Semester Long Internship Report

On

Modeling, Simulation and Development of Thermodynamic model and Functions in OpenModelica

Submitted by

Patel Bharvi ParulKumar

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

July 24, 2021

Acknowledgement

I would like to extend my deepest thanks to my Internship guide Prof. Kanan M Moudgalya, Department of Chemical Engineering, IIT Bombay, for his support and insight. I would also like to thank our mentors Priyam Nayak and Malapati Sree Harsha with my whole heart, without whom I would not have had the opportunity to work on and experience a fellowship of this caliber. Their continuous support and encouragement has made this learning experience unforgettable.

Contents

1 Introduction to OpenModelica

OpenModelica is a free and open-source modelling environment that uses "Modelica" modelling language. It follows equation oriented approach. OpenModelica can be used for modelling, simulation, optimization and analysis of complex steady state and dynamic systems. Modelica modelling language allows users to express a system in the form of equations. OpenModelica complies expressions, equations, functions and algorithms into C code. The generated C code is combined with a library of utility functions, a run-time library, and a numerical Differential-Algebric Equation (DAE) solver. OpenModelica Connection Editor, called as OMEdit is the integrated Graphical User Interface (GUI) in OpenModelica for graphical modelling and editing. 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 diagrams. OMEdit can be used for creating custom models and for editing or drawing connections between the model interfaces. It also allows users to plot graphs between parameters of the model simulated.

2 Functions

2.1 Replaceable Function

Description:
In OpenModelica replaceable function is used to identify components in a model whose type can be changed (or "redeclared") in the future.
The advantage of using the replaceable keyword is that it allows new models to be created without having to reconnect anything.

In our Simulator, few Unit Operations have to call in flow sheet separately as a model because we have to define their thermodynamic package separately so overcome this problem we use a replaceable function for thermodynamic packages in those Unit operations and used as redeclared in our flow sheet.

I used this function in following Unit Operations;
Flash Separator,
Adiabatic Compressor,
Adiabatic Turbine,
and also in Material Stream.

Example Syntax:
replaceable model thermo = Simulator.Files.ThermodynamicPackages.RaoultsLaw;
redeclare model thermo = Simulator.Files.ThermodynamicPackages.NRTL;

From above syntax we saied that it replace thermodynamic package Raoults law to NRTL.

I also checked this function in flowsheets, it works properly.

2.2 Round Function

Description: Rounding real input signals to the given number of digits after the decimal point. The output is a real. This function may be used in conjunction with lookup tables that must not be interpolated. The table data must then be provide with the set precision.

Instead of a round function, I used direct equations for rounding mole fractions in the material stream.

In our Simulator, sometimes there is an error encountered in a convergence of the flowsheet so I tried to solve this problem with round function and its work for some flowsheet but not for all.

Out.x_pc[1,:] = integer(x_pc[1,:].*10000)./10000;
Out.x_pc[2,:] = x_pc[2,:];
Out.x_pc[3,:] = x_pc[3,:];
I added the following equations in the material stream code instead of this equation,
Out.x_pc=x_pc;
I checked this in one extractive distillation flowsheet.

3 Thermodynamic Packages

3.1 Chao Seader Package

Description:

The Chao seader correlation mainly liquid or vapor H2O because they include special correlations that accurately represent the steam tables. The Chao seader method can be used for light hydrocarbon mixtures.
The Chao Seader (CS) method for heavy hydrocarbons, where the pressure is less than 10342 kPa (1500 psia), and temperatures range between -17.78 and 260°C (0-500°F).

The CS property package is used for the steam systems. The CS property package can also be used for three-phase flashes, but is restricted to the use of pure H2O for the second liquid phase.

The Chao Seader uses the CS-RK Method for VLE calculation. The vapor phase fugacity coefficients are calculated with the Redlich Kwong equation of state.

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

Here,

$$a_i = \frac{0.42748 * R^2 * T_c^{2.5}}{(P_c * T^{0.5})}$$

$$b_i = 0.08664 * R * (T_c / P_c)$$

The pure liquid fugacity coefficients are calculated using the principle of corresponding states.

Development:

- Equations are founded for a thermodynamic model from the DWSIM GitHub code by me.
- The Chao seader model developed as follows;
  - Two separate models are made for fugacity coefficients by me.
  - Liquid phase fugacity coefficient model:
    Equations used in this model:
    * A separate function is made for this model by me for avoiding more equations in the model. Equations used in this function are as follows;
    * The value of Liquid fugacity(ν_i) of component is calculated by this equation;

$$\log \nu_i = \log \nu(0) + (\omega_i * \log \nu(1))$$

Here, ωi is Chao seader accentric factor.

$$\log \nu(0) = A0+A1/Tr+A2Tr+A3Tr^2+A4Tr^3+(A5+A6Tr+A7Tr^2)Pr + (A8 + A9Tr)Pr^2 - logPr$$

$$\log \nu(1) = -4.23893+8.65808Tr-1.22060/Tr-3.15224Tr^3-0.025(Pr-0.6)$$

    * Then the value of activity coefficient(γ_i) is calculated by this equation;

$$\gamma_i = \frac{V_i(\delta_i - \bar{\delta})^2}{RT}$$

Here,

$$\bar{\delta} = \frac{\sum(x_i * V_i * \delta_i)}{\sum(x_i * V_i)}$$

In these equations V_i is Chao seader liquid molar volume and δ_i is solubility parameter.

    * Then the value of the liquid fugacity coefficient(ϕ_{liq}) is calculated by this equation;

$$\phi_{liq} = \nu_i * \gamma_i$$

– Vapour phase fugacity coefficient model:
Equations used in this model:

* Constants of the equation of state are called from their function in and used as input for vapor fugacity coefficient, and their equations are as follows;

$$a_i = \frac{0.42748 * R^2 * T_c^{2.5}}{(P_c * T^{0.5})}$$

$$b_i = 0.08664 * R * (T_c/P_c)$$

$$a_m = \sum(x_i * x_j * a_{ij})$$

$$b_m = \sum(x_i * b_i)$$

* The value of the compressibility factor(Z) is calculated by finding the roots of the following equation,

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

Here,

$$A = \frac{a_m * P}{R^2 * T^2}$$

$$B = \frac{b_m * P}{RT}$$

* Then the value of the vapor fugacity coefficient(ϕ_{vap}) is calculated by this equation;

$$\phi_{vap} = \exp((b_i * (Z - 1)/b_m) + (\ln(Z - B))$$
$$+ (\frac{A}{(B * 1.4142)} * (\frac{b_i}{b_m} - 2 * (\frac{a_i}{a_m})^{0.5})) * (\ln(\frac{(2 * Z + B * (2.4142))}{(2 * Z + B * 0.4142))})$$

* Assumed activity and fugacity coefficients as one at bubble and dew point.
* In the main model first called both models and get values for both fugacity coefficients.
* Then calculate an equilibrium constant by this equation,

$$Ki = \frac{y_i}{x_i} = \frac{\phi_{liq}}{\phi_{vap}}$$

Error Encountered:

- The values of Bubble pressure and Dew pressure at stream temperature are not converged properly.

