Semester Long Internship

On

Development of Unit Operations and Binary Phase Diagrams

Submitted by

Aniruddh Mukunth

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

August 23, 2021

Acknowledgment

I wish to express my deepest gratitude to my internship guide Dr. Kannan M. Moudgalya ,Professor, Department of Chemical Engineering IIT Bombay for his support, guidance and supervision throughout this internship. I am forever grateful for providing me with this opportunity to learn develop and grow in writing and analyzing codes.
I would like to thank my mentors Priyam Nayak and Malapati Sree Harsha Department of Chemical Engineering, IIT Bombay and the FOSSEE team for their support and guidance.
I would also like to thank my mentor Dr. P.R.Naren, Associate Professor, SASTRA University for encouraging and guiding me throughout the process and helped me in realizing my passion for coding.

Contents

Chapter 1

Introduction

Open Modelica is an open-source software which is based on the Modelica Language. It has various tools like OpenModelica Complier(OMC),Open Modelica Connection Editor (OMEdit), Open Modelica Shell(OMShell) and OpenModelica Python.Open Modelica is used for modelling, simulating, optimizing and analyzing steady state and dynamic systems. Open Modelica allows the users to write equations for a particular system and the solver solves the equations provided the number of equations is equal to the number of unknown variable. Open Modelica compiles the equations and functions into C code. The C code is combined with utility functions, run time library and a numerical Differential Algebraic Equation solver. OMEdit consists of several libraries for various domains like Electrical, Magnetic, Math, Thermal, etc. It provides various user friendly features like representation of a model in the form of block diagram. The Chemical process simulator(OMChemSim) which is being developed by FOSSEE Team at IIT Bombay provides an alternative to the various commercial software like ASPEN. OMChem has various unit operations and thermodynamic packages which makes it a very efficient tool for analysis.

Chapter 2

Distillation Column

2.1 Introduction

Distillation column is an unit operation which is used to physically separate a mixture into two or more fractions based on volatility of the components. The distillation column consist of trays for contacting vapour and liquid stream for effective mass transfer between the two phases. The vapour leaving from the top plate enters a condenser. The condenser can be a partial or total depending on the users preference. The condensate is collected in an accumulator and is separated into two streams L(Reflux) and D(Distillate). The amount of liquid reflux depends on the Reflux Ratio(RR) or L/D ratio. The liquid that leaves from the bottom tray enters the reboiler .In the reboiler the liquid is partially vapourised and the vapour is allowed to flow back into the distillation column and the liquid is withdrawn from the reboiler which is called as the bottom product(B).

The existing models for Distillation column provided in OMChem requires the user to write separate models to use condenser, reboiler, trays and distillation column. Each of the model is accompanied with thermodynamic package. In the existing model a rigorous procedure was employed using energy balances. This particular method made it difficult for the solver to solve the equation and would often result in convergence error. This is also made the model to be time consuming and less efficient. The primary objective of this work is to reduce the number of equations and improve the efficiency of the model.The distillation column that is build can be used for thermodynamic packages such as Raoults Law and NRTL.

2.2 Modelling

The distillation column analysis is being done by Lewis Matheson Method. In this method to compute the composition and temperature of each stage in the distillation column it is necessary to obtain a solution to the following equations:
1. Equilibrium Relationships
2. Component Material Balances
3. Total Material Balances

The analysis of distillation column starts from the condenser and proceeds till the reboiler. The analysis is as follows:

At the Condenser:

The Lewis Matheson method by starts by assuming guess values for the distillate composition $x_{Di,guess}$ where i=1,2,3,...,Nc.

The vapour leaving from stage 1 has the same composition as the distillate $x_{Di,guess} = y_{1i}$ where i=1,2,3,...,Nc.

The liquid leaving from the condenser has the same composition as the distillate $x_{0i} = x_{Di,guess}$.

The vapour flow rate in mol/s leaving from the stage 1 is calculated by

$$V_1 = D + L_0$$

The reflux in mol/s that is entering the column is calculated by

$$L_0 = RR * D$$

The temperature at the condenser is determined using the bubble point equation :

$$\sum_{1}^{Nc} x_{Di,guess} * K_{Di} = 1$$

Where
K_{Di} depends on which thermodynamic package is chosen. In general it is given by

$$K_{Di} = \frac{y_{Di}}{x_{Di}} = \frac{\phi_{Di,L}}{\phi_{Di,V}}$$

Nc is the number of components.
D is the distillate flow rate in mol/s.
RR is the reflux ratio.
V_1 is the vapour flow rate in mol/s leaving from the stage 1.
L_0 is the reflux in mol/s entering the column.
$x_{Di,guess}$ is the guess composition value of distillate.
x_{0i} is the reflux composition.
y_{1i} is the composition of the stream leaving from stage 1.

Before proceeding with further analysis the vapour flow rates and liquid flow rates can be determined by using
Total Material Balance :
Across the entire column

$$\sum_{k=1}^{Ni} F_k = D + B + \sum_{k=1}^{Nout} S_k$$

Across each stage

$$f_j + V_{j+1} + L_{j-1} = V_j + L_j + s_j$$

j=1,2,3,....,Nt Where

F is the feed flow rate in mol/s
D is the distillate flow rate in mol/s
B is the bottoms flow rate in mol/s
S is the side flow rates in mol/s.
Nt is the number of stages in the distillation column
Ni is the number of feed Streams
Nout is the number of side streams.
f_j is the feed flow rate in mol/s that enters a particular stage.
V_{j+1} is the vapour flow rate in mol/s that enters a particular stage.
L_{j-1} is the liquid flow rate in mol/s that enters a particular stage.
V_j is the vapour flow rate in mol/s that leaves a particular stage.
L_j is the liquid flow rate in mol/s that leaves a particular stage.
s_j is the side flow rate in mol/s that leaves a particular stage.

In Lewis Matheson method the following assumption is valid that is equimolar flow of vapour

$$V_j + V_{fj} = V_{j+1}$$

where j =1,2,3,. . . ,Nt-1
Where V_f is the vapour present in feed that enters the particular stage.

From the above the vapour flow rate from each stage is found out and the liquid flow rate from each stage is calculated by using total material balance for each stage.

The following procedure is used to calculate from stage 1 to reboiler

The temperature for a particular stage is being determined from dew point temperature formula.

$$\sum_{i=1}^{Nc} x_{ji} = 1$$

The equilibrium relation for each stage is given as follows:

$$y_{ji} = K_{ji} x_{ji}$$

Where
j=1,2,3,...,Nt and i=1,2,3,..,Nc.
y_{ji} is the vapour mole fraction of a particular component at a particular stage
x_{ji} is the liquid mole fraction of a particular component at a particular stage
K_{ji} depends on which thermodynamic package is chosen. In general it is given by

$$K_{ji} = \frac{y_{ji}}{x_{ji}} = \frac{\phi_{ji,L}}{\phi_{ji,V)}}$$

The composition of the entering vapour stream can be found out using the component material balance of each stage.

$$f_j z_j + V_{j+1} y_{j+1,i} + L_{j-1} x_{j-1,i} = V_j y_{ji} + L_j x_{ji} + s_j w_{ji}$$

Where
j=1,2,3,. . . ,Nt and i=1,2,3,. . . ,Nc
z_j is the composition of the feed that is entering a particular stage
$y_{j+1,i}$ is the vapour composition that is entering a particular stage.
$x_{j-1,i}$ is the liquid composition that is entering a particular stage.
y_{ji} is the vapour composition that is leaving a particular stage.
x_{ji} is the liquid composition that is leaving a particular stage.
w_{ji} is the composition of the side stream that is leaving a particular stage.

Since an initial guess for the distillate composition is taken the condition to obtain the correct composition is given as follows:

The bottoms composition is calculated by using the guessed distillate composition with the help of overall component material balance

$$\sum_{k=1}^{Ni} F_k z_{ki} = D x_{Di} + B x_{obtBi} + \sum_{j=1}^{Nout} S_j w_{ji}$$

where i=(1,2,3,. . . ,Nc-1)

The condition is imposed as follows where the difference between the calculated value from the analysis and the obtained value from the total component material balance should be 0.

The condenser and the reboiler duty is found as follows by energy balances:
Condenser duty is found by energy balance across the condenser

$$V_1 H_1 = L_1 h_1 + D H_D + Q_C$$

Reboiler duty is found by energy balance across the entire column

$$\sum_{k=1}^{Ni} F_k H_{fk} + Q_R = D H_D + B H_B + \sum_{j=1}^{Nout} S_j H_{sj}$$

Where
Q_C is the condenser duty in W.
Q_R is the condenser duty in W.
H_1 is the enthalpy of vapour in J/mol leaving the 1st stage.
h_1 is the enthalpy of liquid in J/mol leaving the condenser.
H_D is the enthalpy of the distillate in J/mol.
H_f is the enthalpy of the feed in J/mol.
H_B is the enthalpy of the bottoms in J/mol.
H_s is the enthalpy of the side stream in J/mol.
The enthalpy per mole of vapour and liquid leaving a particular stage can be expressed as

$$H_j = \sum_{i=1}^{Nc} H_{ji} y_{ji}$$

$$h_j = \sum_{i=1}^{Nc} h_{ji} x_{ji}$$

Where
j= 1,2,3,.....,Nt and i = 1,2,3,...,Nc
H is the enthalpy of vapour in J/mol.
h is the enthalpy of liquid in J/mol.
Since we have made an assumption of the vapour flow rate. The correct flow rates can be found by using the energy balances and the total material balance across each stage.The energy balance for each stage is given by

$$f_j H_{fj} + V_{j+1} H_{j+1} + L_{j-1} h_{j-1,i} = V_j H_{ji} + L_j h_{ji} + s_j H_{sj}$$

Where j=1,2,3,.....,Nt
H_f is the enthalpy of feed in J/mol.
H_{j+1} is the enthalpy of vapour entering a particular stage in J/mol.
h_{j-1} is the enthalpy of liquid entering a particular stage in J/mol.
H_j is the enthalpy of vapour leaving a particular stage in J/mol.
h_j is the enthalpy of liquid leaving a particular stage in J/mol.
h_{sj} is the enthalpy of side stream leaving a particular stage in J/mol.

2.3 Results

The following example was simulated and the results were compared with DWSIM.
System: Benzene and Toluene

The distillation column parameters and the feed conditions is given below

Condenser Pressure (Pa)	101325
Distillate(mol/s)	50
Feed stage location	3
Number of inlet	1
Number of side streams	0
Number of stages	6
Reflux Ratio RR	2
Reboiler Pressure (Pa)	101325
Guess Temperature Tguess(K)	360

Table 2.1: Distillation column parameters

Temperature(K)	298.15
Pressure(Pa)	101325
Flow Rate(mol/s)	100
Composition	
Benzene	0.5
Toluene	0.5

Table 2.2: Feed Conditions

The following results were obtained after simulation

Stages	x(-) Benzene	x(-) Toluene	y(-) Benzene	y(-) Toluene
Condenser	0.838931004	0.161068996	0.930988105	0.069011895
Stage 1	0.671505378	0.328494622	0.838931004	0.161068996
Stage 2	0.515678254	0.484321746	0.72731392	0.27268608
Stage 3	0.401205332	0.598794668	0.623429171	0.376570829
Stage 4	0.276230784	0.723769216	0.481250777	0.518749223
Reboiller	0.161068996	0.838931004	0.314618047	0.685381953

Table 2.3: Component Fractions in each stage

Stages	OM	DWSIM	Error percentage
	T(K)	T(K)	
Condenser	356.7318411	356.661	0.019862314
Stage 1	360.7146698	360.584	0.036238391
Stage 2	364.902247	364.713	0.051889304
Stage 3	368.3318454	368.067	0.07195576
Stage 4	372.4806425	372.494	0.003585974
Reboiler	376.7407085	376.866	0.033245639

Table 2.4: Temperature in each stage

Stages	Correct L(mol/s)	Correct V(mol/s)
Condenser	0	100
Stage 1	150	97.67614065
Stage 2	147.6761407	96.04156743
Stage 3	146.0415674	227.5145322
Stage 4	177.5145322	226.0838528
Reboiler	176.0838528	50

Table 2.5: Correct flow rates

The obtained condenser and reboiler duty is

	OM	DWSIM	Error Percentage
Condenser Duty Qc(W)	4748530	4744980	0.074815911
Reboiler Duty Qr(W)	-5865240	-5862210	0.051686992

Table 2.6: Condenser And Reboiler Duty

Chapter 3

Binary Phase Diagrams using Peng Robinson

3.1 Introduction

The Peng Robinson equation of state was developed by Ding-Yu-Peng and Donald Robinson in 1976.The Peng Robinson provides good accuracy near the critical point and for liquid molar volumes.It can be used to predict vapour liquid equilibria with good accuracy when combined with appropriate mixing rules.

3.2 Modelling

Peng Robinson (PR) Equation of state is given by

$$P = \frac{RT}{V-b} - \frac{a(T)}{V(V+b)+b(V-b)}$$

$$\alpha_i = [1 + \kappa_i * (1 - \sqrt{\frac{T}{T_c}})]^2$$

$$a_i = \frac{0.45724 \alpha R^2 T_c^2}{P_c}$$

$$b_i = \frac{0.0778 R T_c}{Pc}$$

$$\kappa_i = 0.37464 + 1.54226\omega - 0.26992\omega^2$$

For mixture consisting more than one component

$$a_{ij} = (1 - k_{ij}\sqrt{a_i a_j})$$

$$b = \sum_{i=1}^{n} x_i b_i$$

$$a = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j a_{ij}$$

Peng Robinson Equation of State based on compressibility factor:

$$Z^3 - (1-B)Z^2 + (A - 3B^2 - 2B)Z - (AB - B^2 - B^3) = 0$$

$$A = \frac{aP}{R^2T^2}$$

$$B = \frac{bP}{RT}$$

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

The fugacity coefficient is calculated as follows:

$$\ln \phi_i = \frac{b_i(1-Z)}{b} - \ln(Z-B) - \frac{A}{2B\sqrt{2}} \ln(\frac{Z + 2.414B}{Z - 0.414B})(\frac{2\sum_{j=1}^{n} x_j a_{ji}}{a} - \frac{b_i}{b})$$

The equilibrium relations is given as follows

$$K_i = \frac{\phi_{i,L}}{\phi_{i,V}}$$

$$y_i = K_i x_i \qquad (i = 1, 2, 3. . . ..., Nc)$$

For Pxy:
Pxy is a binary phase diagram where the temperature is held constant. For a particular liquid phase mole fraction the pressure and vapour phase mole fraction is obtained by equilibrium equations.
Conditions for determining the pressure and vapour composition is

$$\sum_{i=1}^{Nc} y_i = 1$$

Initial pressure guess values is given by calculating the vapour pressure of the pure components at the specified temperature.
For Txy:
Txy is a binary phase diagram where the pressure is held constant. For a particular liquid phase mole fraction the temperature and vapour phase mole fraction is obtained by equilibrium equations.
Conditions for determining the temperature and vapour composition is

$$\sum_{i=1}^{Nc} y_i = 1$$

Initial temperature guess values is given by calculating the boiling point of the pure components at the specified pressure.
Nomenclature:
P is the Pressure (Pa)
V is the molar volume in m^3/mol
R is the gas constant 8.314 $Pa.m^3/mol.K$
T is the temperature (K)
P_c is the critical pressure for a particular component(Pa).
T_c is the critical pressure for a particular component(K).
ω_i is the acentric factor a particular component.
a is the Peng Robinson attraction parameter for the mixture.
a_i is the Peng Robinson attraction parameter for a paticular component.
b is the Peng Robinson co-volume for the mixture.
b_i is the Peng Robinson co-volume for a particular component.
k_{ij} is the Peng Robinson binary interaction parameter.
Z is the compressibility factor
ϕ_i is the fugacity coefficient for a particular component.
a_{ij} is the cross Peng Robinson attraction parameter for components i and j.

3.3 Results

Txy Results:
System: Ethane and Propane
Pressure(Pa): 101325
Number of Points : 6
The result obtained from the code is as follows:

x(-)(Ethane)	T(K) Temperature	y (-) Ethane
0	230.9289515	0
0.2	211.6561003	0.675565119
0.4	200.673366	0.867060687
0.6	193.5330955	0.942473334
0.8	188.3975618	0.979300095
1	184.4292755	1

Table 3.1: Txy Result for Ethane,Propane System

Pxy Results:
System: Propane and N-Butane
Temperature(K): 323
Number of Points : 6
The result obtained from the code is as follows:

x(-)(Propane)	P(Pa) Pressure	y (-) Propane
0	494268.6284	0
0.2	714942.7833	0.411429053
0.4	943508.0564	0.641867256
0.6	1182673.26	0.794098292
0.8	1436924.421	0.907112012
1	1714147.699	1

Table 3.2: Pxy Result for Propane, N-Butane System

The Txy and Pxy diagram for this particular system is given below in Fig 3.1 and 3.2 respectively.

