Summer Fellowship Report

On

Design of Unit Operations in OpenModelica

Submitted by

M . Sree Harsha

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

June 25, 2020

Acknowledgment

I express my deep sense of gratitude to Dr. Kannan M.Moudgalya, Professor, Department of Chemical Engineering, IIT Bombay for his exemplary guidance, monitoring and constant encouragement throughout the Fellowship .
It is pleasure to express my deep sense of gratitude and indebtedness to my mentors Priyam Nayak,Rahul Nagraj Department of Chemical Engineering, IIT Bombay and the FOSSEE team for their timely help and guidance

Contents

Chapter 1

Introduction

OpenModelica is an open source modelling and simulation environment intended for industrial and academic usage.It is an object oriented declarative multi domain modelling language for complex systems. The OpenModelica environment allows most of the expression, algorithm, and function parts of Modelica to be executed interactively, as well as equation models and Modelica functions to be compiled into efficient C code. The generated C code is combined with a library of utility functions, a run-time library and a numerical DAE solver. OpenModelica Connection Editor is the new Graphical User Interface for graphical model editing in OpenModelica.

Chapter 2

Distillation Column Design

2.1 Introduction

Design of plate column for distillation involves steps of calculation such as determination of number of theoretical plates, column diameter, plate hydraulic design, etc. Steps for Distillation Plate column design is,

1. Calculating overall column efficiency (O connell's correlation).
2. Calculating number of actual stages.
3. Assuming plate spacing.
4. Calculating the height of tower.
5. Calculating column diameter.
6. Calculating ,
    - Down comer area
    - Active area
    - Hole area
    - Hole size
    - weir height
7. Calculating Plate Pressure drop
8. Calculating Number of Holes

2.2 Column Efficiency O Connell's correlation

In a real stage, equilibrium will be attained rarely. The concept of column efficiency is used to link the performance of practical contacting stages to the theoretical equilibrium stage. A quick estimate of the overall column efficiency can be obtained from the correlation given by O'Connell. The overall column efficiency is correlated

with the product of the relative volatility of the light key component (relative to the heavy key) and the molar average viscosity of the feed, estimated at the average column temperature. It has been found to give reliable estimates of the overall column efficiency for hydrocarbon systems and can be used to make an approximate estimate of the efficiency for other systems.

$$E_0 = 51 - 32.5log(\mu_a \alpha_a)$$

where
μ_a = the molar average liquid viscosity, $mNs/m2$,
α_a = average relative volatility of the light key.
from efficiency and Teoritical stages we canl get number of Real stages.

2.3 Plate Spacing

The overall height of the column will depend on the plate spacing. Plate spacings from 0.15 m (6 in) to 1 m (36 in) are normally used. The spacing chosen will depend on the column diameter and operating conditions. The suggested tray spacing with column diameter is appended below

Tower Diameter(m)	Tray Spacing(m)
1 or less	0.5
$1-3$	0.6
$3-4$	0.75
$4-8$	0.9

2.4 Column Diameter

The flooding condition fixes the upper limit of vapour velocity. A high vapour velocity is needed for high plate efficiencies, and the velocity will normally be between 70 to 90 per cent of that which would cause flooding.The column diameter is determined from the flooding correlation for a chosen plate spacing. The superficial vapor velocity at flooding through the net area relates to liquid and vapor densities according to Fair's correlation.

$$V_f = C_f \sqrt{\frac{\rho_L - \rho_V}{\rho_V}}$$

where
V_f = flooding vapour velocity, m/s
ρ_V = vapor density,kg/m^3
ρ_L = liquid density,kg/m^3
C_f = Floding coefficient

$$C_f = 0.0105 + 8.127(10^{-4})(T_t^{0.755})exp(-1.463F_{LG}^{0.842})$$

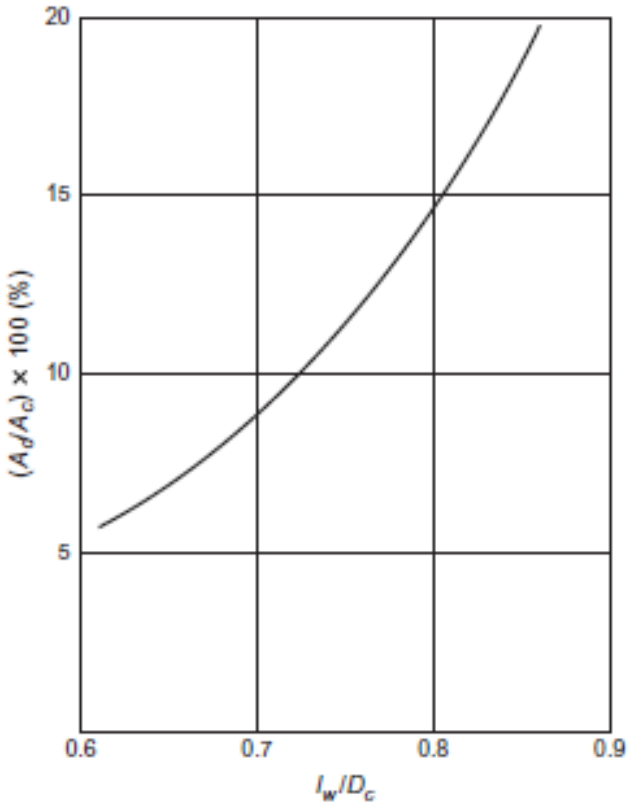
where
T_t = plate spacing, mm.

$$F_{LG} = \frac{L}{G}\sqrt{\frac{\rho_V}{\rho_L}}$$

where
L = liquid mass flow-rate,kg/s,
V =vapour mass flow-rate, kg/s.
Now we got vapour velocity and vapour volumetric flow rate ,so we can calculate the crossectional area and column diameter.

2.5 Weir Dimensions

Weir Length is calculated using this graph



Weir Liquid Crest is calculated using Francis weir formula

$$h_{ow} = 750[\frac{L_w}{\rho_l l_w}]^{\frac{2}{3}}$$

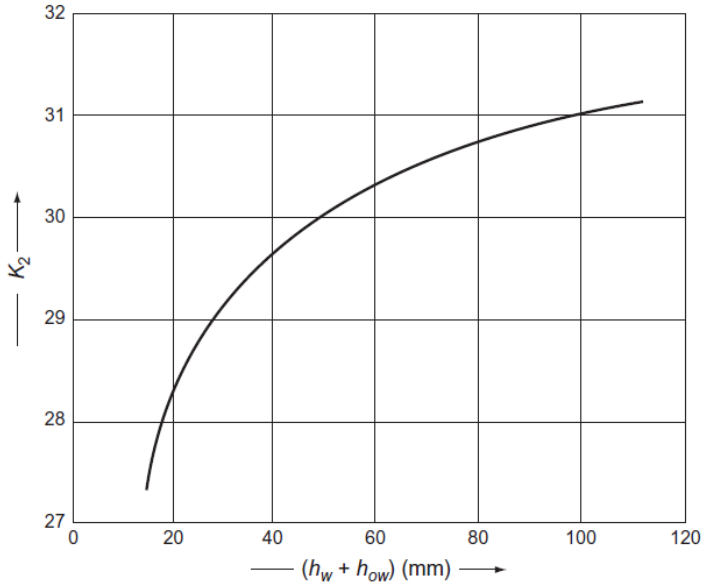
where
h_{ow} =weir crest,mm Liquid
l_w = Weir Length, m
L_w =Liquid FlowRate,Kg/s

2.6 Weep Point

Weep point is calculated using Eduljee correlation.

$$u_h = \frac{K_2 - 0.90(25.4 - d_h)}{(\rho_v)^{(1/2)}}$$

where
u_h =Weeping vapour velocity,m/s
d_h =Hole Diameter,mm
K_2 Value is getting from the Graph,



2.7 Hole Area

Initially it is assumed to 15% of Active area and then the actual minimum vapour velocity is calculated and if it is less than weeping velocity then again Hole area is Reduced to 10% and Verified wethear it satisfies the condition are not,if the condition doesnot satisfies hole area is taken as 7% of active area.