I checked Chao Seader thermodynamic package for Methane, Ethane, N-pentane and also for Methane, N-butane, N-pentane and propane and checked values with DWSIM.

Results are as follows;

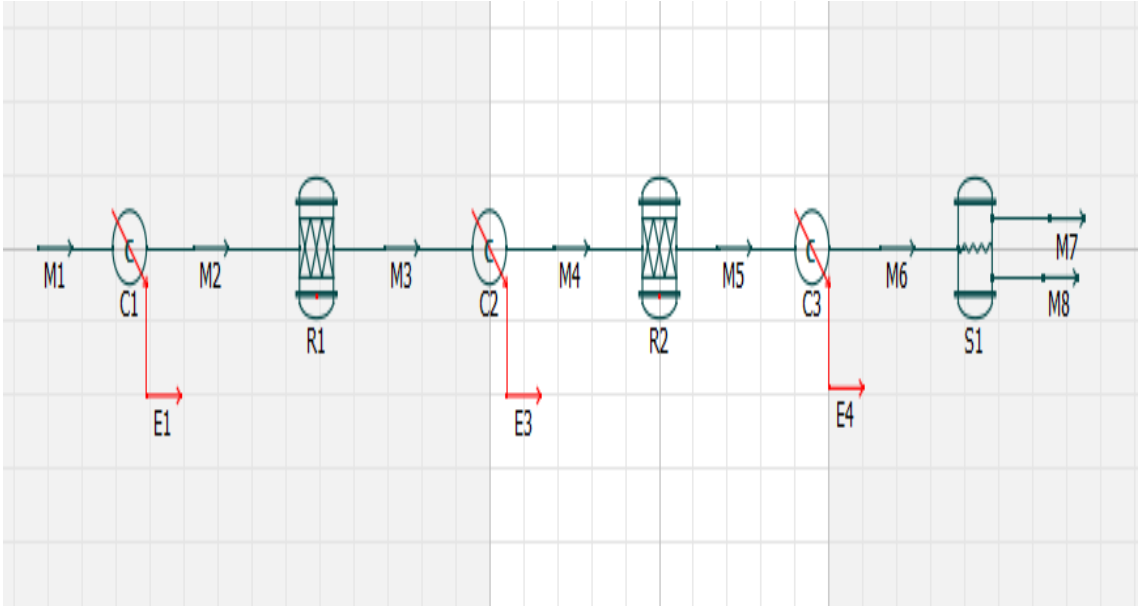
		Fp	100	DWSIM								Open Modelica									
Sr no	Compounds	T	P	Mole fraction			xvap	fugacity coefficient		liquid	vapour	Mole fraction			xvap	fugacity coefficient				Pbubl	Pdew
				mixture	liquid	vapour		liduid	vapour	mole flowrate	mole flowrate	xpc1	xpc2	xpc3		liduid	vapour	fpc2	fpc3		
1	methane	168.751	506625	0.25	0.25	0	0	3.68897				0.25	0.2499	0.973	8.99E-07	3.68898	0.947	25	8.75E-05	5.67E+05	9.40E+00
	ethane	Pbubl	506625	0.25	0.25	0		0.0910947				0.25	0.25	0.0264		0.09109	0.8597	25	2.38E+06		
	n-pentane			0.5	0.5	0		3.63E-05				0.5	0.5	2.66E-05		3.63E-05	0.6811	5.00E+01	2.39E-09		
2	methane	338.485	506625	0.25	0	0.25	1		1.02202			0.25	0.00762	0.25	1	0.10945	1.02202	4.89E-08	2.50E+01	4.13E+07	4.97E+05
	ethane	Pdew	506625	0.25	0	0.25			0.980738			0.25	0.02591	0.25		0.50815	0.98074	1.66E-07	2.50E+01		
	n-pentane			0.5	0	0.5			0.888871			0.5	0.96647	0.5		0.92526	0.88887	6.20E-06	5.00E+01		
3	methane	289.647	506625	0.25	0.01892	0.48108	0.5	25.3175	0.995855			0.25	0.01892	0.48108	0.5	25.317	0.99586	9.46E-01	2.41E+01	1.39E+07	9.70E+04
	ethane			0.25	0.0871	0.4129		4.54883	0.95954			0.25	0.0871	0.4129		4.54883	0.95954	4.35E+00	2.06E+01		
	n-pentane			0.5	0.89397	0.10602		0.104225	0.878827			0.5	0.89398	0.10602		0.10423	0.87883	4.47E+01	5.30108		
4	methane	325.619	506625	0.25	0.0105	0.32981	0.75	31.5875	1.0117			0.25	0.01056	0.32981	0.74999			2.26E-01	24.736	3.09E+07	341311
	ethane			0.25	0.03918	0.32027		7.95436	0.972938			0.25	0.03917	0.32028				9.79E-01	24.0206		
	n-pentane			0.5	0.95203	0.34991		0.326881	0.887726			0.5	0.95203	0.34991				2.38E+01	26.2428		
5	methane	200	883006	0.2	0.2	0	0	4.68075		20		0.2	0.2	0.99188	1.25E-07	4.68075	0.9438	2.00E+01	1.24E-05	883006	428.383
	n-butane	Pbubl	883006	0.2	0.2	0		3.23E-03		20		0.2	0.2	0.00091		0.00323	0.71285	2.00E+01	1.13E-08		
	n-pentane			0.4	0.4	0		4.57E-04		40		0.4	0.4	0.00028		0.00046	0.65638	4.00E+01	3.47E-09		
	propane			0.2	0.2	0		0.0268808		20		0.2	0.2	0.00693		0.02688	0.7763	2.00E+01	8.63E-08		
6	methane	400	2.28E+06	0.2	0	0.2	1		1.14092		20	0.2	0.02498	0.2	0.99998	9.1316	1.1409	4.53E-06	2.00E+01	1.36E+08	2.03E+06
	n-butane	Pdew	2.28E+06	0.2	0	0.2			0.766811		20	0.2	0.19062	0.2		0.80405	0.76681	3.46E-05	2.00E+01		
	n-pentane			0.4	0	0.4			0.68292		40	0.4	0.68025	0.4		0.40157	0.68292	1.23E-04	4.00E+01		
	propane			0.2	0	0.2			0.86474		20	0.2	0.10413	0.2		1.66088	0.86474	1.89E-05	2.00E+01		
7	methane	200	6262.37	0.2	0.00078	0.4918	0.4	641.1064	0.999935	0.046677	19.9538	0.2	0.00078	0.49884	0.4	641.113	0.99935	0.04667	19.9533	1.21E+06	428.383
	n-butane			0.2	0.25953	0.11068		0.424924	0.996394	15.57	4.4274	0.2	0.25953	0.11069		0.42496	0.99639	45.574	4.42758		
	n-pentane			0.4	0.64106	0.03839		0.0596065	0.995339	38.463	1.53558	0.4	0.64107	0.03839		0.05961	0.99534	38.4643	1.5357		
	propane			0.2	0.09863	0.35207		3.56073	0.997477	5.9172	14.083	0.2	0.09862	0.35207		3.561	0.99748	5.91726	14.0827		

