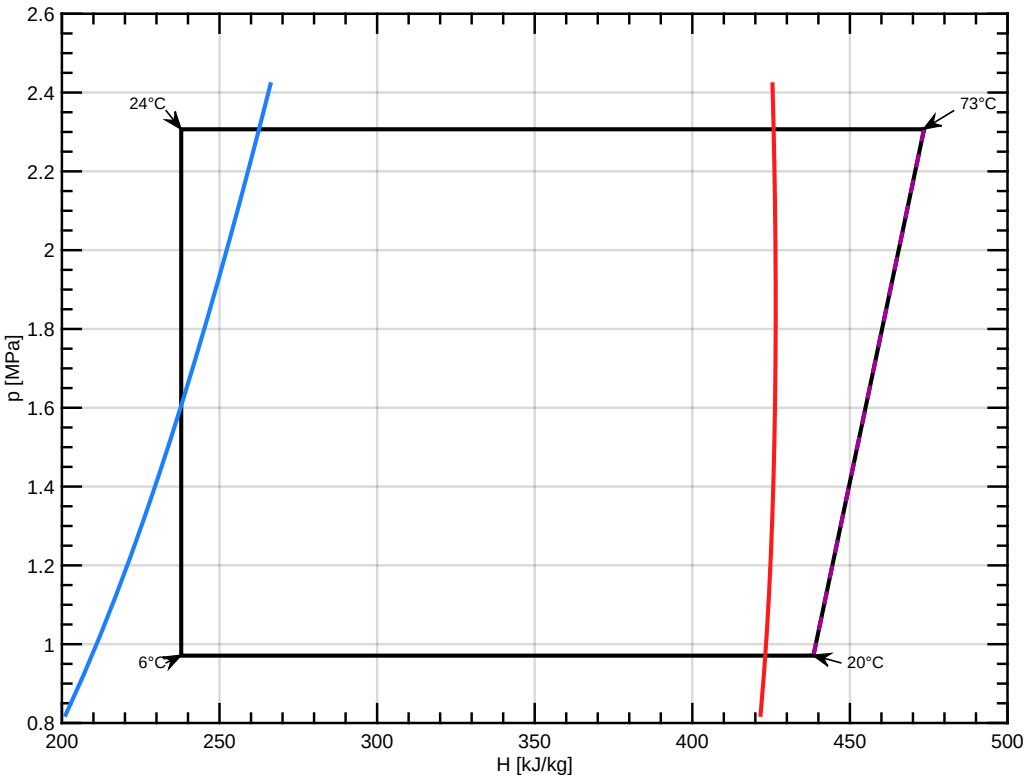


Figure 4.7: Standalone VCC fmincon solution P-H diagram. Blue line is saturated liquid line and red line is saturated vapor line. In each corner of the cycle is refrigerant temperature. This diagram was generated using function $VCC.Model.plot\_ph()$.

4.6 Control Optimizing VCC

Second problem addressed in this chapter is solving VCC and optimizing COP while relaxing control variables. In this problem amount and temperature of air delivered is set and PSOS is used to find optimal control with respect to COP and steady state. Same ambient conditions as in Section 4.3 are used.

Following input is provided:

- Desired output air temperature at condenser $T^{cond}_{a,out,d} = 25\,°\mathrm{C} = 298.15\,\mathrm{K}$ (1 degree more than evaporator input)
- Set $\dot{m}_{c,a} = 0.8\,\mathrm{kg/s}$
- Maximize COP

Output air temperature is not state variable so it will be implemented using additional equation. COP is easily calculated from component outputs according to equation 4.2. Following cost function is used:

$$\begin{aligned} f_1(\mathbf{x}) &= p_{out} - p_{23} = 0 \\ f_2(\mathbf{x}) &= H_{out} - H_1 = 0 \\ f_3(\mathbf{x}) &= \max(0, H^{cond}_{out} - H_{34,ub}) = 0 \\ f_4(\mathbf{x}) &= T^{cond}_{a,out} - T^{cond}_{a,out,d} = 0 \\ F_0(\mathbf{x}) &= \left\| \begin{pmatrix} 10^{-3} \cdot f_2(\mathbf{x}) \\ 10^{-5} \cdot f_3(\mathbf{x}) \\ 10 \cdot f_4(\mathbf{x}) \end{pmatrix} \right\|_1 \\ F_{\min}(\mathbf{x}) &= \frac{1}{COP(\mathbf{x})} \\ F_v &= \|10^{-5} \cdot f_1(\mathbf{x})\|_2 \end{aligned} \tag{4.11}$$

For valid and optimal solution $F_0 \to 0$ and $F_{\min} \to \min$. These two objectives are combined into single cost function which is minimized by PSOS.

$$F(\mathbf{x}) = \ln(1 + F_0(\mathbf{x})) + 5\ln(1 + F_{\min}(\mathbf{x})) \tag{4.12}$$

Several ways of combining F_0 and $F_{\min}$ were tried during this thesis and Equation 4.12 performed decently. Other interesting candidate is $F = F_0 \cdot F_{\min}$. Deeper and sound comparison of possible F candidates can be continuation of this thesis. It is important to note that cost function determine how particles move and converge in PSO. At some point trade off between F_0 and $F_{\min}$ will happen. COP ranges between 1 and 10 in practice. L_1 norm is used and weight for f_3 is set to 10^-5. PSO parameter was increased $\phi_t = 0.5 \to 1.5$.