2.8 Plate Pressure Drop

Plate Pressure Drop is calculated using the formula

$$\Delta P_t = 9.81 \times 10^{-3} h_t \rho_l$$

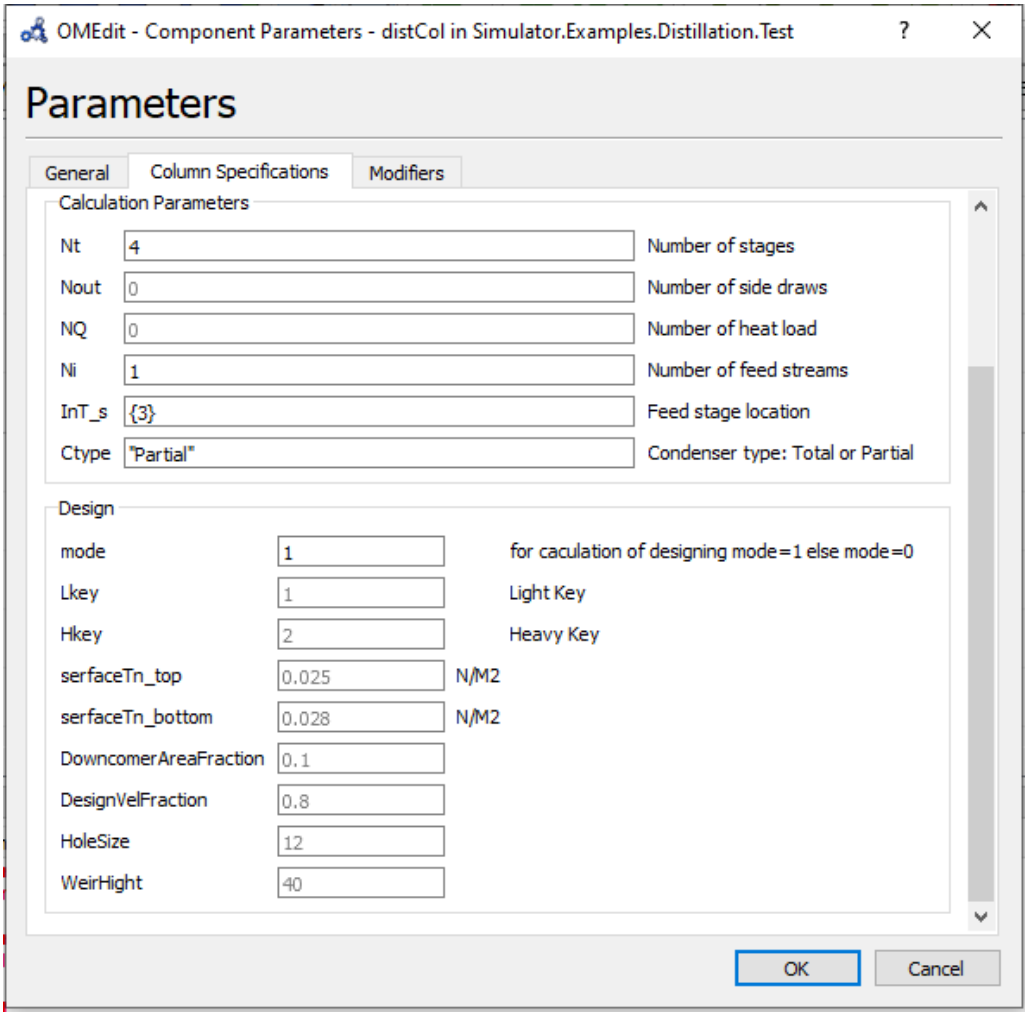
$$h_t = h_d + (h_w + h_{ow}) + h_r$$

$$h_r = \frac{12.5 \times 10^3}{\rho_L}$$

$$h_d = 51(\frac{u_h}{C_o})^2 \frac{\rho_V}{\rho_L}$$

2.9 About Code

The designing of distillation column code is added in the DistCol code ,required designing variables and designing eqations are added in the DistCol code in the simulator. Some Required values lik compositions of top ,bottom ang feed are taken from the code and some extra inputs are added in new group in column specification tab.If the user want to calculate the Designing specification then the value of the mode should be given as 1 and should provide all the remaining paramters. if the mode is 0 then all the designing variables will be assigned with zero value.



2.10 Results

Compounds: Acetone,Water
Feed Conditions:333.1 K,101325 Pa,182.75 Sol/s
Feed Composition(Mole Fraction) : 0.5 Acetone, 0.5 Water
Reflux Ratio : 1.675
Number of Stages: 8
Feed Stage : 6
Bottoms Flow rate :86.5833333Mol/s

	OpenModaica	ChemSep
Top diameter(m)	2.46126	2.46
Bottom diameter(m)	1.85395	1.85
Tower Height(m)	6.6	6.34

Chapter 3

Shell & tube Heat Exchanger Design

3.1 Introduction

Design of a shell and tube heat exchanger typically includes the determination of heat transfer area, number of tubes, tube length and diameter, tube layout, number of shell and tube passes, type of heat exchanger (fixed tube sheet, removable tube bundle etc), tube pitch, number of baffles, its type and size, shell and tube side pressure drop etc.

3.2 Algorithm and Calculation

Step 1 :Obtain the required thermophysical properties of hot and cold fluids at the arithmetic mean temperature.
Step 2 : Calculation of heat duty (Q) of the exchanger.
Step 3 :Decide tentative number of shell and tube passes (N_p).Determine the LMTD and the correction factor F_T.

$$F_T = \frac{\sqrt{R^2+1}ln(\frac{1-S}{1-RS})}{(R-1)ln(\frac{2-S[R+1-\sqrt{R^2+1}]}{2-S[R+1+\sqrt{R^2+1}]})}$$

Step 4 :Assume a reasonable value of overall heat transfer coefficient (U_{ass}).
Step 5 :Calculate heat transfer area (A) required

$$A : \frac{Q}{U_{ass}LMTD(F_T)}$$

Step 6 :Calculate the number of tubes (N_t) required to provide the heat transfer area (A):

$$N_t = \frac{A}{\pi d_o L}$$

Step 7 : Determine the tube side film heat transfer coefficient (h_i)
if the flow is Laminar

$$Nu = 1.86(RePr)^{0.33}(\frac{d}{L})^{0.33}$$

if the flow is Terbulent

$$Nu = 0.023Re^{0.8}Pr^{0.33}$$

if the flow is Transition flow then the coefficient should be evaluated using both the above equations and the lower value taken.
Step 8 :Determine the Shell side film heat transfer coefficient (h_s)

$$Nu = j_h RePr^{0.33}$$

Step 9 :Calculate overall heat transfer coefficient (U_{cal})

$$\frac{1}{U_{cal}} = \frac{1}{h_s} + \frac{1}{f_{hot}} + \frac{D_o ln(\frac{D_o}{D_i})}{2K_w} + \frac{D_o}{D_i}\frac{1}{f_{cold}} + \frac{D_o}{D_i}\frac{1}{h_i}$$