4 Flowsheets in OpenModelica

Production of Hydrogen using Water Gas Shift Reaction (WGSR)

BACKGROND:
Water Gas Shift Reaction is a major part of Steam Methane Reforming process, the flue gases after methanation has Carbon Monoxide and major part of unconverted steam. WGSR increases the yield of Hydrogen produced by reacting carbon monoxide with the remaining steam. This process is important for two reasons, firstly, it increases yield. Secondly, converts CO to CO2, hence the emission of poisonous CO gas is avoided.
WGSR is divided into two parts, High Temperature Shift(HTS) and Low Temperature Shift(LTS).Flues gases from SMR enters HTS Reactor which operates at 500-700K, where it converts remaining steam by 35%, then the overhead is passed through a cooler to bring down the temperature to 400K. Later, the cooled stream is taken into LTS Reactor which operates at 300-400k, this is industrially used to increase the H2 yield by 2-3%. The vapor outlet is cooled and sent to Gas-Liq Separator, where remaining steam is condensed to water and the product gas is taken out, Pressure Swing Adsorption is done and the obtained Hydrogen is liquified.

FLOWSHEET:



PROBLEM DURING SIMULATING FLOWSHEET:
In Open Modelica one problem occurred while simulating the flowsheet, Flash column did not give proper results. When the Flash column was simulated separately it gave perfect results. To overcome the problem, results of separate Flash column were given as inputs of guess model in the Output stream of Flash column.

RESULTS:

High Temperature Shift Reaction:

Open Modelica		DWSIM	
Compound	Amount	Compound	Amount
Methane	0.0161393	Methane	0.016139308
Oxygen	0.00743258	Oxygen	0.0074325759
Nitrogen	0.2835	Nitrogen	0.28349968
Water	0.16564	Water	0.16564026
Hydrogen	0.394988	Hydrogen	0.39498832
Carbon Monoxide	0.0000000034	Carbon Monoxide	2.3093299E-13
Carbon Dioxide	0.1323	Carbon dioxide	0.13229985
Molar Flowrate	4.709	Molar Flowrate	4.70899

Low Temperature Shift Reaction:

Open Modelica		DWSIM	
Compound	Amount	Compound	Amount
Methane	0.0161393	Methane	0.016139308
Oxygen	0.00743258	Oxygen	0.0074325759
Nitrogen	0.2853	Nitrogen	0.28349968
Water	0.1654	Water	0.16564026
Hydrogen	0.394988	Hydrogen	0.39498832
Carbon Monoxide	0.0000000034	Carbon Monoxide	4.9595777E-15
Carbon Dioxide	0.1323	Carbon Dioxide	0.13229985
Molar Flowrate	4.709	Molar Flowrate	4.70899

Product Gas:

Open Modelica		DWSIM	
Compound	Amount	Compound	Amount
Methane	0.019211	Methane	0.019212208
Oxygen	0.00885397	Oxygen	0.0088477164
Nitrogen	0.337785	Nitrogen	0.33747781
Water	0.0077521	Water	0.0068413518
Hydrogen	0.470638	Hydrogen	0.47019373
Carbon Monoxide	4.049E-8	Carbon Monoxide	5.9038776E-15
Carbon Dioxide	0.155764	Carbon Dioxide	0.15742718
Molar Flowrate	3.9520788	Molar Flowrate	3.95581