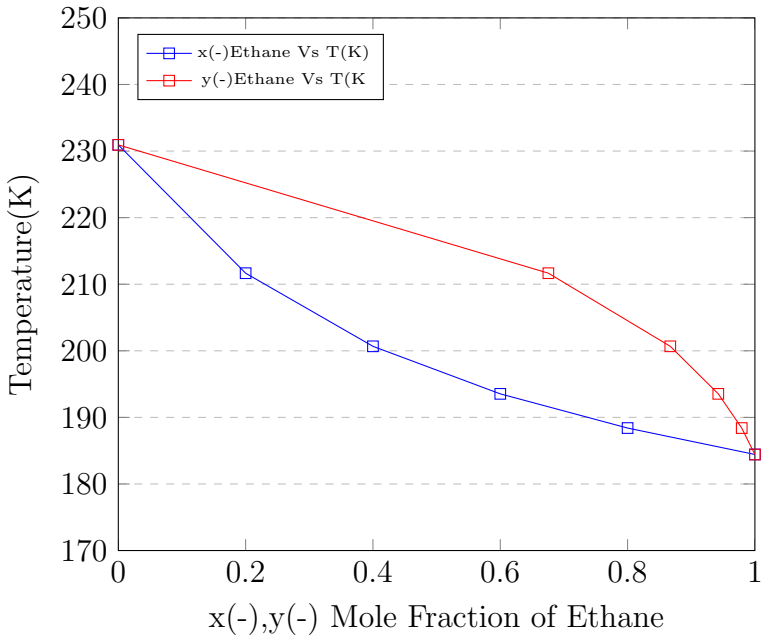

Figure 3.1: Txy Diagram of Ethane Propane System

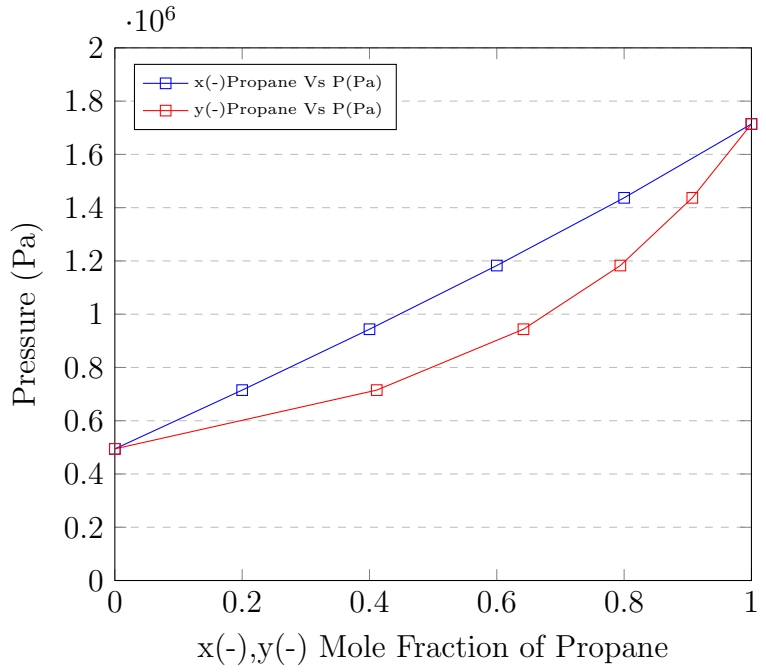

Figure 3.2: Pxy Diagram of Propane N-Butane System

Chapter 4

Equilibrium constant from Gibbs Energy of reaction

4.1 Introduction

The equilibrium reactor is a reactor for which equilibrium reactions can be specified. There are various modes to calculate and specify the equilibrium constant based on which the conversion is being calculated. The various modes are Gibbs energy of reaction, specifying the value directly or by specifying the coefficients of the equation if the equilibrium constant is expressed in terms of function of temperature. The primary objective of this work is to perform the calculation of equilibrium constant from the Gibbs energy of reaction.

4.2 Modelling

The equilibrium constant can be calculated as follows:
The standard state of enthalpy entropy and Gibbs free energy at 298 K is calculated as follows

$$\Delta H_R^0 = \sum_{i=1}^{n} \vartheta_i * \Delta H_{Fi}^0$$

$$\Delta S_R^0 = \sum_{i=1}^{n} \vartheta_i * \Delta S_{Fi}^0$$

$$\Delta G_R^0 = \sum_{i=1}^{n} \vartheta_i * \Delta G_{Fi}^0$$

Where
ϑ_i is the reaction stoichiometry

ΔH_{Fi}^0 is the enthalpy of formation

ΔS^0_{Fi} is the entropy of formation

ΔG^0_{Fi} is the Gibbs energy of formation

n is the number of components.
Enthalpy,Entropy and Gibbs Free Energy at Reaction Conditions:

$$\Delta G_R = \Delta H_R - T * \Delta S_R$$

Where

$$\Delta H_R(T) = \Delta H^0_R + \int_{Tref}^{T} \Delta Cp dT$$
$$\Delta S_R(T) = \Delta S^0_R + \int_{Tref}^{T} \frac{\Delta Cp}{T} dT$$

For the given relations the specific capacity in J/mol.K is given by

$$Cp_i = aT^3 + bT^2 + cT + d$$

The ΔCp is given by

$$\Delta Cp = \Delta a * T^3 + \Delta b * T^2 + \Delta c * T + \Delta d$$

where

$$\Delta a = \sum_{i=1}^{n} \vartheta_i a_i$$

$$\Delta b = \sum_{i=1}^{n} \vartheta_i b_i$$

$$\Delta c = \sum_{i=1}^{n} \vartheta_i c_i$$

$$\Delta d = \sum_{i=1}^{n} \vartheta_i d_i$$

The constants a,b,c,d are predefined for each of the component. The constants are obtained by regression of the data of liquid and vapour specific heat capacity from DWSIM by using python.

$$\int_{Tref}^{T} \Delta CpdT = \frac{\Delta a(T^4 - T_{ref}^4)}{4} + \frac{\Delta b(T^3 - T_{ref}^3)}{3} + \frac{\Delta c(T^2 - T_{ref}^2)}{2} + \Delta d(T - T_{ref})$$

$$\int_{Tref}^{T} \frac{\Delta Cp}{T} dT = \frac{\Delta a(T^3 - T_{ref}^3)}{3} + \frac{\Delta b(T^2 - T_{ref}^2)}{2} + \Delta c(T - T_{ref}) + \Delta d \ln \frac{T}{T_{ref}}$$

Where
T is the reaction temperature in K.
T_{ref} is the reference temperature which is 298.15 K.
Estimation of Equilibrium constant:

$$\Delta G_R = -RT \ln K$$

Where
R is the gas constant which is given by $8.315 J.m^3/mol.K$
T is the reaction temperature in K.
K is the equilibrium constant.
The equilibrium constant K_y for a particular reaction is given by

$$K = K_y * P^{\Delta n}$$

where $\Delta n = \sum_{i=1}^{n} \vartheta_i$

4.3 Results

The reaction is as follows:

$$2H_2 + CO \rightarrow CH_4OH$$

The parameters to the equilibrium reactor is provided as follows:
Basis : Activity
Mode : Isothermal
Phase : Vapour
Rmode : Gibbs
The feed conditions is provided below:

Parameter	Value
Temperature(K)	366.5
Pressure(Pa)	101325
Flow Rate(mol/s)	27.778
Composition	
Hydrogen	0.667
Carbon Monoxide	0.333
Methanol	0

Table 4.1: Feed Conditions

After Simulating the reactor the following results were obtained:

Parameter	DWSIM	OM	Error percentage
Gibbs Energy at 25C(J/mol)	-25170	-25170	0
Conversion			
Hydrogen	74.6635	74.8379	0.233581335
Carbon Monoxide	74.5516	74.9503	0.534797375

Table 4.2: Results

Reference

- Introduction to Chemical Engineering Thermodynamics by Joseph Mauk Smith, Hendrick C. Van Ness, Michael M. Abbott, M. T. Swihart
- Fundamentals of multicomponent distillation by C.D. Holland