Step 10 :If $0 < \frac{U_{Cal}-U_{ass}}{U_{ass}} < 30\%$, go the next step. Otherwise go to step 4, assigning U_{cal} value to U_{ass}
Step 11 : Calculation of Pressure Drops

$$\Delta P_s = 8j_f(\frac{D_s}{D_e})(\frac{L}{l_B})\frac{\rho u_s^2}{2}$$

$$\Delta P_t = N_p[8j_f(\frac{L}{d_i}) + 2.5])\frac{\rho u_t^2}{2}$$

3.3 Results

	OpenModaica	Example Results
Number of tubes	970	918
Bundle diameter(mm)	847.094	826
Shell side Pressure drop(Kpa)	290.247	272
Tube side Pressure drop(Kpa)	30.819	7.2

Chapter 4

Lockhart and Martinelli Correlatin

4.1 Introduction

This correlation is used to calculate two phase pressure drop in the separated phase model.In this model, the phases are considered to be flowing separately in the channel, each occupying a given fraction of the channel cross section and each with a given velocity.The method of Lockhart and Martinelli is the original method that predicted the two-phase frictional pressure drop based on a friction multiplicator for the liquid-phase, or the vapor-phase.Pressure drop for each of the phases are calculated explicitly assuming that either liquid or gas is flowing through the pipe based on procedure provided for single phase flow.

$$\Delta P = f\frac{L\rho V^2}{2D}$$

where
L = Length of Pipe
D = Diameter of Pipe
ρ = Density
V =Velocity of fluid
f = friction factor
Friction factor is calculated based on Reynold's Number(Re).

$$Re = \frac{DV\rho}{\mu}$$

If flow is Laminar flow($Re < 3250$) then,

$$f = \frac{64}{Re}$$

if flow is Terbulentflow($Re > 3250$)then,

$$\frac{1}{\sqrt{f}} = -4log(0.2698\frac{K}{D} - \frac{5.0452}{Re}log(0.3539(\frac{K}{D})^{1.1098} + \frac{5.8506}{Re^{0.8981}}))$$

where
K = Rugosity of Pipe

D = diameter of Pipe
Two-phase frictional pressure drop($\Delta P_{L1}, \Delta P_{G1}$) based on a friction multiplicator for the liquid-phase, or the vapor-phase:

$$\Delta P_{L1} = {\Phi_{Ltt}}^2 \Delta P_L$$

$$\Delta P_{G1} = {\Phi_{Gtt}}^2 \Delta P_G$$

where
$\Delta P_L, \Delta P_G$ are pressure drops for each phase calculated separately,
${\Phi_{Ltt}}^2, {\Phi_{Gtt}}^2$ are two Phase multipliers.

$${\Phi_{Ltt}}^2 = 1 + \frac{C}{X_{tt}} + \frac{1}{X_{tt}^2}$$

$${\Phi_{Gtt}}^2 = 1 + CX_{tt} + X_{tt}^2$$

where $X_{tt}$is the Martinelli's parameter defined as:

$$X_{tt} = [\frac{1-x}{x}]^{0.9} [\frac{\rho_G}{\rho_L}]^{0.5} [\frac{\mu_L}{\mu_G}]^{0.1}$$

where C is taken from

Liquid	Gas	C
Terbulent	Terbulent	20
Laminar	Terbulent	12
Terbulent	Laminar	10
Laminar	Laminar	5

Estimatede two phase pressure drop is maximum of $\Delta P_{L1}, \Delta P_{G1}$
Pressure change due to the hydrostatic head of the vertical component is given by

$$\Delta P_{Elev} = \rho g sin(\theta)$$

The total Pressure drop is the sum of frictional pressure drop and hydrostatic pressure drop.

4.2 About Code

The inputs for Lockhart and Martinelli function are

- Pipe Diameter(m)
- Pipe Length(m)
- Elevation(m)
- Roughness
- Volumetric Flowrate of Vapour(m^3/s)

- Volumetric Flowrate of Liquid(m^3/s)
- Viscosity of Liquid
- Viscosity of Vapour
- Density of Vapour
- Density of Liquid
- Surface tension of liquid

Chapter 5

Pipe

5.1 Introduction

The Pipe model calculates the pressure drop of the fluid flowing in the pipe.It also calculates the temperature drop depending on the amount heat exchanged with surrounding.

5.2 About Code

Pipe in OM Chemical simulator is a package. The PIPE packge has two models ,they are

- Pipe Model
- Increments Model

The pipe model has two material stream Connections(In and Out) and one energy stream connection (Energy exchanged between Pipe and Surrondings).The main task of the Pipe model is to take the inputs and making connections between successive increment models.
Increment model has two material stream Connections(In and Out).The inlet material connection variables are getting from Previous increment. The main task of Increment model is to calculate the Pressure drop and Temperature drop in the increment. The Pressure drop is calculated Using Lockhart and Martinelli corrilation.It also calculates the Phase Properties at the calculated Temperature and Pressure using Flash model in the simulator. Hence all the outlet material connection variables are solved.

5.3 Results

Components : Water
Pressure : 202650 Pa
Temperature : 298.15 K
Flow Rate :166.5 Mol/s

Length : 10m
Heat exchanged:100KW
Number of increments : 50

	OpenModaica	DWSIM
Pressure Drop(Pa)	3208	3205.42
Temperature Drop(K)	-8.041	-8.04144

Reference

- Ray Sinnott,Gavin Towler,Coulson and Richardson's Chemical Engineering Series,Chemical Engineering Design,Sixth Edition.
- D.Q.KERN ,Process Heat Transfer.
- GitHub - Source Code DWSIM