5 Open Modelica Code

5.1 Replaceable function code:

Material stream

```
model MaterialStream "Model representing Material Stream"
//1 -  Mixture, 2 - Liquid phase, 3 - Gas Phase
extends Simulator.Files.Icons.MaterialStream;
replaceable model thermo = Simulator.Files.ThermodynamicPackages.RaoultsLaw;
extends thermo;
import Simulator.Files.*;
parameter Integer Nc "Number of components" annotation(
 Dialog(tab = "Stream Specifications", group = "Component Parameters"));
parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "Component instances array" annotation(
 Dialog(tab = "Stream Specifications", group = "Component Parameters"));
Real P(unit = "Pa", min = 0, start = Pg) "Pressure";
Real T(unit = "K", start = Tg) "Temperature";
Real Pbubl(unit = "Pa", min = 0, start = Pmin) "Bubble point pressure";
Real Pdew(unit = "Pa", min = 0, start = Pmax) "dew point pressure";
Real xliq(unit = "-", start = xliqg, min = 0, max = 1) "Liquid Phase mole fraction";
Real xvap(unit = "-", start = xvapg, min = 0, max = 1) "Vapor Phase mole fraction";
Real xmliq(unit = "-", start = xliqg, min = 0, max = 1) "Liquid Phase mass fraction";
Real xmvap(unit = "-",start =xvapg, min = 0, max = 1) "Vapor Phase Mass fraction";
Real F_p[3](each unit = "mol/s", each min = 0, start={Fg,Fliqg,Fvapg}) "Total molar flow in
```

```
    phase";
20  Real Fm_p[3](each unit = "kg/s", each min = 0,
    each start = Fg) "Total mass flow in phase";
21  Real MW_p[3](each unit = "-", each start = 0,
    each min = 0) "Average Molecular weight in
    phase";
22  Real x_pc[3, Nc](each unit = "-", each min = 0,
    each max = 1, start={xguess,xg,yg}) "Component
    mole fraction in phase";
23  Real xm_pc[3, Nc](each unit ="-", start={xguess,
    xg,yg}, each min = 0, each max = 1) "Component
    mass fraction in phase";
24  Real F_pc[3, Nc](each unit = "mol/s", each start
     = Fg, each min = 0) "Component molar flow in
    phase";
25  Real Fm_pc[3, Nc](each unit = "kg/s", each min =
     0, each start = Fg) "Component mass flow in
    phase";
26  Real Cp_p[3](each unit = "kJ/[kmol.K]",start={
    Hmixg,Hliqg,Hvapg}) "Phase molar specific heat"
    ;
27  Real Cp_pc[3, Nc](each unit = "kJ/[kmol.K]") "
    Component molar specific heat in phase";
28  Real H_p[3](each unit = "kJ/kmol",start={Hmixg,
    Hliqg,Hvapg}) "Phase molar enthalpy";
29  Real H_pc[3, Nc](each unit = "kJ/kmol") "
    Component molar enthalpy in phase";
30  Real S_p[3](each unit = "kJ/[kmol.K]") "Phase
    molar entropy";
31  Real S_pc[3, Nc](each unit = "kJ/[kmol.K]") "
    Component molar entropy in phase";
32  Simulator.Files.Interfaces.matConn In(Nc = Nc)
    annotation(
33   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)));
```

```
34  Simulator.Files.Interfaces.matConn Out(Nc = Nc)
      annotation(
35     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)));
36
37  extends Simulator.GuessModels.InitialGuess;
38
39 equation
40 //Connector equations
41  In.P = P;
42  In.T = T;
43  In.F = F_p[1];
44  In.H = H_p[1];
45  In.S = S_p[1];
46  In.x_pc = x_pc;
47  In.xvap = xvap;
48  Out.P = P;
49  Out.T = T;
50  Out.F = F_p[1];
51  Out.H = H_p[1];
52  Out.S = S_p[1];
53  Out.x_pc = x_pc;
54  Out.xvap = xvap;
55 //
     ==========================================================

56 //Mole Balance
57  F_p[1] = F_p[2] + F_p[3];
58 //   x_pc[1, :] .* F_p[1] = x_pc[2, :] .* F_p[2] +
       x_pc[3, :] .* F_p[3];
59 //component molar and mass flows
60  for i in 1:Nc loop
61    F_pc[:, i] = x_pc[:, i] .* F_p[:];
62  end for;
63  if P >= Pbubl then
```

```
64 //below bubble point region
65    xm_pc[3, :] = zeros(Nc);
66    Fm_pc[1, :] = xm_pc[1, :] .* Fm_p[1];
67    xm_pc[2, :] = xm_pc[1, :];
68  elseif P >= Pdew then
69    for i in 1:Nc loop
70      Fm_pc[:, i] = xm_pc[:, i] .* Fm_p[:];
71    end for;
72  else
73 //above dew point region
74    xm_pc[2, :] = zeros(Nc);
75    Fm_pc[1, :] = xm_pc[1, :] .* Fm_p[1];
76    xm_pc[3, :] = xm_pc[1, :];
77  end if;
78 //phase molar and mass fractions
79  xliq = F_p[2] / F_p[1];
80  xvap = F_p[3] / F_p[1];
81  xmliq = Fm_p[2] / Fm_p[1];
82  xmvap = Fm_p[3] / Fm_p[1];
83 //Conversion between mole and mass flow
84  for i in 1:Nc loop
85    Fm_pc[:, i] = F_pc[:, i] * C[i].MW;
86  end for;
87  Fm_p[:] = F_p[:] .* MW_p[:];
88 //Energy Balance
89  for i in 1:Nc loop
90 //Specific Heat and Enthalpy calculation
91    Cp_pc[2, i] = ThermodynamicFunctions.LiqCpId(C
    [i].LiqCp, T);
92    Cp_pc[3, i] = ThermodynamicFunctions.VapCpId(C
    [i].VapCp, T);
93    H_pc[2, i] = ThermodynamicFunctions.HLiqId(C[i
    ].SH, C[i].VapCp, C[i].HOV, C[i].Tc, T);
94    H_pc[3, i] = ThermodynamicFunctions.HVapId(C[i
    ].SH, C[i].VapCp, C[i].HOV, C[i].Tc, T);
95    (S_pc[2, i], S_pc[3, i]) =
    ThermodynamicFunctions.SId(C[i].VapCp, C[i].HOV
    , C[i].Tb, C[i].Tc, T, P, x_pc[2, i], x_pc[3, i
```

```
    ]);
96  end for;
97  for i in 2:3 loop
98    Cp_p[i] = sum(x_pc[i, :] .* Cp_pc[i, :]) +
    Cpres_p[i];
99    H_p[i] = sum(x_pc[i, :] .* H_pc[i, :]) +
    Hres_p[i];
100   S_p[i] = sum(x_pc[i, :] .* S_pc[i, :]) +
    Sres_p[i];
101 end for;
102 Cp_p[1] = xliq * Cp_p[2] + xvap * Cp_p[3];
103 Cp_pc[1, :] = x_pc[1, :] .* Cp_p[1];
104 H_p[1] = xliq * H_p[2] + xvap * H_p[3];
105 H_pc[1, :] = x_pc[1, :] .* H_p[1];
106 S_p[1] = xliq * S_p[2] + xvap * S_p[3];
107 S_pc[1, :] = x_pc[1, :] * S_p[1];
108 //Bubble point calculation
109 Pbubl = sum(gmabubl_c[:] .* x_pc[1, :] .* exp(C
    [:].VP[2] + C[:].VP[3] / T + C[:].VP[4] * log(T
    ) + C[:].VP[5] .* T .^ C[:].VP[6]) ./
    philiqbubl_c[:]);
110 //Dew point calculation
111 Pdew = 1 / sum(x_pc[1, :] ./ (gmadew_c[:] .* exp
    (C[:].VP[2] + C[:].VP[3] / T + C[:].VP[4] * log
    (T) + C[:].VP[5] .* T .^ C[:].VP[6])) .*
    phivapdew_c[:]);
112 if P >= Pbubl then
113 //below bubble point region
114   x_pc[3, :] = zeros(Nc);
115 //    sum(x_pc[2, :]) = 1;
116   F_p[3] = 0;
117   x_pc[2, :] = x_pc[1, :];
118 elseif P >= Pdew then
119 //VLE region
120   for i in 1:Nc loop
121     x_pc[3, i] = K_c[i] * x_pc[2, i];
122     x_pc[2, i] = x_pc[1, i] ./ (1 + xvap * (K_c[
    i] - 1));
```

```
123     end for;
124     sum(x_pc[3, :]) = 1;
125 //sum y = 1
126  else
127 //above dew point region
128     x_pc[2, :] = zeros(Nc);
129 //     sum(x_pc[3, :]) = 1;
130     F_p[2] = 0;
131     x_pc[3, :] = x_pc[1, :];
132  end if;
133 algorithm
134  for i in 1:Nc loop
135     MW_p[:] := MW_p[:] + C[i].MW * x_pc[:, i];
136  end for;
137     end MaterialStream;
```

