Diplomarbeit

Programming of the Thermodynamic Properties of Ammonia-Water Mixtures

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplomingenieurs der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Heimo Walter

E 302

Institut für Energietechnik und Thermodynamik

Forschungsbereich Thermodynamik und Wärmetechnik

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Paul Trunner

0525075

Leopold-Kunschakgasse 73/8/12

2232 Deutsch Wagram

Wien, im Jänner 2012

Danksagung

Mit dieser Arbeit geht für mich ein aufregender und interessanter Lebensabschnitt zu Ende. An dieser Stelle möchte ich mich bei jenen Menschen bedanken, die mir in den letzten sechs Jahren unterstützend zur Seite standen, und diese Zeit wesentlich mitgeprägt haben.

Mein besonderer Dank gilt meinem Betreuer, **Ao.Univ.Prof.Dipl.Ing.Dr.techn. Heimo Walter**, der mich während meiner Arbeit bestmöglich unterstützt hat und bei auftretenden Problemen stets mit Rat und Tat zur Seite stand. Gleichzeitig gab er mir bei der zeitlichen und inhaltlichen Gestaltung der Arbeit alle Freiheiten und Gestaltungsmöglichkeiten.

Die berufliche Nebentätigkeit bei der RHI-AG war während des Studienalltags stets eine willkommene und interessante Abwechslung. Ich bedanke mich bei meinen ehemaligen Kollegen für die gute Zusammenarbeit, und bei Herrn **DI Michael Klikovich**, der mir die Möglichkeit gegeben hat meine berufliche Tätigkeit studienbegleitend fortzusetzen.

Eine große Bereicherung waren meine Studienkollegen, deren Hilfe und Unterstützung einen nicht unwesentlichen Beitrag zum Erfolg meines Studiums darstellt. Die gemeinsamen Aktivitäten abseits des Studienalltags, gaben mir die nötige Freude und Motivation.

Obwohl diese Arbeit im Alleingang entstanden ist, gab es einige Personen die mir hilfreich zur Seite standen, und mir so die Arbeit maßgeblich erleichterten. Ich bedanke mich bei meinem Freunden, **Gerald Weber**, der mir in programmiertechnischen Fragen den Rücken frei gehalten hat, und bei **Martin Knoglinger**, mit dem ich thermodynamische Sachverhalte ausführlich diskutiert und analysiert habe. Für die Verbesserung der schriftlichen Fassung bedanke ich mich bei Herrn **Dr. Franz Wöhrer**, bei **Matthew Clarke**, sowie bei meiner Schwester **Lourdes Trunner**.

Nicht zuletzt gilt mein besonderer Dank meiner Familie, die stets hinter mir stand und mich bestmöglich unterstützt hat. Mein herzlicher Dank gilt meiner Freundin **Magdalena Lang**, für ihre aufopfernde Unterstützung und Geduld.

Abstract

With this diploma thesis, a computer program was developed, which allows the precise calculation of the thermodynamic properties of {ammonia-water} mixtures, by using the *Helmholtz fundamental equation of state.* The equation of state, used in this thesis, represents the currently available measurements, and is the most accurate representation of the thermodynamic properties of {ammonia-water} mixtures [17].

As a first step, the Helmholtz functions and its derivatives have been implemented. There out, the thermodynamic properties were calculated, by using appropriate combinations of the Helmholtz functions.

The equation of state is based on the three parameters *density* ρ, *temperature* T and *ammonia fraction* ξ. For calculating the properties with an alternative input combination, the standard calculation process had to be solved iteratively, by using a *Newton-Rhapson* algorithm. Furthermore, initial calculations had to be found, in order to set starting values for the iterations.

The next step was the calculation of the vapor-liquid region. Therefore, the fugacity coefficients had been calculated, by using the Helmholtz formulation. Core of the vapor-liquid calculation is an algorithm, which consists of two overlapping *while*-loops. With this algorithm, the vapor and liquid fraction of the mixture are solved iteratively, by recalculating the fugacity coefficients.

Finally, a *.dll* file was compiled, to enable the application as a library. This library can be used within other programs, such as *MATLAB*, *MS-Excel* or *Visual Basic.*

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem and task 2

2 Software **3**

2.1 Visual C++ 3

2.2 Application of the Program 4

2.2.1 MS Excel & Visual Basic 4

2.2.2 Matlab 7

2.2.3 Command prompt window 9

3 Application of the Ammonia-Water Mixture **10**

3.1 Generals to the properties of the mixture 10

3.2 Absorption Heat Pumps 11

3.3 Kalina Cycle 13

4 Thermodynamic properties of fluids **15**

4.1 Fundamentals 15

4.2 Equation of States for NH_3-H_2O mixtures 17

5 Helmholtz Equation of State **18**

5.1 Conception and Structure 18

5.2 Calculation of the dimensionless Helmholtz free energy 19

5.2.1 Ideal Part 20

5.2.2 Residual Part 20

5.2.3 Calculation of thermodynamic properties 23

5.3 VLE Computation 24

5.4 Backward Iteration 27
5.4.1 Newton-Rhapson Iteration Method 27
5.4.2 Newton Method for nonlinear systems of equations 28
5.4.3 Implemented Iteration Methods 28
5.5 Initial Calculations 33
5.5.1 Equations of 'Patek & Klomfar' 33
5.5.2 Approximation of the Density ρ 34
5.5.3 Soave-Redlich-Kwong 36
5.5.4 Mixture critical values 37
5.5.5 Line of triple points 37

6 Software implementation 39
6.1 Conception and Structure 39
6.2 Classes & Methods of the program 41
6.2.1 Definition of basic values 41
6.2.2 Helmholtz free energy 42
6.2.3 Thermodynamic Parameter 44
6.2.4 Iteration Methods 45
6.2.5 Initial Calculations 47
6.2.6 Vapor-liquid equilibrium 47
6.2.7 Determination of the phase 50
6.3 Sequence of the calculation 53

7 Validation of the Properties 55
7.1 Comparison with existing Data 55
7.2 Estimation of uncertainty 58

8 Conclusion 59

A Derivatives of the reduced Helmholtz energy 67
A.0.1 Ideal part 67
A.0.2 Residual part 68

B Coefficients of the Helmholtz function of water 73

C Coefficients of the equations for initial calculations 75

Nomenclature

English Symbols

Symbol	Unit	Meaning
A	–	Coefficient in eq.(5.15)
a	–	Coefficient for Helmholtz functions
a_s	–	Function for the Soave EoS eq.(5.70)
a_c	–	Coefficient for eq.(5.77)
A_ρ	–	Coefficients for eq.(5.64)
B	–	Coefficient in eq.(5.14)
b	–	Coefficient for eq.(5.66)
b_c	–	Coefficient for eq.(5.78)
C	–	Coefficient in eq.(5.16)
c_v	$J/(kg * K)$	Isochoric heat capacity
c_p	$J/(kg * K)$	Isobaric heat capacity
D	–	Coefficient in eq.(5.16)
d	–	Coefficient for Helmholtz functions
e	–	Coefficient for Helmholtz functions
f	J/mol	Molar Helmholtz free Energy
F_φ	–	Derivative needed for Fugacity
f_i	–	Fugacity of component i
h	J/kg	Enthalpy
J	–	Jacobi Matrix eq.(5.38)
k_T	–	Coefficient in eq.(5.10)
K	–	Vaporization equilibrium ratio eq.(5.31)
k_V	–	Coefficient in eq.(5.10)
M	kg/mol	Molar mass
n	–	Coefficient for Helmholtz functions
p	N/m^2	Pressure
R	$J/(mol * K)$	Molar gas constant
S	–	Sum in VLE-calculation $S{=}\sum y_i$
s	$J/(kg * K)$	Entropy
T_m^*	–	Dimensionless Temperature for eq.(5.64)

(to be continued)

Symbol	Unit	Meaning
T_n	K	Temperature function, eq.(5.8)
t	–	Coefficient for Helmholtz functions
T	K	Absolute Temperature (ITS-90)
u	J/kg	Internal energy
U	–	Auxiliary matrix, page 47
V	m^3/mol	Molar volume
v	m^3/kg	Specific volume
w	m/s	Speed of sound
x	–	Mole fraction of ammonia in the liquid phase
y	–	Mole fraction of ammonia in the vapor phase
Z	–	Compressibility factor
z	–	Mole fraction of ammonia for VLE calculation

Greek Symbols

Symbol	Unit	Meaning
α	–	Exponent in eq.(5.8)
α_s	–	Function for the Soave EoS eq.(5.70)
β_n	–	Exponent in eq.(5.9)
β	–	Vapor fraction eq.(5.33)
γ	–	Exponent in eq.(5.11)
δ	–	Reduced density
Δ	–	Expression for Helmholtz-H_2O - eq.(5.13)
$\Delta\Phi$	–	Departure function eq.(5.11)
$\Delta\rho$	kg/m^3	Excess density eq.(5.64) and eq.(5.67)
$\Delta\rho_{max}$	–	Expression of eq.(5.68)
Φ	–	Reduced Helmholtz free energy
φ	–	Fugacity coefficient
Ψ	–	Expression for Helmholtz-H_2O - eq.(5.16)
ρ	mol/m^3	Molar density

(to be continued)

Symbol	Unit	Meaning
τ	–	Reduced inverse temperature
θ	–	Coefficient for Helmholtz functions
θ_W	–	Expression for Helmholtz-H_2O - eq.(5.15)
ξ	–	Mass fraction of ammonia
ω	–	Acentric factor

.

Subscripts	
1	first component, water
2	second component, ammonia
12	combined term for mixture
i	running index
n	mixture residual reducing quantity
c	critical point
δ	derivative with respect to δ
τ	derivative with respect to τ
x	derivative with respect to x
U	upper limit of a region
L	lower limit of a region
B	bubble point
D	dew point
m	median value
V	Values of the Vaporizer, page 11
C	Values of the Condenser, page 11

Superscripts	
o	Ideal gas
r	Residual
(V)	Vapor
(L)	Liquid
M	Molar
(k)	Iteration step

Chapter 1

Introduction

1.1 Motivation

The ammonia and water mixture plays an important role as a working fluid in *absorption heat pumps*. Recently, this mixture has also been used for advanced power cycles, particularly the *Kalina Cycle*[18].

The engineering calculation and simulation of those cycles require the availability of accurate and efficient mathematical methods of the thermodynamic properties of the working fluid [1]. Several thermodynamic models have been released in the past but most of them are only applicable in a restricted range, or are not accurate enough.

This diploma thesis uses a fundamental equation of state for the *Helmholtz free energy* of the NH3-H2O Mixture to describe the entire thermophysical properties. The model is based on the formulation of the reduced Helmholtz free energy

$$\frac{f(\rho, T, x)}{RT} = \Phi = \Phi^\circ(\tau^\circ, \delta^\circ, x) + \Phi^r(\tau, \delta, x)$$

where Φ is split into an Ideal Part Φ° and a residual part Φ^r. All thermodynamic properties can be derived by this single mathematical expression. The formulation covers the thermodynamic space between the solid-liquid-vapor boundary and the critical locus [19]. In the single phases it is valid for pressures up to 40 MPa [17]. The conception of this equation of state can be found in Chapter 5.

1.2 Problem and task

Purpose of this thesis is to implement the Helmholtz fundamental equation of state to an autonomous program. The program is then converted to a program library which can be integrated to other programs, such as *Matlab* or *MS Excel*. The user of the routine is asked to choose a combination of three input parameters, e.g. (p,T,x), from which the remaining properties are derived.

At the first step, the reduced Helmholtz functions have to be implemented. This includes functions of the ideal and residual part, as well as functions of pure fluid components and their derivatives with respect to the reduced temperature and density. Based on these functions, the thermodynamic properties can be computed.

The model computes all properties from the three parameters *density* ρ, *temperature* T and *mole fraction* x. For any other input combination, the standard calculation process has to be solved iteratively. The implementation of these iteration methods was another subtask and is described in chapter 5.4.

Iteration methods require initial values for the iteration processes. Therefore, initial calculations have to be found for a first estimation of the current parameter, such as $T(p, x)$, $y(T, x)$ etc. Such an approximate calculation must be both accurate enough to provide convergence in the iteration process while still being valid over a wide range of the thermodynamic space.

The Helmholtz EoS[1] doesn't give information about the phase of the mixture. Therefore, an algorithm has to be found to calculate properties on the liquid-vapor (dew point and bubble point) equilibrium curves [19]. This is realized by using the fugacity coefficients of the components. With the basic condition for a phase equilibrium

$$y_i \varphi_i^{(V)} = x_i \varphi_i^{(L)}$$

the saturation values can be determined once again with an iteration algorithm (Chapter 5.3).

The programming of this calculation routine requires a clear and structured program layout. For this purpose the task is divided up into subroutines, and the program utilizes object-orientated programming.

[1]EoS=Equation of State

Chapter 2

Software

2.1 Visual C++

The choice of using *Visual C++* as coding language was rather secondary. First consideration was to figure out, which language allows the building of a *library file (.dll)* for MATLAB, which is basically possible with several programming languages, such as *Java*, *C* or *Fortran*. Since object-orientated programming is most suitable with *Java* and *C++*, it was obvious to use one of these.

Visual C++ is a development environment for the coding language C++, which uses the *.NET framework* as CLI[1]. In this work, classical C++ commands are used, as well as specific Visual C++ libraries.

As mentioned before, the implementation of the entire calculation process is modularized. That means that each subtask is implemented in an own *class file*, and each class contains several functions with specific routines. Even though a detailed explanation of the structure of the program can be found in Chapter 6, an outline of the implemented *class files* is given in the following.

- For calculating the **Helmholtz functions**, five classes are made. This includes the computation of the pure fluid components, their derivatives as well as the ideal and residual parts.
- In the next class, **thermodynamic parameters** are derived from the Helmholtz classes.

[1]CLI=common language interface

- Another class determines the **vapor-liquid equilibrium** by using functions of the previous mentioned classes. Calculation of fugacity coefficients are implemented in this class as well.
- **Iteration process** and corresponding **initial calculations** are also made in separate classes
- Beside that there are two more auxiliary classes where constant values, such as critical values and molar masses, are defined, as well as the dimensionless density and temperature.

2.2 Application of the Program

This chapter describes the integration of the NH3H2O-library to *MATLAB* and *MS Excel*, and how to use it. Further the application in the *Command prompt window* is illustrated. For calling a certain function of the NH3H2O-library, input parameter have to be transferred, and the calculated output parameter are returned. In table 2.1, the transfer parameter and the corresponding units are listed.

Label	T	ρ	z	p	u	h	s	c_v	c_p	w
Unit	K	kg/m^3	–	Pa	kJ/kg	kJ/kg	$kJ/(kgK)$	$J/(kgK)$	$J/(kgK)$	m/s

Table 2.1: transfer parameters

2.2.1 MS Excel & Visual Basic

At this point, the building of a .dll-file[2] for the usage within *Visual Basic (VBA)*, is described. The library consists of six functions, regarding the input combinations.

In contrast to standard library files, which are simply used within *C++*, libraries for external programs have to be modified. At first, a definition-file (*NH3H2O.def*) is compiled, for defining the functions of the library. This file has the following structure:

[2]Dynamic linked library

```
LIBRARY "NH3H2O"

EXPORTS
    Trx @1
    Tpx @2
    rpx @3
    Tpr @4
    Thx @5
    phx @6
```

Further, an additional class-file (*dllmain.cpp*) is attached to the project. The following source code defines an entry point for a function, once the library is called from outside.

```
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Finally, the functions are defined in the main-file (NH3H2O.cpp). As shown in the source code below, the function header includes the *_stdcall* convention. This command is used to call *Win32* functions. The transfer parameters are put in brackets, whereas return parameters are defined with the *-operator (e.g. *double *T*). This implies that the variable is identified as a pointer.

```
int _stdcall Trx(double T, double rho, ....)
{
    //...
    // source code
    //...
}
```

In order to call the library from *Visual Basic*, the compiled *.dll* file must be added to the directory of the current *VBA* or *Excel* file. At the beginning of the VBA-routine, the library has to be loaded, by using the following convention.

```
Private Declare Function [Function name] Lib "[dll name]" (Arguments...)
```

The calling of the library is shown by the example of the function 'Trx'. Other functions are following the same principle. Parameters, which are transferred to the function, are called with the '*ByVal*'[3] operator, while the return parameters, defined as *pointer*, are called with the '*ByRef*'[4] operator. Parameters of the library are listed in table 2.2. For calling a

```
Private Declare Function Trx Lib "NH3H2O.dll"
                         (ByVal T As Double, ...,ByRef p As Double,...)
```

function of the library, the parameters have to be read in the same order, as shown in the table.

			Function					
		Data type	Trx	Tpx	rpx	Tpr	Thx	phx
input	1	double	T	T	ρ	T	T	p
	2	double	ρ	p	p	p	h	h
	3	double	ξ	ξ	ξ	ρ	ξ	ξ
return parameters	4	string	*phase*	*phase*	*phase*	*phase*	*phase*	*phase*
	5	double	p	x	T	x	p	T
	6	double	x	$\rho^{(L)}$	x	$\rho^{(L)}$	x	x
	7	double	$\rho^{(L)}$	$h^{(L)}$	$\rho^{(L)}$	$h^{(L)}$	$\rho^{(L)}$	$\rho^{(L)}$
	8	double	$h^{(L)}$	$s^{(L)}$	$h^{(L)}$	$s^{(L)}$	$h^{(L)}$	$h^{(L)}$
	9	double	$s^{(L)}$	$u^{(L)}$	$s^{(L)}$	$u^{(L)}$	$s^{(L)}$	$s^{(L)}$
	10	double	$u^{(L)}$	$c_v^{(L)}$	$u^{(L)}$	$c_v^{(L)}$	$u^{(L)}$	$u^{(L)}$
	11	double	$c_v^{(L)}$	$c_p^{(L)}$	$c_v^{(L)}$	$c_p^{(L)}$	$c_v^{(L)}$	$c_v^{(L)}$
	12	double	$c_p^{(L)}$	$w^{(L)}$	$c_p^{(L)}$	$w^{(L)}$	$c_p^{(L)}$	$c_p^{(L)}$
	13	double	$w^{(L)}$	$Z^{(L)}$	$w^{(L)}$	$Z^{(L)}$	$w^{(L)}$	$w^{(L)}$
	14	double	$Z^{(L)}$	y	$Z^{(L)}$	y	$Z^{(L)}$	$Z^{(L)}$
	15	double	y	$\rho^{(V)}$	y	$\rho^{(V)}$	y	y
	16	double	$\rho^{(V)}$	$h^{(V)}$	$\rho^{(V)}$	$h^{(V)}$	$\rho^{(V)}$	$\rho^{(V)}$
	17	double	$h^{(V)}$	$s^{(V)}$	$h^{(V)}$	$s^{(V)}$	$h^{(V)}$	$h^{(V)}$
	18	double	$s^{(V)}$	$u^{(V)}$	$s^{(V)}$	$u^{(V)}$	$s^{(V)}$	$s^{(V)}$
	19	double	$u^{(V)}$	$c_v^{(V)}$	$u^{(V)}$	$c_v^{(V)}$	$u^{(V)}$	$u^{(V)}$
	20	double	$c_v^{(V)}$	$c_p^{(V)}$	$c_v^{(V)}$	$c_p^{(V)}$	$c_v^{(V)}$	$c_v^{(V)}$
	21	double	$c_p^{(V)}$	$w^{(V)}$	$c_p^{(V)}$	$w^{(V)}$	$c_p^{(V)}$	$c_p^{(V)}$
	22	double	$w^{(V)}$	$Z^{(V)}$	$w^{(V)}$	$Z^{(V)}$	$w^{(V)}$	$w^{(V)}$
	22	double	$Z^{(V)}$	–	$Z^{(V)}$	–	$Z^{(V)}$	$Z^{(V)}$

Table 2.2: Overview of the transfer parameters of the functions

[3]Call by value
[4]Call by reference

2.2.2 Matlab

In order to enable the application of the library in *MATLAB*, a so called MEX-file is compiled. MEX-files are a way to make an interface between C++ functions and *MATLAB*.

Core of the MEX-file is a function, called *mexFunction*. This function is a gateway for *MATLAB* to access a library, written in C++. The *mexFunction* has the following structure:

```
void mexFunction(int nlhs, mxArray* plhs[],
                 int nrhs, const mxArray* prhs[])
{ ... }
```

The parameters of the functions are explained in table 2.3

element	meaning
nlhs	Number of outputs (left hand side)
plhs	Array of pointers to expected outputs
nrhs	Number of inputs (right hand side)
prhs	Array of pointers to input data

Table 2.3: Parameters of *mexFunction*

For using the *mx** and *mex** routines, it is necessary to include the header-file *mex.h*, by using the *include* directive. These routines are used to define input and output data for *MATLAB*.

```
#include "mex.h"
```

As shown in the previous chapter, a definition-file (*NH3H2O.def*) is compiled. With this file, the exporting functions of the library are defined. In this case, the function *mexFunction* is exported.