Chapter 6

OpenModelica Code

```
1  within Simulator.UnitOperations.DistillationColumn;
2
3    model DistCol "Model of a distillation column representing fractionating
         towers where mixture is separated in equilibrium stages"
4      extends Simulator.Files.Icons.DistillationColumn;
5       parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "
            Component instances array" annotation(
6      Dialog(tab = "Column Specifications", group = "Component Parameters"));
7      parameter Integer Nc "Number of components" annotation(
8      Dialog(tab = "Column Specifications", group = "Component Parameters"));
9      import data = Simulator.Files.ChemsepDatabase;
10     parameter Boolean Bin_t[Nt] =
           Simulator.Files.OtherFunctions.colBoolCalc(Nt, Ni, InT_s) "Stream
           stage associations" annotation(
11     Dialog(tab = "Column Specifications", group = "Component Parameters"));
12     parameter Integer Nt = 4 "Number of stages" annotation(
13     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
14     parameter Integer Nout = 0 "Number of side draws" annotation(
15     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
16     parameter Integer NQ = 0 "Number of heat load" annotation(
17     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
18     parameter Integer Ni = 1 "Number of feed streams" annotation(
19     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
20     parameter Integer InT_s[Ni] "Feed stage location" annotation(
21     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
22     parameter String Ctype = "Total" "Condenser type: Total or Partial"
           annotation(
23     Dialog(tab = "Column Specifications", group = "Calculation Parameters")
           );
24     Real RR(min = 0);
25     //
           ================================================================

26     //Design Variables
27     parameter Real GasConstant=8314.7295;
28     parameter Integer mode=0 "for caculation of designing mode=1 else mode
           =0"annotation(Dialog(tab = "Column Specifications", group = "Design
           "));
29     parameter Integer Lkey=1 "Light Key" annotation(Dialog(tab = "Column
           Specifications", group = "Design"));
30     parameter Integer Hkey=2 "Heavy Key" annotation(Dialog(tab = "Column
```

```
        Specifications", group = "Design"));
31      parameter Real serfaceTn_top(unit="N/M2")=0.025 annotation(Dialog(tab =
            "Column Specifications", group = "Design")) ;
32      parameter Real serfaceTn_bottom(unit="N/M2")=0.028 annotation(Dialog(
            tab = "Column Specifications", group = "Design"));
33      parameter Real DowncomerAreaFraction=0.1 annotation(Dialog(tab = "
            Column Specifications", group = "Design"));
34      parameter Real DesignVelFraction=0.8 annotation(Dialog(tab = "Column
            Specifications", group = "Design"));
35      parameter Real HoleSize=12 annotation(Dialog(tab = "Column
            Specifications", group = "Design"));
36      parameter Real WeirHight=40 annotation(Dialog(tab = "Column
            Specifications", group = "Design"));
37      parameter Real CoX[6]={5,7,10,11,15,17};
38      parameter Real CoY[6]={0.8,0.825,0.84,0.85,0.88,0.9};
39      parameter Real WeirLengthx[8]={6,7,9,10,12,14.5,15,19.5};
40      parameter Real WeirLengthy[8]={0.61,0.65,0.7,0.73,0.75,0.8,0.81,0.86};
41      parameter Real K2x[11]={15,20,30,40,50,60,70,80,90,100,110};
42      parameter Real K2y
            [11]={27.3,28.3,29.1,29.6,30,30.3,30.5,30.8,30.9,31,31.1};
43
44      Real q;
45      Real Tray_spacing(start=0.5);
46      Real Top_T(unit = "K") "Temperature of Distillate";
47      Real Bottom_T(unit = "K") "Temperature of Bottom";
48      Real Feed_T(unit = "K") "Temperature of Feed";
49      Real Top_P(unit = "Pa") "Pressure of Distillate";
50      Real Bottom_P(unit = "Pa") "Pressure of Bottoms";
51      Real Feed_P(unit = "Pa") "Pressure of Feed";
52      Real Top_x[Nc] "Component mole fraction in Distillate";
53      Real Bottom_x[Nc] "Component mole fraction in Bottom";
54      Real Feed_x[Nc] "Component mole fraction in Feed";
55      Real feed_molar_flow(unit = "Kmol/s") "Molar Flow rate of Feed";
56      Real Top_molar_flow(unit = "Kmol/s") "Molar Flow rate of Distillate";
57      Real Bottom_Molar_flow(unit = "Kmol/s") "Molar Flow rate of Bottom";
58      Integer Theo_stages "No Of theoratical Stages";
59      Real vis[Nc] "Viscosity";
60      Real Top_Psat[Nc] "Saturated Vepor Pressure of Distillate";
61      Real Bottom_Psat[Nc]"Saturated Vapor Pressure of Bottom";
62      Real Top_Ptotal;
63      Real Bottom_Ptotal;
64      Real Top_y[Nc];
65      Real Bottom_y[Nc];
66      Real Alpha_top "Relative volatility of top";
67      Real Alpha_bottom "Relative volatility of Bottom";
68      Real Alpha_avg "Averge Relative volatility ";
69      Real Mol_wt[Nc] "Molecular Weight of Components";
70      Real avg_Molar_viscosity "Avg Molar Viscosity";
71      Real Eff "Column Efficiency";
72      Integer act_stages "Actual Stages";
73      Real Tower_heigt "Column Height";
74      Real feedAvgMW "Average Molecular Weight Feed";
75      Real TopAvgMWLiq "Average Molecular Weight Distillate";
76      Real TopAvgMWVap "Average Molecular Weight Vapor in Equlibrium with
            distillate";
77      Real BottomAvgMWLiq "Average Molecular Weight Bottom";
78      Real BottomAvgMWVap "Average Molecular Weight Vapor in Equlibrium with
            Bottom";
79      Real feed_mass_flow(unit = "Kg/S") "Mass Flow rate of Feed";
80      Real Top_mass_flow(unit = "Kg/S") "Mass Flow rate of Distillate";
81      Real Bottom_Mass_flow(unit = "Kg/S") "Mass Flow rate of Bottom";
82      Real Vw_top(unit = "Kg/S") "Mass Flow rate of vapor in the
            Rectification section";
83      Real V_top(unit = "Kmol/S") "Molar Flow rate of vapor in the
```

```
        Rectification Section";
84      Real Lw_top(unit = "Kg/S") "Mass Flow rate of liquid in the
        Rectification section";
85      Real L_top(unit = "Kmol/S") "Molar Flow rate of liquid in the
        Rectification section";
86      Real Vw_bottom(unit = "Kg/S") "Mass Flow rate of vapor in the Stripping
         Section";
87      Real V_bottom(unit = "Kmol/S") "Molar Flow rate of vapor in the
        Stripping Section";
88      Real Lw_bottom(unit = "Kg/S") "Mass Flow rate of liquid in the
        Stripping Section";
89      Real L_bottom(unit = "Kmol/S") "Molar Flow rate of liquid in the