Flash Separator

```
model Flash "Model of a flash column to separate vapor and liquid phases from a mixed phase material stream"
//=============================================================

//Header Files and Parameters

extends Simulator.Files.Icons.Flash;
replaceable model thermo = Simulator.Files.ThermodynamicPackages.RaoultsLaw;
extends thermo;
import Simulator.Files.*;
parameter ChemsepDatabase.GeneralProperties C[Nc] "Component instances array" annotation(
  Dialog(tab = "Flash Specifications", group = "Component Parameters"));
parameter Integer Nc "Number of components" annotation(
  Dialog(tab = "Flash Specifications", group = "Component Parameters"));
parameter Boolean BTdef = false "True if flash is operated at temperature other than feed temp else false" annotation(
  Dialog(tab = "Flash Specifications", group = "Calculation Parameters"));
parameter Boolean BPdef = false "True if flash is operated at pressure other than feed pressure else false" annotation(
  Dialog(tab = "Flash Specifications", group = "Calculation Parameters"));
parameter Real Tdef(unit = "K") = 298.15 "Separation temperature if BTdef is true" annotation(
  Dialog(tab = "Flash Specifications", group = "Calculation Parameters"));
parameter Real Pdef(unit = "Pa") = 101325 "
```

```
    Separation pressure if BPdef is true"
    annotation(
20   Dialog(tab = "Flash Specifications", group = "
    Calculation Parameters"));
21 //
    ============================================================

22 //Model Variables
23 Real T(unit = "K", start = Tg, min = 0) "Flash
    column temperature";
24 Real P(unit = "Pa", start = Pg, min = 0) "Flash
    column pressure";
25 Real Pbubl(unit = "Pa", min = 0, start = Pmin) "
    Bubble point pressure";
26 Real Pdew(unit = "Pa", min = 0, start = Pmax) "
    Dew point pressure";
27 Real F_p[3](each unit = "mol/s", each min = 0,
    start = {Fg,Fliqg,Fvapg})"Feed stream mole flow
    ";
28 Real x_pc[3, Nc](each unit = "-", each min = 0,
    each max = 1, start={xguess,xg,yg}) "Component
    mole fraction";
29 Real Cp_pc[3, Nc](each unit = "kJ/[kmol.K]") "
    Component molar specific heat";
30 Real H_pc[3, Nc](each unit = "kJ/kmol") "Comopent
     molar enthalpy";
31 Real S_pc[3, Nc](each unit = "kJ/[kmol.K]") "
    Component molar entropy";
32 Real Cp_p[3](each unit = "kJ/[kmol.K]") "Molar
    specific heat in phase";
33 Real H_p[3](each unit = "kJ/kmol") "Molar
    enthalpy in phase";
34 Real S_p[3](each unit = "kJ/[kmol.K]") "Molar
    entropy in phase";
35 Real xliq(unit = "-", min = 0, max = 1, start =
    xliqg)"Liquid phase mole fraction";
36 Real xvap(unit = "-", min = 0, max = 1, start =
    xvapg) "Vapor phase mole fraction";
```

```
37 //
     ================================================================

38 //Instantiation of Connectors
39 Simulator.Files.Interfaces.matConn In(Nc = Nc)
     annotation(
40   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)));
41 Simulator.Files.Interfaces.matConn Out1(Nc = Nc)
     annotation(
42   Placement(visible = true, transformation(origin
      = {102, 72}, extent = {{-10, -10}, {10, 10}},
     rotation = 0), iconTransformation(origin =
     {100, 80}, extent = {{-10, -10}, {10, 10}},
     rotation = 0)));
43 Simulator.Files.Interfaces.matConn Out2(Nc = Nc)
     annotation(
44   Placement(visible = true, transformation(origin
      = {100, -72}, extent = {{-10, -10}, {10, 10}},
      rotation = 0), iconTransformation(origin =
     {100, -80}, extent = {{-10, -10}, {10, 10}},
     rotation = 0)));
45
46 extends Simulator.GuessModels.InitialGuess;
47
48 equation
49 //
     ================================================================

50 //Connector equation
51 if BTdef then
52   Tdef = T;
53 else
54   In.T = T;
55 end if;
```

```
56 if BPdef then
57   Pdef = P;
58 else
59   In.P = P;
60 end if;
61 In.F = F_p[1];
62 In.x_pc[1, :] = x_pc[1, :];
63 Out2.T = T;
64 Out2.P = P;
65 Out2.F = F_p[2];
66 Out2.x_pc[1, :] = x_pc[2, :];
67 Out1.T = T;
68 Out1.P = P;
69 Out1.F = F_p[3];
70 Out1.x_pc[1, :] = x_pc[3, :];
71 //
     ============================================================

72 //Mole Balance
73 F_p[1] = F_p[2] + F_p[3];
74 x_pc[1, :] .* F_p[1] = x_pc[2, :] .* F_p[2] +
     x_pc[3, :] .* F_p[3];
75 //
     ============================================================

76 //Bubble point calculation
77 Pbubl = sum(gmabubl_c[:] .* x_pc[1, :] .* exp(C
     [:].VP[2] + C[:].VP[3] / T + C[:].VP[4] * log(T
     ) + C[:].VP[5] .* T .^ C[:].VP[6]) ./
     philiqbubl_c[:]);
78 //
     ============================================================

79 //Dew point calculation
80 Pdew = 1 / sum(x_pc[1, :] ./ (gmadew_c[:] .* exp(
     C[:].VP[2] + C[:].VP[3] / T + C[:].VP[4] * log(
     T) + C[:].VP[5] .* T .^ C[:].VP[6])) .*
     phivapdew_c[:]);
```

```
81 if P >= Pbubl then
82   x_pc[3, :] = zeros(Nc);
83   F_p[3] = 0;
84 elseif P >= Pdew then
85 //
     ============================================================

86 //VLE region
87   for i in 1:Nc loop
88     x_pc[2, i] = x_pc[1, i] ./ (1 + xvap * (K_c[i
     ] - 1));
89   end for;
90   sum(x_pc[2, :]) = 1;
91 else
92 //
     ============================================================

93 //above dew point region
94   x_pc[2, :] = zeros(Nc);
95   F_p[2] = 0;
96 end if;
97 //
     ============================================================

98 //Energy Balance / Specific Heat and Enthalpy
     calculation from Thermodynamic Functions
99 for i in 1:Nc loop
100   Cp_pc[2, i] = ThermodynamicFunctions.LiqCpId(C[
     i].LiqCp, T);
101   Cp_pc[3, i] = ThermodynamicFunctions.VapCpId(C[
     i].VapCp, T);
102   H_pc[2, i] = ThermodynamicFunctions.HLiqId(C[i
     ].SH, C[i].VapCp, C[i].HOV, C[i].Tc, T);
103   H_pc[3, i] = ThermodynamicFunctions.HVapId(C[i
     ].SH, C[i].VapCp, C[i].HOV, C[i].Tc, T);
104   (S_pc[2, i], S_pc[3, i]) =
     ThermodynamicFunctions.SId(C[i].VapCp, C[i].HOV
     , C[i].Tb, C[i].Tc, T, P, x_pc[2, i], x_pc[3, i
```

```
    ]);
105 end for;
106 //
    ================================================================

107 //Specific Heat and Enthalpy calculation for
    Liquid and Vapor Phase
108 for i in 2:3 loop
109   Cp_p[i] = sum(x_pc[i, :] .* Cp_pc[i, :]) +
    Cpres_p[i];
110   H_p[i] = sum(x_pc[i, :] .* H_pc[i, :]) + Hres_p
    [i];
111   S_p[i] = sum(x_pc[i, :] .* S_pc[i, :]) + Sres_p
    [i];
112 end for;
113 //
    ================================================================

114 //Specific Heat and Enthalpy calculation for
    Mixture Phase
115 Cp_p[1] = xliq * Cp_p[2] + xvap * Cp_p[3];
116 Cp_pc[1, :] = x_pc[1, :] .* Cp_p[1];
117 H_p[1] = xliq * H_p[2] + xvap * H_p[3];
118 H_pc[1, :] = x_pc[1, :] .* H_p[1];
119 S_p[1] = xliq * S_p[2] + xvap * S_p[3];
120 S_pc[1, :] = x_pc[1, :] * S_p[1];
121 //
    ================================================================

122 //phase molar fractions
123 xliq = F_p[2] / F_p[1];
124 xvap = F_p[3] / F_p[1];
125
126 end Flash;
```