```
LIBRARY      "NH3H2O"
EXPORTS
    mexFunction
```

Before compiling the library, some properties of the project have to be modified, in order to ensure that the library is fully compatible with *MATLAB*. The project properties can be opened, by a right-click on the project name and choosing *Properties*.

- In order to access the *mex.h* file, the directory has to be added to the project.
  $ConfigurationProperties \rightarrow C/C{+}{+} \rightarrow General \rightarrow AdditionalIncludeDirectories$
  Add directory: $\rightarrow C : /ProgramFiles(x86)/MATLAB/R2009a/extern/include$
- The string 'MATLAB_MEX_FILE' must be added to the list of *Preprocessors*.
  $ConfigurationProperties \rightarrow C/C{+}{+} \rightarrow Preprocessor \rightarrow Preprocessordefinitions$
- For using *MATLAB*-libraries, the proper directory has to be added.
  $ConfigurationProperties \rightarrow Linker \rightarrow General \rightarrow AdditionalLibraryDirectories$
  Add directory: $C : /ProgramFiles(x86)/MATLAB/R2009a/extern/lib/win32/microsoft$
- The data type of the compiled library must be changed from *.dll* to *.mexw32*
  $ConfigurationProperties \rightarrow Linker \rightarrow General \rightarrow OutputFile$
  Paste: $\$(OutDir)/\$(ProjectName).mexw32$

For calling the library from *MATLAB*, the compiled file has to be added to the current directory. Alternatively, the working directory can be updated to the one from the library. At first, *MATLAB* has to be told which compiler was used to build the library. This can be done by entering

```
>> mex -setup
```

Subsequently, a list of available compilers appears, from which the C++ compiler can be chosen.
The functions of the NH3H2O-library can now be called, like any other functions, written in *MATLAB*. For calling of a function, the function name and the input parameters are read as follows

```
>> NH3H2O_Trx(T, rho, xi)
```

function	input parameters
NH3H2O_Trx	T, ρ, ξ
NH3H2O_Tpx	T, p, ξ
NH3H2O_rpx	ρ, p, ξ
NH3H2O_Tpr	T, p, ρ
NH3H2O_Thx	T, h, ξ
NH3H2O_phx	p, h, ξ

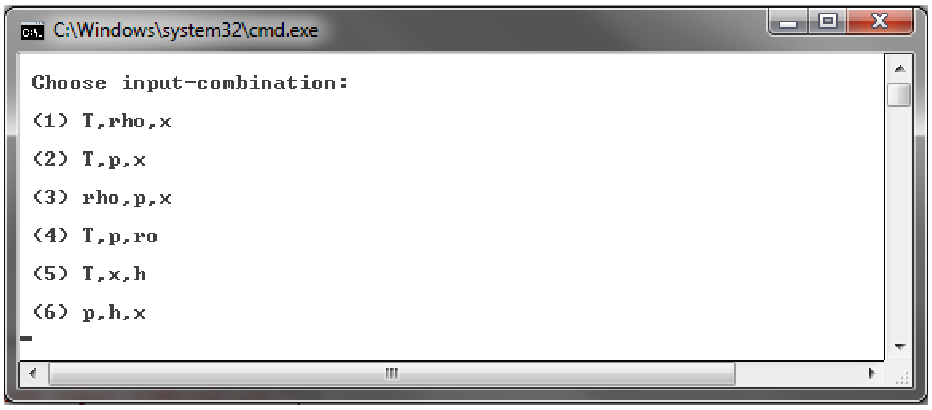
Table 2.4: *MATLAB* functions

A list of available functions is given in table 2.4. After calling a function, an array is returned with the calculated properties. Return parameters are listed in table 2.2.

2.2.3 Command prompt window

There are two ways to execute the application '*NH3H2O.exe*'. The simplest way is, to launch the program with the *Windows-Explorer*. Alternatively, the program can be executed, by calling the *command prompt window* manually. This can simply be done, by holding down the *Shift-key* and right-click on the desktop. Then the directory has to be called, where '*NH3H2O.exe*' program is located.

After calling the application '*NH3H2O.exe*', the *command prompt window* appears. At the beginning, the user has to chose a combination of input parameter, by entering a number from 1 to 6.

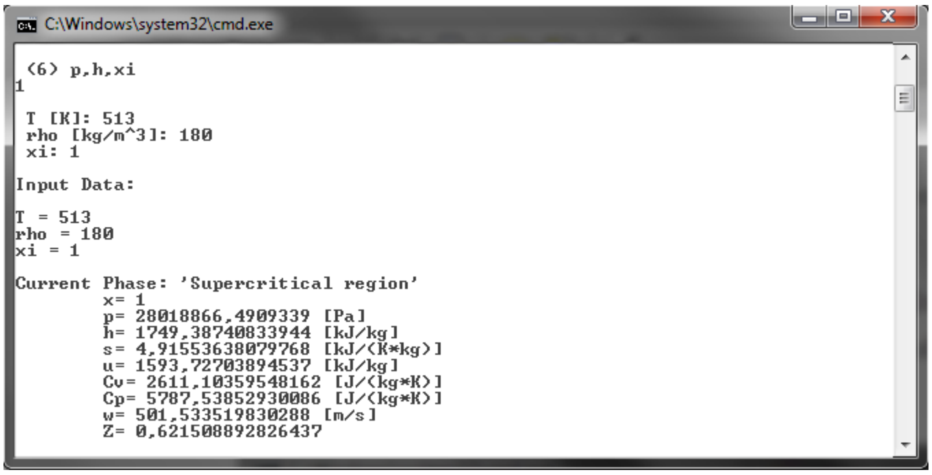


```
C:\Windows\system32\cmd.exe

Choose input-combination:
(1) T,rho,x
(2) T,p,x
(3) rho,p,x
(4) T,p,ro
(5) T,x,h
(6) p,h,x
_
```

Figure 2.1: Command prompt window, after calling *NH3H2O.exe*

Afterwards, the program asks for the corresponding values. The input of the decimal operator have to be made, by using a *comma*, instead of a *dot*. Finally, the calculated parameter are displayed, as shown in fig. 2.2.



```
C:\Windows\system32\cmd.exe

 (6) p,h,xi
1

 T [K]: 513
 rho [kg/m^3]: 180
 xi: 1

Input Data:

T = 513
rho = 180
xi = 1

Current Phase: 'Supercritical region'
        x= 1
        p= 28018866,4909339 [Pa]
        h= 1749,38740833944 [kJ/kg]
        s= 4,91553638079768 [kJ/(K*kg)]
        u= 1593,7270389453? [kJ/kg]
        Cv= 2611,10359548162 [J/(kg*K)]
        Cp= 5787,53852930086 [J/(kg*K)]
        w= 501,533519830288 [m/s]
        Z= 0,621508892826437
```

Figure 2.2: Calculated values, displayed in the command prompt window

Chapter 3

Application of the Ammonia-Water Mixture

3.1 Generals to the properties of the mixture

While the ORC[1] and ordinary heat pumps use pure refrigerants as working fluids, such as chlorofluorocarbons etc, *absorption heat pumps* and the *Kalina cycle* use a binary mixture of {ammonia-water}, instead.

The mixture is a **zeotropic binary mixture**, which means vaporization and condensation processes don't occur at a constant temperature (azeotropic behavior), but rather in a certain temperature range.

Due to non-isothermal phase transition, exergy losses can be reduced. Further, because of the non-isothermal vaporisation temperature, the medium temperature is higher than at ORC. For that reason the thermal efficiency can be raised.

Ammonia is the more volatile component of the mixture, water serves as dissolvent. Phase transitions of vaporization and condensation are replaced by **desorption** and **absorption**. The corresponding devices are indicated as absorber (instead of consensator) and desorber (instead of evaporator) [9].

At this point, reference constants of the pure components which are defined in the IAPWS formulation [19], are given. Thus, calculations described in chapter 5 are based on this definition.

[1]ORC=Organic Rankine cycle

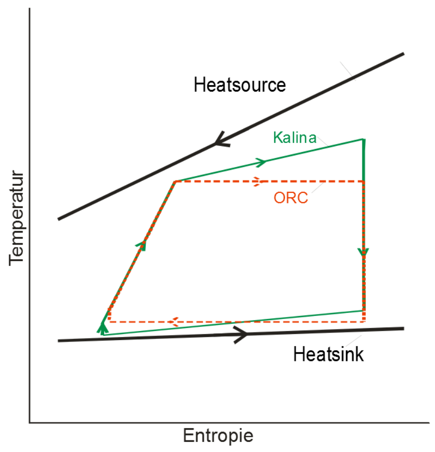


Figure 3.1: T-s Diagram – schematic comparison between ORC and Kalina cycle [9]

Water		Ammonia	
Tc_{H20}	647.096 K	Tc_{NH3}	405.4 K
ρ_{cH2O}	$322\frac{kg}{m^3}$	ρ_{cNH3}	$225\frac{kg}{m^3}$
M_{H20}	$18.015268\ \frac{g}{mol}$	M_{NH3}	$17.03026\ \frac{g}{mol}$

3.2 Absorption Heat Pumps

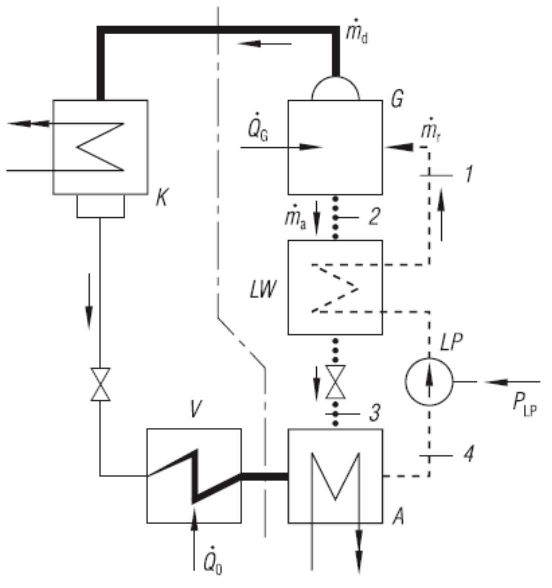
Absorption heat pumps are normally operated with binary mixtures. Most important working fluids are {ammonia-water} and {lithium-bromide-water}.

Such systems are widely used in industrial application. Whereas, for deep temperatures {NH3-H2O} is used as working fluid, while for air-conditioning facilities {H2O-LiBr} is preferred [4]. AHP[2] are **thermally driven** refrigeration machines, where the required exergy for operation is supplied by a heat flow with a higher temperature [3].

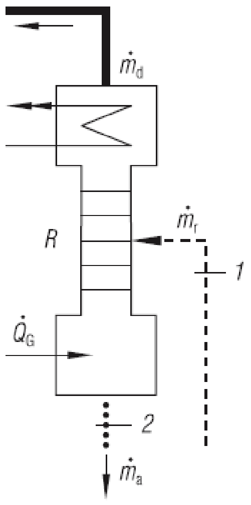
The schematic in fig.3.2a shows up two parts - the **refrigeration part** and the **operator cycle** (divided by a dashed line). Absorption chillers work at two pressure levels

- Condenser (K) and steam generator (G) (sometimes identified as boiler) run at **condenser pressure** p_C
- Evaporator (V) and absorber (A) are working at a lower **vaporize pressure** p_V

[2]Absorption Heat Pumps



(a) Schematic circuit of a single stage AHP



(b) Rectification unit

Figure 3.2: Absorption heat pump

In the Generator the inflowing strong solution is vaporized, by supply of the drive heat flow $\dot{Q}_G$. Thereby, the mass fraction ξ of the off-flowing solution is reduced. The resulting vapor supposed to have a mass fraction of ξ=1. With {ammonia-water} mixture as working fluid this is only obtained by using a rectification device instead of a generator, as shown in fig. 3.2b.

After cooling and condensation in the *condenser* (K) of the refrigeration cycle, the subcooled condensate is throttled to '*vaporize pressure p_V*'. In the *evaporator* (V), the working fluid absorbs the refrigerating capacity $\dot{Q}_0$ at temperature T_V.

In contrast to vapor-compression chiller, where the steam is compressed to the '*condenser pressure p_C*' by a compressor, absorption chillers uses the '*operator cycle*' to achieve this. This means, the weak solution, comming from the generator (G) is cooled in the heat exchanger (LW), before it is throttled to *absorber pressure p_V*.

In the absorber (A), the cool steam is mixed with the weak solution from state (3) and lead to a phase equilibrium. At this point, the dissipated heat flow is absorbed by the cooling water which flows through the absorber. The strong solution is compressed and heated in the heat exchanger (LW). From there it is ducted to the generator or rectification unit [3].

3.3 Kalina Cycle

Power cycles with the feature of {ammonia-water} as working fluid, are called *Kalina cycle*. The cycle has been developed in the 1980s by the Russian '*Alexander I. Kalina*'. Nowadays, this technology is used on the one hand for efficient usage of industrial waste heat, and on the other hand for geothermal power plants [5].

Basically, Kalina cycles corresponds roughly a Rankine cycle (see fig.3.4) with a high rate of heat recovery, and additional distillation and rectification units [9]. It uses the same devices such as turbine and pumps, as a conventional steam power plant, since the molecular weight of the {ammonia-water} mixture differs only slightly from water [13].

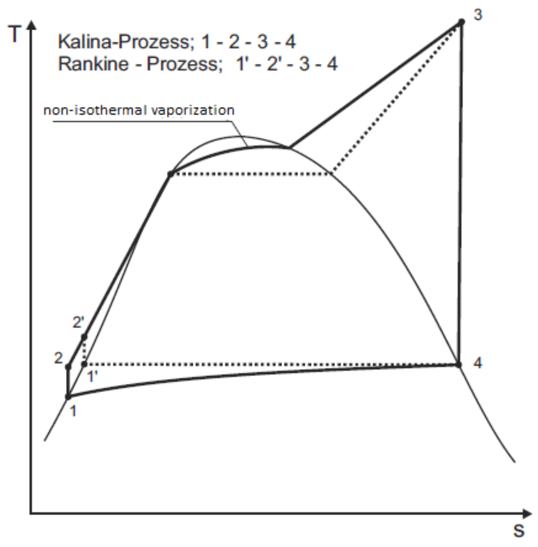

T
Kalina-Prozess; 1 - 2 - 3 - 4
Rankine - Prozess; 1' - 2' - 3 - 4
non-isothermal vaporization
3
2'
2
1'
1
4
s

Figure 3.3: T-s diagramm of a Kalina cycle [5]

Figure 3.4 shows the schematic circuit of a *Kalina Cycle*. Steam from the turbine, which is still superheated, is cooled, diluted with ammonia-weak solution and condensed (2)-(7). The saturated solution from the condenser is then compressed and preheated (7)-(9). Wet steam is ducted to the separator, where ammonia-weak liquid is separated from ammonia-strong steam (12),(13) u.(16). Liquid is cooled (14)-(15), throttled (6) and mixed with the cooled steam (4).

A part of the original condensed fluid (18) is added to the ammonia-strong steam (17), to obtain an ammonia concentration of 70% in the working fluid. Afterwards, this mixture is cooled in the preheater by releasing heat (20) and condensed, before it is ducted by the feed water pump to the evaporator [5].

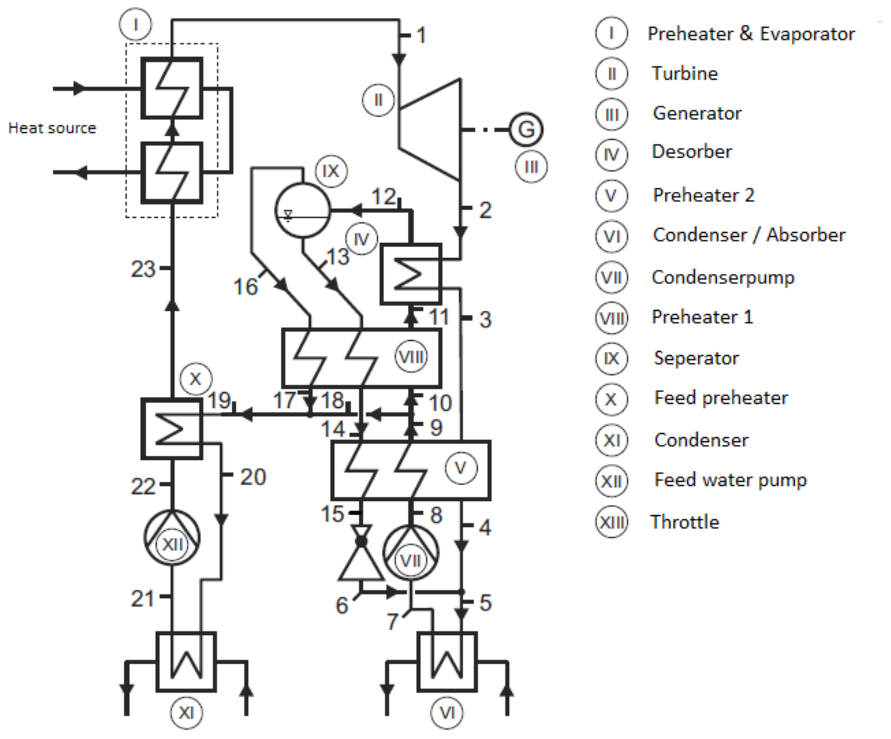


Figure 3.4: Schematic - Kalina cycle [5]

The simplified T-s diagramm in fig.3.3 illustrates, that vaporization and condensation occurs not at same temperatures. These effects have some consequences on exergy and efficiency considerations which are mentioned in chapter 3.1 .

Even though *Kalina cycles* seem to be more efficient as conventional cycles, the control of the process is more difficult, since a Kalina plant is more complex. Especially separator and heat exchangers are difficult to handle with binary mixtures [9].

Chapter 4

Thermodynamic properties of fluids

4.1 Fundamentals

For the determination of thermophysical properties, a wide range of techniques have been developed over time. In this chapter, a survey of the most important equations of states will be given. Theoretical principles which are relevant for the formulation of this thesis, are repeated as well.

The simplest and most common EoS [1] is the **ideal gas law**, which is only valid for small pressures [3].

$$pv = RT$$

Thereout, the *compressibility factor* is defined with

$$Z = \frac{pv}{RT}$$

Basically, this factor describes the discrepancy between a real and an ideal gas. Since the ability of this equation to describe real systems is rather poor, other methods are used to determine real behavior.

The simplest approach is the extension of the ideal gas law to the **Virial equation of state**. In order to get this formulation, the configuration integral is computed as a power series in the density about the zero density limit [12].

$$\frac{p}{\rho_n RT} = 1 + B\rho_n + C\rho_n^2 + D\rho_n^3 + ...$$

[1]EoS=Equation of state

Coefficients of this function, known as *Virial coefficients*, are functions of *temperature* and *composition.* The Virial EoS accurately describes the vapor phase. However, it is not applicable in the liquid phase [3].

Another approach is the use of **cubic equations of state**. Such equations are not very accurate, but therefore available in a wide range of conditions. The simplest and oldest form is the **van der Waals equation**, which assumes that in a fluid each molecule moves independently in its own potential field, which is provided by other molecules.

Over the years, the *van der Waals equation* has been adopted to more accurate formulations, such as the equation of **Redlich-Kwong**, **Soave** or **Peng-Robinson**. Some of the most important cubic equations of state are listed in the table below [12]:

Name	Year	Equation
van der Waals	1873	$p = \frac{RT}{V_m - b} - \frac{a}{V_m^2}$
Redlich-Kwong	1949	$p = \frac{RT}{V_m - b} - \frac{a}{V_m(V_m+b)\sqrt{T}}$
Soave	1972	$p = \frac{RT}{V_m - b} - \frac{a\alpha(T)}{V_m(V_m+b)}$
Peng-Robinson	1976	$p = \frac{RT}{V_m - b} - \frac{a\alpha(T)}{V_m(V_m+b)+b(V_m-b)}$

If experimental data continue to exist over a wide range of conditions, a **fundamental equation of state** can be developed [3]. Based on the experimental data, a state function of choice is set in terms of its natural independent variables. The most common fundamental equation is the **reduced Helmholtz energy**, as a function of *temperature* and *density* [12].

Since this thesis is based on the Helmholtz formulation for the {ammonia-water} mixture, a detailed explanation of this equation of state is given in chapter 5.

4.2 Equation of States for NH_3-H_2O mixtures

Due to its application in absorption heat pumps, {ammonia-water} mixtures have been investigated carefully over the last decades. For configuration and optimization of both absorption refrigeration- and Kalina cycles, an accurate description of the thermodynamic properties for a wide range of conditions is required [21]. In the following, a short overview over the most common models is given.

There are several old models, where experimental data are correlated with graphical methods. The most commonly used study is the diagram of *Merkel & Bosnjakovic*, published in 1929, which provides reliable data of temperature, pressure and enthalpy, as a function of the ammonia fraction. This method was a standard tool for calculating absorption chillers, and is particularly suitable for teaching. However, such methods cannot be used for computer implementation.

Properties for the {ammonia-water} mixture are often obtained by using *tabulated values*, which are based on experimental data. Such tables have been published by *Macriss et al.* [14].

In the early seventies, an equation of state has been published that consists of two separate equations of the *Gibbs free energy* $G = G(p, T, x)$, for both liquid and vapor phase [17]. Phase equilibrium properties can be calculated directly from bubble and dew points. This avoids an iterative solution and reduces numerical costs [21].

There are several other models reported about the calculation of {ammonia-water} mixtures, but most of them are either only valid in a restricted range of conditions, or not very accurate. There is no model available, that has been fitted to new experimental data in single or two-phase regions [17].

A new approach has been developed by *Tillner-Roth & Friend* [17] in 1997, incorporating a *fundamental equation of state for the Helmholtz free energy.* With this method, the representation of the thermodynamic properties over a wide range of fluid states is possible. It is based on highly accurate equations, developed by means of compiled and evaluated experimental data [10].

The Helmholtz formulation covers the thermodynamic space between the solid-liquid-vapor boundary and the critical locus. In the single phase regions it is also valid for pressures up to 40 MPa [17]. Details of the selection and evaluation of experimental data are discussed in *Tillner et al.* [18].

Chapter 5

Helmholtz Equation of State

5.1 Conception and Structure

As mentioned previously, the entire thermodynamic properties are derived from an input combination of three parameters $\{T,\rho,x\}$. In this chapter, the entire calculation process is described. This includes the Helmholtz approach, iteration methods for alternative input combinations, initial calculations to set start values for the iteration, as well as the computation of the vapor-liquid equilibrium.

Fig.5.1 shows the principal procedure of the calculation, whereas detailed drains and coherences are discussed in the following chapters. At first an iteration class is started, regarding the input combination (see chapter 5.4). If the input values are the standard functions of the Helmholtz formulation, thermodynamic parameters can be computed directly. Once the remaining properties are obtained, a VLE[1]-calculation (see chapter 5.3) is initialized, to detect the current phase. According to the detected phase, the properties of the single or the two-phase region can be calculated.

The Helmholtz calculations are not only used to determine the thermodynamic properties, but also for the iteration class, as well as for VLE calculations.

All variables are calculated with standard *SI units*, which are given in the nomenclature. However, the *ammonia fraction x* is used in terms of molar fraction, instead of mass fraction. Since the density is given in $[\rho] = \frac{kg}{m^3}$, it has to be converted to $[\rho^M] = \frac{mol}{m^3}$, by using the

[1]VLE=vapor-liquid equilibrium

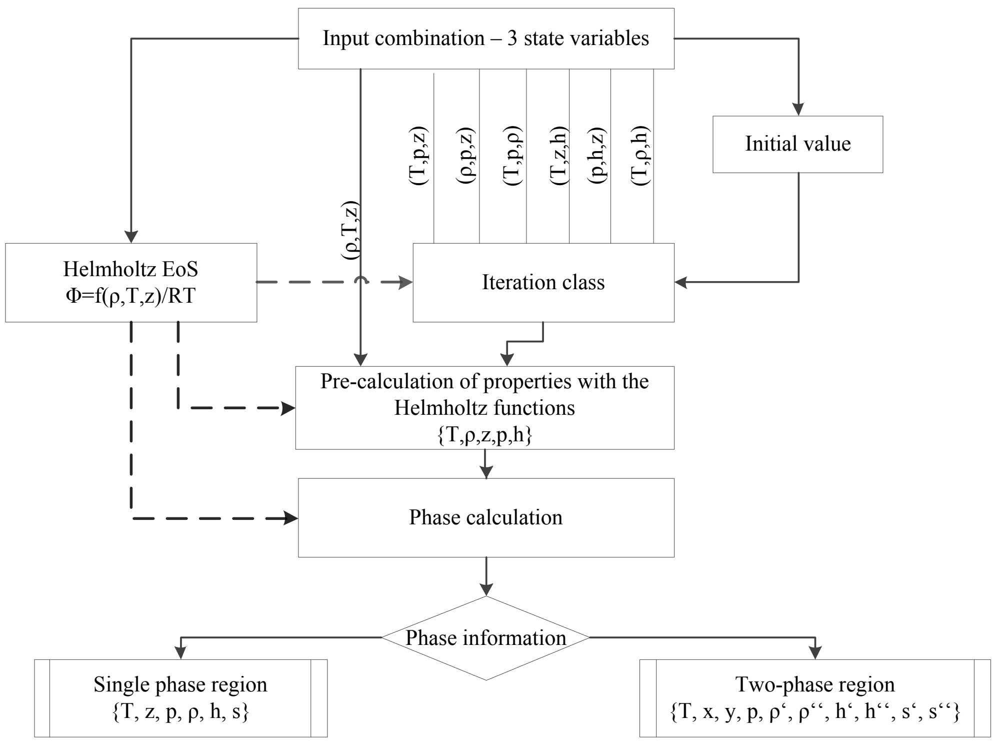


Figure 5.1: Calculation process

molar mass of the mixture.

$$M = (1-x)M_1 + xM_2 \tag{5.1}$$

5.2 Calculation of the dimensionless Helmholtz free energy

The fundamental equation of state is given in terms of the dimensionless Helmholtz free energy as

$$\frac{f(\rho,T,x)}{RT} = \Phi = \Phi^\circ(\tau^\circ,\delta^\circ,x) + \Phi^r(\tau,\delta,x) \tag{5.2}$$

The equation consists of an ideal and a residual part, which depends on the molar fraction x of ammonia and the dimensionless variables τ and δ. The universal gas constant is given with $R = 8,314471\frac{J}{molK}$. Once the Helmholtz functions and their derivatives are calculated, thermodynamic parameters can be computed. The relations are given in section 5.2.3 [19].

5.2.1 Ideal Part

The ideal gas contribution represents the ideal-gas part of the {ammonia-water} mixture, and consists of the ideal-gas expressions for the pure components [17]. Consequently, the function is written as

$$\begin{aligned}\Phi^\circ(\tau^\circ,\delta^\circ,x) = \ln(\delta^\circ) + (1-x)\{a_1^\circ + a_2^\circ\tau^\circ + a_3^\circ\ln(\tau^\circ) + \ln(1-x) + \sum_{i=4}^{8} a_i^\circ \ln(1-e^{-\theta\tau^\circ})\} \\ + x\{a_9^\circ + a_{10}^\circ\tau^\circ + a_{11}^\circ\ln(\tau^\circ) + \ln(x) + \sum_{i=12}^{14} a_i^\circ(\tau^\circ)^{t_i}\}\end{aligned} \tag{5.3}$$

The dimensionless variables τ° and δ° are defined in eq. (5.4), whereas $T^\circ = 500K$ and $\rho^\circ = 15000 mol/m^3$ have been chosen arbitrarily. Coefficients are given in table 5.1.

$$\tau^\circ = \frac{T^\circ}{T} \qquad \delta^\circ = \frac{\rho}{\rho^\circ} \tag{5.4}$$

i	a_i	θ_i	i	a_i	θ_i
1	-7.720 435	–	9	-16.444 285	–
2	8.649 358	–	10	4.036 946	–
3	3.006 320	–	11	-1.0	–
4	0.012 436	1.666	12	10.699 55	1/3
5	0.973 15	4.578	13	-1.775 436	-2/3
6	1.279 500	10.018	14	0.823 740 34	-7/4
7	0.973 15	11.964			
8	0.248 730	35.600			

Table 5.1: Coefficients of the ideal part [19]

The calculation of the thermodynamic properties demands the derivatives of the equation with respect to τ°, which are given in Appendix A.

$$\Phi^\circ_{\tau^\circ} = \left(\frac{\partial\Phi^\circ}{\partial\tau^\circ}\right)_{\delta^\circ,x} \qquad \Phi^\circ_{\tau^\circ\tau^\circ} = \left(\frac{\partial^2\Phi^\circ}{\partial(\tau^\circ)^2}\right)_{\delta^\circ,x} \tag{5.5}$$

5.2.2 Residual Part

The residual part consists, like the ideal part, of contributions of both water Φ^r_1 and ammonia Φ^r_2. Further a departure function $\Delta\Phi^r$ is needed to describe the properties of the mixture accurately [17].

$$\Phi^r = (1-x)\Phi^r_1 + x\Phi^r_2 + \Delta\Phi \tag{5.6}$$

The Helmholtz equations are functions of the same reduced variables τ and δ, given in eq.(5.7). Whereas the reducing functions $T_n(x)$ and $\rho_n(x)$ are functions of the ammonia fraction x.

$$\tau = \frac{T_n(x)}{T} \qquad\qquad \delta = \frac{\rho}{\rho_n(x)} \tag{5.7}$$

The reducing functions are given by [19]

$$T_n(x) = (1-x)^2 T_{c1} + x^2 T_{c2} + 2x(1-x^{\alpha})T_{c12} \tag{5.8}$$

$$V_n(x) = \frac{1}{\rho_n} = (1-x)^2\Big(\frac{1}{\rho_{c1}}\Big) + x^2\Big(\frac{1}{\rho_{c2}}\Big) + 2x(1-x^{\beta_n})\Big(\frac{1}{\rho_{c12}}\Big) \tag{5.9}$$

whereas the critical values of the mixture are defined with

$$T_{c12} = \frac{k_T}{2}\Big(T_{c1} + T_{c2}\Big) \qquad\qquad \frac{1}{\rho_{c12}} = \frac{k_V}{2}\Big(\frac{1}{\rho_{c1}} + \frac{1}{\rho_{c2}}\Big) \tag{5.10}$$

For the **departure function** $\Delta\Phi^r$, the following expression is used