        Stripping Section";
90      Real DensityLiq[Nc](each unit = "Kg/m3") "Densities of components";
91      Real Dens_vap_Top(unit = "Kg/m3") "Avg Density of Vapor in
        Rectification section";
92      Real Dens_Vap_Bottom(unit = "Kg/m3") "Avg Density of Vapor in stripping
         section";
93      Real Dens_liq_Top(unit = "Kg/m3") "Avg Density of liquid in
        Rectification section";
94      Real Dens_liq_Bottom(unit = "Kg/m3") "Avg Density of liquid in
        stripping section";
95      Real Cf_top "Floding Coefficient for Rectification Section";
96      Real Cf_bottom "Floding Coefficient for stripping Section";
97      Real FlodingVel_top(unit = "M/s") "Floding Velocity for Rectification
        Section";
98      Real FlodingVel_bottom(unit = "M/s") "Floding Velocity for stripping
        Section";
99      Real DesignVel_top(unit = "M/s") "Design Velocity for Rectification
        Section";
100     Real DesignVel_bottom(unit = "M/s") "Design Velocity for stripping
        Section";
101     Real NetArea_top(unit= "M2") "Net area for rectification section";
102     Real NetArea_Bottom(unit= "M2") "Net area for stripping section";
103     Real CrossSecArea_top(unit= "M2") "Crosssectional area for
        rectification section";
104     Real CrossSecArea_bottom(unit= "M2") "Crosssectional area for stripping
         section";
105     Real CrossSecArea(unit= "M2") "Crosssectional area for Column";
106     Real Dia_top(unit= "M") "Column diameter for rectification section";
107     Real Dia_bottom(unit= "M") "Column diameter for stripping section";
108     Real Column_dia(unit= "M") "Column diameter";
109     Real WeirLength(unit= "M") "Weir Length";
110     Real DowncomerArea(unit= "M2") "DownComer area";
111     Real ActiveArea(unit= "M2") "Active area";
112     Real HolePitch(unit= "mm") "Holepitch";
113     Real HoleArea(unit= "M2") "Hole area";
114     Real HoleAreaEst[3];
115     Real MinimumTurnDown(unit ="Kg/s");
116     Real WeirCrust(unit="mm")"Weir Crust";
117     Real MinWeepingVelocity(unit="M/s") "Weeping Velocity";
118     Real ActMinVapVel(unit="M/s") "Minimum Velocity";
119     Real ActMinVapVelEst[3];
120     Real ActMaxVapVel(unit="M/s") "Maximum Velocity";
121     Real PlateThickness(unit="mm") "Plate Thickness";
122     Real ResidualHead(unit="mm");
123     Real DryPlateDrop(unit="mm");
124     Real TotalPressureHeadDrop(unit="mm");
125     Real TotalpressureDrop(unit="pa");
126     Real OrrificeCoeff;
127     Real NumberOfHoles;
128     Real Theta;//angle subtended by the edge of the plate
129     Real EdgeStripArea(unit="M2");
130     Real ClimingZoneArea(unit="M2");
```

```
131     Real PerforatedArea(unit="M2");
132     Real K2;
133     //
        ================================================================================


134
135
136
137
138
139     Simulator.Files.Interfaces.matConn In_s[Ni](each Nc = Nc) annotation(
140       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)));
141     Simulator.Files.Interfaces.matConn Dist(Nc = Nc) annotation(
142       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)));
143     Simulator.Files.Interfaces.matConn Bot(Nc = Nc) annotation(
144       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)));
145     Simulator.Files.Interfaces.enConn Cduty annotation(
146       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)));
147     Simulator.Files.Interfaces.enConn Rduty annotation(
148       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)));
149     Simulator.Files.Interfaces.matConn Out_s[Nout](each Nc = Nc) annotation
          (
150       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)));
151     Simulator.Files.Interfaces.enConn En[NQ](each Nc = Nc) annotation(
152       Placement(visible = true, transformation(origin = {-34, -54}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {-70, -60}, extent = {{-10, -10}, {10, 10}}, rotation =
            0)));
153
154
155   equation
156   for i in 1:Ni loop
157     if InT_s[i] == 1 then
158       connect(In_s[i], condenser.In);
159     elseif InT_s[i] == Nt then
160       connect(In_s[i], reboiler.In);
161     elseif InT_s[i] > 1 and InT_s[i] < Nt then
162 //this is adjustment done since OpenModelica 1.11 is not handling array
        modification properly
163       In_s[i].P = tray[InT_s[i] - 1].Pdmy1;
164       In_s[i].T = tray[InT_s[i] - 1].Tdmy1;
165       In_s[i].F = tray[InT_s[i] - 1].Fdmy1;
166       In_s[i].x_pc = tray[InT_s[i] - 1].xdmy1_pc;
167       In_s[i].H = tray[InT_s[i] - 1].Hdmy1;
168       In_s[i].S = tray[InT_s[i] - 1].Sdmy1;
169       In_s[i].xvap = tray[InT_s[i] - 1].xvapdmy1;
170     end if;
```

```
171   end for;
172     connect(condenser.Out, Dist);
173     connect(reboiler.Out, Bot);
174     connect(condenser.En, Cduty);
175     connect(reboiler.En, Rduty);
176     for i in 1:Nt - 3 loop
177       connect(tray[i].Out_Liq, tray[i + 1].In_Liq);
178       connect(tray[i].In_Vap, tray[i + 1].Out_Vap);
179     end for;
180     connect(tray[1].Out_Vap, condenser.In_Vap);
181     connect(condenser.Out_Liq, tray[1].In_Liq);
182     connect(tray[Nt - 2].Out_Liq, reboiler.In_Liq);
183     connect(reboiler.Out_Vap, tray[Nt - 2].In_Vap);
184 //tray pressures
185   for i in 1:Nt - 2 loop
186     tray[i].P = condenser.P + i * (reboiler.P - condenser.P) / (Nt - 1);
187   end for;
188
189     for i in 2:Nt - 1 loop
190       tray[i - 1].OutType = "Null";
191       tray[i - 1].Out.x_pc = zeros(3, Nc);
192       tray[i - 1].Out.F = 0;
193       tray[i - 1].Out.H = 0;
194       tray[i - 1].Out.S = 0;
195       tray[i - 1].Out.xvap = 0;
196       tray[i - 1].Q = 0;
197     end for;
198     RR = condenser.Fliqout / condenser.Out.F;
199
200
201     //
        ==========================================================================

202     //for caculation of designing mode=1
203      if (mode==0) then
204       q=0;
205       Top_T=0;
206       Bottom_T=0;
207       Feed_T=0;
208       Top_P=0;
209       Bottom_P=0;
210       Feed_P=0;
211       for i in 1:Nc loop
212         Top_x[i]=0;
213         Bottom_x[i]=0;
214         Feed_x[i]=0;
215       end for;
216       feed_molar_flow=0;
217       Top_molar_flow=0;
218       Bottom_Molar_flow=0;
219       Theo_stages=0;
220       for i in 1:Nc loop
221         vis[i]=0;
222       end for;
223       avg_Molar_viscosity=0;
224       for i in 1:Nc loop
225         Top_Psat[i]=0;
226       end for;
227       for i in 1:Nc loop
228         Bottom_Psat[i]=0;