Adiabatic Compressor

```
1 model AdiabaticCompressor "Model of an adiabatic
    compressor to provide energy to vapor stream in
     form of pressure"
2   extends Simulator.Files.Icons.
    AdiabaticCompressor;
3   replaceable model thermo = Simulator.Files.
    ThermodynamicPackages.RaoultsLaw;
4   extends thermo;
5   extends Simulator.Files.Models.Flash;
6   parameter Simulator.Files.ChemsepDatabase.
    GeneralProperties C[Nc] "Component instances
    array" annotation(
7     Dialog(tab = "Compressor Specifications",
    group = "Component Parameters"));
8   parameter Integer Nc "number of components"
    annotation(
9     Dialog(tab = "Compressor Specifications",
    group = "Component Parameters"));
10
11 //
    ============================================================

12   Real Fin(unit = "mol/s", min = 0, start = Fg) "
    Inlet stream molar flow rate";
13   Real Pin(unit = "Pa", min = 0, start = Pg) "
    Inlet stream pressure";
14   Real Tin(unit = "K", min = 0, start = Tg) "
    Inlet stream temperature";
15   Real Hin(unit = "kJ/kmol",start=Htotg) "Inlet
    stream molar enthalpy";
16   Real Sin(unit = "kJ/[kmol/K]") "Inlet stream
    molar entropy";
17   Real xvapin(unit = "-", min = 0, max = 1, start
     = xvapg) "Inlet stream vapor phase mol
    fraction";
18
19   Real Fout(min = 0, start = Fg) "Outlet stream
```

```
    molar flow rate";
20  Real Q(unit = "W") "Power required";
21  Real Pdel(unit = "Pa") "Pressure increase";
22  Real Tdel(unit = "K") "Temperature increase";
23
24  Real Pout(unit = "Pa", min = 0, start = Pg) "
    Outlet stream pressure";
25  Real Tout(unit = "Pa", min = 0, start = Tg) "
    Outlet stream temperature";
26  Real Hout(unit = "kJ/kmol",start=Htotg) "Outlet
     stream molar enthalpy";
27  Real Sout(unit = "kJ/[kmol.K]") "Outlet stream
    molar entropy";
28  Real xvapout(unit = "-", min = 0, max = 1,
    start = xvapg) "Outlet stream vapor phase mole
    fraction";
29  Real x_c[Nc](each unit = "-", each min = 0,
    each max = 1,start=xg) "Component mole fraction
    ";
30
31  parameter Real Eff(unit = "-") "Efficiency"
    annotation(
32    Dialog(tab = "Compressor Specifications",
    group = "Calculation Parameters"));
33
34 //
    ================================================================

35  Simulator.Files.Interfaces.matConn In(Nc = Nc)
    annotation(
36    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)));
37  Simulator.Files.Interfaces.matConn Out(Nc = Nc)
     annotation(
38    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)));
39  Simulator.Files.Interfaces.enConn En annotation
    (
40    Placement(visible = true, transformation(
    origin = {0, -100}, extent = {{-10, -10}, {10,
    10}}, rotation = 0), iconTransformation(origin
    = {0, -66}, extent = {{-10, -10}, {10, 10}},
    rotation = 0)));
41  //
    ============================================================

42  extends Simulator.GuessModels.InitialGuess;
43
44 equation
45 //connector equations
46  In.P = Pin;
47  In.T = Tin;
48  In.F = Fin;
49  In.H = Hin;
50  In.S = Sin;
51  In.x_pc[1, :] = x_c[:];
52  In.xvap = xvapin;
53  Out.P = Pout;
54  Out.T = Tout;
55  Out.F = Fout;
56  Out.H = Hout;
57  Out.S = Sout;
58  Out.x_pc[1, :] = x_c[:];
59  Out.xvap = xvapout;
60  En.Q = Q;
61 //
    ============================================================

62  Fin = Fout;
63 //material balance
```

```
64    Hout = Hin + (H_p[1] - Hin) / Eff;
65    Q = Fin * (H_p[1] - Hin) / Eff;
66  //energy balance
67    Pin + Pdel = Pout;
68  //pressure calculation
69    Tin + Tdel = Tout;
70  //temperature calculation
71  //
      ============================================================

72  //ideal flash
73    Fin = F_p[1];
74    Pout = P;
75    Sin = S_p[1];
76    x_c[:] = x_pc[1, :];
77      end AdiabaticCompressor;
```

Adiabatic Turbine(Expander)

```
model AdiabaticExpander "Model of an adiabatic expander to extract energy from a vapor stream in form of pressure"
//================================================================

//Header Files and Parameters
extends Simulator.Files.Icons.AdiabaticExpander;
replaceable model thermo = Simulator.Files.ThermodynamicPackages.RaoultsLaw;
extends thermo;
extends Simulator.Files.Models.Flash;
parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "Component instances array" annotation(
Dialog(tab = "Expander Specifications", group = "Component Parameters"));
parameter Integer Nc "Number of components" annotation(
Dialog(tab = "Expander Specifications", group = "Component Parameters"));
parameter Real Eff(unit = "-") "Expander efficiency" annotation(
Dialog(tab = "Expander Specifications", group = "Calculation Parameters"));
//================================================================

//Model Variables
Real Fin(unit = "mol/s", min = 0, start = Fg) "Inlet stream molar flow rate";
Real Tin(unit = "K", min = 0, start = Tg) "Inlet stream temperature";
Real Hin(unit = "kJ/kmol",start=Htotg) "Inlet stream molar enthalpy";
Real xvapin(unit = "-", min = 0, max = 1, start = xvapg) "Inlet stream vapor phase mole fraction
```

```
    ";
20 Real xin_c[Nc](each unit = "-", each min = 0,
    each max = 1, start=xg) "Component mole
    fraction";
21 Real Sin(unit = "kJ/[kmol.K]") "Inlet stream
    molar entropy";
22 Real Pin(unit = "Pa", min = 0, start = Pg) "Inlet
     stream pressure";
23 Real Q(unit = "W") "Generated Power";
24 Real Pdel(unit = "Pa") "Pressure drop";
25 Real Tdel(unit = "K") "Temperature change";
26 Real Pout(unit = "Pa", min = 0, start = Pg) "
    Outlet stream pressure";
27 Real Fout(unit = "mol/s", min = 0, start = Fg) "
    Outlet stream molar flow rate";
28 Real Tout(unit = "K", min = 0, start = Tg) "
    Outlet stream temperature";
29 Real Hout(unit = "kJ/kmol") "Outlet stream molar
    enthalpy";
30 Real Sout(unit = "kJ/[kmol.k]") "Outlet stream
    molar entropy";
31 Real xvapout(unit = "-", min = 0, max = 1, start
    = xvapg) "Outlet stream vapor phase mole
    fraction";
32 //
    ============================================================

33 //Instantiation of connectors
34 Simulator.Files.Interfaces.matConn In(Nc = Nc)
    annotation(
35 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)));
36 Simulator.Files.Interfaces.matConn Out(Nc = Nc)
    annotation(
37 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)));
38 Simulator.Files.Interfaces.enConn En annotation(
39 Placement(visible = true, transformation(origin =
    {0, -100}, extent = {{-10, -10}, {10, 10}},
    rotation = 0), iconTransformation(origin = {0,
    -66}, extent = {{-10, -10}, {10, 10}}, rotation
    = 0)));
40
41 extends Simulator.GuessModels.InitialGuess;
42
43 equation
44 //
    ============================================================

45 //connector equations
46 In.P = Pin;
47 In.T = Tin;
48 In.F = Fin;
49 In.H = Hin;
50 In.S = Sin;
51 In.x_pc[1, :] = xin_c[:];
52 In.xvap = xvapin;
53 Out.P = Pout;
54 Out.T = Tout;
55 Out.F = Fout;
56 Out.H = Hout;
57 Out.S = Sout;
58 Out.x_pc[1, :] = xin_c[:];
59 Out.xvap = xvapout;
60 En.Q = Q;
61 //
    ============================================================

62 //Material and Energy balance
63 Fin = Fout;
```

```
64 Hout = Hin + (H_p[1] - Hin) * Eff;
65 Q = Fin * (H_p[1] - Hin) * Eff;
66 //
     ============================================================

67 //Pressure and Temperature calculation
68 Pin - Pdel = Pout;
69 Tin - Tdel = Tout;
70 //
     ============================================================

71 //Ideal flash
72 Fin = F_p[1];
73 Pout = P;
74 Sin = S_p[1];
75 xin_c[:] = x_pc[1, :];
76
77 end AdiabaticExpander;
```