Chapter 5

OpenModelica Code

```
1  package DistCol
2  model RaoultsLaw
3  //Model Description:
4
5  // The Distillation Column aims to seperate the feed mixture into two
       streams i.e. Distillate and Residue based on the Volatility of the
       components present in the stream.Multiple Feeds and Side Draws can also
        be introduced.
6
7  //Thermodynamics : Raoults Law
8  extends Simulator.Files.Icons.DistillationColumn;
9  //========================= User Input Data
       ==================================//
10
11     parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "
           Component instances array" annotation(
12     Dialog(tab = "Column Specifications", group = "Component Parameters"));
13     parameter Integer Nc "Number of components" annotation(
14     Dialog(tab = "Column Specifications", group = "Component Parameters"));
15     parameter Integer Nt = 4 "Number of stages" annotation(
16     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
17     parameter Integer Nout = 0 "Number of side draws" annotation(
18     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
19     parameter Integer Ni = 1 "Number of feed streams" annotation(
20     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
21     parameter Integer InT_s[Ni] "Feed stage location" annotation(
22     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
23     parameter Integer OutT_s[Nout] "Feed stage location" annotation(
24     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
25     parameter Real D(unit="mol/s") "Distillate flow rate" annotation(
26     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
27     parameter Real RR(unit="-") "Reflux Ratio" annotation(
28     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
29     parameter Real CondP(unit="Pa") "Condenser Pressure" annotation(
30     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
31     parameter Real ReboilerP(unit="Pa") "Reboiler Pressure" annotation(
32     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
```

```
33      parameter Real Tguess(unit="K") "Guess Temperature" annotation(
34      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
35      parameter Real SideF[Nout](each unit="mol/s")"Side stream flow rate"
            annotation(
36      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
37      parameter Real SidePhase1 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
38      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
39      parameter Real SidePhase2 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
40      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
41      parameter Real SidePhase3 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
42      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
43      parameter Real SidePhase4 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
44      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
45      parameter Real SidePhase5 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
46      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
47      parameter Real SidePhase6 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
48      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
49      parameter Real SidePhase7 = 0"Side stream Phase 0 for None 1 for Liquid
             and 2 for Vapour" annotation(
50      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
51      parameter String Ctype = "Total" "Condenser type: Total or Partial"
            annotation(
52      Dialog(tab = "Column Specifications", group = "Calculation Parameters")
            );
53
54
55  //======================== Distillation Column Variables
        ================================//
56
57  //======================== Connecter Variables
        ============================================//
58      Simulator.Files.Interfaces.matConn In[Ni](each Nc = Nc) annotation(
59        Placement(visible = true, transformation(origin = {-248, -40}, extent
              = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {-250, 0}, extent = {{-10, -10}, {10, 10}}, rotation =
            0)));
60      Simulator.Files.Interfaces.matConn Dist(each Nc = Nc) annotation(
61        Placement(visible = true, transformation(origin = {250, 316}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {250, 298}, extent = {{-10, -10}, {10, 10}}, rotation =
            0)));
62      Simulator.Files.Interfaces.matConn Bot(each Nc = Nc) annotation(
63        Placement(visible = true, transformation(origin = {250, -296}, extent
              = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {252, -300}, extent = {{-10, -10}, {10, 10}}, rotation =
             0)));
64      Simulator.Files.Interfaces.enConn Cduty annotation(
65        Placement(visible = true, transformation(origin = {246, 590}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
```

```
            origin = {250, 600}, extent = {{-10, -10}, {10, 10}}, rotation =
            0)));
66      Simulator.Files.Interfaces.enConn Rduty annotation(
67        Placement(visible = true, transformation(origin = {252, -588}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {250, -598}, extent = {{-10, -10}, {10, 10}}, rotation =
            0)));
68      Simulator.Files.Interfaces.matConn Out_s[Nout](each Nc = Nc) annotation
          (
69        Placement(visible = true, transformation(origin = {-36, 32}, extent =
            {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin
            = {-70, 60}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
70      //
          ============================================================================

71    extends Simulator.GuessModels.InitialGuess;
72      //============================Model Variables
          ============================================

73
74      Real Qc(unit="W")"Condenser Duty";
75      Real Qr(unit="W")"Reboiler Duty";
76      Real Hvapcond_c[Nc](each unit="J/mol")"Enthalpy of Components present
          in Vapour from Stage 1";
77      Real Hliqcond_c[Nc](each unit="J/mol")"Enthalpy of Components present
          in Liquid from Condenser";
78      Real Hvapcond(unit="J/mol")"Enthalpy of Vapour from Stage 1";
79      Real Hout1(unit="J/mol")"Enthalpy of the Distillate Stream";
80      Real Hout2(unit="J/mol")"Enthalpy of the Residue Stream";
81      Real Hin[Ni](unit="J/mol")"Enthalpy of the Feed Stream";
82      Real Hliqcond(unit="J/mol")"Enthalpy of Vapour from Stage 1";
83      Real Fin[Ni](each unit = "mol/s", each min = 0) "Inlet stream molar
          flow rate";
84      Real Pin[Ni](each unit = "Pa", each min = 0) "Inlet stream pressure";
85      Real Tin[Ni](each unit = "K", each min = 0) "Inlet stream emperature";
86      Real x_c[Ni,Nc](each unit = "-") "Component mole fraction";
87      Real Feed[Nt](each unit="mol/s")"Flow rate that enters the distillation
           column";
88      Real Fc[Nt,Nc](each unit="mol/s")"Component flow rate that enters the
          column";
89      Real B(unit="mol/s")"Bottom Flow Rate";
90      Real V[Nt+1](each unit="mol/s")"Vapour flow rate at each stage";
91      Real L[Nt](each unit="mol/s")"Liquid flow rate at each stage" ;
92      Real x[Nt,Nc](each unit="-",each start = 1/Nc)"Liquid Composition at
          each stage";
93      Real y[Nt+1,Nc](each unit="-",each start = 1/Nc)"Vapour Composition at
          each stage";
94      Real Pvap_c[Nt,Nc](each unit="Pa")"Saturation Pressure";
95      Real T[Nt](each unit = "K", each min = 0,  each start = Tguess)"
          Temperature at each stage";
96      Real K_c[Nt,Nc](each unit = "-", each min = 1, each start = 1.5);
97      Real P[Nt](each unit="Pa")"Pressure at each stage";
98      Real HSideOut[Nout](each unit="J/mol")"Enthalpy of the Side Streams";
99      Real HFeed[Nt](each unit="J/mol")"Enthalpy of the Feed";
100     Real HSide[Nt](each unit="J/mol")"Enthalpy of the Side Stream";
101     Real FSide[Nt](each unit="mol/s")"Side Stream Flow Rates";
102     Real correctL[Nt](each unit="mol/s")"Correct Liquid Flow Rates";
103     Real correctV[Nt](each unit="mol/s")"Correct Vapour Flow Rates";
104     Real Hvap[Nt,Nc](each unit="J/mol")"Enthalpy of Components present in
          Vapour of each stage";
105     Real Hliq[Nt,Nc](each unit="J/mol")"Enthalpy of Components present in
          Liquid of each stage";
106     Real F[Nt](each unit="mol/s")"Feed that enters the Distillation Column
          at each stage";
107     //
```

```
108       Real xdel[Nc-1];
109       Real a11,a12,a13,a14,a15,a16,a17,a18,a19,a10;
110       Real xvapin[Ni];
111       Real pervap[Nt];
112       Integer b11;
113       Integer b12,b13,b14,b15,b16,b17;
114     equation
115
116     //============================ Pressure at Each Stage
            ==========================================
117
118     // Pressure at each stage is calculated by interpolation.
119     P[1]=CondP;
120     P[Nt]=ReboilerP;
121     for i in 2:Nt-1 loop
122       if CondP == ReboilerP then
123         P[i]=CondP;
124       else
125         if ReboilerP > CondP then
126           P[i]=CondP + ((i - 1)/(Nt -1))*(ReboilerP -CondP);
127         else
128           P[i]=CondP + ((i - 1)/(Nt -1))*(CondP -ReboilerP);
129         end if;
130       end if;
131     end for;
132
133   //
          ==========================================================================
134
135   //====================Determing the Location of the Feed and the Side
          Stream====================
136     if 1==Ni then
137       a11=InT_s[1]+1;
138       a12=0;
139       a13=0;
140       a14=0;
141       a15=0;
142       a16=0;
143       a17=0;
144       a18=0;
145       a19=0;
146       a10=0;
147     elseif 2==Ni then
148       a11=InT_s[1]+1;
149       a12=InT_s[2]+1;
150       a13=0;
151       a14=0;
152       a15=0;
153       a16=0;
154       a17=0;
155       a18=0;
156       a19=0;
157       a10=0;
158     elseif 3==Ni then
159       a11=InT_s[1]+1;
160       a12=InT_s[2]+1;
161       a13=InT_s[3]+1;
162       a14=0;
163       a15=0;
164       a16=0;
165       a17=0;
```

```
166         a18=0;
167         a19=0;
168         a10=0;
169       elseif 4==Ni then
170         a11=InT_s[1]+1;
171         a12=InT_s[2]+1;
172         a13=InT_s[3]+1;
173         a14=InT_s[4]+1;
174         a15=0;
175         a16=0;
176         a17=0;
177         a18=0;
178         a19=0;
179         a10=0;
180       elseif 5==Ni then
181         a11=InT_s[1]+1;
182         a12=InT_s[2]+1;
183         a13=InT_s[3]+1;
184         a14=InT_s[4]+1;
185         a15=InT_s[5]+1;
186         a16=0;
187         a17=0;
188         a18=0;
189         a19=0;
190         a10=0;
191       elseif 6==Ni then
192         a11=InT_s[1]+1;
193         a12=InT_s[2]+1;
194         a13=InT_s[3]+1;
195         a14=InT_s[4]+1;
196         a15=InT_s[5]+1;
197         a16=InT_s[6]+1;
198         a17=0;
199         a18=0;
200         a19=0;
201         a10=0;
202       elseif 7==Ni then
203         a11=InT_s[1]+1;
204         a12=InT_s[2]+1;
205         a13=InT_s[3]+1;
206         a14=InT_s[4]+1;
207         a15=InT_s[5]+1;
208         a16=InT_s[6]+1;
209         a17=InT_s[7]+1;
210         a18=0;
211         a19=0;
212         a10=0;
213       elseif 8==Ni then
214         a11=InT_s[1]+1;
215         a12=InT_s[2]+1;
216         a13=InT_s[3]+1;
217         a14=InT_s[4]+1;
218         a15=InT_s[5]+1;
219         a16=InT_s[6]+1;
220         a17=InT_s[7]+1;
221         a18=InT_s[8]+1;
222         a19=0;
223         a10=0;
224       elseif 9==Ni then
225         a11=InT_s[1]+1;
226         a12=InT_s[2]+1;
227         a13=InT_s[3]+1;
228         a14=InT_s[4]+1;
229         a15=InT_s[5]+1;
```

```
230         a16=InT_s[6]+1;
231         a17=InT_s[7]+1;
232         a18=InT_s[8]+1;
233         a19=InT_s[9]+1;
234         a10=0;
235       else
236         a11=InT_s[1]+1;
237         a12=InT_s[2]+1;
238         a13=InT_s[3]+1;
239         a14=InT_s[4]+1;
240         a15=InT_s[5]+1;
241         a16=InT_s[6]+1;
242         a17=InT_s[7]+1;
243         a18=InT_s[8]+1;
244         a19=InT_s[9]+1;
245         a10=InT_s[10]+1;
246       end if;
247       if 1==Nout then
248         b11=OutT_s[1]+1;
249         b12=0;
250         b13=0;
251         b14=0;
252         b15=0;
253         b16=0;
254         b17=0;
255       elseif 2==Nout then
256         b11=OutT_s[1]+1;
257         b12=OutT_s[2]+1;
258         b13=0;
259         b14=0;
260         b15=0;
261         b16=0;
262         b17=0;
263       elseif 3==Nout then
264         b11=OutT_s[1]+1;
265         b12=OutT_s[2]+1;
266         b13=OutT_s[3]+1;
267         b14=0;
268         b15=0;
269         b16=0;
270         b17=0;
271       elseif 4==Nout then
272         b11=OutT_s[1]+1;
273         b12=OutT_s[2]+1;
274         b13=OutT_s[3]+1;
275         b14=OutT_s[4]+1;
276         b15=0;
277         b16=0;
278         b17=0;
279       elseif 5==Nout then
280         b11=OutT_s[1]+1;
281         b12=OutT_s[2]+1;
282         b13=OutT_s[3]+1;
283         b14=OutT_s[4]+1;
284         b15=OutT_s[5]+1;
285         b16=0;
286         b17=0;
287       elseif 6==Nout then
288         b11=OutT_s[1]+1;
289         b12=OutT_s[2]+1;
290         b13=OutT_s[3]+1;
291         b14=OutT_s[4]+1;
292         b16=OutT_s[6]+1;
293         b15=0;
```

```
294       b17=0;
295     elseif 7==Nout then
296       b11=OutT_s[1]+1;
297       b12=OutT_s[2]+1;
298       b13=OutT_s[3]+1;
299       b14=OutT_s[4]+1;
300       b15=OutT_s[5]+1;
301       b16=OutT_s[6]+1;
302       b17=OutT_s[7]+1;
303     else
304       b11=0;
305       b12=0;
306       b13=0;
307       b14=0;
308       b15=0;
309       b16=0;
310       b17=0;
311     end if;
312     //
          ==========================================================================

313     //====================Determining the Feed Flow Rate at each stage
          ================================
314     //If the Feed is enetring that paticular stage it will be assigned the
          inlet flow rate else it will be assigned the value zero. If a side
          stream is leaving from a paticular tray provided that the phase is
          liquid it will be assigned the value provided by the user.
315
316     for i in 1:Nt loop
317       if i==a11 then
318       Feed[i]=Fin[1];
319       elseif i==a12 then
320       Feed[i]=Fin[2];
321       elseif i==a13 then
322       Feed[i]=Fin[3];
323       elseif i==a14 then
324       Feed[i]=Fin[4];
325       elseif i==a15 then
326       Feed[i]=Fin[5];
327       elseif i==a16 then
328       Feed[i]=Fin[6];
329       elseif i==a17 then
330       Feed[i]=Fin[7];
331       elseif i==a18 then
332       Feed[i]=Fin[8];
333       elseif i==a19 then
334       Feed[i]=Fin[9];
335       elseif i==a10 then
336       Feed[i]=Fin[10];
337       elseif i==b11 and SidePhase1 == 1  then
338        Feed[i]=(-1*SideF[1]);
339       elseif i==b12 and SidePhase2 == 1  then
340        Feed[i]=(-1*SideF[2]);
341       elseif i==b13 and SidePhase3 == 1  then
342        Feed[i]=(-1*SideF[3]);
343       elseif i==b14 and SidePhase4 == 1  then
344        Feed[i]=(-1*SideF[4]);
345       elseif i==b15 and SidePhase5 == 1  then
346        Feed[i]=(-1*SideF[5]);
347       elseif i==b16 and SidePhase6 == 1  then
348        Feed[i]=(-1*SideF[6]);
349       elseif i==b17 and SidePhase7 == 1  then
350        Feed[i]=(-1*SideF[7]);
351       else
```

```
352       Feed[i]=0;
353       end if;
354     end for;
355     //========== Determing the Percentage Vapour of the Feed enetering the
          paticular stream================
356     for i in 1:Nt loop
357       if i==a11 then
358       pervap[i]=xvapin[1];
359       elseif i==a12 then
360       pervap[i]=xvapin[2];
361       elseif i==a13 then
362       pervap[i]=xvapin[3];
363       elseif i==a14 then
364       pervap[i]=xvapin[4];
365       elseif i==a15 then
366       pervap[i]=xvapin[5];
367       elseif i==a16 then
368       pervap[i]=xvapin[6];
369       elseif i==a17 then
370       pervap[i]=xvapin[7];
371       elseif i==a18 then
372       pervap[i]=xvapin[8];
373       elseif i==a19 then
374       pervap[i]=xvapin[9];
375       elseif i==a10 then
376       pervap[i]=xvapin[10];
377       else
378       pervap[i]=0;
379       end if;
380       end for;
381       for i in 1:Nc loop
382       Fc[1,i]=0;
383       end for;
384     //
          ================================================================================

385     //================================Overall Material Balance
          ==========================================
386     if Nout > 0 then
387     sum(Fin[:])=B+D+sum(SideF[:]);
388     else
389     sum(Fin[:])=B+D;
390     end if;
391     //
          ================================================================================

392     //connector equations
393     for i in 1:Ni loop
394     In[i].P = Pin[i];
395     In[i].T = Tin[i];
396     In[i].H = Hin[i];
397     In[i].F = Fin[i];
398     In[i].x_pc[1, :] = x_c[i,:];
399     In[i].xvap=xvapin[i];
400     end for;
401     Dist.P = CondP;
402     Dist.T = T[1];
403     Dist.F = D;
404     Dist.H = Hout1;
405     Dist.x_pc[1, :] = (integer(x[1,:] .*10000)) ./10000;
406     Bot.P = P[Nt];
407     Bot.T = T[Nt];
408     Bot.F = B;
409     Bot.H = Hout2;
```

```
410     Bot.x_pc[1, :] = (integer(x[Nt,:] .*10000)) ./10000;
411     Cduty.Q=Qc;
412     Rduty.Q=Qr;
413     //
            ================================================================================

414     //=========================================== At Condenser
            ==========================================
415     V[1]=L[1]+D;
416     RR=L[1]/D;
417      for i in 1:Nc loop
418         Pvap_c[1,i] = Simulator.Files.ThermodynamicFunctions.Psat(C[i].VP, T
                [1]);
419       end for;
420      for j in 1:Nc loop
421         K_c[1,j] = Pvap_c[1,j] / P[1];
422      end for;
423      sum(x[1,:].*K_c[1,:])=1;//Bubble Point Equation
424     //
            ================================================================================

425     //============================Assumption For Vapor Flow : Equimolal Flow
            ================================
426     for k in 2:Nt+1 loop
427       if Nout > 0 and SidePhase1 == 2 and k== b11 then
428         V[k]=V[k-1]-SideF[1];
429       elseif Nout > 0 and SidePhase2 == 2 and k== b12 then
430         V[k]=V[k-1]-SideF[2];
431       elseif Nout > 0 and SidePhase3 == 2 and k== b13 then
432         V[k]=V[k-1]-SideF[3];
433       elseif Nout > 0 and SidePhase4 == 2 and k== b14 then
434         V[k]=V[k-1]-SideF[4];
435       elseif Nout > 0 and SidePhase5 == 2 and k== b15 then
436         V[k]=V[k-1]-SideF[5];
437       elseif Nout > 0 and SidePhase6 == 2 and k== b16 then
438         V[k]=V[k-1]-SideF[6];
439       elseif Nout > 0 and SidePhase7 == 2 and k== b17 then
440        V[k]=V[k-1]-SideF[7];
441      else
442        V[k]=V[k-1]+(Feed[k-1]*pervap[k-1]);
443     end if;