229       end for;
230       Top_Ptotal=0;
231       Bottom_Ptotal=0;
232       for i in 1:Nc loop
```

```
233             Top_y[i]=0;
234             Bottom_y[i]=0;
235           end for;
236           Alpha_top=0;
237           Alpha_bottom=0;
238           Alpha_avg=0;
239           for i in 1:Nc loop
240             Mol_wt[i]=0;
241           end for;
242           act_stages=0;
243           Tower_heigt=0;
244           feedAvgMW=0;
245           TopAvgMWLiq=0;
246           BottomAvgMWLiq=0;
247           TopAvgMWVap=0;
248           BottomAvgMWVap=0;
249           feed_mass_flow=0;
250           Top_mass_flow=0;
251           Bottom_Mass_flow=0;
252           V_top=0;
253           Vw_top=0;
254           L_top=0;
255           Lw_top=0;
256           L_bottom=0;
257           Lw_bottom=0;
258           V_bottom=0;
259           Vw_bottom=0;
260           for i in 1:Nc loop
261             DensityLiq[i]=0;
262           end for;
263           Dens_vap_Top=0;
264           Dens_Vap_Bottom=0;
265           Dens_liq_Top=0;
266           Dens_liq_Bottom=0;
267           Cf_top=0;
268           Cf_bottom=0;
269           FlodingVel_top=0;
270           FlodingVel_bottom=0;
271           DesignVel_top=0;
272           DesignVel_bottom=0;
273           NetArea_top=0;
274           NetArea_Bottom=0;
275           CrossSecArea_top=0;
276           CrossSecArea_bottom=0;
277           Dia_top=0;
278           Dia_bottom=0;
279           Column_dia=0;
280           CrossSecArea=0;
281           Tray_spacing=0;
282           DowncomerArea=0;
283           ActiveArea=0;
284           WeirLength=0;
285           MinimumTurnDown=0;
286           WeirCrust=0;
287           K2=0;
288           MinWeepingVelocity=0;
289           HoleAreaEst[1]=0;
290           HoleAreaEst[2]=0;
291           HoleAreaEst[3]=0;
292           ActMinVapVelEst[1]=0;
293           ActMinVapVelEst[2]=0;
294           ActMinVapVelEst[3]=0;
295           HoleArea=0;
296           ActMinVapVel=0;
```

```
297         ActMaxVapVel=0;
298         HolePitch=0;
299         Eff=0;
300         PlateThickness=0;
301         ResidualHead=0;
302         OrrificeCoeff=0;
303         DryPlateDrop=0;
304         TotalPressureHeadDrop=0;
305         TotalpressureDrop=0;
306         NumberOfHoles=0;
307         Theta=0;
308         EdgeStripArea=0;
309         ClimingZoneArea=0;
310         PerforatedArea=0;
311
312
313       else
314         q=1-In_s[1].xvap;
315         Top_T=Dist.T;
316         Bottom_T=Bot.T;
317         Feed_T=In_s[1].T;
318         Top_P=Dist.P;
319         Bottom_P=Bot.P;
320         Feed_P=In_s[1].P;
321         Top_x=Dist.x_pc[1,:];
322         Bottom_x=Bot.x_pc[1,:];
323         Feed_x=In_s[1].x_pc[1,:];
324         feed_molar_flow=In_s[1].F/1000;
325         Top_molar_flow=Dist.F/1000;
326         Bottom_Molar_flow=Bot.F/1000;
327         Theo_stages=Nt;
328         // calculation of Viscosity
329         for i in 1:Nc loop
330           vis[i]=Simulator.Files.TransportProperties.LiqVis(C[i].LiqVis,((
                Top_T+Bottom_T)/2));
331         end for;
332         //calculation of Saturated Vapor pressures
333         for i in 1:Nc loop
334           Top_Psat[i]=Simulator.Files.ThermodynamicFunctions.Psat(C[i].
                VP,Top_T);
335         end for;
336         for i in 1:Nc loop
337           Bottom_Psat[i]=Simulator.Files.ThermodynamicFunctions.Psat(C[i].
                VP,Bottom_T);
338         end for;
339
340         Top_Ptotal=sum(Top_Psat.*Top_x);
341         Bottom_Ptotal=sum(Bottom_Psat.*Bottom_x);
342
343         for i in 1:Nc loop
344           Top_y[i]=(Top_Psat[i]*Top_x[i])/Top_Ptotal;
345           Bottom_y[i]=(Bottom_Psat[i]*Bottom_x[i])/Bottom_Ptotal;
346         end for;
347         //Calculation of Reltive Volatility
348         Alpha_top=(Top_y[Lkey]/Top_x[Lkey])/(Top_y[Hkey]/Top_x[Hkey]);
349         Alpha_bottom=(Bottom_y[Lkey]/Bottom_x[Lkey])/(Bottom_y[Hkey]/Bottom_x
              [Hkey]);
350         Alpha_avg=(Alpha_top*Alpha_bottom)^(0.5);
351         for i in 1:Nc loop
352           Mol_wt[i]=C[i].MW;
353         end for;
354
355         //Caclulation of Avg Viscosity
356         avg_Molar_viscosity=sum(vis.*Feed_x)*100;
```

```
357         //calculation of column Efficience using O CONNELLS CORRELATION
358         Eff=51-(32.5*log10(avg_Molar_viscosity*Alpha_avg));
359         //calculation of actual stages
360         act_stages=(Theo_stages*100)/Eff;
361         //calculation of Column Height
362         Tower_heigt=(act_stages+1)*Tray_spacing;
363         //calculation of Average Molecular Weights
364         feedAvgMW=sum(Mol_wt.*Feed_x);
365         TopAvgMWLiq=sum(Mol_wt.*Top_x);
366         BottomAvgMWLiq=sum(Mol_wt.*Bottom_x);
367         TopAvgMWVap=sum(Mol_wt.*Top_y);
368         BottomAvgMWVap=sum(Mol_wt.*Bottom_y);
369         //calculation of Mass flow rates of Feed,Distillate,Bottoms
370         feed_mass_flow=feed_molar_flow*feedAvgMW;
371         Top_mass_flow=Top_molar_flow*TopAvgMWLiq;
372         Bottom_Mass_flow=Bottom_Molar_flow*BottomAvgMWLiq;
373         //calulation of flowrates of liquid and vapors in Top and Bottom
              sections
374         V_top=Top_molar_flow*(RR+1);
375         Vw_top=V_top*TopAvgMWVap;
376         L_top=V_top-Top_molar_flow;
377         Lw_top=L_top*TopAvgMWLiq;
378         L_bottom=L_top+(feed_molar_flow*q);
379         Lw_bottom=L_bottom*BottomAvgMWLiq;
380         V_bottom=L_bottom-Bottom_Molar_flow;
381         Vw_bottom=V_bottom*BottomAvgMWVap;
382         //Calculation of Densities of each component at column conditions
383
384
385         for i in 1:Nc loop
386           DensityLiq[i]=(Simulator.Files.ThermodynamicFunctions.Dens(C[i].
                LiqDen,C[i].Tc,((Top_T+Bottom_T)/2),101325))*Mol_wt[i]/1000;
387         end for;
388         //Calculation of average Densities of Vapor and Liquid in
              Rectification and stripping section
389         Dens_vap_Top=(Top_P*TopAvgMWVap)/(GasConstant*Top_T);
390         Dens_Vap_Bottom=(Bottom_P*BottomAvgMWVap)/(GasConstant*Bottom_T);
391         Dens_liq_Top=sum(Top_x.*DensityLiq);
392         Dens_liq_Bottom=sum(Bottom_x.*DensityLiq);
393
394         Cf_top=(0.0105+(8.127*0.0001*((Tray_spacing*1000)^0.755)
              *(2.71828^((-1.463)*(((Lw_top/Vw_top)*((Dens_vap_Top/Dens_liq_Top
              )^0.5))^0.842)))))*((serfaceTn_top/0.02)^0.2);
395
396         Cf_bottom=(0.0105+(8.127*0.0001*((Tray_spacing*1000)^0.755)