$$\frac{\Delta\Phi^r_{(\tau^r,\delta^r,x)}}{x(1-x^{\gamma})} = a_1\tau^{t_1}\delta^{d_1} + \sum_{i=2}^{6} a_i\tau^{t_i}\delta^{d_i}e^{-\delta^{e_i}} + x\sum_{i=7}^{13} a_i\tau^{t_i}\delta^{d_i}e^{-\delta^{e_i}} + a_{14}x^2\tau^{t_{14}}\delta^{d_{14}}e^{-\delta^{e_{14}}} \tag{5.11}$$

with the coefficients, given in table 5.2

i	a_i	t_i	d_i	e_i	i	a_i	t_i	d_i	e_i
1	-1.855 822E-02	3/2	4	–	8	-1.368 072E-08	4	15	1
2	5.258 010E-02	1/2	5	1	9	1.226 146E-02	7/2	4	1
3	3.552 874E-10	13/2	15	1	10	-7.181 443E-02	0	5	1
4	5.451 379E-06	7/4	1	1	11	9.970 849E-02	-1	6	2
5	-5.998 546E-13	15	12	1	12	1.058 408 6E-03	8	10	2
6	-3.687 808E-06	6	15	2	13	-0.196 368 7	15/2	6	2
7	0.258 619 2	-1	4	1	14	-0.777 789 7	4	2	2
α=1.125455			β=0.8978069				k_T=0.9648407		
k_V=1.2395117					γ=0.5248379				

Table 5.2: Coefficients of the departure function [19]

The Helmholtz formulation of pure **ammonia** is taken by *Baehr and Tillner-Roth* [7] or *R. Tillner-Roth and F. Harms-Watzenberg* [16]. The ideal part is already included in eq.(5.3), the residual part is given by

$$\Phi_2^r = \sum_{i=1}^{5} a_i\tau^{t_i}\delta^{d_1} + \sum_{i=6}^{21} a_i\tau^{t_1}\delta^{d_i}e^{-\delta^{e_i}} \tag{5.12}$$

i	a_i	t_i	d_i	e_i	i	a_i	t_i	d_i	e_i
1	-1.858 814 0E+00	1.5	1	–	12	2.397 852 0E-02	3	1	2
2	4.554 431 0E-02	-1/2	2	–	13	-4.085 375 0E-02	6	1	2
3	7.238 548 0E-01	1/2	1	–	14	2.379 275 0E-01	8	2	2
4	1.229 470 0E-02	1	4	–	15	-3.548 972 0E-02	8	3	2
5	2.141 882 0E-11	3	15	–	16	-1.823 729 0E-01	10	2	2
6	-1.430 020 0E-02	0	3	1	17	2.281 556 0E-02	10	4	2
7	3.441 324 0E-01	3	3	1	18	-6.663 444 0E-03	5	3	3
8	-2.873 571 0E-01	4	1	1	19	-8.847 486 0E-03	7.5	1	3
9	2.352 589 0E-05	4	8	1	20	2.272 635 0E-03	15	2	3
10	-3.497 111 0E-02	5	2	1	21	-5.588 655 0E-04	30	4	3
11	1.831 117 0E-03	5	8	2					

Table 5.3: Coefficients of the ammonia part [19]

Coefficients of eq.(5.12) are given in table 5.3.

The residual contribution of **water** is taken from *Pruß and Wagner* [20], and is given below.

$$\Phi_1^r = \sum_{i=1}^{7} n_i \tau^{t_i} \delta^{d_i} + \sum_{i=8}^{51} n_i \tau^{t_i} \delta^{d_i} e^{-\delta^{c_i}} + \sum_{i=52}^{54} n_i \tau^{t_i} \delta^{d_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2} + \sum_{i=55}^{56} n_i \Delta^{b_i} \delta \Psi \tag{5.13}$$

where the functions Δ and Ψ in the last term are defined with

$$\Delta = \Theta_W^2 + B_i \left[(\delta-1)^2\right]^{a_i} \tag{5.14}$$

$$\Theta_W = (1-\tau) + A_i \left[(\delta-1)^2\right]^{\frac{1}{2\beta_i}} \tag{5.15}$$

$$\Psi = e^{-C_i(\delta-1)^2 - D_i(\tau-1)^2} \tag{5.16}$$

A list of coefficients and exponents of the functions is annexed to Appendix B.

For the calculation of the thermodynamic properties, derivatives of the residual parts with respect to τ, δ and x had to be calculated. These derivatives are given in Appendix A.

$$\Phi_\delta^r = \left(\frac{\partial \Delta\Phi^r}{\partial \delta}\right)_{\tau,x} \qquad \Phi_{\delta\delta}^r = \left(\frac{\partial^2 \Delta\Phi^r}{\partial \delta^2}\right)_{\tau,x} \qquad \Phi_{\delta\tau}^r = \left(\frac{\partial^2 \Delta\Phi^r}{\partial \delta \partial \tau}\right)_{x}$$

$$\Phi_\tau^r = \left(\frac{\partial \Phi^r}{\partial \tau}\right)_{\delta,x} \qquad \Phi_{\tau\tau}^r = \left(\frac{\partial^2 \Phi^r}{\partial \tau^2}\right)_{\delta,x} \qquad \Phi_x^r = \left(\frac{\partial \Phi^r}{\partial x}\right)_{\delta,\tau}$$

5.2.3 Calculation of thermodynamic properties

All thermodynamic properties can be derived by using the appropriate combination of the reduced Helmholtz free energy and their derivatives [19].

- Pressure

$$p_{(\tau,\delta,x)} = \frac{\rho}{MRT}\left(1 + \delta\Phi^r_\delta\right) \tag{5.17}$$

- Compressibility Factor

$$Z_{(\tau,\delta,x)} = \frac{pM}{\rho RT} \tag{5.18}$$

- Internal Energy

$$U_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = \frac{R}{MT}\left(\tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau\right) \tag{5.19}$$

- Enthalpy

$$H_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = \frac{R}{MT}\left(1 + \delta\Phi^r_\delta + \tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau\right) \tag{5.20}$$

- Entropy

$$S_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = \frac{R}{M}\left(\tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau - \Phi^\circ - \Phi^r\right) \tag{5.21}$$

- Isochoric Heat Capacity

$$cv_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = \frac{R}{M}\left(-\tau^{\circ 2}\Phi^\circ_{\tau^\circ\tau^\circ} - \tau^2\Phi^r_{\tau\tau}\right) \tag{5.22}$$

- Isobaric Heat Capacity

$$cp_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = cv + \frac{R}{M}\frac{\left[1 + \delta\Phi^r_\delta - \delta\tau\Phi^r_{\delta\tau}\right]^2}{\left[1 + 2\delta\Phi^r_\delta + \delta^2\Phi^r_{\delta\delta}\right]^2} \tag{5.23}$$

- Speed of Sound

$$w^2_{(\tau,\delta,\tau^\circ,\delta^\circ,x)} = \frac{RT}{M}\left(1 + 2\delta\Phi^r_\delta + \delta^2\Phi^r_{\delta\delta}\right) + \frac{\left[1 + \delta\Phi^r_\delta - \delta\tau\Phi^r_{\delta\tau}\right]^2}{CvM/R} \tag{5.24}$$

- Fugacity of Components

$$\ln\left[Z\varphi_1(\tau,\delta,x)\right] = \Phi^r + \delta\Phi^r_\delta - xF_\varphi \tag{5.25}$$

$$\ln\left[Z\varphi_2(\tau,\delta,x)\right] = \Phi^r + \delta\Phi^r_\delta + (1-x)F_\varphi \tag{5.26}$$

$$F_\varphi = \Phi^r_x - \frac{\delta}{\rho_n}\frac{\partial\rho_n}{\partial x}\Phi^r_\delta + \frac{\tau}{T_n}\frac{\partial T_n}{\partial x}\Phi^r_\tau \tag{5.27}$$

The molar mass is composed of pure fluid values

$$M = (1-x)M_1 + xM_2 \tag{5.28}$$

5.3 VLE Computation

So far, an algorithm was introduced which determines several thermodynamic properties accurately. However, there is no information available about the phase behavior. In this chapter, the determination of vapor-liquid equilibrium conditions is discussed [12].

The vapor-liquid equilibrium is solved by introducing an algorithm which uses the fugacity coefficient model. The basic condition for the equilibrium between liquid and vapor phase is

$$f_i^{(V)} = f_i^{(L)} \tag{5.29}$$

By introducing fugacity coefficients, the equation becomes

$$y_i \varphi_i^{(V)} = x_i \varphi_i^{(L)} \tag{5.30}$$

This equation is more suitable, since fugacity coefficients are much easier to determine than fugacity f_i. The calculation of the fugacity coefficients is given in table 6.3. At this point, the *vaporization equilibrium ratio* K is introduced, given by

$$K_i = y_i / x_i \tag{5.31}$$

If two phases are in an equilibrium state, K is given by [12]

$$K_i = \varphi_i^{(L)} / \varphi_i^{(V)} \tag{5.32}$$

In the following, two basic algorithms are described, in order to determine bubble- and dew-point of the mixture. The algorithm is based on eq.(5.30) to eq.(5.32), and is given by ref. [12].

1. There are two types of bubble-point algorithm implemented. The first one uses the input values $\{T, x\}$ and the pressure p is iterated, while the input values of the second one are $\{p, x\}$ by varying temperature T.
2. The unknown parameter (T or p) is pre-calculated with the *Helmholtz EoS*.
3. An initial value for the vapor fraction y of ammonia is set.
4. In the next step the molar densities and the fugacity coefficients of each component are calculated in both phases.
5. A new estimation of the vapor fraction y is made eq.(5.31) by using the *vaporisation equilibrium ratio K*.

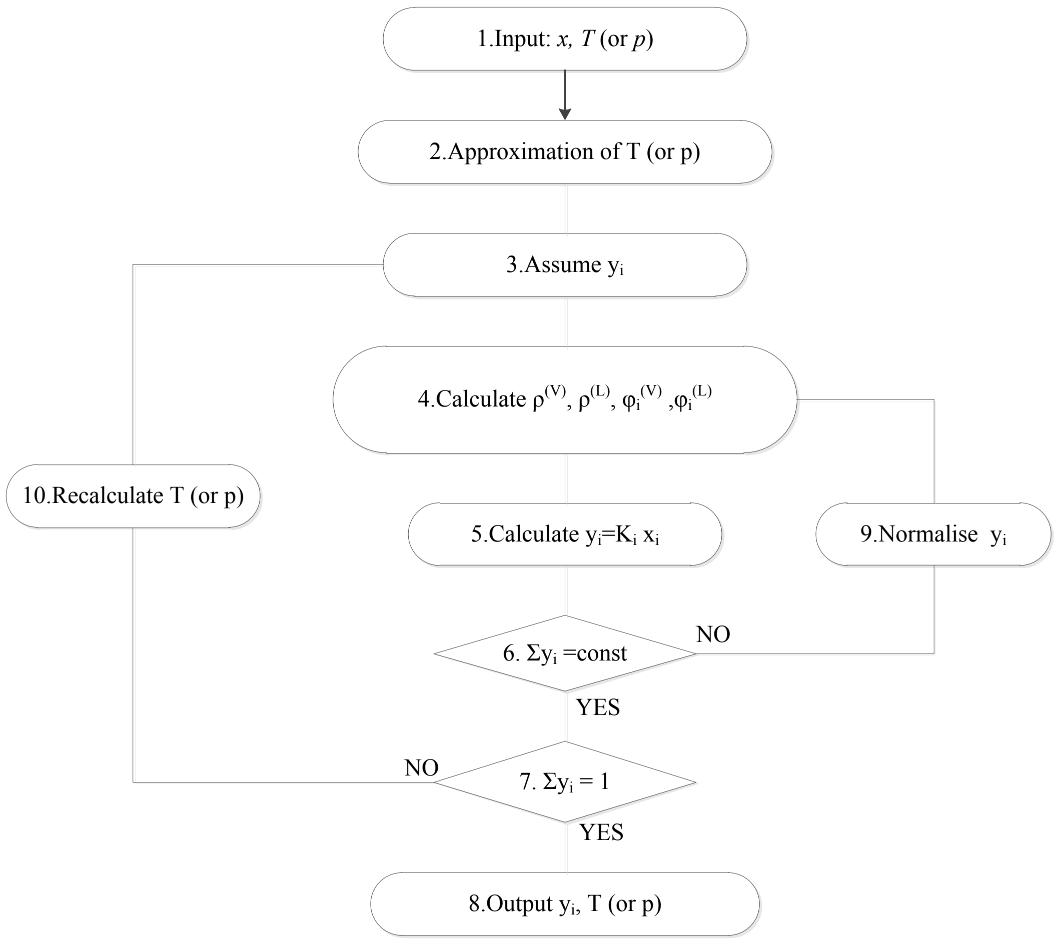


Figure 5.2: Bubble-point & dew-point algorithm [12]

6. If the sum $S = \Sigma y_i$ is equal to the one from the previous iteration step, the calculation can proceed to step 7. Otherwise a new assumption of y has to be made (see step 9.)
7. The next step is intended to verify, if the condition $S = 1$ is fulfilled. If not, a new estimation of the unknown parameter (T or p) has to be made (see step 10.), otherwise a solution has been found.
8. Output of the required parameter.
9. With the calculated sum s, a new vapor fraction y is calculated $y^{(k+1)} = y^{(k)}/S$. Thereafter, a new iteration step is initialized.
10. The unknown parameter has to be modified by means of a bisection algorithm. If $S > 1$, the assumed temperature T is too high, respectively the assumed pressure is too low. If $S > 1$, the assumed temperature T is too low, while the assumed pressure is too high. In case of the dew-point calculation the reverse applies [12].

The algorithm allows the determination of the saturation properties of the mixture. This is important for the detection of the existing phase (see chapter 6.2.7). However, for calculating the bubble and dew point at a specified temperature or pressure, the algorithm has to be modified [12].

In order to find a proper algorithm, the vapor fraction β is introduced. With the overall mole fraction z of ammonia, the equilibrium condition may be written [12]

$$\sum_{i=1}^{2} \frac{z_i}{1+\beta(K_i-1)} - 1 = 0 \tag{5.33}$$

by applying the Newton-Rhapson algorithm (5.4.1), β can be approximated with

$$\beta^{(k+1)} = \beta^{(k)} + \left[\sum_{i=1}^{2}\left(\frac{z_i}{1+\beta(K_i-1)}\right)-1\right]\left[\sum_{i=1}^{2}\left(\frac{(K_i-1)z_i}{[1+\beta(K_i-1)]^2}\right)\right]^{-1} \tag{5.34}$$

In order to provide successive convergence, a starting value with $\beta = 1$ has been chosen. Hence, the phase composition can be derived by

$$x_i = \frac{z_i}{1+\beta(K_i-1)} \qquad\qquad y_i = K_i x_i \tag{5.35}$$

Consequently, an algorithm for calculating the vapor and liquid composition is given in figure 5.3.

1. The function loads the temperature, pressure and the overall composition of the mixture.
2. Initial values for the vapor and liquid fraction are assumed.
3. Like in the previous algorithm, the *fugacity coefficients* φ_i and the *vaporization equilibrium ratio* K are calculated. This involves the calculation of the *density* of both phases.
4. A new value for β is calculated.
5. Hence, the composition of each phase is determined.
6. The fractions x_i and y_i are normalized with $x_i^{(k+1)} = \frac{x_i^{(k)}}{s}$.
7. In the next step, the function tests if the vapor composition differs from the previous iteration. If it does, a new approximation has to be made, otherwise the calculation proceeds with the next step.
8. Output of the parameter [12].

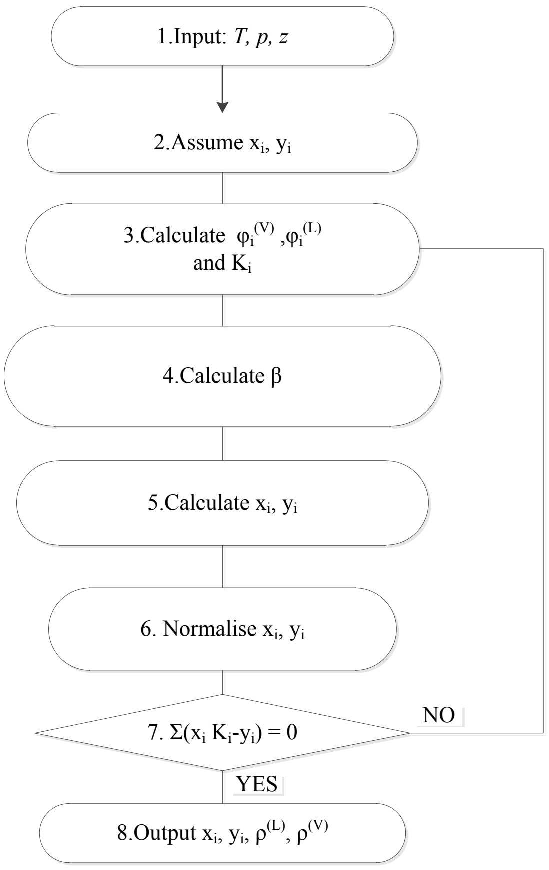

1.Input: *T, p, z*
2.Assume x_i, y_i
3.Calculate $\varphi_i^{(V)}$, $\varphi_i^{(L)}$ and K_i
4.Calculate β
5.Calculate x_i, y_i
6. Normalise x_i, y_i
7. $\Sigma(x_i K_i - y_i) = 0$
NO
YES
8.Output x_i, y_i, $\rho^{(L)}$, $\rho^{(V)}$

Figure 5.3: Algorithm for determining vapor-liquid equilibrium properties [12]

5.4 Backward Iteration

The Helmholtz formulation for calculating properties of {ammonia-water} mixtures requires an input combination of the three parameters ρ, T and x. For any other given parameter, the calculation process has to be solved iteratively. In order to do this, standard iteration methods are applied.

5.4.1 Newton-Rhapson Iteration Method

The *Newton-Rhapson Method* is an algorithm for finding the root of a function. The idea is to approximate the function f by a *Taylor series* [6]

$$f(x) = f(x_o) + (x - x_o)f'(x_o) + ...$$

The approximation of the zero of the function $f(x) = 0$ results in

$$f(x_o) + (x_1 - x_o)f'(x_o) = 0$$

this is transformed into

$$x_1 = x_0 - \frac{f(x_o)}{f'(x_o)}$$

Consequently, the Newton-iteration can be written as

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \qquad n = 0, 1, 2, ... \tag{5.36}$$

Even though the *Newton-Raphson Method* convergences rapidly, the initial value has to be close enough to the actual zero of the function, to provide convergence [6].

5.4.2 Newton Method for nonlinear systems of equations

The *Newton-Method* described in chapter 5.4.1, can be extended to determine the roots of a system of nonlinear functions. The derivation of this method can be found in [6].

The *Taylor series* can be written in a compact form as

$$J^{(k)} * \vec{\delta}^{(k)} = -\vec{f}^{(k)} \tag{5.37}$$

whereby the *Jacobi-matrix* J is defined as

$$J^{(k)} = \left.\frac{\partial f_i}{\partial x_j}\right|_{\vec{x}^{(k)}} = \begin{pmatrix} \frac{\partial f_1(\vec{x}^{(k)})}{\partial x_1} & \cdots & \frac{\partial f_1(\vec{x}^{(k)})}{\partial x_N} \\ \cdots & & \cdots \\ \frac{\partial f_N(\vec{x}^{(k)})}{\partial x_1} & \cdots & \frac{\partial f_N(\vec{x}^{(k)})}{\partial x_N} \end{pmatrix} \tag{5.38}$$

Consequently, the Newton-algorithm is [6]

- Calculate $\vec{\delta}^{(k)}$ out of $J^{(k)} * \vec{\delta}^{(k)} = -\vec{f}^{(k)}$
- Solve the equation system $\vec{x}^{(k+1)} = \vec{x}^{(k)} + \vec{\delta}^{(k)}$
- In case of convergence, $\vec{\delta}$ stays constant and $\vec{f} = 0$

5.4.3 Implemented Iteration Methods

In the following, the implemented iteration methods are explained. The software implementation can be found in chapter 6.2.4. For all iteration methods, an initial value has to be committed to the respective method, from which the iteration starts. The calculation of the initial values is given in chapter 5.5.

Input combination: $\{T, p, x\}$

This iteration is based on eq.(5.17), where the density is determined iteratively by the Newton-Method. According to eq.(5.36) the algorithm is

$$\delta^{(k+1)} = \delta^{(k)} - \frac{f(p,\delta,\tau,x)}{f'(\delta,\tau,x)} \tag{5.39}$$

Since the dimensionless density δ is evaluated, the density ρ has to be substituted by $\delta = \rho_n \delta$. The function and its derivative is given in eq.(5.40) and eq.(5.41)

$$f_{(p,\tau,\delta,x)} = 0 = \delta \frac{\rho_n}{MRT}\left(1 + \delta \Phi^r_\delta\right) - p \tag{5.40}$$

$$f'_{(\tau,\delta,x)} = \left.\frac{\partial f}{\partial \delta}\right|_{(\tau,x)} = \frac{\rho_n}{MRT}\left(1 + 2\delta\Phi^r_\delta + \delta^2 \Phi^r_{\delta\delta}\right) \tag{5.41}$$

Once δ has been found, the density can be recalculated from the equation $\rho = \delta \rho_n M$

The algorithm is shown by figure 5.4. Other iteration methods are following the same principle.

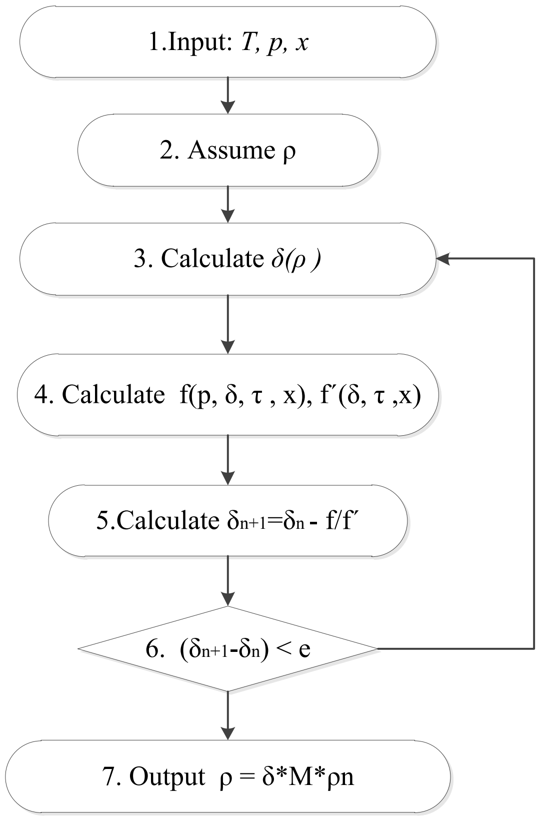


Figure 5.4: Algorithm for ‘T,p,x’-Iteration

Input combination: $\{\rho, p, x\}$

For this iteration method the temperature is computed from eq.(5.17) as well. Whereby, the function is derived with respect to the temperature.

$$T^{(k+1)} = T^{(k)} - \frac{f(p, \delta, T, x)}{f'(\delta, T, x)} \tag{5.42}$$

The dimensionless temperature τ is substituted by

$$\tau = \frac{T_n}{T} \qquad\qquad \tau' = \frac{\partial \tau}{\partial T} = \frac{T_n}{T^2}$$

The function and its derivative with respect to T is given by

$$f_{(p,T,\delta,x)} = 0 = \frac{\rho}{M} RT\big(1 + \delta \Phi_\delta^r\big) - p \tag{5.43}$$

$$f'_{(T,\delta,x)} = \frac{\partial f}{\partial T}\bigg|_{(\delta,x)} = \frac{\rho}{M} R\big(1 - \delta \Phi_{\delta\tau}^r \frac{T_n}{T} + \delta \Phi_\delta^r\big) \tag{5.44}$$

Input combination: $\{T, p, \rho\}$

In order to determine the ammonia fraction x, eq.(5.17) is derived with respect to x. This is more sophisticated, since the inner derivatives with respect to the dimensionless density and temperature have to be made, for this equation. Further, the derivative of the molar mass M is required (see appendix A).

$$x^{(k+1)} = x^{(k)} - \frac{f(p, \delta, \tau, x)}{f'(p, \delta, \tau, x)} \tag{5.45}$$

whereby the function is given by

$$f_{(p,\tau,\delta,x)} = 0 = \frac{\rho}{M} RT\big(1 + \delta \Phi_\delta^r\big) - p \tag{5.46}$$

$$f'_{(\tau,\delta,x)} = \frac{\partial f}{\partial x}\bigg|_{(\tau,\delta)} = -\frac{\rho}{M^2} RTM_x\big(1 + \delta \Phi_\delta^r\big) + \frac{\rho}{M} RT\big(\delta_x \Phi_\delta^r + \delta \Phi_{\delta x}^r\big) \tag{5.47}$$

The derivative of the molar mass (eq.(5.28)) is given by

$$M_x = -M_1 + M_2$$

Once the derivatives of the pure components are obtained, the residual part is composed as follows

$$\Phi_{\delta x} = -\Phi 1 + (1-x)\Phi 1_{\delta x}^r + \Phi 2 + x\Phi 2_{\delta x}^r + \Delta \Phi_{\delta x}^r \tag{5.48}$$

Input combination: $\{T, h, x\}$

For this iteration method, the density ρ is determined by using the Helmholtz equation for the enthalpy (5.20). As done in previous iterations, the density is substituted by the dimensionless value δ, and is resubstituted once a solution has been found.

$$\delta^{(k+1)} = \delta^{(k)} - \frac{f(h, \delta, \tau, \tau^\circ, x)}{f'(\delta, \tau, x)} \tag{5.49}$$

The function is given by

$$f_{(h,\delta,\tau,\tau^\circ,x)} = 0 = RT\big(1 + \delta\Phi^r_\delta + \tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau\big) - h \tag{5.50}$$

$$f'_{(\delta,\tau,x)} = \frac{\partial f}{\partial \delta}\bigg|_{(\tau,x)} = RT\big(\Phi^r_\delta + \delta\Phi^r_{\delta\delta} + \tau\Phi_{\tau\delta}\big) \tag{5.51}$$

Input combination: $\{p, h, x\}$

For this input combination, the two parameters (ρ, T) have to be calculated. In order to determine the missing values, the *Newton Method for nonlinear systems of equations* (5.4.2) is applied.

The Helmholtz equations for *pressure* p, eq.(5.17) and *enthalpy* h, eq.(5.20) are derived in order to build the Jacobi-matrix. This implies the derivatives with respect to the temperature and density, for both equations, whereas the density is substituted by the dimensionless density δ.

Figure 5.5 shows the algorithm for solving the iteration. The functions and its derivatives are given by