444     end for;
445     //
            ================================================================================

446     //================Calculation of Vapour Composition at Condenser and 1st
            Stage=========================
447    y[1,:]=x[1,:] .*K_c[1,:];
448    sum(y[2,:])=1;
449    for i in 1:Nc-1 loop
450    y[2,i]=x[1,i];
451    end for;
452    //
            ================================================================================

453    //============================Calculation from 2nd Stage to Reboiler
            ========================================
454
455    for m in 2:Nt loop
456    //Liquid Flow Rate at the paticular stage calculated from material balance
            at that stage
457     if m==Nt then
458       L[m]=B;
459     else
```

```
460       L[m]=(Feed[m]*(1-pervap[m]))+L[m-1];
461     end if;
462     //
        ================================================================================

463     //Thermodynamic Calculation at that Stage
464     for n in 1:Nc loop
465         Pvap_c[m,n] = Simulator.Files.ThermodynamicFunctions.Psat(C[n].VP, T[
            m]);
466     end for;
467     for p in 1:Nc loop
468         K_c[m,p] = Pvap_c[m,p] / P[m];
469     end for;
470     for q in 1:Nc loop
471         y[m,q]-(x[m,q]*K_c[m,q])=0;
472     end for;
473     sum(x[m,:])=1;
474     //Assigning the Inlet Component Flow Rates and Leaving Side Streams
        Component Flow Rates
475     if m==a11 then
476        for k in 1:Nc loop
477        Fc[m,k]=Fin[1]*x_c[1,k];
478        end for;
479     elseif m==a12 then
480        for k in 1:Nc loop
481        Fc[m,k]=Fin[2]*x_c[2,k];
482        end for;
483     elseif m==a13 then
484        for k in 1:Nc loop
485        Fc[m,k]=Fin[3]*x_c[3,k];
486        end for;
487     elseif m==a14 then
488        for k in 1:Nc loop
489        Fc[m,k]=Fin[4]*x_c[4,k];
490        end for;
491     elseif m==a15 then
492        for k in 1:Nc loop
493        Fc[m,k]=Fin[5]*x_c[5,k];
494        end for;
495     elseif m==a16 then
496        for k in 1:Nc loop
497        Fc[m,k]=Fin[6]*x_c[6,k];
498        end for;
499     elseif m==a17 then
500        for k in 1:Nc loop
501        Fc[m,k]=Fin[7]*x_c[7,k];
502        end for;
503     elseif m==a18 then
504        for k in 1:Nc loop
505        Fc[m,k]=Fin[8]*x_c[8,k];
506        end for;
507     elseif m==a19 then
508        for k in 1:Nc loop
509        Fc[m,k]=Fin[9]*x_c[9,k];
510        end for;
511     elseif m==a10 then
512        for k in 1:Nc loop
513        Fc[m,k]=Fin[10]*x_c[10,k];
514        end for;
515     elseif m==b11 then
516        if SidePhase1==2 then
517         for k in 1:Nc loop
518           Fc[m,k]=(-1*SideF[1]*y[m,k]);
519         end for;
```

```
520         else
521         for k in 1:Nc loop
522         Fc[m,k]=(-1*SideF[1]*x[m,k]);
523         end for;
524         end if;
525       elseif m==b12 then
526         if SidePhase2==2 then
527         for k in 1:Nc loop
528         Fc[m,k]=(-1*SideF[2]*y[m,k]);
529         end for;
530         else
531         for k in 1:Nc loop
532         Fc[m,k]=(-1*SideF[2]*x[m,k]);
533         end for;
534         end if;
535       elseif m==b13 then
536         if SidePhase3==2 then
537         for k in 1:Nc loop
538         Fc[m,k]=(-1*SideF[3]*y[m,k]);
539         end for;
540         else
541         for k in 1:Nc loop
542         Fc[m,k]=(-1*SideF[3]*x[m,k]);
543         end for;
544         end if;
545       elseif m==b14 then
546         if SidePhase4==2 then
547         for k in 1:Nc loop
548         Fc[m,k]=(-1*SideF[4]*y[m,k]);
549         end for;
550         else
551         for k in 1:Nc loop
552         Fc[m,k]=(-1*SideF[4]*x[m,k]);
553         end for;
554         end if;
555       elseif m==b15 then
556         if SidePhase5==2 then
557         for k in 1:Nc loop
558         Fc[m,k]=(-1*SideF[5]*y[m,k]);
559         end for;
560         else
561         for k in 1:Nc loop
562         Fc[m,k]=(-1*SideF[5]*x[m,k]);
563         end for;
564         end if;
565       elseif m==b16 then
566         if SidePhase6==2 then
567         for k in 1:Nc loop
568         Fc[m,k]=(-1*SideF[6]*y[m,k]);
569         end for;
570         else
571         for k in 1:Nc loop
572         Fc[m,k]=(-1*SideF[6]*x[m,k]);
573         end for;
574         end if;
575       elseif m==b17 then
576         if SidePhase7==2 then
577         for k in 1:Nc loop
578         Fc[m,k]=(-1*SideF[7]*y[m,k]);
579         end for;
580         else
581         for k in 1:Nc loop
582         Fc[m,k]=(-1*SideF[7]*x[m,k]);
583         end for;
```

```
584       end if;
585     else
586       for l in 1:Nc loop
587       Fc[m,l]=0;
588       end for;
589       end if;
590       //
            ==========================================================================

591       //=======Calculating the Composition of Vapour Flow Rate Entering the
              Stage==========
592     for r in 1:Nc-1 loop
593       if (((V[m]*y[m,r])+(L[m]*x[m,r])-Fc[m,r]-(L[m-1]*x[m-1,r]))/V[m+1]) < 0
            or (((V[m]*y[m,r])+(L[m]*x[m,r])-Fc[m,r]-(L[m-1]*x[m-1,r]))/V[m
            +1]) > 1 then
594       y[m+1,r]=y[m,r];
595       else
596       (L[m-1]*x[m-1,r]) + (V[m+1]*y[m+1,r])+ Fc[m,r]-(V[m]*y[m,r])- (L[m]*x[
            m,r])=0;
597       end if;
598     end for;
599     sum(y[m+1,:])=1;
600     end for;
601     //=============================================
602     //Conditions:
603     //Assigning the Condition that the difference between the reboiler
          composition from the calculated value and the obtained value should
          be zero.
604     for h in 1:Nc-1 loop
605     xdel[h]=((sum(Fc[:,h]))-(D*x[1,h]))/B;
606     end for;
607     for w in 1:Nc-1 loop
608     x[Nt,w]-xdel[w]=0;
609     end for;
610     sum(x[1,:])=1;
611   //
        ==========================================================================

612   //=========================Energy Balance Calculation
        ========================
613     for i in 1:Nc loop
614       Hvapcond_c[i] = Simulator.Files.ThermodynamicFunctions.HVapId(C[i].SH,
            C[i].VapCp, C[i].HOV, C[i].Tc, T[2]);
615       Hliqcond_c[i] = Simulator.Files.ThermodynamicFunctions.HLiqId(C[i].SH,
            C[i].VapCp, C[i].HOV, C[i].Tc, T[1]);
616     end for;
617     if Ctype == "Total" then
618       Hliqcond = Hout1;
619     elseif Ctype == "Partial" then
620       Hliqcond = sum(y[2,:] .* Hliqcond_c[:]);
621     end if;
622     Hvapcond = sum(y[2,:] .* Hvapcond_c[:]);
623     if Nout > 0 then
624     sum(Fin[:] .* Hin[:]) + Qr - Qc = B * Hout2 + D * Hliqcond + sum(SideF
          [:].*HSideOut[:]);
625     else
626     sum(Fin[:] .* Hin[:]) - Qr  = B * Hout2 + D * Hliqcond + Qc;
627     end if;
628      V[1] * Hvapcond = Qc + D * Hliqcond + L[1] * Hout1;
629   //==Assigning the Obtained Values to the Side Streams if Selected
        ==============
630     if Nout == 1 then
631         Out_s[1].P = P[b11];
632         Out_s[1].T = T[b11];
```

```
633         Out_s[1].F = (-1*Feed[b11]);
634         Out_s[1].H= HSideOut[1];
635         if SidePhase1==1 then
636           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
637         else
638           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
639         end if;
640       end if;
641       if Nout==2 then
642         Out_s[1].P = P[b11];
643         Out_s[1].T = T[b11];
644         Out_s[1].F = (-1*Feed[b11]);
645         Out_s[1].H= HSideOut[1];
646         if SidePhase1==1 then
647           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
648         else
649           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
650         end if;
651         Out_s[2].P = P[b12];
652         Out_s[2].T = T[b12];
653         Out_s[2].F = (-1*Feed[b12]);
654         Out_s[2].H= HSideOut[2];
655         if SidePhase2==1 then
656           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
657         else
658           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
659         end if;
660       end if;
661       if Nout==3 then
662         Out_s[1].P = P[b11];
663         Out_s[1].T = T[b11];
664         Out_s[1].F = (-1*Feed[b11]);
665         Out_s[1].H= HSideOut[1];
666         if SidePhase1==1 then
667           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
668         else
669           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
670         end if;
671         Out_s[2].P = P[b12];
672         Out_s[2].T = T[b12];
673         Out_s[2].F = (-1*Feed[b12]);
674         Out_s[2].H= HSideOut[2];
675         if SidePhase2==1 then
676           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
677         else
678           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
679         end if;
680         Out_s[3].P = P[b13];
681         Out_s[3].T = T[b13];
682         Out_s[3].F = (-1*Feed[b13]);
683         Out_s[3].H= HSideOut[3];
684         if SidePhase3==1 then
685           Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
686         else
687           Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
688         end if;
689       end if;
690       if Nout==4 then
691         Out_s[1].P = P[b11];
692         Out_s[1].T = T[b11];
693         Out_s[1].F = (-1*Feed[b11]);
694         Out_s[1].H= HSideOut[1];
695         if SidePhase1==1 then
696           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
```

```
697         else
698           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
699         end if;
700         Out_s[2].P = P[b12];
701         Out_s[2].T = T[b12];
702         Out_s[2].F = (-1*Feed[b12]);
703         Out_s[2].H= HSideOut[2];
704         if SidePhase2==1 then
705           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
706         else
707           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
708         end if;
709         Out_s[3].P = P[b13];
710         Out_s[3].T = T[b13];
711         Out_s[3].F = (-1*Feed[b13]);
712         Out_s[3].H= HSideOut[3];
713         if SidePhase3==1 then
714           Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
715         else
716           Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
717         end if;
718         Out_s[4].P = P[b14];
719         Out_s[4].T = T[b14];
720         Out_s[4].F = (-1*Feed[b12]);
721         Out_s[4].H= HSideOut[4];
722         if SidePhase4==1 then
723           Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
724         else
725           Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
726         end if;
727       end if;
728       if Nout ==5 then
729         Out_s[1].P = P[b11];
730         Out_s[1].T = T[b11];
731         Out_s[1].F = (-1*Feed[b11]);
732         Out_s[1].H= HSideOut[1];
733         if SidePhase1==1 then
734           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
735         else
736           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
737         end if;
738         Out_s[2].P = P[b12];
739         Out_s[2].T = T[b12];
740         Out_s[2].F = (-1*Feed[b12]);
741         Out_s[2].H= HSideOut[2];
742         if SidePhase2==1 then
743           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
744         else
745           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
746         end if;
747         Out_s[3].P = P[b13];
748         Out_s[3].T = T[b13];
749         Out_s[3].F = (-1*Feed[b13]);
750         Out_s[3].H= HSideOut[3];
751         if SidePhase3==1 then
752           Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
753         else
754           Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
755         end if;
756         Out_s[4].P = P[b14];
757         Out_s[4].T = T[b14];
758         Out_s[4].F = (-1*Feed[b12]);
759         Out_s[4].H= HSideOut[4];
760         if SidePhase4==1 then
```

```
761           Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
762         else
763           Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
764         end if;
765         Out_s[5].P = P[b15];
766         Out_s[5].T = T[b15];
767         Out_s[5].F = (-1*Feed[b15]);
768         Out_s[5].H= HSideOut[5];
769         if SidePhase5==1 then
770           Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
771         else
772           Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
773         end if;
774       end if;
775       if Nout==6 then
776         Out_s[1].P = P[b11];
777         Out_s[1].T = T[b11];
778         Out_s[1].F = (-1*Feed[b11]);
779         Out_s[1].H= HSideOut[1];
780         if SidePhase1==1 then
781           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
782         else
783           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
784         end if;
785         Out_s[2].P = P[b12];
786         Out_s[2].T = T[b12];
787         Out_s[2].F = (-1*Feed[b12]);
788         Out_s[2].H= HSideOut[2];
789         if SidePhase2==1 then
790           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
791         else
792           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
793         end if;
794         Out_s[3].P = P[b13];
795         Out_s[3].T = T[b13];
796         Out_s[3].F = (-1*Feed[b13]);
797         Out_s[3].H= HSideOut[3];
798         if SidePhase3==1 then
799           Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
800         else
801           Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
802         end if;
803         Out_s[4].P = P[b14];
804         Out_s[4].T = T[b14];
805         Out_s[4].F = (-1*Feed[b12]);
806         Out_s[4].H= HSideOut[4];
807         if SidePhase4==1 then
808           Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
809         else
810           Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
811         end if;
812         Out_s[5].P = P[b15];
813         Out_s[5].T = T[b15];
814         Out_s[5].F = (-1*Feed[b15]);
815         Out_s[5].H= HSideOut[5];
816         if SidePhase5==1 then
817           Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
818         else
819           Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
820         end if;
821         Out_s[6].P = P[b16];
822         Out_s[6].T = T[b16];
823         Out_s[6].F = (-1*Feed[b16]);
824         Out_s[6].H= HSideOut[6];
```

```
825         if SidePhase6==1 then
826           Out_s[6].x_pc[1, :] = (integer(x[b16,:] .*10000)) ./10000;
827         else
828           Out_s[6].x_pc[1, :] = (integer(y[b16,:] .*10000)) ./10000;
829         end if;
830       end if;
831       if Nout==7 then
832          Out_s[1].P = P[b11];
833         Out_s[1].T = T[b11];
834         Out_s[1].F = (-1*Feed[b11]);
835         Out_s[1].H= HSideOut[1];
836         if SidePhase1==1 then
837           Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
838         else
839           Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
840         end if;
841         Out_s[2].P = P[b12];
842         Out_s[2].T = T[b12];
843         Out_s[2].F = (-1*Feed[b12]);
844         Out_s[2].H= HSideOut[2];
845         if SidePhase2==1 then
846           Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
847         else
848           Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
849         end if;
850         Out_s[3].P = P[b13];
851         Out_s[3].T = T[b13];
852         Out_s[3].F = (-1*Feed[b13]);
853         Out_s[3].H= HSideOut[3];
854         if SidePhase3==1 then
855           Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
856         else
857           Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
858         end if;
859         Out_s[4].P = P[b14];
860         Out_s[4].T = T[b14];
861         Out_s[4].F = (-1*Feed[b12]);
862         Out_s[4].H= HSideOut[4];
863         if SidePhase4==1 then
864           Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
865         else
866           Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
867         end if;
868         Out_s[5].P = P[b15];
869         Out_s[5].T = T[b15];
870         Out_s[5].F = (-1*Feed[b15]);
871         Out_s[5].H= HSideOut[5];
872         if SidePhase5==1 then
873           Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
874         else
875           Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
876         end if;
877         Out_s[6].P = P[b16];
878         Out_s[6].T = T[b16];
879         Out_s[6].F = (-1*Feed[b16]);
880         Out_s[6].H= HSideOut[6];
881         if SidePhase6==1 then
882           Out_s[6].x_pc[1, :] = (integer(x[b16,:] .*10000)) ./10000;
883         else
884           Out_s[6].x_pc[1, :] = (integer(y[b16,:] .*10000)) ./10000;
885         end if;
886         Out_s[7].P = P[b17];
887         Out_s[7].T = T[b17];
888         Out_s[7].F = (-1*Feed[b12]);
```

```
889         Out_s[7].H= HSideOut[7];
890         if SidePhase7==1 then
891           Out_s[7].x_pc[1, :] = (integer(x[b17,:] .*10000)) ./10000;
892         else
893           Out_s[7].x_pc[1, :] = (integer(y[b17,:] .*10000)) ./10000;
894         end if;
895         end if;
896   //=============== Correct Flow Rates through Energy Balance
897       for i in 1:Nt loop
898        if i==a11 then
899        HFeed[i]=Hin[1];
900        F[i]=Fin[1];
901        elseif i==a12 then
902        HFeed[i]=Hin[2];
903        F[i]=Fin[2];
904        elseif i==a13 then
905        HFeed[i]=Hin[3];
906        F[i]=Fin[3];
907        elseif i==a14 then
908        HFeed[i]=Hin[4];
909        F[i]=Fin[4];
910        elseif i==a15 then
911        HFeed[i]=Hin[5];
912        F[i]=Fin[5];
913        elseif i==a16 then
914        HFeed[i]=Hin[6];
915        F[i]=Fin[6];
916        elseif i==a17 then
917        HFeed[i]=Hin[7];
918        F[i]=Fin[7];
919        elseif i==a18 then
920        HFeed[i]=Hin[8];
921        F[i]=Fin[8];
922        elseif i==a19 then
923        HFeed[i]=Hin[9];
924        F[i]=Fin[9];
925        elseif i==a10 then
926        HFeed[i]=Hin[10];
927        F[i]=Fin[10];
928        else
929        HFeed[i]=0;
930        F[i]=0;
931        end if;
932        end for;
933        for i in 1:Nt loop
934        if i==b11 then
935        HSide[i]=HSideOut[1];
936        FSide[i]=SideF[1];
937        elseif i==b12 then
938        HSide[i]=HSideOut[2];
939        FSide[i]=SideF[2];
940        elseif i==b13 then
941        HSide[i]=HSideOut[3];
942        FSide[i]=SideF[3];
943        elseif i==b14 then
944        HSide[i]=HSideOut[4];
945        FSide[i]=SideF[4];
946        elseif i==b15 then
947        HSide[i]=HSideOut[5];
948        FSide[i]=SideF[5];
949        elseif i==b16 then
950        HSide[i]=HSideOut[6];
951        FSide[i]=SideF[6];
952        elseif i==b17 then
```

```
953       HSide[i]=HSideOut[7];
954       FSide[i]=SideF[7];
955       else
956       HSide[i]=0;
957       FSide[i]=0;
958       end if;
959       end for;
960     for j in 1:Nt loop
961     for i in 1:Nc loop