              *(2.71828^((-1.463)*(((Lw_bottom/Vw_bottom)*((Dens_Vap_Bottom/
              Dens_liq_Bottom)^0.5))^0.842)))))*((serfaceTn_bottom/0.02)^0.2);
397         //Flodin Velocity Calculation
398         FlodingVel_top=Cf_top*(((Dens_liq_Top-Dens_vap_Top)/Dens_vap_Top)
              ^0.5);
399         FlodingVel_bottom=Cf_bottom*(((Dens_liq_Bottom-Dens_Vap_Bottom)/
              Dens_Vap_Bottom)^0.5);
400         //Design Velocity Calculation
401         DesignVel_top=DesignVelFraction*FlodingVel_top;
402         DesignVel_bottom=DesignVelFraction*FlodingVel_bottom;
403         //Calculation of NetArea
404         NetArea_top=Vw_top/(Dens_vap_Top*DesignVel_top);
405         NetArea_Bottom=Vw_bottom/(Dens_Vap_Bottom*DesignVel_bottom);
406         //Calculating CrossSection Area of Top And Bottom
407         CrossSecArea_top=NetArea_top/(1-DowncomerAreaFraction);
408         CrossSecArea_bottom=NetArea_Bottom/(1-DowncomerAreaFraction);
409         //Calculation Of column Dia
410         Dia_top=((CrossSecArea_top*4)/3.14)^0.5;
411         Dia_bottom=((CrossSecArea_bottom*4)/3.14)^0.5;
```

```
412         if (Dia_top>Dia_bottom) then
413           Column_dia=Dia_top;
414         else
415           Column_dia=Dia_bottom;
416         end if;
417         CrossSecArea=3.14*(Column_dia^2)/4;
418         if(Column_dia<=1) then
419           Tray_spacing=0.5;
420         elseif(Column_dia>1 and Column_dia<=3) then
421           Tray_spacing=0.6;
422         elseif(Column_dia>3 and Column_dia<=4) then
423           Tray_spacing=0.75;
424         else
425           Tray_spacing=0.9;
426         end if;
427
428         DowncomerArea=DowncomerAreaFraction*CrossSecArea;
429         ActiveArea=CrossSecArea_top-(2*DowncomerArea);
430         WeirLength=(Modelica.Math.Vectors.interpolate(
              WeirLengthx,WeirLengthy,(DowncomerArea*100/CrossSecArea)))*
              Column_dia;
431
432         MinimumTurnDown=0.7*Top_mass_flow;
433         WeirCrust=750*((MinimumTurnDown/(Dens_liq_Bottom*WeirLength))^(2/3));
434
435         K2=Modelica.Math.Vectors.interpolate(K2x,K2y,(WeirHight+WeirCrust));
436
437         MinWeepingVelocity=(K2-(0.9*(25.4-HoleSize)))/((Dens_Vap_Bottom)^0.5)
              ;
438
439         HoleAreaEst[1]=0.15*ActiveArea;
440         HoleAreaEst[2]=0.1*ActiveArea;
441         HoleAreaEst[3]=0.07*ActiveArea;
442         ActMinVapVelEst[1]=(0.7*(Vw_bottom/Dens_Vap_Bottom))/HoleAreaEst[1];
443         ActMinVapVelEst[2]=(0.7*(Vw_bottom/Dens_Vap_Bottom))/HoleAreaEst[2];
444         ActMinVapVelEst[3]=(0.7*(Vw_bottom/Dens_Vap_Bottom))/HoleAreaEst[3];
445
446         if (ActMinVapVelEst[1]>MinWeepingVelocity) then
447           HoleArea=HoleAreaEst[1];
448           ActMinVapVel=ActMinVapVelEst[1];
449         elseif (ActMinVapVelEst[2]>MinWeepingVelocity) then
450           HoleArea=HoleAreaEst[2];
451           ActMinVapVel=ActMinVapVelEst[2];
452         else
453           HoleArea=HoleAreaEst[3];
454           ActMinVapVel=ActMinVapVelEst[3];
455         end if;
456
457         HoleArea=(0.907*((HoleSize/HolePitch)^2))*ActiveArea;//Calculation of
              HolePitch
458
459         ActMaxVapVel=(Vw_bottom/Dens_Vap_Bottom)/HoleArea;
460         PlateThickness=HoleSize;
461         ResidualHead=(12.5*1000)/Dens_liq_Bottom;
462         OrrificeCoeff=Modelica.Math.Vectors.interpolate(CoX,CoY,(HoleArea
              *100/PerforatedArea));
463         DryPlateDrop=51*((MinWeepingVelocity/OrrificeCoeff)^2)*(
              Dens_Vap_Bottom/Dens_liq_Bottom);
464         TotalPressureHeadDrop=DryPlateDrop+ResidualHead+WeirHight+WeirCrust;
465         TotalpressureDrop=9.81*TotalPressureHeadDrop*Dens_liq_Bottom/1000;
466         NumberOfHoles=HoleArea/(3.14*(HoleSize*HoleSize)/(4*1000*1000));
467         Theta=180-((acos(((2*((Dia_top/2)^2))-(WeirLength^2))/(2*((Dia_top/2)
              ^2))))*180/3.14);
468         EdgeStripArea=((Dia_top-0.05)*3.14*Theta/180)*0.05;
```

```
469         ClimingZoneArea=2*(WeirLength+0.05)*0.05;
470         PerforatedArea=ActiveArea-ClimingZoneArea-EdgeStripArea;
471
472
473
474       end if;
475
476     end DistCol;
```

```
within Simulator.Files.TransportProperties;

function LockhartMartinelli
extends Modelica.Icons.Function;
input Real D;
input Real L;
input Real H;
input Real K;
input Real Qv;
input Real Ql;
input Real mul;
input Real muv;
input Real rhov;
input Real rhol;
input Real sft;

output Real dpt;

protected  Real g=9.8;
protected Real Pi=3.14;
Real Theta;
Real A;
Real Vsl;
Real Vsg;
Real Vm;
Real Cg;
Real Cl;
Real Re_SL;
Real Re_SG;
Real fsg;
Real fsl;
Real a1;
Real b1;
Real dP_SL;
Real dP_SG;
Real X;
Real const;
Real fi_L;
Real fi_G;
Real dP_SL1;
Real dP_SG1;
Real dPg;
Real dPf;
algorithm
if Ql==0 then
  Theta:= atan(H / (L ^ 2 - H ^ 2) ^ 0.5)*180/Pi;
  A:=Pi*D*D/4;
  Vsg:=Qv/A;
  Re_SG:= rhov * Vsg * D / muv;
  if Re_SG>3250 then
    a1 := log(((K / D) ^ 1.1096) / 2.8257 + (7.149 / Re_SG) ^ 0.8961) / log
        (10.0);
    b1 := -2 * log((K / D) / 3.7065 - 5.0452 * a1 / Re_SG) / log(10.0);
    fsg:= (1 / b1) ^ 2;
  else
    fsg := 64/Re_SG;
  end if;
  dP_SG := fsg * Vsg ^ 2 * L * rhov / (D * 2);
  dPg:= rhov * g * sin(Theta*Pi/180) * L;
  dpt:=dP_SG+dPg;
elseif Qv==0 then
  Theta:= atan(H / (L ^ 2 - H ^ 2) ^ 0.5)*180/Pi;
  A:=Pi*D*D/4;
  Vsl:=Ql/A;
```

```
64    Re_SL:= rhol * Vsl * D / mul;
65    if Re_SL>3250 then
66      a1 := log(((K / D) ^ 1.1096) / 2.8257 + (7.149 / Re_SL) ^ 0.8961) / log
          (10.0);
67      b1 := -2 * log((K / D) / 3.7065 - 5.0452 * a1 / Re_SL) / log(10.0);
68      fsl:= (1 / b1) ^ 2;
69    else
70      fsl := 64/Re_SL;
71    end if;
72    dP_SL := fsl * Vsl ^ 2 * L * rhol / (D * 2);
73    dPg:= rhol * g * sin(Theta*Pi/180) * L;
74    dpt:=dP_SL+dPg;
75  else
76    Theta:= atan(H / (L ^ 2 - H ^ 2) ^ 0.5)*180/Pi;
77    A:=Pi*D*D/4;
78    Vsl:=Ql/A;
79    Vsg:=Qv/A;
80    Vm:=Vsg+Vsl;
81    Cg:=Vsg/Vm;
82    Cl:=1-Cg;
83    Re_SL:= rhol * Vsl * D / mul;
84    Re_SG:= rhov * Vsg * D / muv;
85    if Re_SG>3250 then
86      a1 := log(((K / D) ^ 1.1096) / 2.8257 + (7.149 / Re_SG) ^ 0.8961) / log
          (10.0);
87      b1 := -2 * log((K / D) / 3.7065 - 5.0452 * a1 / Re_SG) / log(10.0);
88      fsg:= (1 / b1) ^ 2;
89    else
90      fsg := 64/Re_SG;
91    end if;
92    if Re_SL>3250 then
93      a1 := log(((K / D) ^ 1.1096) / 2.8257 + (7.149 / Re_SL) ^ 0.8961) / log
          (10.0);
94      b1 := -2 * log((K / D) / 3.7065 - 5.0452 * a1 / Re_SL) / log(10.0);
95      fsl:= (1 / b1) ^ 2;
96    else