$$J^{(k)} * \vec{\delta}^{(k)} = -\vec{f}^{(k)}$$

$$f1_{(p,T,\delta,x)} = 0 = \rho_n RT\big(\delta + \delta^2\Phi^r_\delta\big) - p \tag{5.52}$$

$$\frac{\partial f1}{\partial \delta}\bigg|_{(T,x)} = \rho_n RT\big(1 + 2\delta\Phi^r_\delta + \delta^2\Phi^r_{\delta\delta}\big) \tag{5.53}$$

$$\frac{\partial f1}{\partial T}\bigg|_{(\delta,x)} = \rho_n R\big(\delta + \delta^2\Phi^r_\delta\big) + \rho_n RT\left(-\delta^2\Phi^r_{\delta\tau}\frac{T_n}{T^2}\right) \tag{5.54}$$

$$f2_{(h,T,\delta,x)} = 0 = \frac{R}{MT}\big(1 + \delta\Phi^r_\delta + \tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau\big) - h \tag{5.55}$$

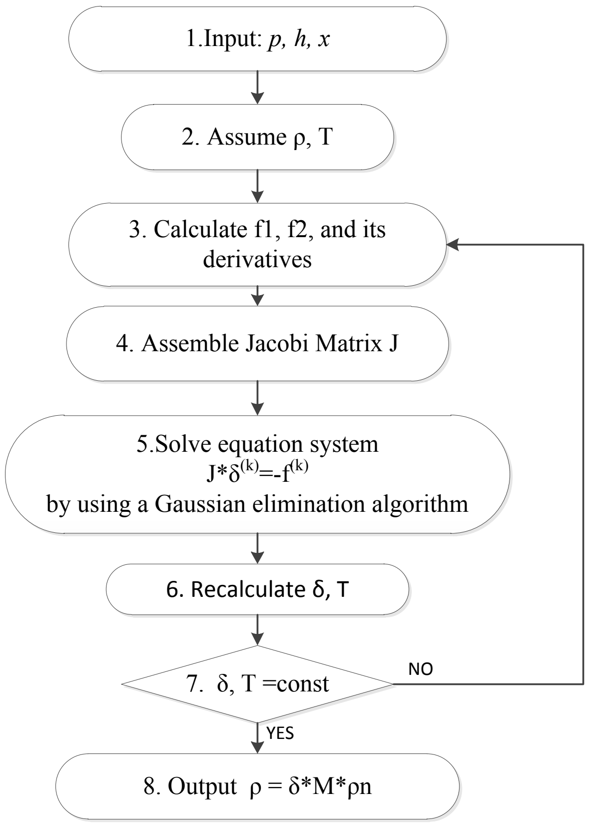

1.Input: p, h, x
2. Assume ρ, T
3. Calculate f1, f2, and its derivatives
4. Assemble Jacobi Matrix J
5.Solve equation system J*δ(k)=-f(k) by using a Gaussian elimination algorithm
6. Recalculate δ, T
7. δ, T =const
NO
YES
8. Output ρ = δ*M*ρn

Figure 5.5: Algorithm for 'p,h,x'-Iteration

$$\left.\frac{\partial f2}{\partial \delta}\right|_{(T,x)} = \frac{1}{M} RT\big(\Phi^r_\delta + \delta\Phi^r_{\delta\delta} + \tau\Phi^r_{\tau\delta}\big) \tag{5.56}$$

$$\begin{aligned}\left.\frac{\partial f2}{\partial T}\right|_{(\delta,x)} &= \frac{1}{M} R\big(1 + \delta\Phi^r_\delta + \tau^\circ\Phi^\circ_{\tau^\circ} + \tau\Phi^r_\tau\big) \\ &+ \frac{1}{M} RT\big(-\delta\Phi^r_{\tau\delta}\frac{T_n}{T^2} - \tau^\circ\Phi^\circ_{\tau^\circ}\frac{T_o}{T^2} - \tau^\circ\Phi^\circ_{\tau^\circ\tau^\circ}\frac{T_o}{T^2} - \Phi^r_\tau\frac{T_n}{T^2} - \tau\Phi^r_{\tau\tau}\frac{T_n}{T^2}\big)\end{aligned} \tag{5.57}$$

Jacobi-matrix J, the *solution-vector* $\vec{\delta}$ and *vector* $\vec{f}$ can be written as

$$J = \begin{pmatrix} \frac{\partial f1}{\partial \delta}\big|_{(T,x)} & \frac{\partial f1}{\partial T}\big|_{(\delta,x)} \\ \\ \frac{\partial f2}{\partial \delta}\big|_{(T,x)} & \frac{\partial f2}{\partial T}\big|_{(\delta,x)} \end{pmatrix} \qquad \vec{\delta} = \begin{pmatrix} \delta \\ T \end{pmatrix} \qquad \vec{f} = \begin{pmatrix} f1 \\ f2 \end{pmatrix}$$

The algorithm can be solved according to chapter 5.4.2. For each iteration step, the equation system $J^{(k)} * \vec{\delta}^{(k)} = -\vec{f}^{(k)}$ is solved by using a **Gaussian elimination algorithm**. The software implementation of this algorithm is explained in chapter 6.2.4.

5.5 Initial Calculations

Iteration methods, as well as vapor-liquid algorithms, require initial values from which the missing parameter can be determined accurately. To set a start value, a fast approximation of the current parameter has to be made. Such calculations are intended to meet the following requirements:

- **Adequate accuracy** to provide convergence of the corresponding algorithm.
- The formulation must be **valid** over a wide range of the thermodynamic surface. This implies both liquid and vapor phase.
- Furthermore, the function should not have more than **two dependent variables** (although with *Soave-Redlich-Kwong*, a formulation with three dependent variables is applied).

Acceptable formulations are given by *Patek & Klomfar* [8]. The set of equations has been especially developed for describing vapor-liquid equilibrium properties. Single phase properties are approximated with a sufficient accuracy [1].

Simplified formulations for the density can be obtained by *Saul and Wagner* [2], which has been originally made for describing properties of pure water. A modified version for {ammonia-water} mixtures is given in *M.Conde Engineering* [1].

Another approach for calculating the mixture density is the application of a cubic EoS (see chapter 4). The *Soave-Redlich-Kwong* equation generates good results in the vapor phase. However, for some iterations this formulation is not applicable, since three input parameters (T, p, y) are required.

5.5.1 Equations of 'Patek & Klomfar'

The equations of '*Patek and Klomfar*' allow the determination of the temperature T, vapor fraction y and enthalpy h. As already mentioned, the equations have been developed for fast approximation, especially for vapor-liquid equilibrium [8]. The functions have the following structure.

$$T(p,x) = T_o \sum_{i=1}^{14} a_i (1-x)^{m_i} \left[ln\left(\frac{p_o}{p}\right)\right]^{n_i} \tag{5.58}$$

$$T(p,y) = T_o \sum_{i=1}^{17} a_i (1-y)^{m_i/4} \left[ln\left(\frac{p_o}{p}\right)\right]^{n_i} \tag{5.59}$$

$$y(p,x) = 1 - exp\left[ln(1-x) \sum_{i=1}^{14} a_i \left(\frac{p}{p_o}\right)^{m_i} x^{n_i/3}\right] \tag{5.60}$$

$$h_L(T,x) = h_o \sum_{i=1}^{16} a_i \left(\frac{T}{T_o} - 1\right)^{m_i} x^{n_i} \tag{5.61}$$

$$h_G(T,y) = h_o \sum_{i=1}^{17} a_i \left(1 - \frac{T}{T_o}\right)^{m_i} (1-y)^{n_i/4} \tag{5.62}$$

Coefficients of the equations can be found in Appendix C.

5.5.2 Approximation of the Density ρ

Fast approximation of the mixture density is given by *Conde engineering* [1]. There are two separate formulations for the liquid and vapor phase. Both equations consist of the pure component densities, and an additional 'excess part' [1].

Liquid mixture density

The liquid density is calculated with the equation [1]

$$\rho^{(L)} = x\rho_2^{(L)} + (1-x)\rho_1^{(L)} + \Delta\rho(T_m^*, x) \tag{5.63}$$

The excess term is approximated with

$$\Delta\rho(T_m^{*i}, x) = \big[x(1-x) - A_\rho x^2 (1-x)\big] \rho_2^{0,5} \rho_1^{0,5} \tag{5.64}$$

The parameter A_ρ, which is a function of the critical temperature of water and the ammonia fraction x, can be calculated with

$$A_\rho = \sum_{i=0}^{2} A_{\rho 1,i} T_m^{*i} + \frac{\sum_{i=0}^{2} A_{\rho 2,i} T_m^{*i}}{x} \qquad T_m^* = \frac{T}{T_{c1}} \tag{5.65}$$

Coefficients are given in Appendix C.

The pure component densities are calculated with the equation [1]

$$\rho_{1,2}^{(L)} = \rho_{c1} \sum_{i=0}^{6} A_{\rho_i} \Big[1 - \frac{T}{T_{c1}}\Big]^{b_i} \tag{5.66}$$

Vapor mixture density

The vapor density is calculated with the following equation

$$\rho^{(V)} = y\rho_2^{(V)} + (1-y)\rho_1^{(V)} + \Delta\rho(T_m^*, y) \tag{5.67}$$

According to [1]. The excess term is approximated with

$$\Delta\rho(T_m^{*i}, y) = B_1(1-y)^{B_2}\left(1 - e^{B_3 y^{B_4}}\right)\Delta\rho_{max}(T_m^*) \tag{5.68}$$

$$\Delta\rho_{max}(T_m^*) = e^{B_5 - B_6/T_m^*} \qquad\qquad T_m^* = \frac{T}{T_{c1}}$$

The pure component densities of the vapor part are calculated with the equation [1]

$$ln\left(\frac{\rho_{1,2}^{(V)}}{\rho_{c1}}\right) = \sum_{i=0}^{6} A_{\rho_i}\left[1 - \frac{T}{T_{c1}}\right]^{b_i} \tag{5.69}$$

All coefficients needed for this formulation are given in Appendix C.

5.5.3 Soave-Redlich-Kwong

The approximation of the vapor density with eq.(5.67) has shown a significant deviation to the real value. This might causes convergence problems in some cases. A more accurate approach is the calculation of the vapor density with the Soave equation of state (see chapter 4). Even though this approximation is not applicable in all cases, since three input parameters are required, a precise starting value is ensured [12].

The Soave EoS describes the pressure as a function of temperature and molar volume.

$$p = \frac{RT}{V_m - b} - \frac{a_s \alpha_s}{V_m(V_m + b)} \tag{5.70}$$

The factors a_s, α_s and b are modified functions, incorporating the acentric factor ω and the vapor composition y [12].

$$a_s\alpha_s = \sum_{i=1}^{2}\sum_{i=1}^{2} y_i y_j (1 - k_{ij})\sqrt{(a_i\alpha_i)(a_j\alpha_j)} \qquad b = \sum_{i=1}^{2} y_i b_i \tag{5.71}$$

with the functions for the pure components

$$a_i = 0,42747\frac{(RT_{ci})^2}{p_{ci}} \qquad b_i = 0.08664\frac{RT_{ci}}{p_{ci}} \tag{5.72}$$

$$\alpha_i = \left[1 + n_i(1 - \sqrt{T_{r,i}})\right]^2 \qquad n_i = 0,48508 + 1,55171\ \omega_i - 0,15613\ \omega_i^2 \tag{5.73}$$

The acentric factor ω_{H2O} and ω_{NH3} is set to $0,344$ and $0,25$, respectively [12].

To calculate the mixture density, the equation of state has to be solved iteratively, as described in chapter 5.4.1, by finding the root of the function (5.70). The algorithm for this iteration is

$$\rho_{n+1} = \rho_n + \frac{f(p,T,\rho,y)}{f'(T,\rho,y)} \tag{5.74}$$

The molar volume of eq.(5.70) is replaced by $V_m = M/\rho$. The function f and its derivative is given with

$$f(p,T,\rho,y) = 0 = \frac{RT}{V_m - b} - \frac{a_s\alpha_s}{V_m(V_m+b)} - p \tag{5.75}$$

$$f'(T,\rho,y) = \frac{\partial f}{\partial \rho} = \frac{RTM}{\left(M/\rho - b\right)^2\rho^2} - \frac{a_s\alpha_s}{M\left(M/\rho + b\right)} - \frac{a_s\alpha_s}{\rho\left(M/\rho + b\right)^2} \tag{5.76}$$

The software implementation of the algorithm is shown in chapter 6.2.5

5.5.4 Mixture critical values

The determination of the current phase requires information about the critical locus of the mixture. Thus, approximations of the critical pressure and temperature are required. The equations, given by *Conde Engineering* [1], were established by using experimental data from *Sassen et al.* [15]. Coefficients are given in Appendix C.

$$T_{c,12} = \sum_{i=0}^{4} a_{ci} z^i \tag{5.77}$$

$$p_{c,12} = \sum_{i=0}^{4} b_{ci} z^i \tag{5.78}$$

The mixture critical density is a linear function of the critical values of the single components, by incorporating a factor k_V [19]. The factor k_V is given in table 5.2. Thus, the density is defined with

$$\frac{1}{\rho_{c,12}} = \frac{k_V}{2}\left(\frac{1}{\rho_{c1}} + \frac{1}{\rho_{c2}}\right) \tag{5.79}$$

5.5.5 Line of triple points

The range of validity covers the space between the solid-liquid-vapor boundary and the critical point [19]. The line of triple points of {ammonia-water} mixtures has three eutectic points.

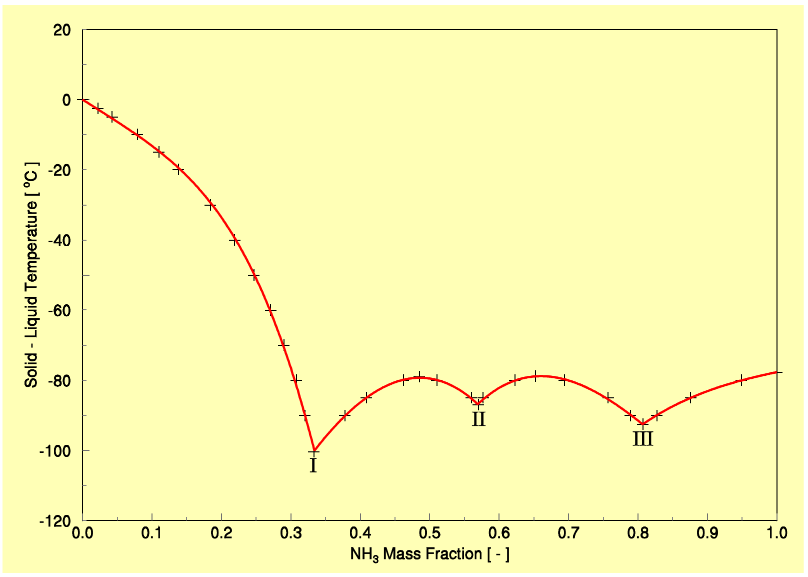


Figure 5.6: Line of triple points [1]

Figure 5.6 shows the three eutectic points at $x = 0.334$, $x = 0.584$ and $x = 0.815$. Furthermore, the triple-point line shows two maxima at $x = 0.5$ and $x = 2/3$ [18]. The expression for the triple point temperature is given by [19].

$$T_{tr} = 273.16(1 + c_1 x + c_2 x^2 + c_3 x^7) \qquad (0 < x < 0.33367)$$

$$T_{tr} = 193.549(1 + c_4 (x - 0.5)^2) \qquad (0.33367 < x < 0.58396)$$

$$T_{tr} = 194.380(1 + c_5 (x - 2/3)^2 + c_6 (x - 2/3)^3) \qquad (0.58396 < x < 0.814)$$

$$T_{tr} = 195.495(1 + c_7 (1 - x) + c_8 (1 - x)^4) \qquad (0.81473 < x < 1)$$

The coefficients for calculating the triple point temperature, can be found in table 5.4.

c_1	-0.343 982 3	c_5	-4.886 151
c_2	-1.327 427 1	c_6	10.372 98
c_3	-274.973	c_7	-0.323 998
c_4	-4.987 368	c_8	-15.875 60

Table 5.4: Coefficients for the triple-point calculation

Chapter 6

Software implementation

6.1 Conception and Structure

In this chapter the programming of the entire calculation procedure is described. This includes the layout of the program as well as the implementation of the individual routines.

The program is partitioned into subtasks, which are independently compiled in separate *class files*. Each class contains several *methods*. All classes are linked by interacting *objects*. In order to do this, an object of a certain class is initialized within another class by transferring parameters [11].

To begin, the procedure was subdivided into five main tasks. Each task consists of one or more class files.

- Helmholtz functions (5 classes)
- Thermodynamic properties (1 class)
- Initial calculation (1 class)
- Iteration methods (1 class)
- Vapor-liquid equilibrium (2 classes)

Further, two more classes were made for definition of constant values and auxiliary functions. A class consists of two separate files, the *header file* and the *source code file*. In the header file, all variables and subroutines of the class are declared, while the real programming is made in the source code file [11]. Table 6.1 gives an overview of all implemented classes.

No.	Header file	Brief description	No. of Methods	Integrated classes
1.	Const.h	Definition of general parameter	15	–
2.	IdealPart.h	Calculation of the ideal part of the Helmholtz energy	3	1.)
3.	Rp_H2O.h	Calculation of the residual part of water	29	1.)
4.	Rp_NH3.h	Calculation of the residual part of ammonia	9	1.)
5.	Rp_MIX.h	Calculation of the departure function	9	1.)
6.	ResidualPart.h	Composition of all residual parts	8	1.) 3.) 4.) 5.)
7.	TD_Properties.h	Relations between Helmholtz functions and TD-properties	9	1.) 2.) 6.)
8.	Initial.h	Definition of initial calculations	9	1.)
9.	Iteration.h	Implementation of iteration methods	5	1.) 8.) 2.) 6.) 7.)
10.	VLE.h	Algorithm for vapor-liquid equilibrium	9	1.) 6.) 7.) 8.) 9.) 12.)
11.	Phase.h	Determination of the present phase	5	1.) 7.) 8.) 9.) 10.)
12.	Mole_fraction.h	Conversion of molar to mass fraction	2	1.)

Table 6.1: List of implemented classes

The correlation of the classes is illustrated in figure 6.1, whereas a detailed data flow is given in chapter 6.3.

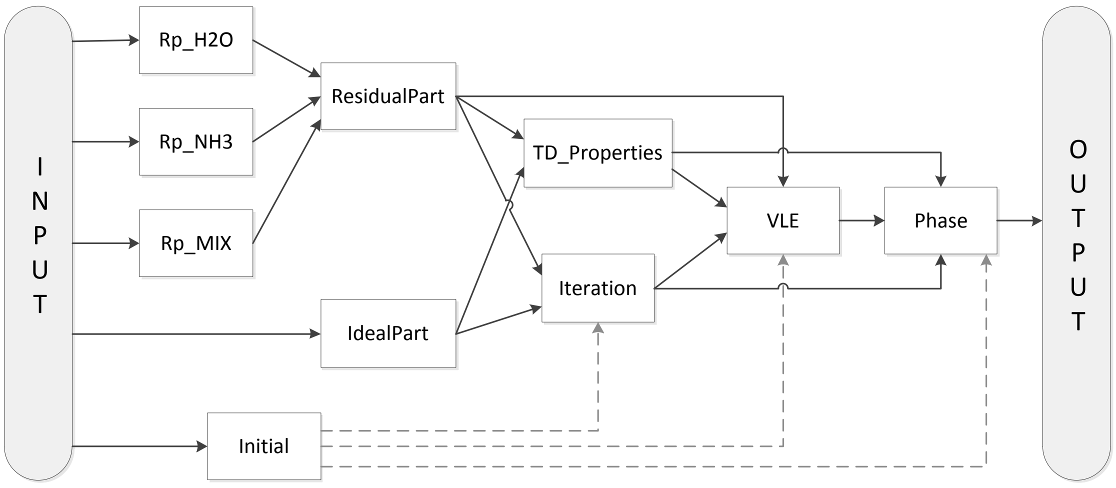


Figure 6.1: Coherences of the classes

6.2 Classes & Methods of the program

6.2.1 Definition of basic values

The standard functions and global variables, which are used in several other classes, where implemented in a separate class file, for added convenience. In the class **Const**, pure fluid components are defined, as well as the dimensionless temperature and density, described in chapter 5. All methods of the class *Const* are listed in table 6.2.

In class **Mole_fraction**, two functions are implemented to convert the mass fraction ξ into molar fraction x, and vice versa. These functions are needed at the beginning and the end of the calculation process, as the user is asked to input mass fraction while the procedure is expecting molar values.

The correlation between mass- and molar fraction is given in [5] with

$$\xi = \frac{M_2}{M}x \tag{6.1}$$

by using eq.(5.28), the fractions can be converted to

$$x = \frac{\xi M_1}{M_2 + \xi M_1 - \xi M_2} \qquad \xi = x\frac{M_2}{(1-x)M_1 + xM_2} \tag{6.2}$$

Each calculation is implemented in separate methods.

Name	Description	Label	Unit	Equation
M	molar mass	M	kg/mol	(5.28)
Mx	derivative of the molar mass	M_x	–	(A.31)
tau	dimensionless temperature	τ	–	(5.7)
delta	dimensionless density	δ	–	(5.7)
tau_o	dimensionless temperature	τ°	–	(5.4)
delta_o	dimensionless density	δ°	–	(5.4)
Vn	critical molar volume	V_n	m^3/mol	(5.9)
Vn_x	molar volume, derivative	$V_{n,x}$	–	(A.33)
Tn	temperature function	T_n	K	(5.8)
Tn_x	temperature function, derivative	$T_{n,x}$	–	(A.32)
delta_x	dimensionless density, derivative	δ_x	–	(A.34)
tau_x	dimensionless temperature, derivative	τ_x	–	$(A.35)$
Tc	mixture critical temperature	T_c	K	(5.77)
pc	mixture critical pressure	p_c	N/m^2	(5.78)
rho_c	mixture critical density	ρ_c	kg/m^3	$\rho_c = M/V_c$

Table 6.2: List of implemented Methods of class *Const*

6.2.2 Helmholtz free energy

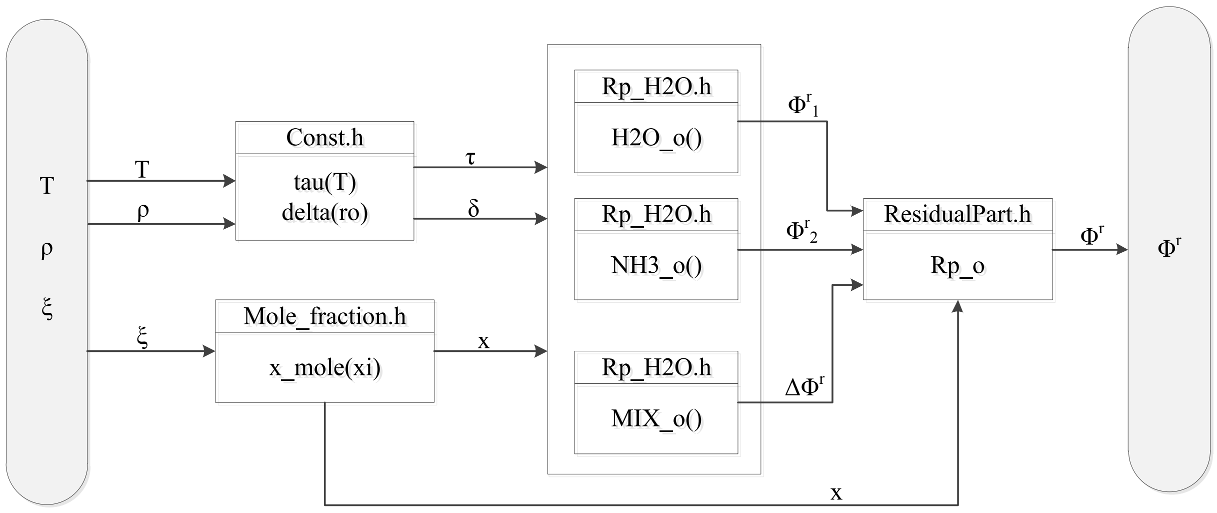


Figure 6.2: Procedure for the calculation of Φ^r

Core of the routine is the calculation of the Helmholtz functions. As shown previously, the calculation of the Helmholtz functions, is subdivided into five classes. Each term of the residual part (Φ^r_1, Φ^r_2, $\Delta\Phi^r$) and the ideal part (Φ°) is implemented in a separate class, containing their respective functions. Within these classes, the respective function and its derivatives, are again implemented in own methods. Further, an additional method is initialized as a container for the coefficients (*iniarray()*), in terms of arrays.

In figure 6.2, the calculation of the residual part of the Helmholtz free energy is illustrated. The derivatives of this function, are following the same procedure.

Rp_H2O.h			Rp_NH3.h			Rp_MIX.h		
Name	Label	Eq.	Name	Label	Eq.	Name	Label	Eq.
H2O_o	Φ_1^r	(5.13)	NH3_o	Φ_2^r	(5.12)	MIX_o	$\Delta\Phi^r$	(5.11)
H2O_d	$\Phi_{1,\delta}^r$	(A.17)	NH3_d	$\Phi_{2,\delta}^r$	(A.11)	MIX_d	$\Delta\Phi_\delta^r$	(A.3)
H2O_t	$\Phi_{1,\tau}^r$	(A.19)	NH3_t	$\Phi_{2,\tau}^r$	(A.13)	MIX_t	$\Delta\Phi_\tau^r$	(A.6)
H2O_x	$\Phi_{1,x}^r$	(A.22)	NH3_x	$\Phi_{2,x}^r$	(A.16)	MIX_x	$\Delta\Phi_x^r$	(A.9)
H2O_tt	$\Phi_{1,\tau\tau}^r$	(A.20)	NH3_tt	$\Phi_{2,\tau\tau}^r$	(A.14)	MIX_tt	$\Delta\Phi_{\tau\tau}^r$	(A.7)
H2O_dd	$\Phi_{1,\delta\delta}^r$	(A.18)	NH3_dd	$\Phi_{2,\delta\delta}^r$	(A.12)	MIX_dd	$\Delta\Phi_{\delta\delta}^r$	(A.4)
H2O_dt	$\Phi_{1,\delta\tau}^r$	(A.21)	NH3_dt	$\Phi_{2,\delta\tau}^r$	(A.15)	MIX_dt	$\Delta\Phi_{\delta\tau}^r$	(A.8)

Table 6.3: List of Methods of the classes of the residual part

Table 6.3 gives a summary of the methods implemented in the classes of the residual part. However, in class '*Rp_H2O*' additional methods are made, to outsource functions from eq.(5.13).

In the class **'ResidualPart'**, the single terms of the residual part are summarized, by using eq.(5.6). Hence, each derivative is implemented in a separate method.

In contrast to the residual part, the ideal part class has only three methods, which include the derivatives with respect to τ (Table 6.4).

Name	Label	Equation
Ip_o	Φ°	(5.3)
Ip_t	Φ_τ°	(A.1)
Ip_tt	$\Phi_{\tau\tau}^\circ$	(A.2)

Table 6.4: List of Methods of the class '*IdealPart*'

In the following the programming of the Helmholtz function Φ_2^r (eq. 5.12) is illustrated. All the other Helmholtz functions are conforming to the same principle.

Considering C++ routine, a function is called with *'Class::Method'*, in this case *'double*