Valve is solved separately after VCC solution is found. This requires additional PSOS run with 50 particles and cost function F_v (only optimized variable is x the rest of $\mathbf{x}$ is fixed). This converges rapidly (1 second). Removing valve from main PSOS run brings additional speedup. With valve 50 seconds (Listing 4.1). Without valve 15 iterations took 35 seconds (Listing 4.2.

Example of found solution is shown in Figures 4.8 and 4.9. Desired output temperature was reached as well as decent total error. COP value 4.3 is reasonable but there is no certainty that found solution is the best solution.

Overall behaviour of PSOS is similar as observed during solving Standalone VCC problem. Several solver executions lead to scattered solutions. Same pattern as in Figure 4.5 was observed. This is again proving extreme initial particle sampling sensitivity. However once this sensitivity will be mitigated there are no clear benefits of using step based optimizer on top of Standalone VCC solver instead of this combined approach using PSOS.

`fmincon` was used to compare found solution. Behaviour is similar as with Standalone VCC but in this case fmincon failed to meet desired output temperature with error of 0.33 °C while reaching COP 4.52.

Listing 4.2: Example usage of Control optimizing VCC Solver

```
m=VCC.Model();
%( cond_Ta_out, cond_dotma
% evap_Ta_in, evap_p_a, evap_R,
% cond_Ta_in, cond_p_a, cond_R)
m.evaluate2(...
    273.15+25, 0.8, ...
    273.15+20, 101325, 0.4,...
    273.15+24, 101325, 0.4...
);

%Initiating PSOS to find solution
%Particle Team Sizes = [ 84  72  94 ]
%PSOS done with 15 iterations, f(X) = [1.0459], X = [ ... ]
%PSO was terminated by MaxIter condition.
%Total duration 34.713 seconds.
%Initiating PSOS to find valve solution
%Particle Team Sizes = [ 17  18  15 ]
%PSOS done with 10 iterations, f(X) = [6.8743e-08], X =
   [0.075181]
%PSO was terminated by Objective Limit condition.
%Total duration 0.616 seconds.

%m.print();
%m.plot_ph();
%m.print_bounds();
```

Variable	Unit	Value	Error
Solution			
p41	MPa	0.7527	0.0000
p23	MPa	1.6593	-
H1	kJ/kg	443.280	0.0001
comp.n	Hz	5.000	-
valve.x	%	7.518	-
evap.dotma	kg/s	0.102	-
cond.dotma	kg/s	0.800	-
Quality			
comp.P	W	116.79	-
evap.P	W	8.47	-
cond.P	W	66.41	-
Total P	W	191.68	-
COP	-	4.30	-
Internal			
comp.dotm	g/s	3.45	-
Desirables			
cond.Ta_out	C	25.00	0.0000
Air		**In**	**Out**
cond.Ta	C	24.00	25.00
evap.Ta	C	20.00	13.24
Error			
Total Error	-	6.4e-05	-

Figure 4.8: Control optimizing VCC PSOS solution and other interesting variables. Output generated using function $VCC.Model.print()$.