5.2 Chao Seader model code:

```
model ChaoSeader
  //Header Files and Parameters
  import Simulator.Files.Thermodynamic_Functions.*;
  parameter Real R(each unit = "J.K^-1.mol^-1") = 8.314 "Ideal Gas Constant";
  parameter Real u = 1;
  import Simulator.Files.*;
  parameter Real W_c[Nc] = C.ChaoSeadAF "Chao-Seader Accentric Factor";
  parameter Real SP_c[Nc](each unit = "(J/m3)^0.5") = C.ChaoSeadSP "Chao-Seader Solubility Parameter";
  parameter Real V_c[Nc](each unit = "m3/kmol") = C.ChaoSeadLV "Chao-Seader Liquid Volume";
  parameter Real T_c[Nc] = C.Tc;
  parameter Real Pc_c[Nc] = C.Pc;
  parameter Real Rgas(each unit = "erg.K^-1.mol^-1") = 8314470 "Ideal Gas Constant";
  //============================================================

  //Model Variables
  //Excess Energy Properties
  Real Cpres_p[3], Hres_p[3], Sres_p[3];
  Real K_c[Nc] "Equlibrium constant of compound i";
  Real gma_c[Nc] "Activity Coefficent";
  Real philiq_c[Nc] "Liquid Phase Fugacity Coefficent", phivap_c[Nc] "Vapour Phase Fugacity Coefficient";
  //Fugacity coefficient  at the Bubble and Dew Points
  Real philiqbubl_c[Nc], phivapdew_c[Nc];
  //Activity Coefficient at the Bubble and Dew Points
```

```
23   Real gmabubl_c[Nc], gmadew_c[Nc];
24
25   Real gmaliq_c[Nc];
26
27   //
     ============================================================

28   phivapCS Phivap(Nc=Nc, C=C, P=P, T=T, T_c=T_c,
     Pc_c=Pc_c, x_c=x_pc[3,:]);
29   philiqCS Philiq(Nc=Nc, C=C, P=P, T=T, T_c=T_c,
     Pc_c=Pc_c, x_c=x_pc[2,:], W_c=W_c, SP_c=SP_c,
     V_c=V_c);
30
31 equation
32
33   for i in 1:Nc loop
34     philiqbubl_c[i] = 1;
35     phivapdew_c[i] = 1;
36     gmabubl_c[i] = 1;
37     gmadew_c[i] = 1;
38   end for;
39 //
     ============================================================

40 //Calculation Routine for Liquid Phase Fugacity
     Coefficient
41     gma_c =Philiq.gma_c;
42     philiq_c = Philiq.philiq_c;
43     gmaliq_c = Philiq.gmaliq_c;
44 //
     ============================================================

45
46   Cpres_p[:] = zeros(3);
47   Hres_p[:] = zeros(3);
48   Sres_p[:] = zeros(3);
49
50   phivap_c =Phivap.phivap_c;
```

```
51
52   for i in 1:Nc loop
53     K_c[i] = philiq_c[i] / phivap_c[i];
54   end for;
55 end ChaoSeader;
```

Liquid fugacity coefficient model

```
1 model philiqCS
2  parameter Integer Nc;
3     parameter Simulator.Files.ChemsepDatabase.
    GeneralProperties C[Nc];
4 import Simulator.Files.*;
5     parameter Real W_c[Nc];
6     parameter Real SP_c[Nc](each unit = "(J/m3)
    ^0.5") ;
7     parameter Real V_c[Nc](each unit = "m3/kmol")
     ;
8     parameter Real T_c[Nc];
9     parameter Real Pc_c[Nc];
10    constant Real Rgas(each unit = "erg.K^-1.mol
    ^-1") = 8.314470 "Ideal Gas Constant";
11    Real x_c[Nc](each min = 0, each max = 1, each
     start = 1 / (Nc + 1)), T(min = 0, start =
    273.15), P;
12    Real S, gma_c[Nc] "Activity Coefficent";
13    Real gmaliq_c[Nc], Pvap_c[Nc];
14    Real Sp_c[Nc](each unit = "(cal/ml)^0.5"),
    LV_c[Nc](each unit = "ml/gmol");
15    Real philiq_c[Nc] "Liquid Phase Fugacity
    Coefficent";
16 equation
17 //Caculation for changing Unit of Solubility
    Parameter and Liquid molal volume
18 Sp_c = SP_c/2046;
19 LV_c = V_c*1000;
20
21 //Calculation Routine for Liquid Phase Fugacity
    Coefficient
```

```
22
23      S = Simulator.Files.ThermodynamicFunctions.
    SolublityParameter(Nc, LV_c, Sp_c, x_c);
24      for i in 1:Nc loop
25        gma_c[i] = exp(LV_c[i] * (Sp_c[i] - S) ^ 2
    / (Rgas * T));
26      end for;
27      philiq_c = liquidfugasitycoefficient(Nc, T_c,
     Pc_c, W_c, T, P, LV_c, S, gma_c);
28      for i in 1:Nc loop
29        Pvap_c[i] = Simulator.Files.
    ThermodynamicFunctions.Psat(C[i].VP, T);
30        gmaliq_c[i] = philiq_c[i] * (P / Pvap_c[i])
    ;
31      end for;
32  end philiqCS;
```