```
double Rp_NH3::NH3_o()
{
    Rp_NH3 ^getarray = gcnew Rp_NH3(T,ro,x);
    getarray->iniarray();       //get coefficients from Rp_NH3::iniarray()

    double S1 = 0;              // 1st term
    int i;
    for (i=0; i<=4; i++)
        S1 += ai[i]*Math::Pow(tau, ti[i])*Math::Pow(delta, di[i]);

    double S2 = 0;              // 2nd term
    int j;
    for (j=5; j<=20; j++)
        S2 += ai[j]*Math::Pow(tau, ti[j])*Math::Pow(delta, di[j])
            *Math::Exp(-Math::Pow(delta,ei[j]));

    return S1 + S2;
}
```

Rp_NH3::NH3_o()'. The commands are then written between two braces { ...}. As a first step, the coefficients, needed for the equation, are initialized by calling the container class *iniarray()*. Each of the two terms, is calculated by *for-loops*. Thus, two variables of the type *'double'* are initialized (*S1, S2*), where the terms of the sums are added.

6.2.3 Thermodynamic Parameter

Based on the Helmholtz functions, the thermophysical properties can be calculated. In doing so, a new class has been created where each single parameter is computed in a separate method.

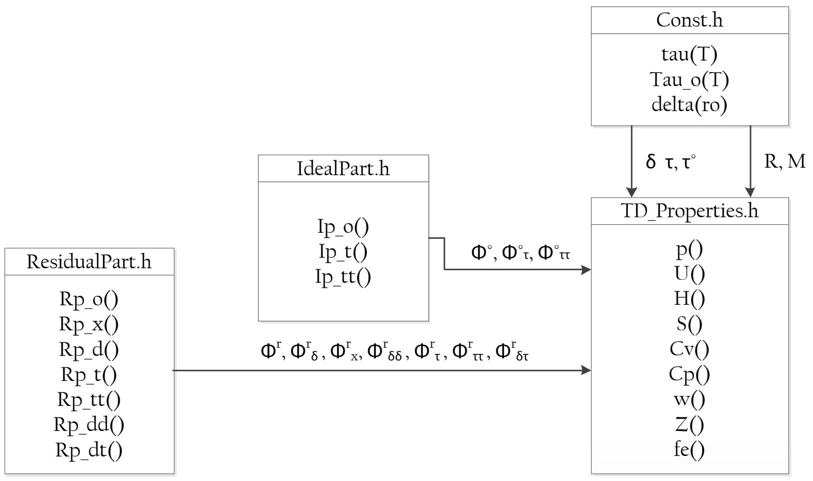


Figure 6.3: program sequence of class '*TD_Properties*'

Figure 6.3 shows the program flow within the class. The required parameters are obtained from the Helmholtz classes, as well as from class '*Const*', by initializing objects.

Name	Description	Label	Unit	Equation
p	pressure	p	N/m^2	(5.17)
U	internal energy	u	J/kg	(5.19)
H	enthalpy	h	J/kg	(5.20)
S	entropy	s	$J/(kgK)$	(5.21)
Cv	Isochoric heat capacity	c_v	$J/(kgK)$	(5.22)
Cp	Isobaric heat capacity	c_p	$J/(kgK)$	(5.23)
w	Speed of sound	w	m/s	(5.24)
Z	Compressibility factor	Z	–	(5.18)
fe	Helmholtz free energy	f	J/mol	(5.2)

Table 6.5: Methods of the class 'TD_Properties'

Table 6.5 gives an outline of the properties which are computed in the class '*TD_Properties*'.

6.2.4 Iteration Methods

The iteration algorithms are implemented in class '*Iteration*'. Thus, each iteration method, described in chapter 5.4, is compiled in an own method.

Iteration	Given parameters	Return value	Algorithm
Tpx	T, p, x	ρ	(5.39)
rpx	ρ, p, x	T	(5.42)
Tpr	T, p, ρ	x	(5.45)
Txh	T, x, h	ρ	(5.52)
phx	p, h, x	T, ρ	(5.52)

Table 6.6: Iteration methods

The programming of the iteration algorithms is shown by the example of of the following code. Except iteration method '*phx*', all iteration methods are following the same principle.

For calculating the varying parameter δ, an array (Xn) is initialized. The first position of the array is the start value, which is committed to the function. Each time a solution has

```
    int k=0;
    do
    {
        ResidualPart ^functions = gcnew ResidualPart(T,(Xn[k]*M*ro_n),x);
        double Rp_d = functions->Rp_d();
        double Rp_dd = functions->Rp_dd();

        double f = ro_n*R*T* (Xn[k] + Math::Pow(Xn[k],2)*Rp_d) - p;

        double f_d = ro_n*R*T* (1 + 2*Xn[k]*Rp_d + Math::Pow(Xn[k],2)*Rp_dd);

        Xn[k+1] = Xn[k] - f/f_d;

        delta = Xn[k+1];

        e = Math::Abs(Xn[k+1]-Xn[k]);
        k = k+1;

    }while(e>=1E-5);

    double rho = delta*ro_n*M;
    return rho;
```

been found, the new δ is added to the array (Xn). Once a constant value for δ is obtained, the iteration is stopped. Finally the density can be recalculated by $\rho = \delta M \rho_n$.

In contrast to other iterations, the algorithm for the '*phx*'-iteration is based on the *newton method for non linear equation systems*, and is presented by the following code:

```
  ...
  // assembling of the auxiliary matrix U
  int i;
  int j;
  for(i=0;i<=1;i++)
      for(j=0;j<=1;j++)
          U[i][j] = J[i][j];
  U[0][2]=-f1; U[1][2]=-f2;

  // Gaussian elimination algorithm
  int w; int l; int m;
  double a; double b; double c; double d;
  for(w=0; w<=1; w++)
  {
     a = U[w][w];
     for(l=w+1; l<=1; l++)
      {
          b = U[l][w];
          for(m=w; m<=2; m++)
          {
              c = U[l][m];
              d = U[w][m];
              U[l][m] = c-(b/a*d);
          }
      }
  }
  ...
```

The single steps are depicted in figure 5.5. Accordingly, the derived functions have to be assembled to a Jacobi Matrix J. Further, the *Gaussian elimination method* is applied. In order to solve the equation system, an auxiliary matrix U is initialized. This matrix is then solved with the elimination algorithm, as shown in the source code above.

6.2.5 Initial Calculations

Initial calculations are compiled in class '*Initial*'. An overview of the implemented methods, is given in table 6.7.

Name	Description	Unit	Equation
T_px	temperature - liquid phase	K	(5.58)
T_py	temperature - vapor phase	K	(5.59)
y_px	ammonia fraction-vapor phase	–	(5.60)
hL	enthalpy - liquid phase	kJ/kg	(5.61)
hG	enthalpy - vapor phase	kJ/kg	(5.62)
ro_lq	density - liquid phase	kg/m^3	(5.63)
ro_vp	density - vapor phase (SRK)	kg/m^3	(5.67)
ro_vp2	density - vapor phase	kg/m^3	(5.74)
SRK	Soave - Redlich - Kwong	–	(5.70)

Table 6.7: Methods of class '*Initial*'

All methods transfer the given inputs and return the actual parameter. Coefficients, needed for the calculations, are outsourced to method '*iniarray()*'.

In case of the method '*SRK*', two values are returned. In fact, the pressure and its derivative with respect to ρ. These parameter are needed for method '*ro_vp*'.

6.2.6 Vapor-liquid equilibrium

The calculation of the vapor-liquid equilibrium is an important subtask of this work. As described in chapter 5.3, information about vapor-liquid boundaries are necessary for the detection of the existing phase, as well as for determining the properties in the two-phase region.

The VLE-algorithms, described in chapter 5.3, are compiled in the class '*VLE*'. The class consists of four different algorithm, which are needed for further calculations. Additional functions are implemented for auxiliary calculations. Table 6.8 gives an overview of the methods of class '*VLE*'

Name	Brief description	Input values	Return parameter
phase	Calculation of properties in the vapor liquid area	T, p, z	$x, y, \rho^{(L)}, \rho^{(V)}, phase$
transition	Determination of the vapor liquid boundary at $z = 0$ and $z = 1$	p, z	$T, \rho^{(V)}, phase$
bubble	Determination of the bubble point of the mixture	p, z, T_U, T_L	$T_B, \rho^{(L)}, phase$
dew	Determination of the dew point of the mixture	p, z, T_U, T_L	$T_B, \rho^{(V)}, phase$
beta	Returns a new approximation for the vapor fraction β	z, β, K_1, K_1	β
set_pB	Returns a new approximation for the bubble point pressure p_B	i, s, p	p_L, p_U
set_pD	Returns a new approximation for the dew point pressure p_D	i, s, p	p_L, p_U
set_TB	Returns a new approximation for the bubble point temperature T_B	i, s, T	T_L, T_U
set_TD	Returns a new approximation for the dew point temperature T_D	i, s, T	T_L, T_U

Table 6.8: Methods of class '*VLE*'

The computation of thermodynamic properties in the vapor-liquid area is made with the function '*phase*'. The implemented algorithm is shown in fig.5.3. Extracts of the source code are given below.

After initializing local variables, a start value for y_i is set, by using eq.(5.60). The iteration is then started with a *do-while* loop. The iteration of the density requires initial values, for both vapor and liquid phase. Once the fugacity coefficients are obtained, a new vapor fraction β is approximated. This is done by function '*beta*'. Finally, y_i and x_i can be recalculated.

Another feature of the methods of class '*VLE*' is a Boolean value (*phase*). This value is set with *true*, unless the iteration yields an invalid value. In order to switch to *false*, several exceptions had been implemented.

```
do
{   ...
    // Calculation of density and fugacity coefficients
    ...
    K1 = phi1_lq / phi1_vp;      // Calculate K1, K2
    K2 = phi2_lq / phi2_vp;

    //Calculate beta
    Beta[i+1] = setbeta->beta(z,Beta[i],K1,K2);

    //Calculate xi,yi
    x1 = (1-z)/(1+Beta[i+1]*(K1-1));
    x2 = z/(1+Beta[i+1]*(K2-1));
    y1 = K1*x1;
    y2 = K2*x2;

    //Normalize xi,yi
    X1[i+1]=x1/(x1+x2);
    ...

    xi=X2[i+1];
    yi=Y2[i+1];
    // set break condition
    h1 = X1[i+1]*K1-Y1[i+1];     h2 = X2[i+1]*K2-Y2[i+1];
    if(Double::IsNaN(h1)==true || Double::IsNaN(h2)==true)
    {
        *phase = false;
        break;
    }
    i=i+1;
}while(Math::Abs(h1+h2)>=e  && i<=499);
```

In order to determine the existing phase, the vapor-liquid boundaries of the pure components, have to be calculated (see chapter 6.2.7). This is made by method '*transition*'. The algorithm of this function is given in figure 5.2.

At first, the temperature is pre-calculated by using eq.(5.58). The temperature is then multiplied with an empirical factor to set iteration limits. At the beginning of the first loop, the median temperature is computed. Afterwards, the inner loop is initialized, which is basically the same as in function '*phase*'. This implies the calculation of the fugacity coefficients, as well as a new approximation of the ammonia fraction.

Once a new approximation for the fraction is made, the temperature is recalculated by means of a bisection algorithm [12]. This is outsourced to function '*set_TB*', which returns new iteration limits. Extracts of the source code are given below.

Methods '*bubble*' and '*dew*' are calculating the saturation properties of the mixture. The algorithm can be found in figure 5.2 as well. In contrast to method '*transition*', where the

```
try
{
    int i=0;
    do
    {
        Tb[i] = (TU+TL)/2;
        int k=0;
        do
        {
            ...
            // Calculate phi, rho, xi, yi,...
            ...