962       Hvap[j,i] = Simulator.Files.ThermodynamicFunctions.HVapId(C[i].SH, C[i
              ].VapCp, C[i].HOV, C[i].Tc, T[j]);
963       Hliq[j,i] = Simulator.Files.ThermodynamicFunctions.HLiqId(C[i].SH, C[i
              ].VapCp, C[i].HOV, C[i].Tc, T[j]);
964     end for;
965     end for;
966     correctL[1]=L[1];
967     correctV[1]=0;
968     correctV[2]=V[1];
969     correctL[Nt]=B;
970     //==================Energy Balance at each stage
          ==========================================
971     for i in 2:Nt-1 loop
972       F[i]+correctL[i-1]+correctV[i+1]-correctV[i]-correctL[i]-FSide[i]=0;
973      (F[i]*HFeed[i])+(correctL[i-1]*sum(Hliq[i-1,:].*x[i-1,:]))+(correctV[i
          +1]*sum(Hvap[i+1,:].*y[i+1,:]))-(correctV[i]*sum(Hvap[i,:].*y[i,:]))
          -       (correctL[i]*sum(Hliq[i,:].*x[i,:]))-(FSide[i]*HSide[i])=0;
974   end for;
975   //
        ================================================================================

976   end RaoultsLaw;
977     model NRTL
978     //Model Description:
979
980     // The Distillation Column aims to seperate the feed mixture into two
          streams i.e. Distillate and Residue based on the Volatility of the
          components present in the stream.Multiple Feeds and Side Draws can
          also be introduced.
981
982     //Thermodynamics : Raoults Law
983     extends Simulator.Files.Icons.DistillationColumn;
984     //========================= User Input Data
          ==================================//
985
986         parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "
              Component instances array" annotation(
987         Dialog(tab = "Column Specifications", group = "Component Parameters")
              );
988         parameter Integer Nc "Number of components" annotation(
989         Dialog(tab = "Column Specifications", group = "Component Parameters")
              );
990         parameter Integer Nt = 4 "Number of stages" annotation(
991         Dialog(tab = "Column Specifications", group = "Calculation Parameters
              "));
992         parameter Integer Nout = 0 "Number of side draws" annotation(
993         Dialog(tab = "Column Specifications", group = "Calculation Parameters
              "));
994         parameter Integer Ni = 1 "Number of feed streams" annotation(
995         Dialog(tab = "Column Specifications", group = "Calculation Parameters
              "));
996         parameter Integer InT_s[Ni] "Feed stage location" annotation(
997         Dialog(tab = "Column Specifications", group = "Calculation Parameters
              "));
998         parameter Integer OutT_s[Nout] "Feed stage location" annotation(
```

```
999         Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1000        parameter Real D(unit="mol/s") "Distillate flow rate" annotation(
1001        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1002        parameter Real RR(unit="-") "Reflux Ratio" annotation(
1003        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1004        parameter Real CondP(unit="Pa") "Condenser Pressure" annotation(
1005        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1006        parameter Real ReboilerP(unit="Pa") "Reboiler Pressure" annotation(
1007        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1008        parameter Real Tguess(unit="K") "Guess Temperature" annotation(
1009        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1010        parameter Real SideF[Nout](each unit="mol/s")"Side stream flow rate"
                annotation(
1011        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1012        parameter Real SidePhase1 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1013        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1014        parameter Real SidePhase2 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1015        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1016        parameter Real SidePhase3 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1017        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1018        parameter Real SidePhase4 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1019        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1020        parameter Real SidePhase5 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1021        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1022        parameter Real SidePhase6 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1023        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1024        parameter Real SidePhase7 = 0"Side stream Phase 0 for None 1 for
                Liquid and 2 for Vapour" annotation(
1025        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1026        parameter String Ctype = "Total" "Condenser type: Total or Partial"
                annotation(
1027        Dialog(tab = "Column Specifications", group = "Calculation Parameters
                "));
1028
1029
1030    //========================= Distillation Column Variables
            ================================//
1031
1032    //========================= Connecter Variables
            ========================================//
1033        Simulator.Files.Interfaces.matConn In[Ni](each Nc = Nc) annotation(
1034          Placement(visible = true, transformation(origin = {-248, -40},
                  extent = {{-10, -10}, {10, 10}}, rotation = 0),
                  iconTransformation(origin = {-250, 0}, extent = {{-10, -10},
```

```
        {10, 10}}, rotation = 0)));
1035    Simulator.Files.Interfaces.matConn Dist(each Nc = Nc) annotation(
1036      Placement(visible = true, transformation(origin = {250, 316},
          extent = {{-10, -10}, {10, 10}}, rotation = 0),
          iconTransformation(origin = {250, 298}, extent = {{-10, -10},
          {10, 10}}, rotation = 0)));
1037    Simulator.Files.Interfaces.matConn Bot(each Nc = Nc) annotation(
1038      Placement(visible = true, transformation(origin = {250, -296},
          extent = {{-10, -10}, {10, 10}}, rotation = 0),
          iconTransformation(origin = {252, -300}, extent = {{-10, -10},
          {10, 10}}, rotation = 0)));
1039    Simulator.Files.Interfaces.enConn Cduty annotation(
1040      Placement(visible = true, transformation(origin = {246, 590},
          extent = {{-10, -10}, {10, 10}}, rotation = 0),
          iconTransformation(origin = {250, 600}, extent = {{-10, -10},
          {10, 10}}, rotation = 0)));
1041    Simulator.Files.Interfaces.enConn Rduty annotation(
1042      Placement(visible = true, transformation(origin = {252, -588},
          extent = {{-10, -10}, {10, 10}}, rotation = 0),
          iconTransformation(origin = {250, -598}, extent = {{-10, -10},
          {10, 10}}, rotation = 0)));
1043    Simulator.Files.Interfaces.matConn Out_s[Nout](each Nc = Nc)
          annotation(
1044      Placement(visible = true, transformation(origin = {-36, 32}, extent
          = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
          origin = {-70, 60}, extent = {{-10, -10}, {10, 10}}, rotation =
          0)));
1045  //
      ==========================================================================

1046  extends Simulator.GuessModels.InitialGuess;
1047  //============================Model Variables
      ==========================================

1048
1049    Real Qc(unit="W")"Condenser Duty";
1050    Real Qr(unit="W")"Reboiler Duty";
1051    Real Hvapcond_c[Nc](each unit="J/mol")"Enthalpy of Components present
        in Vapour from Stage 1";
1052    Real Hliqcond_c[Nc](each unit="J/mol")"Enthalpy of Components present
        in Liquid from Condenser";
1053    Real Hvapcond(unit="J/mol")"Enthalpy of Vapour from Stage 1";
1054    Real Hout1(unit="J/mol")"Enthalpy of the Distillate Stream";
1055    Real Hout2(unit="J/mol")"Enthalpy of the Residue Stream";
1056    Real Hin[Ni](unit="J/mol")"Enthalpy of the Feed Stream";
1057    Real Hliqcond(unit="J/mol")"Enthalpy of Vapour from Stage 1";
1058    Real Fin[Ni](each unit = "mol/s", each min = 0) "Inlet stream molar
        flow rate";
1059    Real Pin[Ni](each unit = "Pa", each min = 0) "Inlet stream pressure";
1060    Real Tin[Ni](each unit = "K", each min = 0) "Inlet stream emperature"
        ;
1061    Real x_c[Ni,Nc](each unit = "-") "Component mole fraction";
1062    Real Feed[Nt](each unit="mol/s")"Flow rate that enters the
        distillation column";
1063    Real Fc[Nt,Nc](each unit="mol/s")"Component flow rate that enters the
        column";
1064    Real B(unit="mol/s")"Bottom Flow Rate";
1065    Real V[Nt+1](each unit="mol/s")"Vapour flow rate at each stage";
1066    Real L[Nt](each unit="mol/s")"Liquid flow rate at each stage" ;
1067    Real x[Nt,Nc](each unit="-",each start = 1/Nc)"Liquid Composition at
        each stage";
1068    Real y[Nt+1,Nc](each unit="-",each start = 1/Nc)"Vapour Composition
        at each stage";
1069    Real Pvap_c[Nt,Nc](each unit="Pa")"Saturation Pressure";
1070    Real T[Nt](each unit = "K", each min = 0,  each start = Tguess)"
```

```
        Temperature at each stage";
1071    Real K_c[Nt,Nc](each unit = "-", each min = 1, each start = 1.5);
1072    Real P[Nt](each unit="Pa")"Pressure at each stage";
1073    Real HSideOut[Nout](each unit="J/mol")"Enthalpy of the Side Streams";
1074    Real HFeed[Nt](each unit="J/mol")"Enthalpy of the Feed";
1075    Real HSide[Nt](each unit="J/mol")"Enthalpy of the Side Stream";
1076    Real FSide[Nt](each unit="mol/s")"Side Stream Flow Rates";
1077    Real correctL[Nt](each unit="mol/s")"Correct Liquid Flow Rates";
1078    Real correctV[Nt](each unit="mol/s")"Correct Vapour Flow Rates";
1079    Real Hvap[Nt,Nc](each unit="J/mol")"Enthalpy of Components present in
          Vapour of each stage";
1080    Real Hliq[Nt,Nc](each unit="J/mol")"Enthalpy of Components present in
          Liquid of each stage";
1081    Real F[Nt](each unit="mol/s")"Feed that enters the Distillation
        Column at each stage";
1082    //
          ================================================================

1083    Real xdel[Nc-1];
1084    Real a11,a12,a13,a14,a15,a16,a17,a18,a19,a10;
1085    Real xvapin[Ni];
1086    Real pervap[Nt];
1087    Integer b11;
1088    Integer b12,b13,b14,b15,b16,b17;
1089    Real a[Nc+1];
1090    //Thermodynamic Variables:
1091    constant Real R = 1.98721;
1092    Real tau[Nt,Nc, Nc], G[Nt,Nc, Nc], alpha[Nc, Nc], A[Nc, Nc], BIPS[Nc,
          Nc, 2];
1093    Real sum1[Nt,Nc](each start = 1), sum2[Nt,Nc](each start = 1);
1094    Real gma_c[Nt,Nc](each start = 1);
1095
1096  equation
1097  BIPS = Simulator.Files.ThermodynamicFunctions.BIPNRTL(Nc, C.CAS);
1098  A = BIPS[:, :, 1];
1099  alpha = BIPS[:, :, 2];
1100  //============================ Pressure at Each Stage
        ========================================

1101
1102  // Pressure at each stage is calculated by interpolation.
1103  P[1]=CondP;
1104  P[Nt]=ReboilerP;
1105  for i in 2:Nt-1 loop
1106    if CondP == ReboilerP then
1107      P[i]=CondP;
1108    else
1109      if ReboilerP > CondP then
1110        P[i]=CondP + ((i - 1)/(Nt -1))*(ReboilerP -CondP);
1111      else
1112        P[i]=CondP + ((i - 1)/(Nt -1))*(CondP -ReboilerP);
1113      end if;
1114    end if;
1115  end for;
1116
1117  //
        ================================================================

1118
1119  //====================Determing the Location of the Feed and the Side
        Stream====================
1120  if 1==Ni then
1121    a11=InT_s[1]+1;
1122    a12=0;
1123    a13=0;
```

```
1124          a14=0;
1125          a15=0;
1126          a16=0;
1127          a17=0;
1128          a18=0;
1129          a19=0;
1130          a10=0;
1131        elseif 2==Ni then
1132          a11=InT_s[1]+1;
1133          a12=InT_s[2]+1;
1134          a13=0;
1135          a14=0;
1136          a15=0;
1137          a16=0;
1138          a17=0;
1139          a18=0;
1140          a19=0;
1141          a10=0;
1142        elseif 3==Ni then
1143          a11=InT_s[1]+1;
1144          a12=InT_s[2]+1;
1145          a13=InT_s[3]+1;
1146          a14=0;
1147          a15=0;
1148          a16=0;
1149          a17=0;
1150          a18=0;
1151          a19=0;
1152          a10=0;
1153        elseif 4==Ni then
1154          a11=InT_s[1]+1;
1155          a12=InT_s[2]+1;
1156          a13=InT_s[3]+1;
1157          a14=InT_s[4]+1;
1158          a15=0;
1159          a16=0;
1160          a17=0;
1161          a18=0;
1162          a19=0;
1163          a10=0;
1164        elseif 5==Ni then
1165          a11=InT_s[1]+1;
1166          a12=InT_s[2]+1;
1167          a13=InT_s[3]+1;
1168          a14=InT_s[4]+1;
1169          a15=InT_s[5]+1;
1170          a16=0;
1171          a17=0;
1172          a18=0;
1173          a19=0;
1174          a10=0;
1175        elseif 6==Ni then
1176          a11=InT_s[1]+1;
1177          a12=InT_s[2]+1;
1178          a13=InT_s[3]+1;
1179          a14=InT_s[4]+1;
1180          a15=InT_s[5]+1;
1181          a16=InT_s[6]+1;
1182          a17=0;
1183          a18=0;
1184          a19=0;
1185          a10=0;
1186        elseif 7==Ni then
1187          a11=InT_s[1]+1;
```

```
1188          a12=InT_s[2]+1;
1189          a13=InT_s[3]+1;
1190          a14=InT_s[4]+1;
1191          a15=InT_s[5]+1;
1192          a16=InT_s[6]+1;
1193          a17=InT_s[7]+1;
1194          a18=0;
1195          a19=0;
1196          a10=0;
1197        elseif 8==Ni then
1198          a11=InT_s[1]+1;
1199          a12=InT_s[2]+1;
1200          a13=InT_s[3]+1;
1201          a14=InT_s[4]+1;
1202          a15=InT_s[5]+1;
1203          a16=InT_s[6]+1;
1204          a17=InT_s[7]+1;
1205          a18=InT_s[8]+1;
1206          a19=0;
1207          a10=0;
1208        elseif 9==Ni then
1209          a11=InT_s[1]+1;
1210          a12=InT_s[2]+1;
1211          a13=InT_s[3]+1;
1212          a14=InT_s[4]+1;
1213          a15=InT_s[5]+1;
1214          a16=InT_s[6]+1;
1215          a17=InT_s[7]+1;
1216          a18=InT_s[8]+1;
1217          a19=InT_s[9]+1;
1218          a10=0;
1219        else
1220          a11=InT_s[1]+1;
1221          a12=InT_s[2]+1;
1222          a13=InT_s[3]+1;
1223          a14=InT_s[4]+1;
1224          a15=InT_s[5]+1;
1225          a16=InT_s[6]+1;
1226          a17=InT_s[7]+1;
1227          a18=InT_s[8]+1;
1228          a19=InT_s[9]+1;
1229          a10=InT_s[10]+1;
1230        end if;
1231        if 1==Nout then
1232          b11=OutT_s[1]+1;
1233          b12=0;
1234          b13=0;
1235          b14=0;
1236          b15=0;
1237          b16=0;
1238          b17=0;
1239        elseif 2==Nout then
1240          b11=OutT_s[1]+1;
1241          b12=OutT_s[2]+1;
1242          b13=0;
1243          b14=0;
1244          b15=0;
1245          b16=0;
1246          b17=0;
1247        elseif 3==Nout then
1248          b11=OutT_s[1]+1;
1249          b12=OutT_s[2]+1;
1250          b13=OutT_s[3]+1;
1251          b14=0;
```

```
1252        b15=0;
1253        b16=0;
1254        b17=0;
1255      elseif 4==Nout then
1256        b11=OutT_s[1]+1;
1257        b12=OutT_s[2]+1;
1258        b13=OutT_s[3]+1;
1259        b14=OutT_s[4]+1;
1260        b15=0;
1261        b16=0;
1262        b17=0;
1263      elseif 5==Nout then
1264        b11=OutT_s[1]+1;
1265        b12=OutT_s[2]+1;
1266        b13=OutT_s[3]+1;
1267        b14=OutT_s[4]+1;
1268        b15=OutT_s[5]+1;
1269        b16=0;
1270        b17=0;
1271      elseif 6==Nout then
1272        b11=OutT_s[1]+1;
1273        b12=OutT_s[2]+1;
1274        b13=OutT_s[3]+1;
1275        b14=OutT_s[4]+1;
1276        b16=OutT_s[6]+1;
1277        b15=0;
1278        b17=0;
1279      elseif 7==Nout then
1280        b11=OutT_s[1]+1;
1281        b12=OutT_s[2]+1;
1282        b13=OutT_s[3]+1;
1283        b14=OutT_s[4]+1;
1284        b15=OutT_s[5]+1;
1285        b16=OutT_s[6]+1;
1286        b17=OutT_s[7]+1;
1287      else
1288        b11=0;
1289        b12=0;
1290        b13=0;
1291        b14=0;
1292        b15=0;
1293        b16=0;
1294        b17=0;
1295      end if;
1296      //
          ==========================================================================

1297      //====================Determining the Feed Flow Rate at each stage
          ============================
1298      //If the Feed is enetring that paticular stage it will be assigned the
          inlet flow rate else it will be assigned the value zero. If a side
          stream is leaving from a paticular tray provided that the phase is
          liquid it will be assigned the value provided by the user.
1299
1300      for i in 1:Nt loop
1301        if i==a11 then
1302        Feed[i]=Fin[1];
1303        elseif i==a12 then
1304        Feed[i]=Fin[2];
1305        elseif i==a13 then
1306        Feed[i]=Fin[3];
1307        elseif i==a14 then
1308        Feed[i]=Fin[4];
1309        elseif i==a15 then
```

```
1310        Feed[i]=Fin[5];
1311        elseif i==a16 then
1312        Feed[i]=Fin[6];
1313        elseif i==a17 then
1314        Feed[i]=Fin[7];
1315        elseif i==a18 then
1316        Feed[i]=Fin[8];
1317        elseif i==a19 then
1318        Feed[i]=Fin[9];
1319        elseif i==a10 then
1320        Feed[i]=Fin[10];
1321        elseif i==b11 and SidePhase1 == 1  then
1322         Feed[i]=(-1*SideF[1]);
1323        elseif i==b12 and SidePhase2 == 1  then
1324         Feed[i]=(-1*SideF[2]);
1325        elseif i==b13 and SidePhase3 == 1  then
1326         Feed[i]=(-1*SideF[3]);
1327        elseif i==b14 and SidePhase4 == 1  then
1328         Feed[i]=(-1*SideF[4]);
1329        elseif i==b15 and SidePhase5 == 1  then
1330         Feed[i]=(-1*SideF[5]);
1331        elseif i==b16 and SidePhase6 == 1  then
1332         Feed[i]=(-1*SideF[6]);
1333        elseif i==b17 and SidePhase7 == 1  then
1334         Feed[i]=(-1*SideF[7]);
1335        else
1336        Feed[i]=0;
1337        end if;
1338      end for;
1339      //========== Determing the Percentage Vapour of the Feed enetering the
          paticular stream==========
1340      for i in 1:Nt loop
1341        if i==a11 then
1342        pervap[i]=xvapin[1];
1343        elseif i==a12 then
1344        pervap[i]=xvapin[2];
1345        elseif i==a13 then
1346        pervap[i]=xvapin[3];
1347        elseif i==a14 then
1348        pervap[i]=xvapin[4];
1349        elseif i==a15 then
1350        pervap[i]=xvapin[5];
1351        elseif i==a16 then
1352        pervap[i]=xvapin[6];
1353        elseif i==a17 then
1354        pervap[i]=xvapin[7];
1355        elseif i==a18 then
1356        pervap[i]=xvapin[8];
1357        elseif i==a19 then
1358        pervap[i]=xvapin[9];
1359        elseif i==a10 then
1360        pervap[i]=xvapin[10];
1361        else
1362        pervap[i]=0;
1363        end if;
1364        end for;
1365        for i in 1:Nc loop
1366        Fc[1,i]=0;
1367        end for;
1368      //
          ==========

1369      //==========Overall Material Balance
          ==========
```

```
1370      if Nout > 0 then
1371      sum(Fin[:])=B+D+sum(SideF[:]);
1372      else
1373      sum(Fin[:])=B+D;