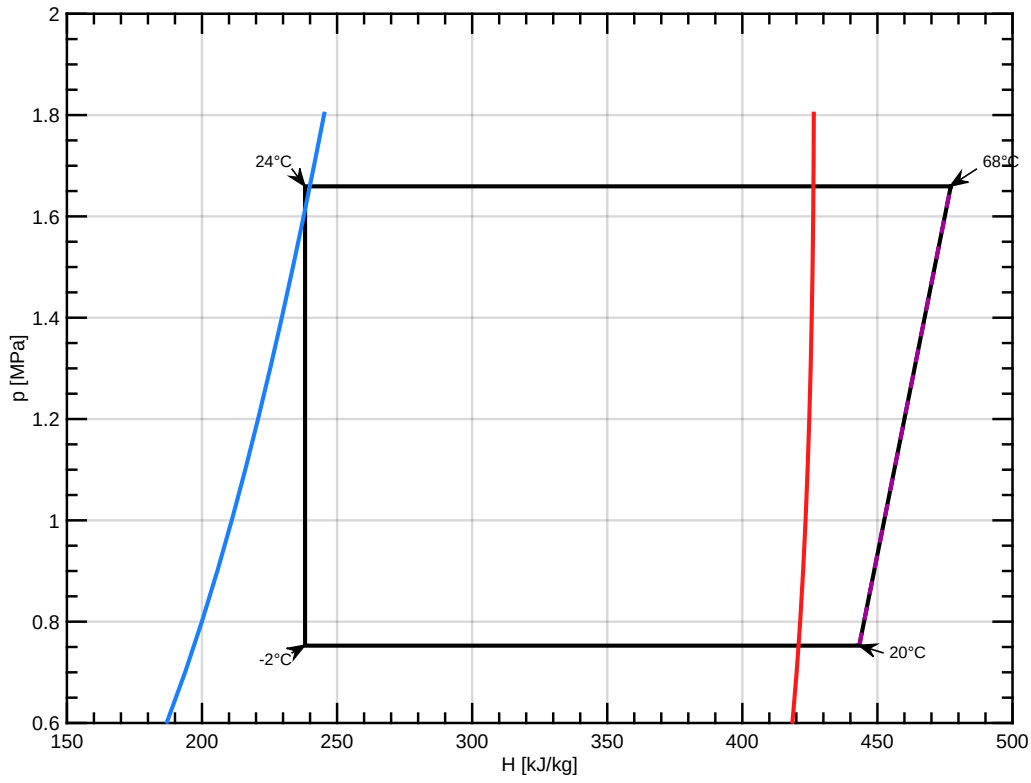


Figure 4.9: Control optimizing VCC PSOS solution P-H diagram. Blue line is saturated liquid line and red line is saturated vapor line. In each corner of the cycle is refrigerant temperature. This diagram was generated using function $VCC.Model.plot\_ph()$.

Chapter 5

Conclusion

This thesis was set out to examine possibility of using PSO algorithm to solve VCC equations. It was shown that implemented PSOS is suitable however computationally resource intensive. Further development is required for real world application. Number of underlying model evaluations is considerably large.

Important step was to prepare computational model of VCC components. Since PSO is supplying rather random values to all variables model must handle all possibilities. Computational performance of used model was optimized. Especially using tabular interpolation for material library and using model configuration which minimizes number of state variables.

Implemented PSOS suffered considerable initial particle sampling sensitivity. As PSO is stochastic in nature it sometimes converges and sometimes not. Comparable behaviour was observed with `fmincon` as initially provided solution affects found optimum. PSOS converged consistently for simple problems. This suggests that observed sensitivity is caused by sneaky behaviour of objective function (behaviour of the model).

Various approaches were tried to mitigate this issue however none appeared as a go to solution. Deeper analysis of PSO parameters and objective function should be done to asses this sensitivity. One approach that could easily help is setting narrow bounds for optimized variables based on guessed solution.

During research it was found that large scale PSO tuning should be done for standard implementation as results would be comparable to other peers. Because of using custom PSO implementation and computationally intensive cost function, tuning was performed subjectively according to intuition and small scale experiments.

Using PSO to solve VCC with relaxed control variables while optimizing COP arised as interesting and innovative approach. Two problems neglect this advantage. Choosing multi objective cost function is not trivial and trade-off between VCC loop closing and COP has to be made. Second problem is already mentioned initial particle sampling sensitivity.

VCC Model and PSOS are implemented in Matlab®and are available on attached DVD. Possible continuation of this thesis should focus on using standard PSO implementation and evaluating different objective functions and PSO parameters. Used VCC Model should be compared to real world to verify observed behaviour.

Bibliography

[1] M. Kolovratník et al. *Pokročilé řízení a optimalizace provozu tepelných čerpadel - závěrečná zpráva za rok 2013 (část projektu řešená na ČVUT v Praze)*. Tech. rep. Z-582/2013. Praha: ČVUT FS Ústav energetiky, 2013.

[2] I. H. Bell et al. "Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp". In: *Industrial & Engineering Chemistry Research* 53.6 (2014), pp. 2498–2508. DOI: 10.1021/ie4033999. eprint: http://pubs.acs.org/doi/pdf/10.1021/ie4033999. URL: http://pubs.acs.org/doi/abs/10.1021/ie4033999.

[3] *DuPont(TM) Suva(R) 407C (R-407C) and DuPont(TM) Suva(R) 410A (R-410A) Properties, Uses, Storage, and Handling*. Tech. rep. E. I. du Pont de Nemours and Company. URL: https://www.chemours.com/Refrigerants/en_US/assets/downloads/h65905_Suva407C_410A_push.pdf.

[4] H. Wilson. *Pressure Drop in Pipe Fittings and Valves*. Tech. rep. Katmar Software, 2012. URL: http://www.katmarsoftware.com/articles/pipe-fitting-pressure-drop.htm.