        }while(Math::Abs(ya2[k-1]-ya2[k]) >=e);
        if(*phase==false)
            break;
        setpressure->set_TB(i,s,Tb[i], &TL, &TU);   //recalculate temperature
        TB=Tb[i];
        i=i+1;
    }while(Math::Abs(s-1)>=e);
    *T = TB;
    *ro_vp = ro_V;
}
catch (Exception ^e)
{
    *phase = false;
}
```

temperature is readjusted with each iteration step, methods '*bubble*' and '*dew*' readjust the pressure.

The calculation of the fugacity coefficients out of the Helmholtz functions is made in class '*Fugacity*'. The class consists of three methods, given in table 6.9

Name	Description	Label	Equation
phi1	Fugacity coefficient of water	φ_1	(5.25)
phi2	Fugacity coefficient of ammonia	φ_2	(5.26)
F	auxiliary function	F_φ	(5.27)

Table 6.9: Methods of class '*Fugacity*'

6.2.7 Determination of the phase

In order to detect the existing phase, an algorithm has been implemented, which uses the parameters T, p and z to compute the bubble- and dew point of the mixture.

At first, the temperatures at the liquid-vapor boundary of the pure components are determined, by using function '*VLE::transition*'. Thereafter, a median temperature T_m, as a function of the overall composition is calculated with the linear equation.

$$T_m = T_{D(z=1)} + (T_{D(z=0)} - T_{D(z=1)})(1 - z) \tag{6.3}$$

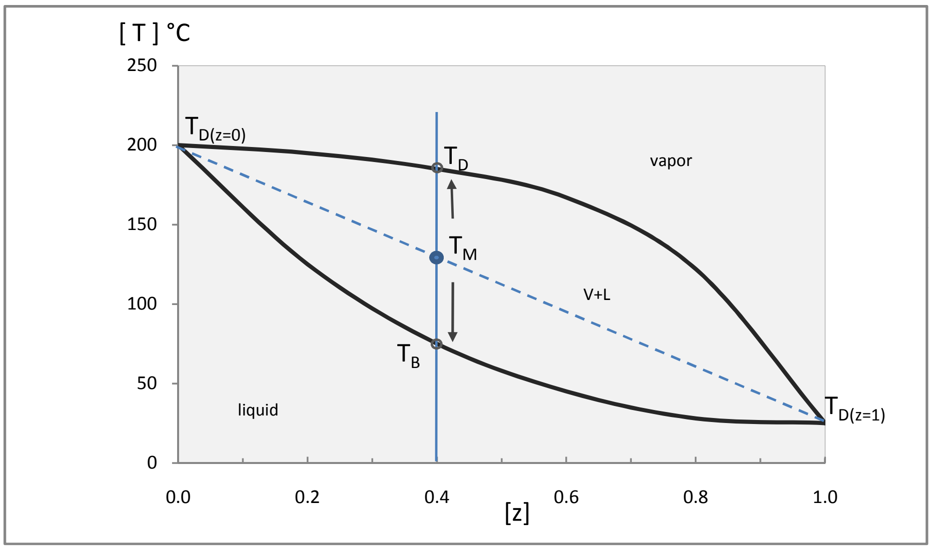


Figure 6.4: T-x Diagram at p=1,5 bar

T_m is then used to set iteration limits for the calculation of dew and bubble point. The placement of the iteration limits is set dynamically. At the beginning, a narrow margin is set, in order to provide rapid convergence. Each time the VLE-calculation returns an invalid value (identified with the boolean operator *'phase'* of class *VLE.h*), the iteration limits are redefined. Figure 6.5 shows the procedure of the calculation.

Similar to class *'VLE'*, a boolean operator is defined, in order to identify iteration errors. Finally, the temperature is compared with the bubble- and dew point of the mixture. Thus, the following cases are distinguished.

$T < T_B$	$\rightarrow$ liquid phase
$T > T_D$	$\rightarrow$ vapor phase
$T > T_B$ & $T < T_D$	$\rightarrow$ two-phase region
$T > T_c$ & $p >= p_c$	$\rightarrow$ supercritical region
dim==*false*	$\rightarrow$ no phase information available

For any other input combination, the missing parameter (T, p or z) has to be determined before the phase detection algorithm is initialized.

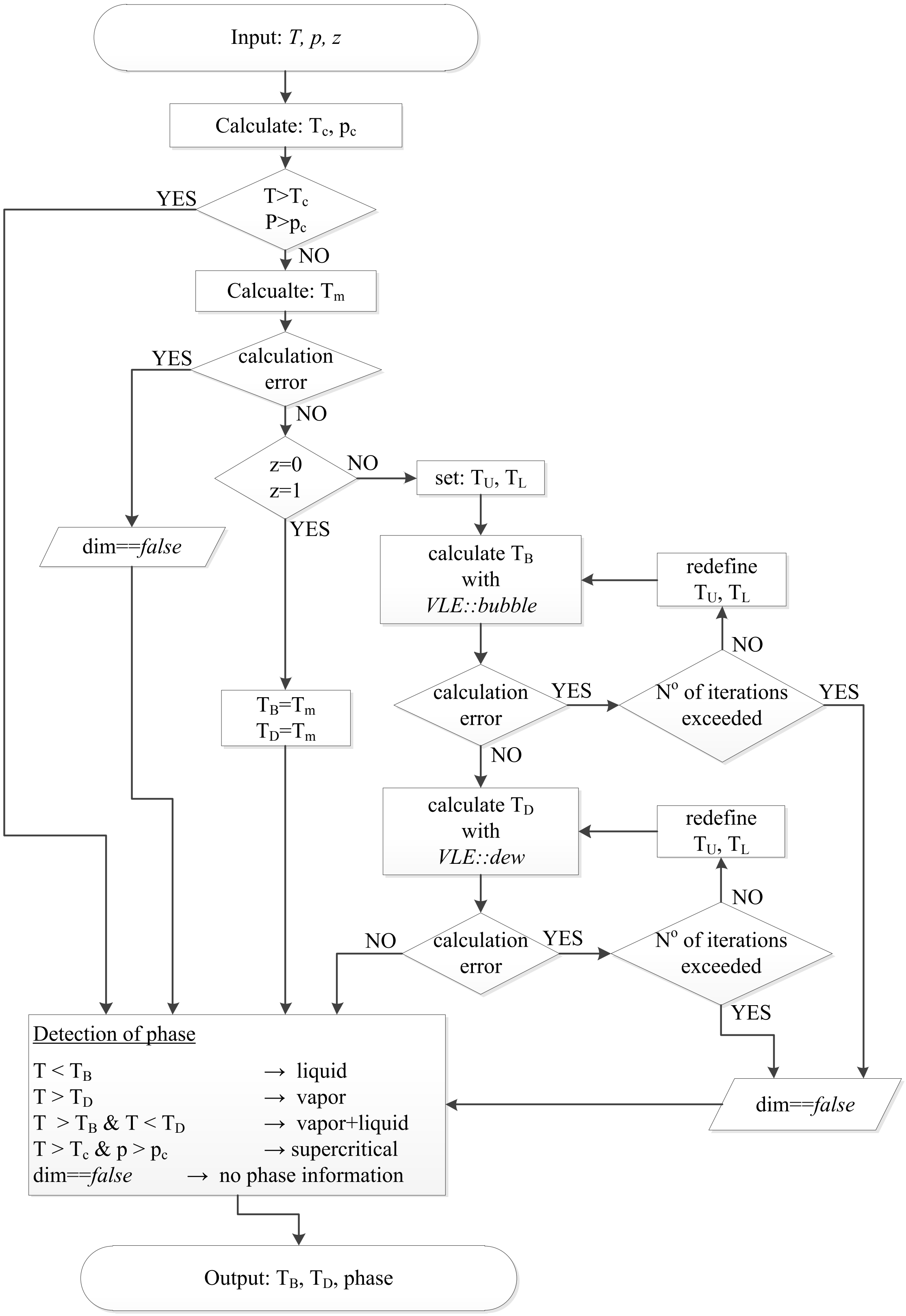


Figure 6.5: procedure of method '*getphase*'

6.3 Sequence of the calculation

In this section, the procedure of the entire calculation process is discussed. If the routine is used by an external program (MATLAB, Excel), a function considering the favored input combination (e.g. $\{T,p,z\}$) is called, and the corresponding parameters are transferred to the function. If the program is called via *command prompt window*, the user must input the appropriate parmeter.

As a first step, the input parameters are verified. If the values are outside the scope, the process is cancelled and the error massage '*Input error*' is returned. The routine also tests, if the input is within the solid region of the mixture. In this case, the string '*Solid phase*' is returned. The range of validity of the single input parameter is given in table 6.10.

Parameter	Unit	Upper limit	Lower limit
T	K	650	0
ρ	kg/m^3	1030	0
z	–	1	0
p	Pa	$40 * 10^6$	100
h	kJ/kg	3300	-400

Table 6.10: range of validity of the input parameters

Before the thermodynamic parameters can be calculated, the current phase has to be determined. In case of the input combination $\{T,p,z\}$, the phase is computed directly with function *Phase::getphase*. Otherwise, the missing parameter has to be calculated iteratively (see chapter 6.2.7).

If the phase determination returns an invalid value, the routine returns a string, '*no phase information available*'. Additionally, the thermodynamic properties are calculated and returned.

If the current input values are within the two phase region, a VLE-calculation is initialized. Once a phase composition is detected, the properties can be calculated using the Helmholtz equations.

In case of a single phase, the properties can be computed directly. Finally, the current phase (*vapor phase*, *liquid phase* or *supercritical region*) and the thermodynamic properties are returned. The sequence of the entire calculation process is illustrated in figure 6.6.

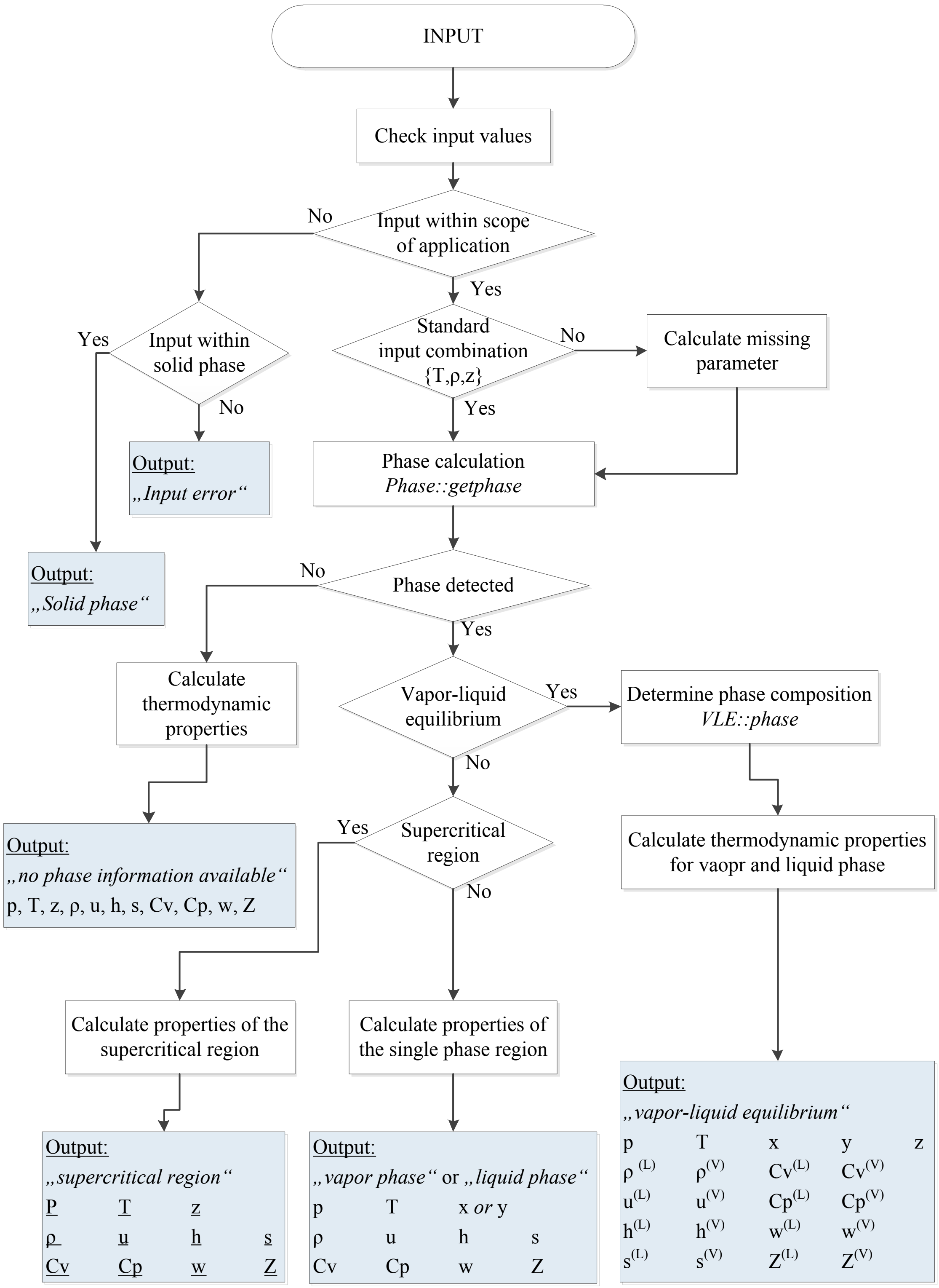

INPUT
Check input values
Input within scope of application
No
Yes
Input within solid phase
Yes
No
Output: „Input error“
Output: „Solid phase“
Standard input combination {T,ρ,z}
No
Calculate missing parameter
Yes
Phase calculation Phase::getphase
Phase detected
No
Yes
Calculate thermodynamic properties
Output: „no phase information available“ p, T, z, ρ, u, h, s, Cv, Cp, w, Z
Vapor-liquid equilibrium
Yes
Determine phase composition VLE::phase
No
Calculate thermodynamic properties for vaopr and liquid phase
Supercritical region
Yes
No
Calculate properties of the supercritical region
Calculate properties of the single phase region
Output: „supercritical region“ P T z ρ u h s Cv Cp w Z
Output: „vapor phase“ or „liquid phase“ p T x or y ρ u h s Cv Cp w Z
Output: „vapor-liquid equilibrium“ p T x y z ρ(L) ρ(V) Cv(L) Cv(V) u(L) u(V) Cp(L) Cp(V) h(L) h(V) w(L) w(V) s(L) s(V) Z(L) Z(V)

Figure 6.6: sequence of the calculation process

Chapter 7

Validation of the Properties

7.1 Comparison with existing Data

In this chapter, the results of the calculation process are discussed, and compared with existing data. In tables 7.1 to 7.7, the calculated properties of the particular input combination are compared with the properties given by '*Tillner-Roth & Friend*' [17].

Tables, which are listed in [17], provide properties of the {ammonia-water} mixture, based on the Helmholtz equation of state.

For both single- and two-phase region, two state points of choice are investigated. In doing so, the values of the parameters T, ρ, z, p, h and s are compared. The given parameters are highlighted.

There is a slight deviation between the reported values [17], and the calculated values. The small difference may occur, because the values, given in [17], for a certain pressure, are given with only two places after the decimal point.

Further deviations may occur within the vapor-liquid equilibrium. Since the properties of the two-phase region are obtained iteratively by calculating the fugacity of the components, a variance of $\pm 0.5\%$ is noticed.

Apart from that the calculated properties of this routine agree completely with the values given in [17].

	T	ρ	x	p	h	s
	$°C$	kg/m^3	-	MPa	kJ/kg	$kJ/(kgK)$
'Tillner-Roth & Friend' [17]	25	923.60	0.2	0.2	19.16	0.5779
input {T, ρ, z}	25	923.60	0.2	0.202	19.169	0.5779
input {T, p, z}	25	923.60	0.2	0.2	19.158	0.57788
input {ρ, p, z}	25.01	923.60	0.2	0.2	19.19	0.5780
input {T, p, ρ}	25	923.60	0.1999	0.2	19.155	0.5779
input {T, z, h}	25	923.60	0.2	0.206	19.16	0.57787
input {p, h, z}	25.00	923.60	0.2	0.2	19.16	0.5779

Table 7.1: Liquid phase properties at p=0,2MPa, $T = 25°C$, $\xi = 0.2$

	T	ρ	x	p	h	s
	$°C$	kg/m^3	-	MPa	kJ/kg	$kJ/(kgK)$
'Tillner-Roth & Friend [17]	175	707.26	0.4	20	740.89	2.6853
input {T, ρ, z}	175	707.26	0.4	20.008	740.90	2.6853
input {T, p, z}	175	707.258	0.4	20	740.89	2.68528
input {ρ, p, z}	174.99	707.26	0.4	20	740.89	2.68527
input {T, p, ρ}	175	707.26	0.3996	20	740.888	2.68527
input {T, z, h}	175	707.251	0.4	19.989	740.89	2.6853
input {p, h, z}	174.99	707.257	0.4	20	740.89	2.68527

Table 7.2: Liquid phase properties at p=20MPa, $T = 175°C$, $\xi = 0.4$

	T	ρ	x	p	h	s
	$°C$	kg/m^3	-	MPa	kJ/kg	$kJ/(kgK)$
'Tillner-Roth & Friend [17]	250	63.52	0.8	12	2118.06	5,8704
input {T, ρ, z}	250	63.52	0.8	11.999	2118.08	5.87042
input {T, p, z}	250	63,523	0.8	12	2118.05	5.87037
input {ρ, p, z}	250,00	63.52	0.8	12	2118.08	5.8704
input {T, p, ρ}	250	63.52	0.7996	12	2118.04	5.87039
input {T, z, h}	250	63,523	0.8	11.999	2118.06	5.87036
input {p, h, z}	249.99	63.523	0.8	12	2118.06	5.87039

Table 7.3: Vapor properties at p=12MPa, $T = 250°C$, $\xi = 0.8$

	T	ρ	x	p	h	s
	$°C$	kg/m^3	-	MPa	kJ/kg	$kJ/(kgK)$
'Tillner-Roth & Friend [17]	325	77.72	0.4	15	2480.77	5,9588
input {T, ρ, z}	325	77.72	0.4	14.999	2480.79	5.9589
input {T, p, z}	325	77.723	0.4	15	2480.773	5.95882
input {ρ, p, z}	325	77.72	0.4	15	2480.79	5.9589
input {T, p, ρ}	325	77.72	0.3998	15	2480.75	5.9589
input {T, z, h}	325	77.726	0.4	15	2480.77	5.9588
input {p, h, z}	324.998	77.724	0.4	15	2480.77	5.95881

Table 7.4: Vapor properties at p=15MPa, $T = 325°C$, $\xi = 0.4$

	T	p	ρ^L	ρ^V	x	y	h^L	h^V	s^L	s^V
	$°C$	kPa	kg/m^3		–		kJ/kg		$kJ/(kgK)$	
[17]	60	578.22	827.49	3.713	0.4	0.98333	165.67	1758.1	1.2706	6.4533
{T, ρ, z}	60	585.4	827.49	3.81	0.4	0.9832	165.73	1758.01	1.2706	6.4496
{T, p, z}	60	578.22	827.49	3.713	0.4	0.833	165.67	1758.13	1.2706	6.4533
{ρ, p, z}	59.9	578.22	827.49	3.713	0.4	0.9833	165.66	1758.13	1.2706	6.4533
{T, p, ρ}	60	578.22	827.49	3.713	0.399	0.9833	165.69	1758.14	1.2706	6.4533
{T, z, h}	60	579.1	827.49	3.74	0.4	0.9833	165.67	1757.91	1.2706	6.4487
{p, h, z}	59.9	578.22	827.49	3.713	0.4	0.9833	165.67	1758.13	1.2706	6.4533

Table 7.5: Saturation properties at $T = 60°C$, p=578.22 kPa, $\xi = 0.4$

	T	p	ρ^L	ρ^V	x	y	h^L	h^V	s^L	s^V
	$°C$	kPa	kg/m^3		–		kJ/kg		$kJ/(kgK)$	
[17]	310	11218	663.59	64.455	0.03448	0.1	1416	2623.9	3.4734	5.707
{T, ρ, z}	310	11218.6	663.99	64.455	0.03443	0.1	1415.8	2623.7	3.4728	5.7068
{T. p, z}	310	11218	664	64.481	0.03444	0.1	1415.75	2623.7	3.4729	5.7067
{ρ, p, z}	310.1	11218	664.1	64.455	0.0341	0.1	1417.6	2625.7	3.4719	5.6979
{T, p, ρ}	310	11218	664.1	64.455	0.03441	0.1	1415.8	2623.7	3.4729	5.7067
{T, z, h}	310	11217.8	664.2	64.56	0.03451	0.1	1416	2623.5	3.4731	5.7063
{p, h, z}	310	11218	664.1	64.5	0.03445	0.1	1415.79	2623.66	3.4725	5.707

Table 7.6: Saturation properties at $T = 310°C$, p=11218.0 kPa, $\xi = 0.1$

	T	ρ	x	p	h	s
	$°C$	kg/m^3	-	MPa	kJ/kg	$kJ/(kgK)$
‘Tillner-Roth & Friend [17]	325	214.69	0.8	40	2003.56	52.770
input {T, ρ, z}	325	214.69	0.8	40,0023	2003.58	5,277
input {T, p, z}	325	214.69	0.8	40	2003.56	5,277
input {ρ, p, z}	325	214.69	0.8	40	2003.55	5,277
input {T, p, ρ}	325	214.69	0.799	40	2003.58	5,277
input {T, z, h}	325	214.69	0.8	39,999	2002.56	5,277
input {p, h, z}	325	214.69	0.8	40	2002.56	5,277

Table 7.7: Supercritical properties at p=40Mpa, $T = 325°C$, $\xi = 0.8$

7.2 Estimation of uncertainty

In chapter 7.1, the calculated data had been validated, by comparing with the data from ‘*Tillner-Roth & Friend*’ [17], which are based on the Helmholtz formulation.

Statements about the uncertainty of the Helmholtz equation of state, are mainly based on comparison with existing data. A survey of experimental measurements of the {ammonia-water} mixture is given in [18].

Experimental data are available for the liquid phase up to $420K$ and $40MPa$, and up to $10Mpa$ for the vapor phase. The uncertainties are around $\pm 0.3\%$ for the density, and $\pm 200J/mol$ for the enthalpy [17]. In the vapor region, the majority of available data is found at high ammonia fractions, since the vapor phase is dominated by ammonia [18].

The variation of the mole fraction is estimated with ± 0.01. Whereas, near the critical locus it can be up to ± 0.04.

For the supercritical region, no experimental data are available. For the present formulation, the data from the single regions are extrapolated. Thus, the accuracy within the supercritical region is unknown [17].

In the two-phase region, experimental data are only available for a limited range. However, enthalpies and densities show acceptable conformity [19].

However, an improvement of the current equation of state requires further measurements. Especially, in the supercritical region, but also in the vapor phase, reliable experimental data would be highly desirable [17].

Chapter 8

Conclusion

In this diploma thesis, a computer program has been developed, which allows the calculation of the thermophysical properties of {ammonia-water} mixtures. The equation of state, used in this study, represents the measurements currently available, and is the most accurate representation of the thermodynamic properties of {ammonia-water} mixtures to date [17].

The building of the routine can be subdivided into four parts. The first main task encompassed the implementation of the Helmholtz functions. This includes the building of the derivatives, as well as their implementation. The resulting thermodynamic properties were evaluated carefully in order to avoid introducing errors into the Helmholtz functions.

After implementing and testing the Helmholtz functions, the iteration methods had to be made by using a Newton-Rhapson algorithm. For this purpose, the thermodynamic functions had to be derived with respect to the unknown parameter.

The quality of these iteration algorithms is closely linked with the corresponding initial value. For this reason, an extensive study of literature was made in order to find appropriate formulations. The resulting equations had to be tested carefully, to provide convergence of the iteration methods, for a wide range of the thermodynamic surface.

The third part of this thesis is concerned with the calculation of the vapor-liquid equilibrium, as well as with the detection of the existing phase. At first, the fugacity coefficients of the pure components had to be calculated, by using the Helmholtz formulation. The core of the vapor-liquid calculation is an algorithm, which consists of two overlapping *while*-loops. With this algorithm, the vapor and liquid fraction of the mixture are solved iteratively, by recalculating the fugacity coefficients.

For the determination of the phase, a new algorithm has been proposed and integrated into the program. In doing so, the vapor-liquid boundary of the pure components, has been determined. From this, a median temperature was calculated to set a proper start value for the iteration of dew- and bubble-point of the mixture.

The last subtask was the compilation of the program. In addition, an interface for *MATLAB* and *MS-Excel* was created. In order to provide a stable and error-free operation, several exceptions had to be implemented. These exceptions were intended to break off the routine once a calculation error occurs. Additionally, limits for the input parameters were placed at the beginning of the routine.

Finally, the compiled program had to be converted to a *.dll* file, to enable the application as a library. At first, a standard library was created, which allows the usage within *C++* and *Visual Basic*. To call MATLAB functions from C++, the program had to be converted to a so called *mex*-file. A *mex*-file provides an interface between MATLAB and C++.

The present program represents a proper tool for calculating and constructing *Kalina cycles* and *absorption heat pumps*. With the Helmholtz fundamental equation of state, the most accurate formulation, for calculating the thermodynamic properties of mixtures of ammonia and water has been used.

List of Figures

2.1 Command prompt window, after calling *NH3H2O.exe* 9
2.2 Calculated values, displayed in the command prompt window 9

3.1 T-s Diagram – schematic comparison between ORC and Kalina cycle [9] . . 11
3.2 Absorption heat pump 12
3.3 T-s diagramm of a Kalina cycle [5] 13
3.4 Schematic - Kalina cycle [5] 14

5.1 Calculation process 19
5.2 Bubble-point & dew-point algorithm [12] 25
5.3 Algorithm for determining vapor-liquid equilibrium properties [12] 27
5.4 Algorithm for 'T,p,x'-Iteration 29
5.5 Algorithm for 'p,h,x'-Iteration 32
5.6 Line of triple points [1] 37

6.1 Coherences of the classes 41
6.2 Procedure for the calculation of Φ^r 42
6.3 program sequence of class '*TD_Properties*' 44
6.4 T-x Diagram at p=1,5 bar 51
6.5 procedure of method '*getphase*' 52
6.6 sequence of the calculation process 54

B.1 Coefficients and exponents of eg.(5.13) [20] 74

C.1 Coefficients for the initial approximation of density, given by [1] 75

C.2 Coefficients for the equations of 'Patek & Klomfar' (5.5.1), given by [1] . . . 76

List of Tables

2.1 transfer parameters 4

2.2 Overview of the transfer parameters of the functions 6

2.3 Parameters of *mexFunction* 7

2.4 *MATLAB* functions 8

5.1 Coefficients of the ideal part [19] 20

5.2 Coefficients of the departure function [19] 21

5.3 Coefficients of the ammonia part [19] 22

5.4 Coefficients for the triple-point calculation 38

6.1 List of implemented classes 40

6.2 List of implemented Methods of class *Const* 42

6.3 List of Methods of the classes of the residual part 43

6.4 List of Methods of the class '*IdealPart*' 43

6.5 Methods of the class 'TD_Properties' 45

6.6 Iteration methods 45

6.7 Methods of class '*Initial*' 47

6.8 Methods of class '*VLE*' 48

6.9 Methods of class '*Fugacity*' 50

6.10 range of validity of the input parameters 53

7.1 Liquid phase properties at p=0,2MPa, $T = 25°C$, $\xi = 0.2$ 56

7.2 Liquid phase properties at p=20MPa, $T = 175°C$, $\xi = 0.4$ 56

7.3 Vapor properties at p=12MPa, $T = 250°C$, $\xi = 0.8$ 56

7.4 Vapor properties at p=15MPa, $T = 325°C$, $\xi = 0.4$ 57

7.5 Saturation properties at $T = 60°C$, p=578.22 kPa, $\xi = 0.4$ 57

7.6 Saturation properties at $T = 310°C$, p=11218.0 kPa, $\xi = 0.1$ 57

7.7 Supercritical properties at p=40Mpa, $T = 325°C$, $\xi = 0.8$ 58

Bibliography

[1] *Thermophysical Properties of {NH3+H2O} Mixtures for the Industrial Design of Absorption Refrigeration Equipment.* M.Conde Engineering, 2006.

[2] A.Saul und W.Wagner. *International Equations for the Saturation Properties of Ordinary Water substance.* Institut für Thermo- und Fluiddynamik, Ruhr Universität Bochum, 1987.

[3] Hans Dieter Baehr und Stephan Kabelac. *Thermodynamik, Grundlagen und technische Anwendungen.* Springer-Verlag Berlin Heidelberg, 2006.

[4] C.Hainbach und S.Schädlich. *Dubbel, Taschenbuch für den Maschinenbau; M: Kälte-, Klima- und Heizungstechnik.* Springer-Verlag Berlin Heidelberg, 2011.

[5] Prof. M. Haider. *Angewandte Thermodynamik, Skriptung zur Vorlesung.* Institut für Thermodynamik und Energiewandlung, TU Wien, 2011.

[6] H.C.Kuhlmann. *Nummerische Methoden der Ingenieurwissenschaften, Skriptum zur Vorlesung.* Institut for Strömungslehre, TU Wien, 2009.

[7] H.D.Baehr und Reiner Tillner-Roth. *Thermodynamische Eigenschaften umweltverträglicher Kältemittel.* Springer-Verlag Berlin Heidelberg, 1994.

[8] J.Patek und J.Klomfar. *Simple functions for fast calculations of selected thermodynamic properties of the ammonia-water system.* Institute of Thermodynamics, Academy of Science of Czech Republic, 1994.

[9] Silke Köhler. *Geothermisch angetriebene Dampfkraftprozesse.* Technische Univesität Berlin, Dissertation, 2005.

[10] Eric W. Lemmon und Reiner Tillner-Roth. *A Helmholtz energy equation of state for calculating the TD Properties of fluid mixtures.* Elsevier Science B.V.-Fluid, 1999.

[11] Dirk Louis. *Visual C++ 2008.* Markt + Technik Verlag, 2008.

[12] J.P.M Trusler M.J. Assael und T.F. Tsolakis. *Thermophysical Properties of Fluids.* Imperial College Press, Covent Garden, London, 1996.

[13] P.A.Lolos und E.D.Rogdakis. *A Kalina power cycle driven by renewable energy.* Elsevier - Energy, 2009.

[14] R.T.Ellington J.Huebler R.A.Macriss, B.E.Eakin. *Physical and Thermodynamic Properties of Ammonia-Water Mixtures.* Institute for Gas techn., Chicago, Illinois, 1964.

[15] van der Kooi Sassen, Kwartel und de Swan Arons. *Vapor-Liquid Equilibria for the System Ammonia+Water up to the Critical Region.* J.Chem.Eng.Data,35, 1990.

[16] R. Tillner-Roth und F. Harms-Watzenberg. *Eine neue Fundamentalgleichung für Ammoniak.* 20th DKV-Tagung Heidelberg, Germany, Vol. II, 1993.

[17] Reiner Tillner-Roth und Daniel G.Friend. *A Helmholtz free energy formulation of the TD Properties of the Mixture {Water+Ammonia}.* NIST, Phys & Chem Prop. Division, Colorado, 1997.