Liquid fugacity coefficient fuction

```
1  function liquidfugasitycoefficient
2     extends Modelica.Icons.Function;
3
4     input Integer Nc;
5     input Real Tc[Nc];
6     input Real Pc[Nc];
7     input Real W_c[Nc];
8     input Real T,P;
9     input Real V_c[Nc];
10    input Real S;
11    input Real gma_c[Nc];
12
13    output Real Philiq_c[Nc](each start = 2);
14    protected Real Tr_c[Nc];
15    protected Real Pr_c[Nc];
16    protected Real v0_c[Nc](each start=2),v1_c[Nc
    ](each start=2),v_c[Nc];
17    protected Real A[10];
18
19    algorithm
```

```
20
21
22      for i in 1:Nc loop
23      Tr_c[i] := T / Tc[i];
24      Pr_c[i] := P / Pc[i];
25
26      if(Tc[i] == 33.19) then
27        A[1] := 1.96718;
28        A[2] := 1.02972;
29        A[3] := -0.054009;
30        A[4] := 0.0005288;
31        A[5] := 0;
32        A[6] := 0.008585;
33        A[7] := 0;
34        A[8] := 0;
35        A[9] := 0;
36        A[10] :=0;
37
38            v0_c[i] := 10^(A[1] + (A[2]/Tr_c[i]) +
    (A[3]*Tr_c[i])+(A[4] *Tr_c[i] *Tr_c[i])+(A[5] *
    Tr_c[i]*Tr_c[i]*Tr_c[i])+((A[6] +(A[7] *Tr_c[i
    ]) +(A[8]*Tr_c[i]*Tr_c[i]))*Pr_c[i])+((A[9] +(A
    [10]*Tr_c[i]))*(Pr_c[i]*Pr_c[i])) - (log10(Pr_c
    [i])));
39
40       elseif(Tc[i] == 190.56) then
41        A[1] := 2.4384;
42        A[2] := -2.2455;
43        A[3] := -0.34084;
44        A[4] := 0.00212;
45        A[5] := -0.00223;
46        A[6] := 0.10486;
47        A[7] := -0.03691;
48        A[8] := 0;
49        A[9] := 0;
50        A[10] := 0;
51
52       v0_c[i] := 10^(A[1] + (A[2]/Tr_c[i]) + (A
```

```
[3]*Tr_c[i])+(A[4] *Tr_c[i] *Tr_c[i])+(A[5] *
Tr_c[i]*Tr_c[i]*Tr_c[i])+((A[6] +(A[7] *Tr_c[i
]) +(A[8]*Tr_c[i]*Tr_c[i]))*Pr_c[i])+((A[9] +(A
[10]*Tr_c[i]))*(Pr_c[i]*Pr_c[i])) - (log10(Pr_c
[i])));
53
54      else
55        A[1] := 5.75748;
56        A[2] := -3.01761;
57        A[3] := -4.985;
58        A[4] := 2.02299;
59        A[5] := 0;
60        A[6] := 0.08427;
61        A[7] := 0.26667;
62        A[8] := -0.31138;
63        A[9] := -0.02655;
64        A[10] := 0.02883;
65
66     v0_c[i] := 10^(A[1] + (A[2]/Tr_c[i]) + (A
[3]*Tr_c[i])+(A[4] *Tr_c[i] *Tr_c[i])+(A[5] *
Tr_c[i]*Tr_c[i]*Tr_c[i])+((A[6] +(A[7] *Tr_c[i
]) +(A[8]*Tr_c[i]*Tr_c[i]))*Pr_c[i])+((A[9] +(A
[10]*Tr_c[i]))*(Pr_c[i]*Pr_c[i])) - (log10(Pr_c
[i])));
67
68    end if;
69
70      v1_c[i] := 10^(-4.23893 + (8.65808 * Tr_c[i
]) - (1.2206 / Tr_c[i]) - (3.15224 * Tr_c[i] ^
3) - 0.025 * (Pr_c[i] - 0.6));
71
72    if(v1_c[i] == 0) then
73      v_c[i]  := 10^(log10(v0_c[i]) );
74    else
75      v_c[i]  := 10^(log10(v0_c[i]) + (W_c[i] *
log10(v1_c[i])));
76    end if;
77     Philiq_c[i] := v_c[i] * gma_c[i];
```

```
78     end for;
79
80 end liquidfugasitycoefficient;
```

Vapor fugacity coefficient model

```
model phivapCS
import Simulator.Files.Thermodynamic_Functions.*;
    constant Real R(each unit = "J.K^-1.mol^-1")
   = 8.314 "Ideal Gas Constant";
    constant Real u = 1;
    parameter Integer Nc;
    parameter Simulator.Files.ChemsepDatabase.
   GeneralProperties C[Nc];
    parameter Real T_c[Nc] ;
    parameter Real Pc_c[Nc] ;
    Real x_c[Nc](each min = 0, each max = 1, each
     start = 1 / (Nc + 1)), T(min = 0, start =
   273.15), P;
    Real a_c[Nc], b_c[Nc];
    Real aij_c[Nc, Nc];
    Real amv , bmv;
    Real Avap , Bvap;
    Real t1_c[Nc], t3_c[Nc], t4, t2;
    Real C_c[Nc], D_c[Nc];
    Real Cvap[4];
    Real Z_RV[3, 2];
    Real Zvap[3], Zvv;
    Real phivap_c[Nc];
equation
//Calculation of Equation of State Constants
    a_c = Simulator.Files.ThermodynamicFunctions.
   EOSConstants(Nc, T_c, Pc_c, T);
    b_c = Simulator.Files.ThermodynamicFunctions.
   EOSConstantII(Nc, T_c, Pc_c, T);
    aij_c = Simulator.Files.
   ThermodynamicFunctions.EOSConstantIII(Nc, a_c);
    amv = Simulator.Files.ThermodynamicFunctions.
   EOSConstant1V(Nc, x_c, aij_c);
    bmv = sum(x_c .* b_c[:]);
    Avap = amv * P / (R * T) ^ 2;
    Bvap = bmv * P / (R * T);

```

```
30 //Caclulation of Compressibility factor
31      Cvap[1] = 1;
32     Cvap[2] = - 1;
33     Cvap[3] = (Avap-Bvap-(Bvap^2));
34     Cvap[4] = -(Avap*Bvap);
35     Z_RV = Modelica.Math.Vectors.Utilities.roots(
   Cvap);
36     Zvap = {Z_RV[i, 1] for i in 1:3};
37     Zvv = max({Zvap});
38
39 //Calculation of Vapour Phase Fugacity
   Coefficient
40     for i in 1:Nc loop
41       if bmv == 0 then
42         C_c[i] = 0;
43       else
44         C_c[i] = b_c[i] / bmv;
45       end if;
46     end for;
47     for i in 1:Nc loop
48       if amv == 0 then
49         D_c[i] = 0;
50       else
51         D_c[i] = a_c[i] / amv;
52       end if;
53     end for;
54     for i in 1:Nc loop
55       t1_c[i] = b_c[i] * (Zvv - 1) / bmv;
56       t3_c[i] = Avap / (Bvap * u ^ (2 ^ 0.5)) * (
   C_c[i] - 2 * D_c[i] ^ 0.5);
57     end for;
58     t4 = log((2 * Zvv + Bvap * (u + u ^ (2 ^ 0.5)
   )) / (2 * Zvv + Bvap * (u - u ^ (2 ^ 0.5))));
59     t2 = -log(Zvv - Bvap);
60       for i in 1:Nc loop
61       phivap_c[i] = exp(t1_c[i] + t2 + t3_c[i] *
   t4);
62       end for;
```

```
63 end phivapCS;
```