[5] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=488968.

[6] R. Eberhart and J. Kennedy. "A New Optimizer Using Particle Swarm Theory". In: *Sixth International Symposium on Micro Machine and Human Science, 1995*. IEEE, Oct. 1995, pp. 39–43. DOI: 10.1109/MHS.1995.494215. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=494215.

[7] D. Bratton and J. Kennedy. "Defining a Standard for Particle Swarm Optimization". In: *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*. 2007. URL: http://www.cil.pku.edu.cn/resources/pso_paper/src/2007SPSO.pdf.

[8] M. Clerc. *Standard Particle Swarm Optimisation*. Tech. rep. Particle Swarm Central, Sept. 2012. URL: http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf.

[9] M. E. H. Pedersen. *Good Parameters for Particle Swarm Optimization*. Tech. rep. Hvass Laboratories, 2010. URL: http://hvass-labs.org/people/magnus/publications/pedersen10good-pso.pdf.

[10] A. Carlisle and G. Dozier. "An off-the-shelf PSO". In: *Proceedings of the Workshop on Particle Swarm Optimization*. IUPUI, 2001, pp. 1–6. URL: http://antho.huntingdon.edu/publications/Off-The-Shelf_PSO.pdf.

Appendix A

List of Abbreviations

COP Coefficient of Performance
HVAC Heating, Ventilating and Air Conditioning
PSO Particle Swarm Optimization
PSOS Particle Swarm Optimization Solver
SPSO Standardized PSO
VCC Vapor Compression Cycle

Appendix B

Content of attached DVD

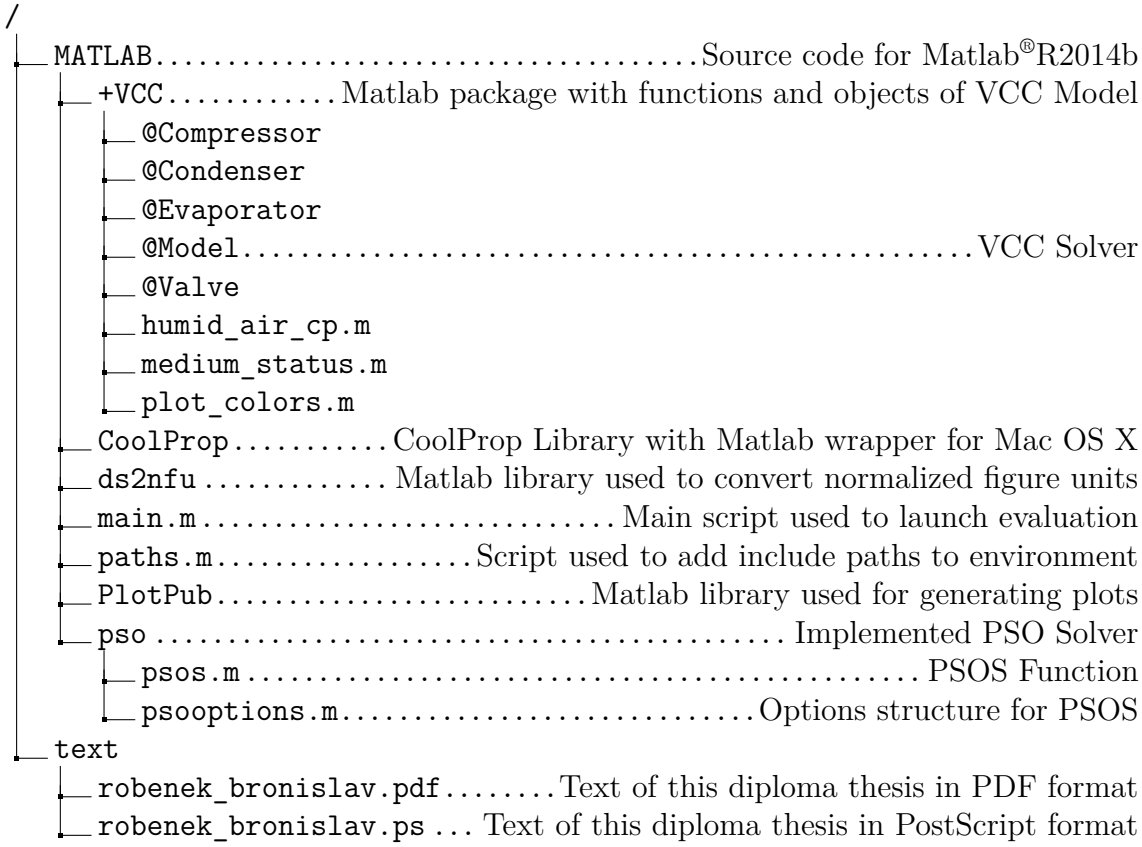


Figure B.1: Content of attached DVD