[18] Reiner Tillner-Roth und Daniel G.Friend. *Survey and Assessment of Available Measurements on TD Properties of the Mixture {Water+Ammonia}.* NIST, Phys & Chem Prop. Division, Colorado, 1997.

[19] Reiner Tillner-Roth und Daniel G.Friend. *Guideline on the IAPWS Formulation 2001 for the TD Properties of Ammonia-Water Mixtures.* International Association for the Properties of Water and Steam, Gaithersburg, Maryland, USA, 2001.

[20] W.Wagner und A.Pruß. *The IAPWS Formulation 1995 for the TD Properties of Ordinary Water Substance for General and Scientific Use.* Lehrstuhl für Thermodynamik, Ruhr-Universität Bochum, D-44780 Bochum, Germany, 2002.

[21] Feng Xu und D.Yogi Goswami. *Thermodynamic Properties of ammonia-water mixtures for power cycle applications.* Elsevier-Energy, 1997.

Appendix A

Derivatives of the reduced Helmholtz energy

In chapter 5 the Helmholtz functions for ideal and residual part, as well as for both components are given. In the following, the derivatives of these functions are presented, which are required for the determination of the thermodynamic properties.

A.0.1 Ideal part

$$\Phi^\circ_\tau = (1-x)\{a_2 + \frac{a_2}{\tau^\circ} + \sum_{i=4}^{8}(\frac{a_i\theta e^{-\theta\tau^\circ}}{1-e^{-\theta\tau^\circ}})\} + x\{a_{10} + \frac{a_{11}}{\tau^\circ} + \sum_{i=12}^{14}(a_i t_i (\tau^\circ)^{t_i-1})\} \tag{A.1}$$

$$\Phi^\circ_{\tau\tau} = (1-x)\{-\frac{a_3}{(\tau^\circ)^2} + \sum_{i=4}^{8} a_i\theta^2 e^{-\theta\tau^\circ}(1-e^{-\theta\tau^\circ})^{-2}\} + x\{-\frac{a_{11}}{\tau^2} + \sum_{i=12}^{14} a_i t_i (t_i-1)(\tau^\circ)^{t_i-2}\} \tag{A.2}$$

A.0.2 Residual part

Derivatives of the departure function

$$\begin{aligned}\Delta\Phi^r_\delta = x(1-x^\gamma)\Big\{a_1 d_1 \tau^{t_1}\delta^{(d_1-1)} + \sum_{i=2}^{6}\big(a_i d_i \tau^{t_i}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i \tau^{t_i}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \\ + x\sum_{i=7}^{13}\big(a_i d_i \tau^{t_i}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i \tau^{t_i}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \\ + x^2\big(a_{14}d_{14}\tau^{t_{14}}\delta^{(d_{14}-1)}e^{-\delta^{e_{14}}} - e_{14}a_{14}\tau^{t_{14}}\delta^{(d_{14}+e_{14}-1)}e^{-\delta^{e_{14}}}\big)\Big\}\end{aligned} \tag{A.3}$$

$$\Delta\Phi^r_{\delta\delta} = x(1-x^\gamma)\Big\{a_1 d_1(d_1-1)\tau^{t_1}\delta^{(d_1-2)} + \sum_{i=2}^{6}\theta_i + x\sum_{i=7}^{13}\theta_i + x^2\theta_{14}\Big\} \tag{A.4}$$

with

$$\begin{aligned}\theta_i = (a_i d_i(d_i-1)\tau^{t_i}\delta^{(d_i-2)}e^{-\delta^{e_i}} - e_i a_i d_i \tau^{t_i}\delta^{(d_i+e_i-2)}e^{-\delta^{e_i}} \\ - (d_i+e_i-1)e_i a_i \tau^{t_i}\delta^{(d_i+e_i-2)}e^{-\delta^{e_i}} + a_i e_i^2 \tau^{t_i}\delta^{(d_i+2e_i-2)}e^{-\delta^{e_i}}\end{aligned} \tag{A.5}$$

$$\begin{aligned}\Delta\Phi^r_\tau = x(1-x^\gamma)\Big\{a_1 t_1 \tau^{(t_1-1)}\delta^{d_1} + \sum_{i=2}^{6}\big(a_i t_i \tau^{(t_i-1)}\delta^{d_i}e^{-\delta^{e_i}}\big) \\ + x\sum_{i=7}^{13}\big(a_i t_i \tau^{(t_i-1)}\delta^{d_i}e^{-\delta^{e_i}}\big) + x^2\big(a_{14}t_{14}\tau^{(t_{14}-1)}\delta^{d_{14}}e^{-\delta^{e_{14}}}\big)\Big\}\end{aligned} \tag{A.6}$$

$$\begin{aligned}\Delta\Phi^r_{\tau\tau} = x(1-x^\gamma)\Big\{a_1 t_1(t_1-1)\tau^{(t_1-2)}\delta^{d_1} + \sum_{i=2}^{6}\big(a_i t_i(t_i-1)\tau^{(t_i-2)}\delta^{d_i}e^{-\delta^{e_i}}\big) \\ + x\sum_{i=7}^{13}\big(a_i t_i(t_i-1)\tau^{(t_i-2)}\delta^{d_i}e^{-\delta^{e_i}}\big) + x^2\big(a_{14}t_{14}(t_{14}-1)\tau^{(t_{14}-2)}\delta^{d_{14}}e^{-\delta^{e_{14}}}\big)\Big\}\end{aligned} \tag{A.7}$$

$$\begin{aligned}\Delta\Phi^r_{\delta\tau} = x(1-x^\gamma)\Big\{a_1 d_1 t_1 \tau^{(t_1-1)}\delta^{(d_1-1)} + \sum_{i=2}^{6}\big(a_i d_i t_i \tau^{(t_i-1)}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i t_i \tau^{(t_i-1)}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \\ + x\sum_{i=7}^{13}\big(a_i d_i t_i \tau^{(t_i-1)}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i t_i \tau^{(t_i-1)}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \\ + x^2\big(a_{14}d_{14}t_{14}\tau^{(t_{14}-1)}\delta^{(d_{14}-1)}e^{-\delta^{e_{14}}} - e_{14}a_{14}t_{14}\tau^{(t_{14}-1)}\delta^{(d_{14}+e_{14}-1)}e^{-\delta^{e_{14}}}\big)\Big\}\end{aligned} \tag{A.8}$$

$$\Delta\Phi^r_x = (1 - x^\gamma - x^\gamma\gamma)\frac{\Delta\Phi^r}{x(1-x^\gamma)}\Big[a_i t_i \tau^{(t_i-1)}\delta^{d_i}\tau_x + a_i d_i \tau^{t_i}\delta^{(d_i-1)}\delta_x + \sum_{i=2}^{6}\theta_{xi} + \sum_{i=7}^{13}\big(\theta_i + x\theta_{xi}\big) + \big(2x\theta_1 4 + x^2\theta_{x14}\big)\Big] \tag{A.9}$$

with eq.(A.5) and

$$\theta_{xi} = a_i t_i \tau^{(t_i-1)}\delta^{d_i}e^{-\delta^{e_i}}\tau_x + a_i d_i \tau^{t_i}\delta^{(d_i-1)}e^{-\delta^{e_i}}\delta_x - a_i e_i \tau^{t_i}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\delta_x \tag{A.10}$$

Derivatives of the ammonia part

$$\Phi^r_{2\delta} = \sum_{i=1}^{5} a_i d_i \tau^{t_i}\delta^{(d_i-1)} + \sum_{i=6}^{21}\big(a_i d_i \tau^{t_i}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i \tau^{t_i}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \tag{A.11}$$

$$\Phi^r_{2\delta\delta} = \sum_{i=1}^{5} a_i d_i (d_i-1)\tau^{t_i}\delta^{(d_i-2)} + \sum_{i=6}^{21}\big(a_i d_i(d_i-1)\tau^{t_i}\delta^{(d_i-2)}e^{-\delta^{e_i}} - e_i a_i d_i \tau^{t_i}\delta^{(d_i+e_i-2)}e^{-\delta^{e_i}} - (d_i+e_i-1)e_i a_i \tau^{t_i}\delta^{(d_i+e_i-2)}e^{-\delta^{e_i}} + e_i^2 a_i \tau^{t_i}\delta^{(d_i+2e_i-2)}e^{-\delta^{e_i}}\big) \tag{A.12}$$

$$\Phi^r_{2\tau} = \sum_{i=1}^{5} a_i t_i \tau^{(t_i-1)}\delta^{d_i} + \sum_{i=6}^{21}\big(a_i t_i \tau^{(t_i-1)}\delta^{d_i-1}e^{-\delta^{e_i}}\big) \tag{A.13}$$

$$\Phi^r_{2\tau\tau} = \sum_{i=1}^{5} a_i t_i (t_i-1)\tau^{(t_i-2)}\delta^{d_i} + \sum_{i=6}^{21}\big(a_i t_i (t_i-1)\tau^{(t_i-2)}\delta^{d_i-1}e^{-\delta^{e_i}}\big) \tag{A.14}$$

$$\Phi^r_{2\delta\tau} = \sum_{i=1}^{5} a_i d_i t_i \tau^{(t_i-1)}\delta^{(d_i-1)} + \sum_{i=6}^{21}\big(a_i d_i t_i \tau^{(t_i-1)}\delta^{(d_i-1)}e^{-\delta^{e_i}} - e_i a_i t_i \tau^{(t_i-1)}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\big) \tag{A.15}$$

$$\Phi^r_{2x} = \sum_{i=1}^{5}\Big(a_i t_i \tau^{(t_i-1)}\delta^{d_i}\tau_x + a_i d_i \tau^{t_i}\delta^{(d_i-1)}\delta_x\Big) + \sum_{i=6}^{21}\Big(a_i t_i \tau^{(t_i-1)}\delta^{d_i}e^{-\delta^{e_i}}\tau_x + a_i d_i \tau^{t_i}\delta^{(d_i-1)}e^{-\delta^{e_i}}\delta_x - a_i e_i \tau^{t_i}\delta^{(d_i+e_i-1)}e^{-\delta^{e_i}}\delta_x\Big) \tag{A.16}$$

Derivatives of the water part. In contrast to the ammonia part and the departure function where the derivatives had to be calculated, the equations for the water part are given by *Pruß and Wagner*[20]

$$\begin{aligned}\Phi^r_{1\delta} = &\sum_{i=1}^{7} n_i d_i \delta^{d_i-1}\tau^{t_i} + \sum_{i=8}^{51} n_i e^{-\delta_i^c}\left[\delta^{d_i-1}\tau^{t_i}(d_i - c_i\delta^{c_i})\right] \\ &+ \sum_{i=52}^{54} n_i d_i \tau^{t_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2}\left[d_i/\delta - 2\alpha_i(\delta-\epsilon_i)\right] + \sum_{i=55}^{56} n_i\left[\Delta^{b_i}\left(\Psi + \delta\frac{\partial\Psi}{\partial\delta}\right) + \frac{\partial\Delta^{b_i}}{\partial\delta}\delta\Psi\right]\end{aligned} \tag{A.17}$$

$$\begin{aligned}\Phi^r_{1\delta} = &\sum_{i=1}^{7} n_i d_i(d_i-1)\delta^{d_i-2}\tau^{t_i} + \sum_{i=8}^{51} n_i e^{-\delta^{c_i}}\left[\delta^{d_i-2}\tau^{t_i}((d_i-c_i\delta^{c_i})(d_i-1-c_i\delta^{c_i}) - c_i^2\delta^{c_i})\right] \\ &+ \sum_{i=52}^{54} n_i \tau^{t_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2}\left[-2\alpha_i\delta^{d_i} + 4\alpha_i^2\delta^{d_i}(\delta-\epsilon_i)^2 - 4d_i\alpha_i\delta^{d_i-1}(\delta-\epsilon_i) + d_i(d_i-1)\delta^{d_i-2}\right] \\ &+ \sum_{i=55}^{56} n_i\left[\Delta^{b_i}\left(2\frac{\partial\Psi}{\partial\delta} + \delta\frac{\partial^2\Psi}{\partial\delta^2}\right) + 2\frac{\partial\Delta^{b_i}}{\partial\delta}\left(\Psi + \delta\frac{\partial\Psi}{\partial\delta}\right) + \frac{\partial^2\Delta^{b_i}}{\partial\delta^2}\delta\Psi\right]\end{aligned} \tag{A.18}$$

$$\begin{aligned}\Phi^r_{1\tau} = &\sum_{i=1}^{7} n_i t_i \delta^{d_i}\tau^{t_i-1} + \sum_{i=8}^{51} n_i t_i \delta^{d_i}\tau^{t_i-1} e^{-\delta_i^c} \\ &+ \sum_{i=52}^{54} n_i \delta^{d_i}\tau^{t_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2}\left[t_i/\tau - 2\beta_i(\tau-\gamma_i)\right] + \sum_{i=55}^{56} n_i\delta\left[\frac{\partial\Delta^{b_i}}{\partial\tau}\Psi + \Delta^{b_i}\frac{\partial\Psi}{\partial\tau}\right]\end{aligned} \tag{A.19}$$

$$\begin{aligned}\Phi^r_{1\tau\tau} = &\sum_{i=1}^{7} n_i t_i(t_i-1)\delta^{d_i}\tau^{t_i-2} + \sum_{i=8}^{51} n_i t_i(t_i-1)\delta^{d_i}\tau^{t_i-2} e^{-\delta_i^c} \\ &+ \sum_{i=52}^{54} n_i \delta^{d_i}\tau^{t_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2}\left[\left(t_i/\tau - 2\beta_i(\tau-\gamma_i)\right)^2 - \frac{t_i}{\tau^2} - 2\beta_i\right] \\ &+ \sum_{i=55}^{56} n_i\delta\left[\frac{\partial^2\Delta^{b_i}}{\partial\tau^2}\Psi + 2\frac{\partial\Delta^{b_i}}{\partial\tau}\frac{\partial\Psi}{\partial\tau} + \Delta^{b_i}\frac{\partial^2\Psi}{\partial\tau^2}\right]\end{aligned} \tag{A.20}$$

$$\begin{aligned}\Phi^r_{1\delta\tau\tau} = & \sum_{i=1}^{7} n_i d_i t_i \delta^{d_i-1}\tau^{t_i-1} + \sum_{i=8}^{51} n_i t_i \delta^{d_i-1}\tau^{t_i-1}(d_i - c_i\delta^{c_i})e^{-\delta^{c_i}} \\ & + \sum_{i=52}^{54} n_i \delta^{d_i}\tau^{t_i} e^{-\alpha_i(\delta-\epsilon_i)^2-\beta_i(\tau-\gamma_i)^2}\left[d_i/\delta - 2\alpha_i(\delta-\epsilon_i)\right]\left[t_i/\tau - 2\beta_i(\tau-\gamma_i)\right] \\ & + \sum_{i=55}^{56} n_i \left[\Delta^{b_i}\left(\frac{\partial\Psi}{\partial\tau} + \delta\frac{\partial^2\Psi}{\partial\delta\partial\tau}\right) + \delta\frac{\partial\Delta^{b_i}}{\partial\delta}\frac{\partial\Psi}{\partial\tau} + \frac{\partial\Delta^{b_i}}{\partial\delta}\left(\Psi + \delta\frac{\partial\Psi}{\partial\delta}\right) + \frac{\partial^2\Delta^{b_i}}{\partial\delta\partial\tau}\delta\Psi\right]\end{aligned} \tag{A.21}$$

Auxiliary functions for the derivatives of the water part

$$\frac{\partial\Delta^{b_i}}{\partial\delta} = b_i\Delta^{b_i-1}\frac{\partial\Delta}{\partial\delta} \qquad \frac{\partial\Delta^{b_i}}{\partial\tau} = -2\Theta b_i\Delta^{b_i-1} \tag{A.22}$$

$$\frac{\partial^2\Delta^{b_i}}{\partial\tau^2} = 2b_i\Delta^{b_i-1} + 4\Theta^2 b_i(b_i-1)\Delta^{b_i-2} \tag{A.23}$$

$$\frac{\partial^2\Delta^{b_i}}{\partial\delta\partial\tau} = -A_i b_i 2/\beta_i \Delta^{b_i-1}(\delta-1)[(\delta-1)^2]^{1/2\beta_i-1} - 2\Theta b_i(b_i-1)\Delta^{b_i-2}\frac{\partial\Delta}{\partial\delta} \tag{A.24}$$

$$\frac{\partial^2\Delta^{b_i}}{\partial\delta^2} = b_i\left\{\Delta^{b_i-1}\frac{\partial^2\Delta}{\partial\delta^2} + (b_i-1)\Delta^{b_i-2}\left(\frac{\partial\Delta}{\partial\delta}\right)^2\right\} \tag{A.25}$$

$$\frac{\partial\Delta}{\partial\delta} = (\delta-1)\left[A_i\Theta 2/\beta_i[(\delta-1)^2]^{1/2\beta_i-1} + 2B_i a_i[(\delta-1)^2]^{a_i-1}\right] \tag{A.26}$$

$$\begin{aligned}\frac{\partial^2\Delta}{\partial\delta^2} = & \frac{1}{(\delta-1)}\frac{\partial\Delta}{\partial\delta} + (\delta-1)^2\left[4B_i a_i(a_i-1)[(\delta-1)^2]^{a_i-2} + 2A_i^2\left(2/\beta_i\right)^2\left[[(\delta-1)^2]^{1/2\beta_i-1}\right]^2\right. \\ & \left. + A_i\Theta 4/\beta_i\left(\frac{1}{2\beta_i} - 1\right)[(\delta-1)^2]^{1/2\beta_i-2}\right]\end{aligned} \tag{A.27}$$

$$\frac{\partial\Psi}{\partial\delta} = -2C_i(\delta-1)\Psi \qquad \frac{\partial^2\Psi}{\partial\delta^2} = \{2C_i(\delta-1)^2 - 1\}2C_i\Psi \tag{A.28}$$

$$\frac{\partial\Psi}{\partial\tau} = -2D_i(\tau-1)\Psi \qquad \frac{\partial^2\Psi}{\partial\tau^2} = \{2D_i(\tau-1)^2 - 1\}2D_i\Psi \tag{A.29}$$

$$\frac{\partial^2 \Psi}{\partial \delta \partial \tau} = 4C_i D_i(\delta - 1)(\tau - 1)\Psi \tag{A.30}$$

Derivatives with respect to x. For the calculation of the fugacity coefficients as well as for iteration methods, the derivative of some Helmholtz functions with respect to the ammonia fraction is needed.

$$M_x = \frac{\partial M}{\partial x} = -M_1 + M_2 \tag{A.31}$$

$$T_{n,x} = \frac{\partial T_n}{\partial x} = -2(1-x)T_{c1} + 2xT_{c2} + 2(1-x^\alpha)T_{c12} - 2x^\alpha \alpha T_{c12} \tag{A.32}$$

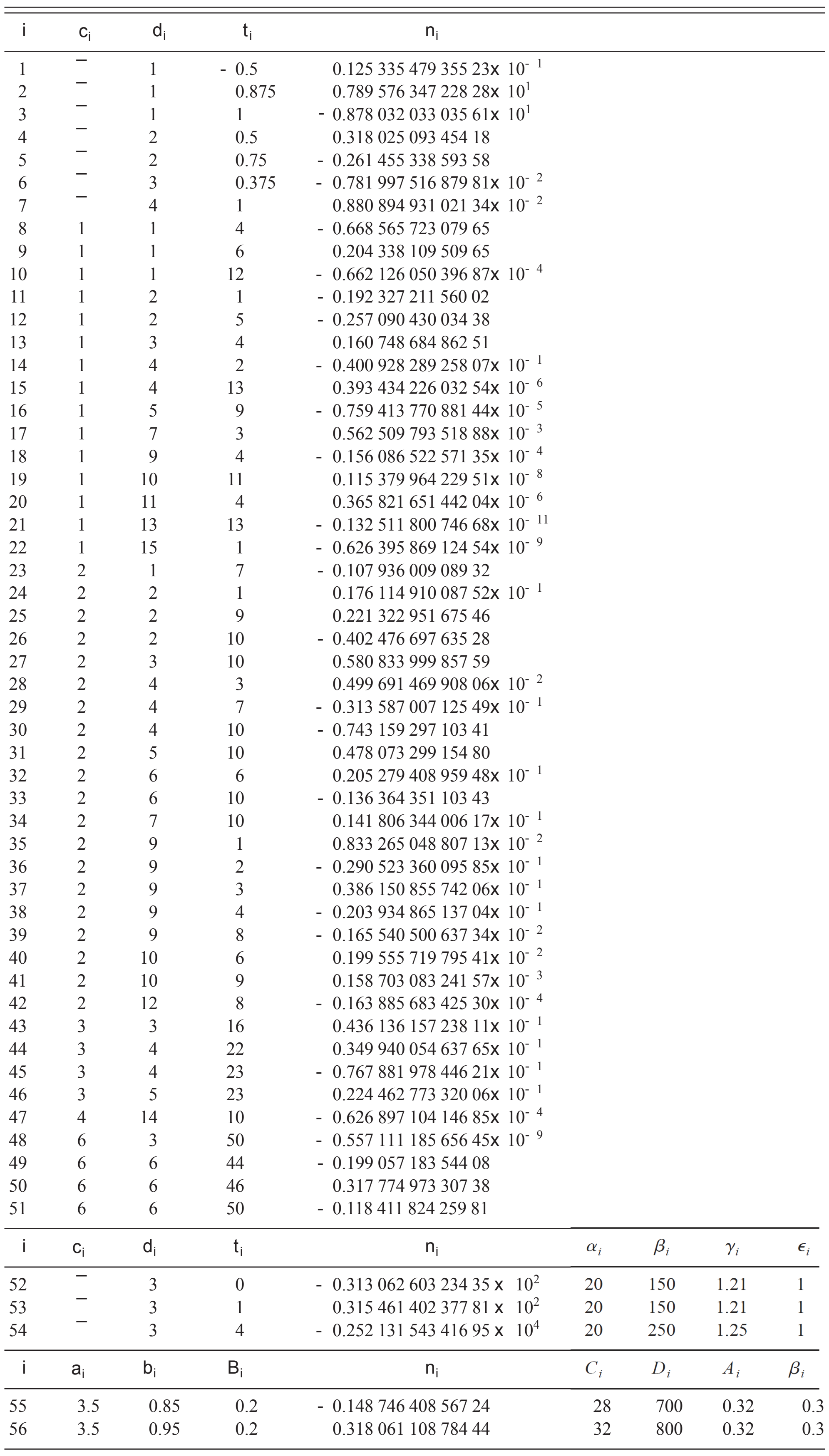
$$V_{m,x} = \frac{\partial V_m}{\partial x} = -2(1-x)\Big(\frac{1}{\rho_{c1}}\Big) + 2x\Big(\frac{1}{\rho_{c2}}\Big) + 2(1-x^\beta)\Big(\frac{1}{\rho_{c12}}\Big) - 2\beta(1-x^\beta)\Big(\frac{1}{\rho_{c12}}\Big) \tag{A.33}$$

$$\delta_x = \frac{\partial \delta}{\partial x} = \frac{\rho V_{m,x} * M - \rho V_m M_x}{M^2} \tag{A.34}$$

$$\tau_x = \frac{\partial \tau}{\partial x} = \frac{T_{n,x}}{T} \tag{A.35}$$

Appendix B

Coefficients of the Helmholtz function of water



i	c_i	d_i	t_i	n_i
1	–	1	-0.5	$0.125\,335\,479\,355\,23 \times 10^{-1}$
2	–	1	0.875	$0.789\,576\,347\,228\,28 \times 10^{1}$
3	–	1	1	$-0.878\,032\,033\,035\,61 \times 10^{1}$
4	–	2	0.5	$0.318\,025\,093\,454\,18$
5	–	2	0.75	$-0.261\,455\,338\,593\,58$
6	–	3	0.375	$-0.781\,997\,516\,879\,81 \times 10^{-2}$
7	–	4	1	$0.880\,894\,931\,021\,34 \times 10^{-2}$
8	1	1	4	$-0.668\,565\,723\,079\,65$
9	1	1	6	$0.204\,338\,109\,509\,65$
10	1	1	12	$-0.662\,126\,050\,396\,87 \times 10^{-4}$
11	1	2	1	$-0.192\,327\,211\,560\,02$
12	1	2	5	$-0.257\,090\,430\,034\,38$
13	1	3	4	$0.160\,748\,684\,862\,51$
14	1	4	2	$-0.400\,928\,289\,258\,07 \times 10^{-1}$
15	1	4	13	$0.393\,434\,226\,032\,54 \times 10^{-6}$
16	1	5	9	$-0.759\,413\,770\,881\,44 \times 10^{-5}$
17	1	7	3	$0.562\,509\,793\,518\,88 \times 10^{-3}$
18	1	9	4	$-0.156\,086\,522\,571\,35 \times 10^{-4}$
19	1	10	11	$0.115\,379\,964\,229\,51 \times 10^{-8}$
20	1	11	4	$0.365\,821\,651\,442\,04 \times 10^{-6}$
21	1	13	13	$-0.132\,511\,800\,746\,68 \times 10^{-11}$
22	1	15	1	$-0.626\,395\,869\,124\,54 \times 10^{-9}$
23	2	1	7	$-0.107\,936\,009\,089\,32$
24	2	2	1	$0.176\,114\,910\,087\,52 \times 10^{-1}$
25	2	2	9	$0.221\,322\,951\,675\,46$
26	2	2	10	$-0.402\,476\,697\,635\,28$
27	2	3	10	$0.580\,833\,999\,857\,59$
28	2	4	3	$0.499\,691\,469\,908\,06 \times 10^{-2}$
29	2	4	7	$-0.313\,587\,007\,125\,49 \times 10^{-1}$
30	2	4	10	$-0.743\,159\,297\,103\,41$
31	2	5	10	$0.478\,073\,299\,154\,80$
32	2	6	6	$0.205\,279\,408\,959\,48 \times 10^{-1}$
33	2	6	10	$-0.136\,364\,351\,103\,43$
34	2	7	10	$0.141\,806\,344\,006\,17 \times 10^{-1}$
35	2	9	1	$0.833\,265\,048\,807\,13 \times 10^{-2}$
36	2	9	2	$-0.290\,523\,360\,095\,85 \times 10^{-1}$
37	2	9	3	$0.386\,150\,855\,742\,06 \times 10^{-1}$
38	2	9	4	$-0.203\,934\,865\,137\,04 \times 10^{-1}$
39	2	9	8	$-0.165\,540\,500\,637\,34 \times 10^{-2}$
40	2	10	6	$0.199\,555\,719\,795\,41 \times 10^{-2}$
41	2	10	9	$0.158\,703\,083\,241\,57 \times 10^{-3}$
42	2	12	8	$-0.163\,885\,683\,425\,30 \times 10^{-4}$
43	3	3	16	$0.436\,136\,157\,238\,11 \times 10^{-1}$
44	3	4	22	$0.349\,940\,054\,637\,65 \times 10^{-1}$
45	3	4	23	$-0.767\,881\,978\,446\,21 \times 10^{-1}$
46	3	5	23	$0.224\,462\,773\,320\,06 \times 10^{-1}$
47	4	14	10	$-0.626\,897\,104\,146\,85 \times 10^{-4}$
48	6	3	50	$-0.557\,111\,185\,656\,45 \times 10^{-9}$
49	6	6	44	$-0.199\,057\,183\,544\,08$
50	6	6	46	$0.317\,774\,973\,307\,38$
51	6	6	50	$-0.118\,411\,824\,259\,81$

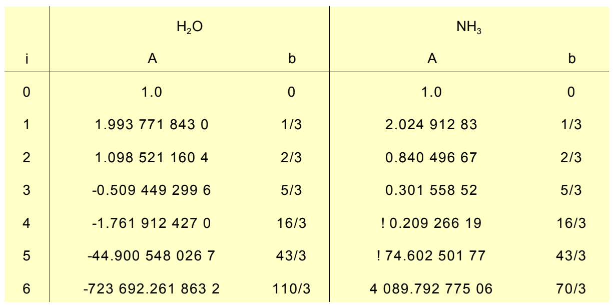
i	c_i	d_i	t_i	n_i	α_i	β_i	γ_i	ϵ_i
52	–	3	0	$-0.313\,062\,603\,234\,35 \times 10^{2}$	20	150	1.21	1
53	–	3	1	$0.315\,461\,402\,377\,81 \times 10^{2}$	20	150	1.21	1
54	–	3	4	$-0.252\,131\,543\,416\,95 \times 10^{4}$	20	250	1.25	1

i	a_i	b_i	B_i	n_i	C_i	D_i	A_i	β_i
55	3.5	0.85	0.2	$-0.148\,746\,408\,567\,24$	28	700	0.32	0.3
56	3.5	0.95	0.2	$0.318\,061\,108\,784\,44$	32	800	0.32	0.3

Figure B.1: Coefficients and exponents of eg.(5.13) [20]

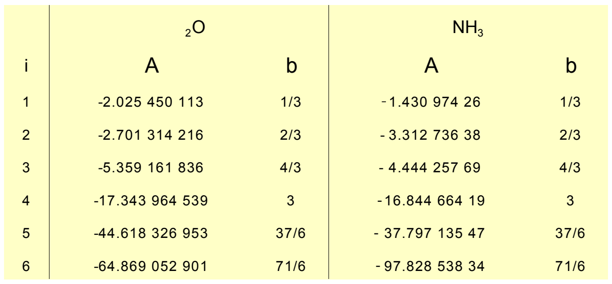
Appendix C

Coefficients of the equations for initial calculations



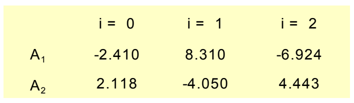
	H_2O		NH_3	
i	A	b	A	b
0	1.0	0	1.0	0
1	1.993 771 843 0	1/3	2.024 912 83	1/3
2	1.098 521 160 4	2/3	0.840 496 67	2/3
3	-0.509 449 299 6	5/3	0.301 558 52	5/3
4	-1.761 912 427 0	16/3	! 0.209 266 19	16/3
5	-44.900 548 026 7	43/3	! 74.602 501 77	43/3
6	-723 692.261 863 2	110/3	4 089.792 775 06	70/3

(a) Coefficients of eq.(5.66)



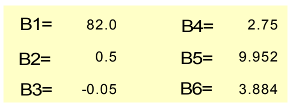
	$_2O$		NH_3	
i	A	b	A	b
1	-2.025 450 113	1/3	- 1.430 974 26	1/3
2	-2.701 314 216	2/3	- 3.312 736 38	2/3
3	-5.359 161 836	4/3	- 4.444 257 69	4/3
4	-17.343 964 539	3	- 16.844 664 19	3
5	-44.618 326 953	37/6	- 37.797 135 47	37/6
6	-64.869 052 901	71/6	- 97.828 538 34	71/6

(b) Coefficients of eq.(5.69)



	i = 0	i = 1	i = 2
A_1	-2.410	8.310	-6.924
A_2	2.118	-4.050	4.443

(c) Coeffs of eq.(5.65)



B1=	82.0	B4=	2.75
B2=	0.5	B5=	9.952
B3=	-0.05	B6=	3.884

(d) Coeffs of eq.(5.68)

Figure C.1: Coefficients for the initial approximation of density, given by [1]

i	m_i	n_i	a_i
1	0	0	$+ 0.322\ 302 \times 10^{1}$
2	0	1	$-0.384\ 206 \times 10^{0}$
3	0	2	$+ 0.460\ 965 \times 10^{-1}$
4	0	3	$-0.,378\ 945 \times 10^{-2}$
5	0	4	$+ 0.135\ 610 \times 10^{-3}$
6	1	0	$+ 0.487\ 755 \times 10^{0}$
7	1	1	$-0.120\ 108 \times 10^{0}$
8	1	2	$+ 0.106\ 154 \times 10^{-1}$
9	2	3	$-0.533\ 589 \times 10^{-3}$
10	4	0	$+ 0.785\ 041 \times 10^{1}$
11	5	0	$-0.115\ 941 \times 10^{2}$
12	5	1	$-0.523\ 150 \times 10^{-1}$
13	6	0	$+ 0.489\ 596 \times 10^{1}$
14	13	1	$+ 0.421\ 059 \times 10^{-1}$
	T_0 = 100 K		p_0 = 2 MPa

(a) Coefficients of eq.(5.58)

i	m_i	n_i	a_i
1	0	0	$+ 0.324\ 004 \times 10^{1}$
2	0	1	$-0.395\ 920 \times 10^{0}$
3	0	2	$+ 0.435\ 624 \times 10^{-1}$
4	0	3	$-0.218\ 943 \times 10^{-2}$
5	1	0	$-0.143\ 526 \times 10^{1}$
6	1	1	$+ 0.105\ 256 \times 10^{1}$
7	1	2	$-0.719\ 281 \times 10^{-1}$
8	2	0	$+ 0.122\ 362 \times 10^{2}$
9	2	1	$-0.224\ 368 \times 10^{1}$
10	3	0	$-0.201\ 780 \times 10^{2}$
11	3	1	$+ 0.110\ 834 \times 10^{1}$
12	4	0	$+ 0.145\ 399 \times 10^{2}$
13	4	2	$+ 0.644\ 312 \times 10^{0}$
14	5	0	$-0.221\ 246 \times 10^{1}$
15	5	2	$-0.756\ 266 \times 10^{0}$
16	6	0	$-0.135\ 529 \times 10^{1}$
17	7	2	$+ 0.183\ 541 \times 10^{0}$
	T_0 = 100 K		p_0

(b) Coefficients of eq.(5.59)

	i	n_i	a_i
1	0	0	$+ 1.980\ 220\ 17 \times 10^{1}$
2	0	1	$-1.180\ 926\ 69 \times 10^{1}$
3	0	6	$+ 2.774\ 799\ 80 \times 10^{1}$
4	0	7	$-2.886\ 342\ 77 \times 10^{1}$
5	1	0	$-5.916\ 166\ 08 \times 10^{1}$
6	2	1	$+ 5.780\ 913\ 05 \times 10^{2}$
7	2	2	$-6.217\ 367\ 43 \times 10^{0}$
8	3	2	$-3.421\ 984\ 02 \times 10^{3}$
9	4	3	$+ 1.194\ 031\ 27 \times 10^{4}$
10	5	4	$-2.454\ 137\ 77 \times 10^{4}$
11	6	5	$+ 2.915\ 918\ 65 \times 10^{4}$
12	7	6	$-1.847\ 822\ 90 \times 10^{4}$
13	7	7	$+ 2.348\ 194\ 34 \times 10^{1}$
14	8	7	$+ 4.803\ 106\ 17 \times 10^{3}$
	p_0		

(c) Coefficients of eq.(5.60)

	i	n_i	a_i
1	0	1	$-0.761\ 080 \times 10^{1}$
2	0	4	$+ 0.256\ 905 \times 10^{2}$
3	0	8	$-0.247\ 092 \times 10^{3}$
4	0	9	$+ 0.325\ 952 \times 10^{3}$
5	0	12	$-0.158\ 854 \times 10^{3}$
6	0	14	$+ 0.619\ 084 \times 10^{2}$
7	1	0	$+ 0.114\ 314 \times 10^{2}$
8	1	1	$+ 0.118\ 157 \times 10^{1}$
9	2	1	$+ 0.284\ 179 \times 10^{1}$
10	3	3	$+ 0.741\ 609 \times 10^{1}$
11	5	3	$+ 0.891\ 844 \times 10^{3}$
12	5	4	$-0.161\ 309 \times 10^{4}$
13	5	5	$+ 0.622\ 106 \times 10^{3}$
14	6	2	$-0.207\ 588 \times 10^{3}$
15	6	4	$-0.687\ 393 \times 10^{1}$
16	8	0	$+ 0.350\ 716 \times 10^{1}$
	h_0 = 100 kJ/kg		T_0 = 273.16 K

(d) Coefficients of eq.(5.61)

i	m_i	n_i	a_i
1	0	0	$+ 0.128\ 827 \times 10^{1}$
2	1	0	$+ 0.125\ 247 \times 10^{0}$
3	2	0	$-0.208\ 748 \times 10^{1}$
4	3	0	$+ 0.217\ 696 \times 10^{1}$
5	0	2	$+ 0.235\ 687 \times 10^{1}$
6	1	2	$-0.886\ 987 \times 10^{1}$
7	2	2	$+ 0.102\ 635 \times 10^{2}$
8	3	2	$-0.237\ 440 \times 10^{1}$
9	0	3	$-0.670\ 155 \times 10^{1}$
10	1	3	$+ 0.164\ 508 \times 10^{2}$
11	2	3	$-0.936\ 849 \times 10^{1}$
12	0	4	$+ 0.842\ 254 \times 10^{1}$
13	1	4	$-0.858\ 807 \times 10^{1}$
14	0	5	$-0.277\ 049 \times 10^{1}$
15	4	6	$-0.961\ 248 \times 10^{0}$
16	2	7	$+ 0.988\ 009 \times 10^{0}$
17	1	10	$+ 0.308\ 482 \times 10^{0}$
	h_0 = 1000 kJ/kg		T_0 = 324 K

(e) Coefficients of eq.(5.77)

i	a_i	b_i
0	647.14	220.64
1	-199.822 371	-37.923 795
2	109.035 522	36.424 739
3	-239.626 217	-41.851 597
4	88.689 691	-63.805 617

(f) Coefficients of eq.(5.77)

Figure C.2: Coefficients for the equations of 'Patek & Klomfar' (5.5.1), given by [1]