1374      end if;
1375      //
          ================================================================================

1376      //connector equations
1377      for i in 1:Ni loop
1378      In[i].P = Pin[i];
1379      In[i].T = Tin[i];
1380      In[i].H = Hin[i];
1381      In[i].F = Fin[i];
1382      In[i].x_pc[1, :] = x_c[i,:];
1383      In[i].xvap=xvapin[i];
1384      end for;
1385      Dist.P = CondP;
1386      Dist.T = T[1];
1387      Dist.F = D;
1388      Dist.H = Hout1;
1389      Dist.x_pc[1, :] = (integer(x[1,:] .*10000)) ./10000;
1390      Bot.P = P[Nt];
1391      Bot.T = T[Nt];
1392      Bot.F = B;
1393      Bot.H = Hout2;
1394      Bot.x_pc[1, :] = (integer(x[Nt,:] .*10000)) ./10000;
1395      Cduty.Q=Qc;
1396      Rduty.Q=Qr;
1397      //
          ================================================================================

1398      //========================================= At Condenser
          ==========================================
1399      V[1]=L[1]+D;
1400      RR=L[1]/D;
1401       for g in 1:Nc loop
1402        for k in 1:Nc loop
1403      tau[1,g,k] = A[g,k]/ (R * T[1]);
1404      end for;
1405      end for;
1406      for i in 1:Nc loop
1407        for j in 1:Nc loop
1408          G[1,i, j] = exp(-alpha[i, j] * tau[1,i, j]);
1409        end for;
1410      end for;
1411      for i in 1:Nc loop
1412        sum1[1,i] = sum(x[1,:] .* G[1,:, i]);
1413        sum2[1,i] = sum(x[1,:] .* tau[1,:, i] .* G[1,:, i]);
1414      end for;
1415      for i in 1:Nc loop
1416      log(gma_c[1,i]) = sum(x[1,:] .* tau[1,:, i] .* G[1,:, i]) / sum(x[1,:]
          .* G[1,:, i]) + sum(x[1,:] .* G[1,i, :] ./ sum1[1,:] .* (tau[1,i,
          :] .- sum2[1,:] ./ sum1[1,:]));
1417      end for;
1418       for i in 1:Nc loop
1419          Pvap_c[1,i] = Simulator.Files.ThermodynamicFunctions.Psat(C[i].VP,
              T[1]);
1420        end for;
1421       for j in 1:Nc loop
1422          K_c[1,j] = gma_c[1,j]*Pvap_c[1,j] / P[1];
1423       end for;
1424       a[1]=0;
1425       for d in 1:Nc loop
```

```
1426        a[d+1]=a[1]+  (x[1,d]*K_c[1,d]);
1427        end for;
1428        sum(a[:])=1;
1429      //
          ================================================================================

1430      //============================Assumption For Vapor Flow : Equimolal Flow
            ==============================
1431      for k in 2:Nt+1 loop
1432        if Nout > 0 and SidePhase1 == 2 and k== b11 then
1433          V[k]=V[k-1]-SideF[1];
1434        elseif Nout > 0 and SidePhase2 == 2 and k== b12 then
1435          V[k]=V[k-1]-SideF[2];
1436        elseif Nout > 0 and SidePhase3 == 2 and k== b13 then
1437          V[k]=V[k-1]-SideF[3];
1438        elseif Nout > 0 and SidePhase4 == 2 and k== b14 then
1439          V[k]=V[k-1]-SideF[4];
1440        elseif Nout > 0 and SidePhase5 == 2 and k== b15 then
1441          V[k]=V[k-1]-SideF[5];
1442        elseif Nout > 0 and SidePhase6 == 2 and k== b16 then
1443          V[k]=V[k-1]-SideF[6];
1444        elseif Nout > 0 and SidePhase7 == 2 and k== b17 then
1445         V[k]=V[k-1]-SideF[7];
1446       else
1447          V[k]=V[k-1]+(Feed[k-1]*pervap[k-1]);
1448      end if;
1449      end for;
1450      //
          ================================================================================

1451      //================Calculation of Vapour Composition at Condenser and 1st
            Stage==========================
1452     y[1,:]=x[1,:] .*K_c[1,:];
1453     sum(y[2,:])=1;
1454     for i in 1:Nc-1 loop
1455     y[2,i]=x[1,i];
1456     end for;
1457     //
          ================================================================================

1458     //============================Calculation from 2nd Stage to Reboiler
            ======================================
1459     for m in 2:Nt loop
1460     //Liquid Flow Rate at the paticular stage calculated from material
            balance at that stage
1461      if m==Nt then
1462      L[m]=B;
1463      else
1464      L[m]=(Feed[m]*(1-pervap[m]))+L[m-1];
1465      end if;
1466      //
          ================================================================================

1467      //Thermodynamic Calculation at that Stage
1468      for n in 1:Nc loop
1469           Pvap_c[m,n] = Simulator.Files.ThermodynamicFunctions.Psat(C[n].VP,
                 T[m]);
1470        end for;
1471      //
1472      for g in 1:Nc loop
1473      for k in 1:Nc loop
1474      tau[m,g,k] = A[g,k]/ (R * T[m]);
1475      end for;
1476      end for;
```

```
1477      for i in 1:Nc loop
1478        for j in 1:Nc loop
1479          G[m,i, j] = exp(-alpha[i, j] * tau[m,i, j]);
1480        end for;
1481      end for;
1482      for i in 1:Nc loop
1483        sum1[m,i] = sum(x[m,:] .* G[m,:, i]);
1484        sum2[m,i] = sum(x[m,:] .* tau[m,:, i] .* G[m,:, i]);
1485      end for;
1486      for i in 1:Nc loop
1487      log(gma_c[m,i]) = sum(x[m,:] .* tau[m,:, i] .* G[m,:, i]) / sum(x[m,:]
          .* G[m,:, i]) + sum(x[m,:] .* G[m,i, :] ./ sum1[m,:] .* (tau[m,i,
          :] .- sum2[m,:] ./ sum1[m,:]));
1488      end for;
1489      //
1490      for p in 1:Nc loop
1491          K_c[m,p] = gma_c[m,p] *Pvap_c[m,p] / P[m];
1492      end for;
1493      for q in 1:Nc loop
1494          y[m,q]-(x[m,q]*K_c[m,q])=0;
1495      end for;
1496      sum(y[m,:] ./K_c[m,:])=1;
1497      ////Assigning the Inlet Component Flow Rates and Leaving Side Streams
          Component Flow Rates
1498      if m==a11 then
1499         for k in 1:Nc loop
1500         Fc[m,k]=Fin[1]*x_c[1,k];
1501         end for;
1502      elseif m==a12 then
1503         for k in 1:Nc loop
1504         Fc[m,k]=Fin[2]*x_c[2,k];
1505         end for;
1506      elseif m==a13 then
1507         for k in 1:Nc loop
1508         Fc[m,k]=Fin[3]*x_c[3,k];
1509         end for;
1510      elseif m==a14 then
1511         for k in 1:Nc loop
1512         Fc[m,k]=Fin[4]*x_c[4,k];
1513         end for;
1514      elseif m==a15 then
1515         for k in 1:Nc loop
1516         Fc[m,k]=Fin[5]*x_c[5,k];
1517         end for;
1518      elseif m==a16 then
1519         for k in 1:Nc loop
1520         Fc[m,k]=Fin[6]*x_c[6,k];
1521         end for;
1522      elseif m==a17 then
1523         for k in 1:Nc loop
1524         Fc[m,k]=Fin[7]*x_c[7,k];
1525         end for;
1526      elseif m==a18 then
1527         for k in 1:Nc loop
1528         Fc[m,k]=Fin[8]*x_c[8,k];
1529         end for;
1530      elseif m==a19 then
1531         for k in 1:Nc loop
1532         Fc[m,k]=Fin[9]*x_c[9,k];
1533         end for;
1534      elseif m==a10 then
1535         for k in 1:Nc loop
1536         Fc[m,k]=Fin[10]*x_c[10,k];
1537         end for;
```

```
1538        elseif m==b11 then
1539           if SidePhase1==2 then
1540            for k in 1:Nc loop
1541              Fc[m,k]=(-1*SideF[1]*y[m,k]);
1542            end for;
1543           else
1544           for k in 1:Nc loop
1545           Fc[m,k]=(-1*SideF[1]*x[m,k]);
1546           end for;
1547           end if;
1548        elseif m==b12 then
1549           if SidePhase2==2 then
1550           for k in 1:Nc loop
1551           Fc[m,k]=(-1*SideF[2]*y[m,k]);
1552           end for;
1553           else
1554           for k in 1:Nc loop
1555           Fc[m,k]=(-1*SideF[2]*x[m,k]);
1556           end for;
1557           end if;
1558        elseif m==b13 then
1559           if SidePhase3==2 then
1560           for k in 1:Nc loop
1561           Fc[m,k]=(-1*SideF[3]*y[m,k]);
1562           end for;
1563           else
1564           for k in 1:Nc loop
1565           Fc[m,k]=(-1*SideF[3]*x[m,k]);
1566           end for;
1567           end if;
1568        elseif m==b14 then
1569          if SidePhase4==2 then
1570           for k in 1:Nc loop
1571           Fc[m,k]=(-1*SideF[4]*y[m,k]);
1572           end for;
1573           else
1574           for k in 1:Nc loop
1575           Fc[m,k]=(-1*SideF[4]*x[m,k]);
1576           end for;
1577           end if;
1578        elseif m==b15 then
1579           if SidePhase5==2 then
1580           for k in 1:Nc loop
1581           Fc[m,k]=(-1*SideF[5]*y[m,k]);
1582           end for;
1583           else
1584           for k in 1:Nc loop
1585           Fc[m,k]=(-1*SideF[5]*x[m,k]);
1586           end for;
1587           end if;
1588        elseif m==b16 then
1589           if SidePhase6==2 then
1590           for k in 1:Nc loop
1591           Fc[m,k]=(-1*SideF[6]*y[m,k]);
1592           end for;
1593           else
1594           for k in 1:Nc loop
1595           Fc[m,k]=(-1*SideF[6]*x[m,k]);
1596           end for;
1597           end if;
1598        elseif m==b17 then
1599           if SidePhase7==2 then
1600           for k in 1:Nc loop
1601           Fc[m,k]=(-1*SideF[7]*y[m,k]);
```

```
1602          end for;
1603          else
1604          for k in 1:Nc loop
1605          Fc[m,k]=(-1*SideF[7]*x[m,k]);
1606          end for;
1607          end if;
1608        else
1609          for l in 1:Nc loop
1610          Fc[m,l]=0;
1611          end for;
1612          end if;
1613        //=========================
1614        for r in 1:Nc-1 loop
1615          if (((V[m]*y[m,r])+(L[m]*x[m,r])-Fc[m,r]-(L[m-1]*x[m-1,r]))/V[m+1]) <
                0 or (((V[m]*y[m,r])+(L[m]*x[m,r])-Fc[m,r]-(L[m-1]*x[m-1,r]))/V[
               m+1]) > 1 then
1616          y[m+1,r]=y[m,r];
1617          else
1618          (L[m-1]*x[m-1,r]) + (V[m+1]*y[m+1,r])+ Fc[m,r]-(V[m]*y[m,r])- (L[m]*x
                [m,r])=0;
1619          end if;
1620        end for;
1621        sum(y[m+1,:])=1;
1622        end for;
1623        //=============================================
1624        //Conditions:
1625        //Assigning the Condition that the difference between the reboiler
              composition from the calculated value and the obtained value should
               be zero.
1626        for h in 1:Nc-1 loop
1627        xdel[h]=((sum(Fc[:,h]))-(D*x[1,h]))/B;
1628        end for;
1629        for w in 1:Nc-1 loop
1630        x[Nt,w]-xdel[w]=0;
1631        end for;
1632        sum(x[1,:])=1;
1633       //
             ==========================================================================

1634       //=========================Energy Balance Calculation
             ==========================
1635        for i in 1:Nc loop
1636          Hvapcond_c[i] = Simulator.Files.ThermodynamicFunctions.HVapId(C[i].
                SH, C[i].VapCp, C[i].HOV, C[i].Tc, T[2]);
1637          Hliqcond_c[i] = Simulator.Files.ThermodynamicFunctions.HLiqId(C[i].
                SH, C[i].VapCp, C[i].HOV, C[i].Tc, T[1]);
1638        end for;
1639        if Ctype == "Total" then
1640          Hliqcond = Hout1;
1641        elseif Ctype == "Partial" then
1642          Hliqcond = sum(y[2,:] .* Hliqcond_c[:]);
1643        end if;
1644        Hvapcond = sum(y[2,:] .* Hvapcond_c[:]);
1645        if Nout > 0 then
1646        sum(Fin[:] .* Hin[:]) + Qr - Qc = B * Hout2 + D * Hliqcond + sum(SideF
             [:].*HSideOut[:]);
1647        else
1648        sum(Fin[:] .* Hin[:]) - Qr  = B * Hout2 + D * Hliqcond + Qc;
1649        end if;
1650         V[1] * Hvapcond = Qc + D * Hliqcond + L[1] * Hout1;
1651       //==Assigning the Obtained Values to the Side Streams if Selected
             ==============
1652        if Nout == 1 then
1653            Out_s[1].P = P[b11];
```

```
1654            Out_s[1].T = T[b11];
1655            Out_s[1].F = (-1*Feed[b11]);
1656            Out_s[1].H= HSideOut[1];
1657            if SidePhase1==1 then
1658              Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1659            else
1660              Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1661            end if;
1662          end if;
1663          if Nout==2 then
1664            Out_s[1].P = P[b11];
1665            Out_s[1].T = T[b11];
1666            Out_s[1].F = (-1*Feed[b11]);
1667            Out_s[1].H= HSideOut[1];
1668            if SidePhase1==1 then
1669              Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1670            else
1671              Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1672            end if;
1673            Out_s[2].P = P[b12];
1674            Out_s[2].T = T[b12];
1675            Out_s[2].F = (-1*Feed[b12]);
1676            Out_s[2].H= HSideOut[2];
1677            if SidePhase2==1 then
1678              Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1679            else
1680              Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1681            end if;
1682          end if;
1683          if Nout==3 then
1684            Out_s[1].P = P[b11];
1685            Out_s[1].T = T[b11];
1686            Out_s[1].F = (-1*Feed[b11]);
1687            Out_s[1].H= HSideOut[1];
1688            if SidePhase1==1 then
1689              Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1690            else
1691              Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1692            end if;
1693            Out_s[2].P = P[b12];
1694            Out_s[2].T = T[b12];
1695            Out_s[2].F = (-1*Feed[b12]);
1696            Out_s[2].H= HSideOut[2];
1697            if SidePhase2==1 then
1698              Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1699            else
1700              Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1701            end if;
1702            Out_s[3].P = P[b13];
1703            Out_s[3].T = T[b13];
1704            Out_s[3].F = (-1*Feed[b13]);
1705            Out_s[3].H= HSideOut[3];
1706            if SidePhase3==1 then
1707              Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
1708            else
1709              Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
1710            end if;
1711          end if;
1712          if Nout==4 then
1713            Out_s[1].P = P[b11];
1714            Out_s[1].T = T[b11];
1715            Out_s[1].F = (-1*Feed[b11]);
1716            Out_s[1].H= HSideOut[1];
1717            if SidePhase1==1 then
```

```
1718            Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1719          else
1720            Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1721          end if;
1722          Out_s[2].P = P[b12];
1723          Out_s[2].T = T[b12];
1724          Out_s[2].F = (-1*Feed[b12]);
1725          Out_s[2].H= HSideOut[2];
1726          if SidePhase2==1 then
1727            Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1728          else
1729            Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1730          end if;
1731          Out_s[3].P = P[b13];
1732          Out_s[3].T = T[b13];
1733          Out_s[3].F = (-1*Feed[b13]);
1734          Out_s[3].H= HSideOut[3];
1735          if SidePhase3==1 then
1736            Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
1737          else
1738            Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
1739          end if;
1740          Out_s[4].P = P[b14];
1741          Out_s[4].T = T[b14];
1742          Out_s[4].F = (-1*Feed[b12]);
1743          Out_s[4].H= HSideOut[4];
1744          if SidePhase4==1 then
1745            Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
1746          else
1747            Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
1748          end if;
1749        end if;
1750        if Nout ==5 then
1751          Out_s[1].P = P[b11];
1752          Out_s[1].T = T[b11];
1753          Out_s[1].F = (-1*Feed[b11]);
1754          Out_s[1].H= HSideOut[1];
1755          if SidePhase1==1 then
1756            Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1757          else
1758            Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1759          end if;
1760          Out_s[2].P = P[b12];
1761          Out_s[2].T = T[b12];
1762          Out_s[2].F = (-1*Feed[b12]);
1763          Out_s[2].H= HSideOut[2];
1764          if SidePhase2==1 then
1765            Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1766          else
1767            Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1768          end if;
1769          Out_s[3].P = P[b13];
1770          Out_s[3].T = T[b13];
1771          Out_s[3].F = (-1*Feed[b13]);
1772          Out_s[3].H= HSideOut[3];
1773          if SidePhase3==1 then
1774            Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
1775          else
1776            Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
1777          end if;
1778          Out_s[4].P = P[b14];
1779          Out_s[4].T = T[b14];
1780          Out_s[4].F = (-1*Feed[b12]);
1781          Out_s[4].H= HSideOut[4];
```

```
1782          if SidePhase4==1 then
1783            Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
1784          else
1785            Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
1786          end if;
1787          Out_s[5].P = P[b15];
1788          Out_s[5].T = T[b15];
1789          Out_s[5].F = (-1*Feed[b15]);
1790          Out_s[5].H= HSideOut[5];
1791          if SidePhase5==1 then
1792            Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1793          else
1794            Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1795          end if;
1796        end if;
1797        if Nout==6 then
1798          Out_s[1].P = P[b11];
1799          Out_s[1].T = T[b11];
1800          Out_s[1].F = (-1*Feed[b11]);
1801          Out_s[1].H= HSideOut[1];
1802          if SidePhase1==1 then
1803            Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1804          else
1805            Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1806          end if;
1807          Out_s[2].P = P[b12];
1808          Out_s[2].T = T[b12];
1809          Out_s[2].F = (-1*Feed[b12]);
1810          Out_s[2].H= HSideOut[2];
1811          if SidePhase2==1 then
1812            Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1813          else
1814            Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1815          end if;
1816          Out_s[3].P = P[b13];
1817          Out_s[3].T = T[b13];
1818          Out_s[3].F = (-1*Feed[b13]);
1819          Out_s[3].H= HSideOut[3];
1820          if SidePhase3==1 then
1821            Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
1822          else
1823            Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
1824          end if;
1825          Out_s[4].P = P[b14];
1826          Out_s[4].T = T[b14];
1827          Out_s[4].F = (-1*Feed[b12]);
1828          Out_s[4].H= HSideOut[4];
1829          if SidePhase4==1 then
1830            Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
1831          else
1832            Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
1833          end if;
1834          Out_s[5].P = P[b15];
1835          Out_s[5].T = T[b15];
1836          Out_s[5].F = (-1*Feed[b15]);
1837          Out_s[5].H= HSideOut[5];
1838          if SidePhase5==1 then
1839            Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1840          else
1841            Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1842          end if;
1843          Out_s[6].P = P[b16];
1844          Out_s[6].T = T[b16];
1845          Out_s[6].F = (-1*Feed[b16]);
```

```
1846            Out_s[6].H= HSideOut[6];
1847            if SidePhase6==1 then
1848              Out_s[6].x_pc[1, :] = (integer(x[b16,:] .*10000)) ./10000;
1849            else
1850              Out_s[6].x_pc[1, :] = (integer(y[b16,:] .*10000)) ./10000;
1851            end if;
1852          end if;
1853          if Nout==7 then
1854             Out_s[1].P = P[b11];
1855            Out_s[1].T = T[b11];
1856            Out_s[1].F = (-1*Feed[b11]);
1857            Out_s[1].H= HSideOut[1];
1858            if SidePhase1==1 then
1859              Out_s[1].x_pc[1, :] = (integer(x[b11,:] .*10000)) ./10000;
1860            else
1861              Out_s[1].x_pc[1, :] = (integer(y[b11,:] .*10000)) ./10000;
1862            end if;
1863            Out_s[2].P = P[b12];
1864            Out_s[2].T = T[b12];
1865            Out_s[2].F = (-1*Feed[b12]);
1866            Out_s[2].H= HSideOut[2];
1867            if SidePhase2==1 then
1868              Out_s[2].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1869            else
1870              Out_s[2].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1871            end if;
1872            Out_s[3].P = P[b13];
1873            Out_s[3].T = T[b13];
1874            Out_s[3].F = (-1*Feed[b13]);
1875            Out_s[3].H= HSideOut[3];
1876            if SidePhase3==1 then
1877              Out_s[3].x_pc[1, :] = (integer(x[b13,:] .*10000)) ./10000;
1878            else
1879              Out_s[3].x_pc[1, :] = (integer(y[b13,:] .*10000)) ./10000;
1880            end if;
1881            Out_s[4].P = P[b14];
1882            Out_s[4].T = T[b14];
1883            Out_s[4].F = (-1*Feed[b12]);
1884            Out_s[4].H= HSideOut[4];
1885            if SidePhase4==1 then
1886              Out_s[4].x_pc[1, :] = (integer(x[b14,:] .*10000)) ./10000;
1887            else
1888              Out_s[4].x_pc[1, :] = (integer(y[b14,:] .*10000)) ./10000;
1889            end if;
1890            Out_s[5].P = P[b15];
1891            Out_s[5].T = T[b15];
1892            Out_s[5].F = (-1*Feed[b15]);
1893            Out_s[5].H= HSideOut[5];
1894            if SidePhase5==1 then
1895              Out_s[5].x_pc[1, :] = (integer(x[b12,:] .*10000)) ./10000;
1896            else
1897              Out_s[5].x_pc[1, :] = (integer(y[b12,:] .*10000)) ./10000;
1898            end if;
1899            Out_s[6].P = P[b16];
1900            Out_s[6].T = T[b16];
1901            Out_s[6].F = (-1*Feed[b16]);
1902            Out_s[6].H= HSideOut[6];
1903            if SidePhase6==1 then
1904              Out_s[6].x_pc[1, :] = (integer(x[b16,:] .*10000)) ./10000;
1905            else
1906              Out_s[6].x_pc[1, :] = (integer(y[b16,:] .*10000)) ./10000;
1907            end if;
1908            Out_s[7].P = P[b17];
1909            Out_s[7].T = T[b17];
```

```
1910          Out_s[7].F = (-1*Feed[b12]);
1911          Out_s[7].H= HSideOut[7];
1912          if SidePhase7==1 then
1913            Out_s[7].x_pc[1, :] = (integer(x[b17,:] .*10000)) ./10000;
1914          else
1915            Out_s[7].x_pc[1, :] = (integer(y[b17,:] .*10000)) ./10000;
1916          end if;
1917          end if;
1918        //================================== Correct Flow Rates through
              Energy Balance ==================================
1919        for i in 1:Nt loop
1920         if i==a11 then
1921         HFeed[i]=Hin[1];
1922         F[i]=Fin[1];
1923         elseif i==a12 then
1924         HFeed[i]=Hin[2];
1925         F[i]=Fin[2];
1926         elseif i==a13 then
1927         HFeed[i]=Hin[3];
1928         F[i]=Fin[3];
1929         elseif i==a14 then
1930         HFeed[i]=Hin[4];
1931         F[i]=Fin[4];
1932         elseif i==a15 then
1933         HFeed[i]=Hin[5];
1934         F[i]=Fin[5];
1935         elseif i==a16 then
1936         HFeed[i]=Hin[6];
1937         F[i]=Fin[6];
1938         elseif i==a17 then
1939         HFeed[i]=Hin[7];
1940         F[i]=Fin[7];
1941         elseif i==a18 then
1942         HFeed[i]=Hin[8];
1943         F[i]=Fin[8];
1944         elseif i==a19 then
1945         HFeed[i]=Hin[9];
1946         F[i]=Fin[9];
1947         elseif i==a10 then
1948         HFeed[i]=Hin[10];
1949         F[i]=Fin[10];
1950         else
1951         HFeed[i]=0;
1952         F[i]=0;
1953         end if;
1954         end for;
1955         for i in 1:Nt loop
1956         if i==b11 then
1957         HSide[i]=HSideOut[1];
1958         FSide[i]=SideF[1];
1959         elseif i==b12 then
1960         HSide[i]=HSideOut[2];
1961         FSide[i]=SideF[2];
1962         elseif i==b13 then
1963         HSide[i]=HSideOut[3];
1964         FSide[i]=SideF[3];
1965         elseif i==b14 then
1966         HSide[i]=HSideOut[4];
1967         FSide[i]=SideF[4];
1968         elseif i==b15 then
1969         HSide[i]=HSideOut[5];
1970         FSide[i]=SideF[5];
1971         elseif i==b16 then
1972         HSide[i]=HSideOut[6];
```

```
1973          FSide[i]=SideF[6];
1974          elseif i==b17 then
1975          HSide[i]=HSideOut[7];
1976          FSide[i]=SideF[7];
1977          else
1978          HSide[i]=0;
1979          FSide[i]=0;
1980          end if;
1981          end for;
1982        for j in 1:Nt loop
1983        for i in 1:Nc loop
1984          Hvap[j,i] = Simulator.Files.ThermodynamicFunctions.HVapId(C[i].SH, C[
                  i].VapCp, C[i].HOV, C[i].Tc, T[j]);
1985          Hliq[j,i] = Simulator.Files.ThermodynamicFunctions.HLiqId(C[i].SH, C[
                  i].VapCp, C[i].HOV, C[i].Tc, T[j]);
1986        end for;
1987        end for;
1988        correctL[1]=L[1];
1989        correctV[1]=0;
1990        correctV[2]=V[1];
1991        correctL[Nt]=B;
1992        //==================Energy Balance at each stage
              ==========================================
1993        for i in 2:Nt-1 loop
1994          F[i]+correctL[i-1]+correctV[i+1]-correctV[i]-correctL[i]-FSide[i]=0;
1995         (F[i]*HFeed[i])+(correctL[i-1]*sum(Hliq[i-1,:].*x[i-1,:]))+(correctV[i
                  +1]*sum(Hvap[i+1,:].*y[i+1,:]))-(correctV[i]*sum(Hvap[i,:].*y[i,
                  :]))-(correctL[i]*sum(Hliq[i,:].*x[i,:]))-(FSide[i]*HSide[i])=0;
1996      end for;
1997      //
            ================================================================================================

1998      end NRTL;
1999    end DistCol;
```

```
1  package PengPxy
2  model Pxy
3  //Model Description:
4
5  //Pxy is a binary phase diagram where the Temperature is held constant and
       the Liquid Phase Mole Fraction is varied. For a paticular liquid phase
       mole fraction the Pressure and Vapour phase mole fraction is obtained
       through thermodynamic realationship at equillbrium conditions.
6
7  //Thermodynamics: Peng Robinson
8
9  //
       ================================================================================

10 //====User Input Data=================================
11   extends Modelica.Icons.Example;
12   import data = Simulator.Files.ChemsepDatabase;
13   parameter data.Propane eth;
14   parameter data.Nbutane prop;
15   parameter Integer Nc = 2"Number of Components";
16   parameter Real Temp(unit="K")=323"Temperature";
17   parameter Integer NOP=6;//Number of Points
18   parameter data.GeneralProperties C[Nc] = {eth, prop};
19   parameter Real step=0.2;//Step Value
20 //=======Model Variables================================
21   points point[NOP](each Nc = Nc, each C = C,each T = Temp);
22   Real x[NOP,Nc](each unit="-")"Liquid Phase Mole Fraction";
23   Real y[NOP,Nc](each unit="-")"Vapour Phase Mole Fraction";
24   Real P[NOP](each unit="Pa")"Pressure";
25 //==========Equation Section============================
26  equation
27  for i in 1:NOP loop
28     x[i, 1] = 0 + (i - 1) * step;
29     x[i, 2]=1-x[i, 1];
30  end for;
31  for i in 1:NOP loop
32  point[i].x=x[i,:];
33  point[i].P=P[i];
34  point[i].y=y[i,:];
35  end for;
36
37 end Pxy;
38 model points
39      parameter Integer Nc"Number of Components";
40      parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc];
41      parameter Real T(unit="K")"Temperature";
42      Real P(unit="Pa",min= Simulator.Files.ThermodynamicFunctions.Psat(C[1].
            VP, T),max=Simulator.Files.ThermodynamicFunctions.Psat(C[2].VP, T),
            start=Simulator.Files.ThermodynamicFunctions.Psat(C[1].VP, T))"
            Pressure";
43      Real x[2](each unit="-")"Liquid Phase Mole Fraction";
44      Real y[2](each min=0,each max=1,each start=0.5)"Vapour Phase Mole
            Fraction";
45      parameter Real kij_c[Nc, Nc](each start = 1) =
            Simulator.Files.ThermodynamicFunctions.BIPPR(Nc, C.name);
46      Real Tr_c[Nc](each start = Tg) "Reduced temperature";
47      Real b_c[Nc];
48      Real a_c[Nc](each start = xg);
49      Real m_c[Nc];
50      Real q_c[Nc];
51      Real aij_c[Nc, Nc];
52      Real K_c[Nc](each start = K_guess);
53      Real aMliq, bMliq;
54      Real Aliq(each start=xliqg), Bliq(each start=xvapg);
```

```
55      Real Cliq[4];
56      Real Z_RL[3, 2](start=xliqg);
57      Real Zliq[3](each start=xliqg), Zll(each start=xvapg);
58      Real sumxliq[Nc];
59      Real aMvap, bMvap;
60      Real Avap(each start=xliqg), Bvap(each start=xvapg);
61      Real Cvap[4];
62      Real Z_RV[3, 2](each start= xvapg);
63      Real Zvap[3](each start=xvapg), Zvv;
64      Real sumxvap[Nc];
65      Real A, B, Cdummy, D_c[Nc], E, F, G, H_c[Nc], I_c[Nc], J_c[Nc];
66      Real R=8.314;
67      Real philiq_c[Nc];
68      Real phivap_c[Nc];
69      extends Simulator.GuessModels.InitialGuess;
70  equation
71      Tr_c = T ./ C.Tc;
72      b_c = 0.0778 * R * C.Tc ./ C.Pc;
73      m_c = 0.37464 .+ 1.54226 * C.AF .- 0.26992 * C.AF .^ 2;
74      q_c = 0.45724 * R ^ 2 * C.Tc .^ 2 ./ C.Pc;
75      a_c = q_c .* (1 .+ m_c .* (1 .- sqrt(Tr_c))) .^ 2;
76      aij_c = {{(1 - kij_c[i, j]) * sqrt(a_c[i] * a_c[j]) for i in 1:Nc} for
            j in 1:Nc};
77    //=============================================================
78  //Liquid_Fugacity Coefficient Calculation Routine
79      aMliq = sum({{x[i] * x[j] * aij_c[i, j] for i in 1:Nc} for j in 1:Nc});
80      bMliq = sum(b_c .* x[:]);
81      Aliq = aMliq * P / (R * T) ^ 2;
82      Bliq = bMliq * P / (R * T);
83      Cliq[1] = 1;
84      Cliq[2] = Bliq - 1;
85      Cliq[3] = Aliq - 3 * Bliq ^ 2 - 2 * Bliq;
86      Cliq[4] = Bliq ^ 3 + Bliq ^ 2 - Aliq * Bliq;
87      Z_RL = Modelica.Math.Vectors.Utilities.roots(Cliq);
88      Zliq = {Z_RL[i, 1] for i in 1:3};
89      Zll = min({Zliq});
90      sumxliq = {sum({x[j] * aij_c[i, j] for j in 1:Nc}) for i in 1:Nc};
91        A = Zll + 2.4142135 * Bliq;
92        B = Zll - 0.414213 * Bliq;
93       Cdummy = log(Zll - Bliq);
94      for i in 1:Nc loop
95          D_c[i] = b_c[i] / bMliq;
96      end for;
97      for i in 1:Nc loop
98          J_c[i] = sumxliq[i] / aMliq;
99      end for;
100     philiq_c = exp(Aliq / (Bliq * sqrt(8)) * log(A / B) .* (D_c .- 2 * J_c)
            .+ (Zll - 1) * D_c .- Cdummy);
101
102   //=============================================================
103 //Vapour Fugacity Calculation Routine
104     aMvap = sum({{y[i] * y[j] * aij_c[i, j] for i in 1:Nc} for j in 1:Nc});
105     bMvap = sum(b_c .* y[:]);
106     Avap = aMvap * P / (R * T) ^ 2;
107     Bvap = bMvap * P / (R * T);
108     Cvap[1] = 1;
109     Cvap[2] = Bvap - 1;
110     Cvap[3] = Avap - 3 * Bvap ^ 2 - 2 * Bvap;
111     Cvap[4] = Bvap ^ 3 + Bvap ^ 2 - Avap * Bvap;
112     Z_RV = Modelica.Math.Vectors.Utilities.roots(Cvap);
113     Zvap = {Z_RV[i, 1] for i in 1:3};
114     Zvv = max({Zvap});
115     sumxvap = {sum({y[j] * aij_c[i, j] for j in 1:Nc}) for i in 1:Nc};
116       E = Zvv + 2.4142135 * Bvap;
```

```
117         F = Zvv - 0.414213 * Bvap;
118         G = log(Zvv - Bvap);
119       for i in 1:Nc loop
120           H_c[i] = b_c[i] / bMvap;
121       end for;
122       for i in 1:Nc loop
123           I_c[i] = sumxvap[i] / aMvap;
124       end for;
125       phivap_c = exp(Avap / (Bvap * sqrt(8)) * log(E / F) .* (H_c .- 2 * I_c)
              .+ (Zvv - 1) * H_c .- G);
126       for i in 1:Nc loop
127           K_c[i] = philiq_c[i] / phivap_c[i];
128       end for;
129       y[:]=x[:].*K_c[:];
130       sum(y[:])=1;
131   end points;
132   end PengPxy;
```

```
1  package PengTxy
2  model Txy
3  //Model Description:
4
5  //Txy is a binary phase diagram where the Pressure is held constant and the
       Liquid Phase Mole Fraction is varied. For a paticular liquid phase
      mole fraction the Temperature and Vapour phase mole fraction is
      obtained through thermodynamic realationship at equillbrium conditions.
6
7  //Thermodynamics: Peng Robinson
8
9  //
       ================================================================================

10 //====User Input Data=================================
11   extends Modelica.Icons.Example;
12   import data = Simulator.Files.ChemsepDatabase;
13   parameter data.Ethane eth;
14   parameter data.Propane prop;
15   parameter Integer Nc = 2"Number of Components";
16   parameter Real Pressure(unit="Pa")=101325"Pressure";
17   parameter Integer NOP=6"Number of Points";
18   parameter data.GeneralProperties C[Nc] = {eth, prop};
19   parameter Real step=0.2;
20
21   //=======Model Variables=================================
22   points point[NOP](each Nc = Nc, each C = C,each P = Pressure,each Tb1=
         Tb1,each Tb2=Tb2);
23   Real x[NOP,Nc];
24   Real y[NOP,Nc];
25   Real T[NOP];
26   extends PengTxy.intialguess;// The intial guess values are the boiling
         point temperature of pure components.The pressure in the intialguess
         should be changed as per requirement.
27 //=========Equation Section=============================
28  equation
29  for i in 1:NOP loop
30     x[i, 1] = 0 + (i - 1) * step;
31     x[i, 2]=1-x[i, 1];
32  end for;
33  for i in 1:NOP loop
34  point[i].x=x[i,:];
35  point[i].T=T[i];
36  point[i].y=y[i,:];
37  end for;
38
39 end Txy;
40 model points
41 parameter Integer Nc;
42 parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc];
43 parameter Real P;
44 parameter Real Tb1;
45 parameter Real Tb2;
46 Real T(min=Tb1,max=Tb2,start=Tb1);
47 Real x[2];
48 Real y[2](each min=0,each max=1,each start=0.4);
49     parameter Real kij_c[Nc, Nc](each start = 1) =
           Simulator.Files.ThermodynamicFunctions.BIPPR(Nc, C.name);
50     Real Tr_c[Nc](each start = Tg) "Reduced temperature";
51     Real b_c[Nc];
52     Real a_c[Nc](start = xg);
53     Real m_c[Nc];
54     Real q_c[Nc];
55     Real aij_c[Nc, Nc];
```

```
56      Real K_c[Nc](start = 1);
57      Real aMliq, bMliq;
58      Real Aliq, Bliq;
59      Real Cliq[4];
60      Real Z_RL[3, 2](start=xliqg);
61      Real Zliq[3], Zll;
62      Real sumxliq[Nc];
63      Real aMvap, bMvap;
64      Real Avap, Bvap;
65      Real Cvap[4];
66      Real Z_RV[3, 2];
67      Real Zvap[3], Zvv;
68      Real sumxvap[Nc];
69      Real A, B, Cdummy, D_c[Nc], E, F, G, H_c[Nc], I_c[Nc], J_c[Nc];
70      Real R=8.314;
71      Real philiq_c[Nc](each start=5);
72      Real phivap_c[Nc](each start=5);
73      extends Simulator.GuessModels.InitialGuess;
74  equation
75      Tr_c = T ./ C.Tc;
76      b_c = 0.0778 * R * C.Tc ./ C.Pc;
77      m_c = 0.37464 .+ 1.54226 * C.AF .- 0.26992 * C.AF .^ 2;
78      q_c = 0.45724 * R ^ 2 * C.Tc .^ 2 ./ C.Pc;
79      a_c = q_c .* (1 .+ m_c .* (1 .- sqrt(Tr_c))) .^ 2;
80      aij_c = {{(1 - kij_c[i, j]) * sqrt(a_c[i] * a_c[j]) for i in 1:Nc} for
            j in 1:Nc};
81    //=============================================================
82  //Liquid_Fugacity Coefficient Calculation Routine
83      aMliq = sum({{x[i] * x[j] * aij_c[i, j] for i in 1:Nc} for j in 1:Nc});
84      bMliq = sum(b_c .* x[:]);
85      Aliq = aMliq * P / (R * T) ^ 2;
86      Bliq = bMliq * P / (R * T);
87      Cliq[1] = 1;
88      Cliq[2] = Bliq - 1;
89      Cliq[3] = Aliq - 3 * Bliq ^ 2 - 2 * Bliq;
90      Cliq[4] = Bliq ^ 3 + Bliq ^ 2 - Aliq * Bliq;
91      Z_RL = Modelica.Math.Vectors.Utilities.roots(Cliq);
92      Zliq = {Z_RL[i, 1] for i in 1:3};
93      Zll = min({Zliq});
94      sumxliq = {sum({x[j] * aij_c[i, j] for j in 1:Nc}) for i in 1:Nc};
95        A = Zll + 2.4142135 * Bliq;
96        B = Zll - 0.414213 * Bliq;
97       Cdummy = log(Zll - Bliq);
98      for i in 1:Nc loop
99          D_c[i] = b_c[i] / bMliq;
100     end for;
101     for i in 1:Nc loop
102         J_c[i] = sumxliq[i] / aMliq;
103     end for;
104     philiq_c = exp(Aliq / (Bliq * sqrt(8)) * log(A / B) .* (D_c .- 2 * J_c)
            .+ (Zll - 1) * D_c .- Cdummy);
105
106   //=============================================================
107 //Vapour Fugacity Calculation Routine
108     aMvap = sum({{y[i] * y[j] * aij_c[i, j] for i in 1:Nc} for j in 1:Nc});
109     bMvap = sum(b_c .* y[:]);
110     Avap = aMvap * P / (R * T) ^ 2;
111     Bvap = bMvap * P / (R * T);
112     Cvap[1] = 1;
113     Cvap[2] = Bvap - 1;
114     Cvap[3] = Avap - 3 * Bvap ^ 2 - 2 * Bvap;
115     Cvap[4] = Bvap ^ 3 + Bvap ^ 2 - Avap * Bvap;
116     Z_RV = Modelica.Math.Vectors.Utilities.roots(Cvap);
117     Zvap = {Z_RV[i, 1] for i in 1:3};
```

```
118       Zvv = max({Zvap});
119       sumxvap = {sum({y[j] * aij_c[i, j] for j in 1:Nc}) for i in 1:Nc};
120         E = Zvv + 2.4142135 * Bvap;
121         F = Zvv - 0.414213 * Bvap;
122         G = log(Zvv - Bvap);
123       for i in 1:Nc loop
124           H_c[i] = b_c[i] / bMvap;
125       end for;
126       for i in 1:Nc loop
127           I_c[i] = sumxvap[i] / aMvap;
128       end for;
129       phivap_c = exp(Avap / (Bvap * sqrt(8)) * log(E / F) .* (H_c .- 2 * I_c)
              .+ (Zvv - 1) * H_c .- G);
130       for i in 1:Nc loop
131           K_c[i] = philiq_c[i] / phivap_c[i];
132       end for;
133       y[:]=x[:].*K_c[:];
134       sum(y[:])=1;
135   end points;
136   model intialguess
137   protected
138   //==========User Input Data==================
139   parameter Real P(unit="Pa")=101325;
140   //==========Intial Guess Variables==========
141   parameter Real Tb1(fixed=false);
142   parameter Real Tb2(fixed=false);
143   initial equation
144   log(P)=(C[1].VP[2] + C[1].VP[3] / Tb1 + C[1].VP[4] * log(Tb1) + C[1].VP[5]
          * Tb1^ C[1].VP[6]);
145   log(P)=(C[2].VP[2] + C[2].VP[3] / Tb2 + C[2].VP[4] * log(Tb2) + C[2].VP[5]
          * Tb2^ C[2].VP[6]);
146   end intialguess;
147   end PengTxy;
```

```
1  package EqReactor
2  model EquilibriumReactor "Model of an equilibrium reactor to calculate the
       outlet stream mole fraction of components"
3      extends Simulator.Files.Icons.EquilibriumReactor;
4      //EquiibriumReactor Code works for all the valid phases and all modes
           available in DWSIM
5      //The reaction basis included are PartialPressure, Activity and
           MoleFraction
6      //The base component need not be specified and is directly calculated
           from an external function
7      //
           ================================================================

8      parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "
           Component instances array" annotation(
9        Dialog(tab = "Reactor Specifications", group = "Component Parameters"
             ));
10     parameter Integer Nc "Number of components" annotation(
11       Dialog(tab = "Reactor Specifications", group = "Component Parameters"
             ));
12     //
           ================================================================

13     extends Simulator.GuessModels.InitialGuess;
14     //Connector Variables
15     Real Pin(unit = "Pa", min = 0, start = Pg) "Inlet stream pressure";
16     Real Tin(unit = "K", min = 0, start = Tg) "Inlet stream temperature";
17     Real Fin(unit = "mol/s", min = 0, start = Fg) "Inlet stream molar flow
           rate";
18     Real Hin(unit = "kJ/kmol", start = Htotg) "Inlet stream molar enthalpy"
           ;
19     Real Sin(unit = "kJ/[kmol.K]") "Inlet stream molar entropy";
20     Real xin_c[Nc](each unit = "K", each min = 0, each max = 1, start = xg)
        "Inlet stream component mole fraction";
21     Real xvapin;
22     Real Pout(unit = "Pa", min = 0, start = Pg) "Outlet stream pressure";
23     Real Tout(unit = "K", min = 0, start = Tg) "Outlet stream temperature";
24     Real Fout(unit = "mol/s", min = 0, start = Fg) "Outlet stream molar
           flow rate";
25     Real Hout(unit = "kJ/kmol", start = Htotg) "Outlet stream molar
           enthalpy";
26     Real Sout(unit = "kJ/[kmol.K]") "Outlet stream molar entropy";
27     Real xout_c[Nc](each unit = "=", each min = 0, each max = 1, start = xg
           ) "Outlet stream component mole fraction";
28     Real xvapout;
29     Real Q;
30     //Model Variables
31     Real Psat[Nc] "Vapour Pressure";
32     Real Kmod[Nr] "Modified Equiibrium Contant";
33     Real Fin_c[Nc] "Component Molar Flow Rates";
34     Real Hr "Reaction Heat";
35     //Model Parameters
36     parameter String Phase = "Vapour" "Required phase:
37       ''Liquid'', ''Vapour''" annotation(
38       Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
             ;
39     parameter String Basis = "Activity" "Required basis: ''MoleFraction'',
           ''Activity'', ''PartialPressure''" annotation(
40       Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
             ;
41     parameter String Mode = "Isothermal" "Required mode of operation: ''
           Isothermal'', ''OutletTemperature'', ''Adiabatic''" annotation(
42       Dialog(tab = "Reactor Specifications", group = "Calculation
             Parameters"));
```

```
43      parameter Real Pdel(unit = "Pa") = 0 "Pressure drop" annotation(
44        Dialog(tab = "Reactor Specifications", group = "Calculation
              Parameters"));
45      parameter Real Tdef(unit = "K") = 300 "Defined outlet temperature,
          applicable if OutletTemperature mode is chosen" annotation(
46        Dialog(tab = "Reactor Specifications", group = "Calculation
              Parameters"));
47      //Reaction Variables
48      Real SC_rc[Nr, Nc] "Stoichiometric coefficients of the components";
49      Integer BC_r[Nr] "Base component of reaction";
50      Real Ndel[Nr];
51      Real Scabs[Nr, Nc] "Relative stoichiometry with respect to base
          component";
52      Real Ext_r[Nr](each start = xvapg) "Reaction Extent";
53      Real X_r[Nr, Nc] "Conversion of reactants";
54      //
            ================================================================

55      extends EqReactor.EquilibriumReaction(Nr = 1, Coef_cr = {{-1}, {-1},
            {1}, {1}}, Rmode = "ConstantK", Kg = {0.5}, T = Tout, P = Pout);
56      Simulator.Files.Interfaces.matConn Out(Nc = Nc) annotation(
57        Placement(visible = true, transformation(origin = {100, 0}, extent =
              {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin
              = {100, 0}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
58      Simulator.Files.Interfaces.enConn enConn annotation(
59        Placement(visible = true, transformation(origin = {2, -100}, extent =
              {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin
              = {0, -130}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
60      Simulator.Files.Interfaces.matConn In(Nc = Nc) annotation(
61        Placement(visible = true, transformation(origin = {-98, -2}, extent =
              {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin
              = {-100, -2}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
62      //
            ================================================================

63    equation
64  //
        ============================================================================

65      In.P = Pin;
66      In.T = Tin;
67      In.F = Fin;
68      In.H = Hin;
69      In.S = Sin;
70      In.x_pc[1, :] = xin_c;
71      In.xvap = xvapin;
72      Out.P = Pout;
73      Out.T = Tout;
74      Out.F = Fout;
75      Out.H = Hout;
76      Out.S = Sout;
77      Out.x_pc[1, :] = xout_c;
78      Out.xvap = xvapout;
79      enConn.Q = Q;
80      Pout = Pin - Pdel;
81      for i in 1:Nc loop
82        Psat[i] = Simulator.Files.ThermodynamicFunctions.Psat(C[i].VP, Tin);
83      end for;
84  //Automated calculation of base component
85      for i in 1:Nc loop
86        Fin_c[i] = Fin * xin_c[i];
87      end for;
88      for i in 1:Nr loop
89        BC_r[i] = Simulator.Files.Models.ReactionManager.BaseCalc(Nc, Fin_c,
```

```
          SC_rc[i, :]);
90      end for;
91      for j in 1:Nr loop
92        for i in 1:Nc loop
93          SC_rc[j, i] = Coef_cr[i, j];
94        end for;
95      end for;
96  //  ================================================================================

97      for i in 1:Nr loop
98        Ndel[i] = sum(SC_rc[i, 1:Nc]);
99      end for;
100     if Mode == "Isothermal" then
101       Tout = Tin;
102       Hr = Hr_r[1] * 1E-3 * (Fin_c[BC_r[1]] * X_r[1, BC_r[1]]) / Coef_cr[
              BC_r[1], 1] * Coef_cr[BC_r[1], 1];
103       Q = Hout * Fout * 1E-3 - Hin * Fin * 1E-3 - Hr;
104     else
105       if Mode == "OutletTemperature" then
106         Tout = Tdef;
107         Hr = Hr_r[1] * 1E-3 * (Fin_c[BC_r[1]] * X_r[1, BC_r[1]]) / Coef_cr[
                BC_r[1], 1] * Coef_cr[BC_r[1], 1];
108         Q = Hout * Fout * 1E-3 - Hin * Fin * 1E-3 - Hr;
109       else
110         Q = 0;
111         Hr = Hr_r[1] * 1E-3 * (Fin_c[BC_r[1]] * X_r[1, BC_r[1]]) / Coef_cr[
                BC_r[1], 1] * Coef_cr[BC_r[1], 1];
112         Q = Hout * Fout * 1E-3 - Hin * Fin * 1E-3 - Hr;
113       end if;
114     end if;
115     for i in 1:Nr loop
116       for j in 1:Nc loop
117         Scabs[i, j] = SC_rc[i, j] / abs(SC_rc[i, BC_r[i]]);
118       end for;
119     end for;
120     if Phase == "Vapour" then
121       if Basis == "MoleFraction" then
122         for i in 1:Nr loop
123           Kmod[i] = K[i];
124           Kmod[i] = product((xin_c + Ext_r * Scabs) .^ SC_rc[i, 1:Nc]) / (1
                + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
125         end for;
126       else
127         if Basis == "Activity" then
128           for i in 1:Nr loop
129             Kmod[i] = K[i] / (Pin / 101325) ^ Ndel[i];
130             Kmod[i] = product((xin_c + Ext_r * Scabs) .^ SC_rc[i, 1:Nc]) /
                  (1 + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
131           end for;
132         else
133           for i in 1:Nr loop
134             Kmod[i] = K[i] / Pout ^ Ndel[i];
135             Kmod[i] = product((xin_c + Ext_r * Scabs) .^ SC_rc[i, 1:Nc]) /
                  (1 + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
136           end for;
137         end if;
138       end if;
139     else
140       if Basis == "MoleFraction" then
141         for i in 1:Nr loop
142           Kmod[i] = K[i];
143           Kmod[i] = product((xin_c + Ext_r * Scabs) .^ SC_rc[i, 1:Nc]) / (1
                + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
```

```
144           end for;
145         else
146           if Basis == "Activity" then
147             for i in 1:Nr loop
148               Kmod[i] = K[i] / Pout ^ (-Ndel[i]);
149               Kmod[i] = product((Psat .* (xin_c + Ext_r * Scabs)) .^ SC_rc[i,
                      1:Nc]) / (1 + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
150             end for;
151           else
152             for i in 1:Nr loop
153               Kmod[i] = K[i];
154               Kmod[i] = product((Psat .* (xin_c + Ext_r * Scabs)) .^ SC_rc[i,
                      1:Nc]) / (1 + sum(Ext_r * Scabs)) ^ sum(SC_rc[i, 1:Nc]);
155             end for;
156           end if;
157         end if;
158       end if;
159       Fout = (1 + sum(Ext_r * Scabs)) * Fin;
160       for i in 1:Nc loop
161         xout_c[i] = (xin_c[i] + Ext_r * Scabs[1:Nr, i]) * (Fin / Fout);
162       end for;
163       for j in 1:Nr loop
164         for i in 1:Nc loop
165           if SC_rc[j, i] < 0 then
166             X_r[j, i] = (Fin * xin_c[i] - Fout * xout_c[i]) / (Fin * xin_c[i
                    ]);
167           else
168             X_r[j, i] = 0;
169           end if;
170         end for;
171       end for;
172   //
      ================================================================================

173     end EquilibriumReactor;
174   model EquilibriumReaction "Model of an equilibrium reaction used in
        equilibrium reactor"
175       //
          ============================================================================

176       import Simulator.Files.*;
177       import data = Simulator.Files.ChemsepDatabase;
178       parameter Integer Nr "Number of reactions" annotation(
179         Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
              ;
180       parameter Real Coef_cr[Nc, Nr] "Stoichiometric coefficient of
            components" annotation(
181         Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
              ;
182       parameter String Rmode = "ConstantK" "Mode of specifying equilibrium
            constant: ''ConstantK'', ''Tempfunc'',''Gibbs''" annotation(
183         Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
              ;
184       parameter Real Kg[Nr] "Equilibrium Constant, applicable if ConstantK is
             chosen in Rmode" annotation(
185         Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
              ;
186       parameter Real A[Nr, 4] "Coefficient of A in equation logk =(A1 + A2*T
            + A3*T^2 + A4*logT)/(B1 + B2*T + B3*T^2 + B4*logT), applicable if
            Tempfunc is chosen in Rmode" annotation(
187         Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
              ;
188       parameter Real B[Nr, 4] "Coefficient of B in equation logk =(A1 + A2*T
            + A3*T^2 + A4*logT)/(B1 + B2*T + B3*T^2 + B4*logT), applicable if
```

```
          Tempfunc is chosen in Rmode" annotation(
189       Dialog(tab = "Reactions", group = "Equilibrium Reaction Parameters"))
            ;
190     Real T;
191     //Stochiometry of reactions
192     Real Schk_r[Nr];
193     //Returns whether the specified stoichiometry is correct
194     Real Hf_c[Nc];
195     Real Hr_r[Nr];
196     //Equilibrium Constant
197     Real K[Nr](start = xliqg);
198     Real N[Nr](each start = Fg), D[Nr](each start = Fg);
199     extends Simulator.GuessModels.InitialGuess;
200     //Variables for Calculating K from Gibbs Free Energy
201     Real a[Nc], b[Nc], c[Nc], d[Nc];
202     Real DelH25[Nr](each unit = "J/mol") "Standard Enthalpy at 298.15 K";
203     Real DelS25[Nr](each unit = "J/mol.K") "Standard Entropy at 298.15 K";
204     Real DelG25[Nr](each unit = "J/mol.K") "Standard Gibbs Free Energy at
          298.15 K";
205     Real Dela[Nr](each unit = "-");
206     Real Delb[Nr](each unit = "-");
207     Real Delc[Nr](each unit = "-");
208     Real Deld[Nr](each unit = "-");
209     Real CpDelT[Nr](each unit = "J/mol") "Value of Integral Cp*dT";
210     Real P(unit = "Pa") "Pressure at the Outlet";
211     Real CpTDelT[Nr](each unit = "J/mol.K") "Value of Integral (Cp/T)*dT";
212     Real DelH[Nr](each unit = "J/mol") "Enthalpy at Reaction Conditions";
213     Real DelS[Nr](each unit = "J/mol.K") "Entropy at Reaction Conditions";
214     Real DelG[Nr](each unit = "J/mol") "Gibbs Energy at Reaction Conditions
          ";
215     Real Ka[Nr](each unit = "-") "Equillbrium constant";
216   equation
217 //Check of stoichiometric balance
218     Schk_r = Simulator.Files.Models.ReactionManager.Stoichiometrycheck(Nr,
          Nc, C[:].MW, Coef_cr);
219 //Calculation of Heat of Reaction
220     Hf_c[:] = C[:].IGHF .* 1E-3;
221 //
      ================================================================================

222     for i in 1:Nr loop
223       Hr_r[i] = sum(Hf_c[:] .* Coef_cr[:, i]) / Coef_cr[BC_r[1], i];
224     end for;
225     if Rmode == "ConstantK" then
226       K = Kg;
227       for i in 1:Nr loop
228         N[i] = 0;
229         D[i] = 0;
230       end for;
231 //Gibbs Energy Values are assigned the value zero
232       for i in 1:Nc loop
233         a[i] = 0;
234         b[i] = 0;
235         c[i] = 0;
236         d[i] = 0;
237       end for;
238       for i in 1:Nr loop
239         DelH25[i] = 0;
240         DelS25[i] = 0;
241         DelG25[i] = 0;
242         Dela[i] = 0;
243         Delb[i] = 0;
244         Delc[i] = 0;
245         Deld[i] = 0;
```

```
246           CpDelT[i] = 0;
247           CpTDelT[i] = 0;
248           DelH[i] = 0;
249           DelS[i] = 0;
250           DelG[i] = 0;
251           Ka[i] = 0;
252         end for;
253       elseif Rmode == "Gibbs" then
254         if Phase == "Vapour" then
255           (a, b, c, d) = EqReactor.VapCpdata(Nc, C[:].SN);
256         else
257           (a, b, c, d) = EqReactor.LiqCpdata(Nc, C[:].SN);
258         end if;
259         for i in 1:Nr loop
260           DelH25[i] = sum(Coef_cr[:, i] .* C[:].IGHF / 1000);
261           DelS25[i] = sum(Coef_cr[:, i] .* C[:].AS / 1000);
262           DelG25[i] = sum(Coef_cr[:, i] .* C[:].GEF / 1000);
263         end for;
264         for i in 1:Nr loop
265           Dela[i] = sum(Coef_cr[:, i] .* a[:]);
266           Delb[i] = sum(Coef_cr[:, i] .* b[:]);
267           Delc[i] = sum(Coef_cr[:, i] .* c[:]);
268           Deld[i] = sum(Coef_cr[:, i] .* d[:]);
269         end for;
270         for i in 1:Nr loop
271   //Cp=aT^3+bT^2+cT+d
272           CpDelT[i] = Dela[i] * (T ^ 4 - 298.15 ^ 4) * 0.25 + Delb[i] * (T ^
                3 - 298.15 ^ 3) * (1 / 3) + Delc[i] * (T ^ 2 - 298.15 ^ 2) *
                0.5 + Deld[i] * (T - 298.15);
273           CpTDelT[i] = Deld[i] * log(T / 298.15) + Delc[i] * (T - 298.15) +
                Delb[i] * (T ^ 2 - 298.15 ^ 2) * 0.5 + Dela[i] * (T ^ 3 -
                298.15 ^ 3) * (1 / 3);
274         end for;
275         for i in 1:Nr loop
276           DelH[i] = DelH25[i] + CpDelT[i];
277           DelS[i] = DelS25[i] + CpTDelT[i];
278           DelG[i] = DelH[i] - T * DelS[i];
279         end for;
280         for i in 1:Nr loop
281           Ka[i] = exp(-DelG[i] / (8.314 * T));
282         end for;
283         for i in 1:Nr loop
284           N[i] = 0;
285           D[i] = 0;
286           K[i] = Ka[i] / (P / 10 ^ 5) ^ sum(Coef_cr[:, i]);
287         end for;
288       elseif Rmode == "Tempfunc" then
289         for i in 1:Nr loop
290           N[i] = A[i, 1] + A[i, 2] * T + A[i, 3] * T ^ 2 + A[i, 4] * log(T);
291           D[i] = B[i, 1] + B[i, 2] * T + B[i, 3] * T ^ 2 + B[i, 4] * log(T);
292         end for;
293         K = exp(N ./ D);
294         for i in 1:Nc loop
295           a[i] = 0;
296           b[i] = 0;
297           c[i] = 0;
298           d[i] = 0;
299         end for;
300         for i in 1:Nr loop
301           DelH25[i] = 0;
302           DelS25[i] = 0;
303           DelG25[i] = 0;
304           Dela[i] = 0;
305           Delb[i] = 0;
```

```
306           Delc[i] = 0;
307           Deld[i] = 0;
308           CpDelT[i] = 0;
309           CpTDelT[i] = 0;
310           DelH[i] = 0;
311           DelS[i] = 0;
312           DelG[i] = 0;
313           Ka[i] = 0;
314         end for;
315       end if;
316       annotation(
317         Icon(coordinateSystem(initialScale = 0)),
318         Documentation(info = "<html><head></head><body><div>The <b>
              Equilibrium Reaction</b> defined is used in the equilibrium
              reactor for the following purposes:</div><div><ul><li>Check if
              the stoichiometry specified for the reaction is balanced</li><li>
              Calculate the heat of formation</li><li>Calculate the heat of
              reaction</li></ul></div><div><br></div>In an equilibrium reaction
               model, following calculation parameters are defined:<div><ol><li
              >Number of Reactions (<b>Nr</b>)</li><li>Stoichiometric
              Coefficient of Components in Reaction (<b>Coef_cr</b>)</li><li>
              Mode of specifying Equilibrium Constant (<b>Rmode</b>)</li><li>
              Equilibrium Constant (<b>Kg</b>) (<!--StartFragment--><span style
              =\"font-size: 12px;\">If Equilibrium Constant mode is ConstantK</
              span>)</li><li>Temperature function coefficients (<b>A</b> and <b
              >B</b>) (<span style=\"font-size: 12px;\">If Equilibrium
              Constant mode is Tempfunc</span>)</li></ol><div><div style=\"
              font-size: 12px; orphans: 2; widows: 2;\"><span style=\"orphans:
              auto; widows: auto;\">All the above variables are of type <i>
              parameter Real </i>except Mode of specifying Equilibrium Constant
               (<b>Rmode</b>) which </span><span style=\"orphans: auto;
              widows: auto;\">is of type </span><i style=\"orphans: auto;
              widows: auto;\">parameter String</i><span style=\"orphans: auto;
              widows: auto;\">. It can have either of the sting values among
              following:</span></div><!--StartFragment--><div style=\"font-size
              : 12px; orphans: 2; widows: 2;\"><ol><li><b>ConstantK</b>: If the
               equilibrium constant is defined directly</li><li><b>Tempfunc</b
              >: If the equilibrium constant is to be calculated from given
              function of temperature</li></ol></div><!--EndFragment--><div
              style=\"font-size: 12px; orphans: 2; widows: 2;\"><br></div><div
              style=\"font-size: 12px; orphans: 2; widows: 2;\">During
              simulation, their values can specified directly under <b>
              Reactions </b>tab<b> </b>by double clicking on the reactor
              model instance.</div></div></div><div style=\"font-size: 12px;
              orphans: 2; widows: 2;\"><br></div><div style=\"font-size: 12px;
              orphans: 2; widows: 2;\"><br></div><div><br></div></body></html>"
              ));
319   end EquilibriumReaction;
320 end EqReactor;
```