97      fsl := 64/Re_SL;
98    end if;
99    dP_SL := fsl * Vsl ^ 2 * L * rhol / (D * 2);
100   dP_SG := fsg * Vsg ^ 2 * L * rhov / (D * 2);
101   X := ((1 - Cg) / Cg) ^ 0.9 * (rhov / rhol) ^ 0.5 * (mul / muv) ^ 0.1;
102
103   if Re_SL<3250 then
104     if Re_SG<3250 then
105       const:=5;
106     else
107       const:=12;
108     end if;
109   else
110     if Re_SG<3250 then
111       const:=10;
112     else
113       const:=20;
114     end if;
115   end if;
116   fi_L:=1+(const/X)+(1/X^200);
117   fi_G:=1+(const*X)+(X^2);
118   dP_SL1:=fi_L*dP_SL;
119   dP_SG1:=fi_G*dP_SG;
120   if dP_SG1>dP_SL1 then
121     dPf:=dP_SG1;
122   else
123     dPf:=dP_SL1;
124   end if;
```

```
125     dPg:= (Cg * rhov + Cl * rhol) * g * sin(Theta*Pi/180) * L;
126     dpt:=dPf+dPg;
127   end if;
128
129
130   end LockhartMartinelli;
```

```
model he2
  import data = Simulator.Files.ChemsepDatabase;
  parameter data.Water Wat;
  parameter data.Methanol Meth;
  parameter Integer Nc = 2;
  parameter data.GeneralProperties C[Nc] = {Wat, Meth};
  parameter Real Hot_x[Nc]={0,1};
  parameter Real Cold_x[Nc]={1,0};
  parameter Real HotT_in=95+273.15;
  parameter Real HotT_Out=40+273.15;
  parameter Real ColdT_in=25+273.15;
  parameter Real ColdT_out=40+273.15;
  parameter Real Di=16;
  parameter Real Do=20;
  parameter Real L=4.83;
  parameter String Flow="CounterFlow";
  parameter String Case="Cold in Tube";
  parameter Real Shell_Dia_x
      [11]={0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2};
  parameter Real Shell_Dia_y[11]={50,52,55,58,62,64,67,69,72,74,78};
  parameter Real jh_x[6]={10,100,1000,10000,100000,1000000};
  parameter Real jh_y[6]={0.19,0.05,0.018,0.0058,0.002,0.0007};
  parameter Real jft_x
      [10]={10,40,100,400,800,1000,4000,10000,100000,1000000};
  parameter Real jft_y
      [10]={0.8,0.2,0.08,0.02,0.0095,0.009,0.006,0.0048,0.0029,0.0018};
  parameter Real jst_x
      [11]={10,100,300,400,600,700,800,1000,10000,100000,1000000};
  parameter Real jst_y
      [11]={2.4,0.25,0.1,0.09,0.08,0.078,0.075,0.07,0.05,0.037,0.025};
  parameter Real Fouling_hot=5000;
  parameter Real Fouling_cold=3000;
  parameter Real Kw=50;
  parameter Integer NP=2 "Number Of Passes";

Real vis_c[Nc] "Viscosity of Each component at Cold Stream Mean Temperature
    ";
Real vis_h[Nc] "Viscosity of Each component at Cold Stream Mean Temperature
    ";
Real VisCold "Viscosity of Cold stream";
Real VisHot  "Viscosity of Hot stream";
Real Dens_C[Nc] "Density of Each component at Cold Stream Mean Temperature"
    ;
Real Dens_H[Nc] "Density of Each component at Cold Stream Mean Temperature"
    ;
Real HotDens "Density of Hot Stream";
Real ColdDens "Density of Hot Stream";
Real Cph  "Heat capacities of Hot Stream";
Real Cpc "Heat capacities of Cold Stream";
Real Cp_c[Nc] "Heat capacities of each component at Cold Stream Mean
    Temperature";
Real Cp_h[Nc] "Heat capacities of each component at Hot Stream Mean
    Temperature";
Real K_c[Nc] "Conductivity of each component at Cold Stream Mean Temperatur
    ";
Real K_h[Nc] "Conductivity of each component at Hot Stream Mean Temperatur"
    ;
Real Cold_K "Conductivity of Cold stream";
Real Hot_K "Conductivity of Hot stream";
Real HotFm,ColdFm;
Real Q;
Real LMTD;
Real LMTDf;
Real R,S;
```

```
52  Real CorrectionFactor;
53  parameter Real Uass1=454.5454545;
54  Real Ucalc,Uass;
55  Real HTA "Heat Transfer Area";
56  Integer Nt;
57
58  Real TCA "Tube Cross Sectional Area";
59  Integer TubesPerPass;
60  Real TFA "Tube Flow Area";
61  Real Tube_MassVel "Tube Mass Vel";
62  Real Tube_Re "Reynolds number in the tube";
63  Real Tube_Pr "Tube side Prandtle number";
64  Real Shell_Re;
65  Real Shell_Pr;
66  Real hi;
67  Real hs;
68  Real Bundle_Dia,Shell_Dia;
69  Real BaffleSpacing;
70  Real TubePitch;
71  Real CrossFlowArea;
72  Real Shell_MassVel;
73  Real Equ_Dia;
74  Real jh;
75  Real jft;
76  Real jst;
77  Real Pt;
78  Real Ps;
79
80  equation
81    //Properties=====================================
82
83    for i in 1:Nc loop
84      vis_c[i]=Simulator.Files.TransportProperties.LiqVis(C[i].LiqVis,(
          ColdT_in+ColdT_out)/2);
85      vis_h[i]=Simulator.Files.TransportProperties.LiqVis(C[i].LiqVis,(
          HotT_in+HotT_Out)/2);
86    end for;
87    VisCold=vis_c*Cold_x;//0.8/1000
88    VisHot=vis_h*Hot_x;//0.34/1000
89
90
91    for i in 1:Nc loop
92          Dens_C[i]=(Simulator.Files.ThermodynamicFunctions.Dens(C[i].
              LiqDen,C[i].Tc,(ColdT_in+ColdT_out)/2,101325))*C[i].MW/1000;
93          Dens_H[i]=(Simulator.Files.ThermodynamicFunctions.Dens(C[i].
              LiqDen,C[i].Tc,(HotT_in+HotT_Out)/2,101325))*C[i].MW/1000;
94    end for;
95    ColdDens=Dens_C*Cold_x;
96    HotDens=Dens_H*Hot_x;
97
98
99    for i in 1:Nc loop
100     Cp_c[i]=Simulator.Files.ThermodynamicFunctions.LiqCpId(C[i].LiqCp,(
          ColdT_in+ColdT_out)/2)/C[i].MW;
101     Cp_h[i]=Simulator.Files.ThermodynamicFunctions.LiqCpId(C[i].LiqCp,(
          HotT_in+HotT_Out)/2)/C[i].MW;
102   end for;
103   Cph=Cp_c*Cold_x;
104   Cpc=Cp_h*Hot_x;
105
106   for i in 1:Nc loop
107    K_c[i]=Simulator.Files.TransportProperties.LiqK(C[i].LiqK,(ColdT_in+
         ColdT_out)/2);
108    K_h[i]=Simulator.Files.TransportProperties.LiqK(C[i].LiqK,(HotT_in+
```

```
        HotT_Out)/2);
109   end for;
110   Cold_K=K_c*Cold_x;//0.59;
111   Hot_K=K_h*Hot_x;//0.19;
112
113
114   //=============================================================
115
116   HotFm=100000/3600;
117   HotFm*Cph*(HotT_in-HotT_Out)=ColdFm*Cpc*(ColdT_out-ColdT_in);
118   if (Flow=="CounterFlow") then
119     LMTD=((HotT_in-ColdT_out)-(HotT_Out-ColdT_in))/log((HotT_in-ColdT_out)
          /(HotT_Out-ColdT_in));
120   else
121     LMTD=((HotT_in-ColdT_in)-(HotT_Out-ColdT_out))/log((HotT_in-ColdT_in)/(
          HotT_Out-ColdT_out));
122   end if;
123
124   if Case=="Cold in Tube" then
125     R=(HotT_in-HotT_Out)/(ColdT_out-ColdT_in);
126     S=(ColdT_out-ColdT_in)/(HotT_in-ColdT_in);
127   else
128     R=(ColdT_in-ColdT_out)/(HotT_Out-HotT_in);
129     S=(HotT_Out-HotT_in)/(ColdT_in-HotT_in);
130   end if;
131
132   CorrectionFactor=((((R^2)-1)^0.5)*log((1-S)/(1-(R*S))))/((R-1)*log((2-(S
        *(R+1-(((R^2)-1)^0.5))))/(2-(S*(R+1+(((R^2)-1)^0.5))))));
133   LMTDf=LMTD*CorrectionFactor;
134   //LMTDf=26;
135   Q=HotFm*(Cph*1000)*(HotT_in-HotT_Out);
136   //Q=4340*1000;
137 algorithm
138   Uass:=Uass1;
139   Ucalc:=Uass*1.32;//not(Uass==Ucalc)//(abs(Uass-Ucalc)/Uass)>0.3
140   while (not(Uass==Ucalc)) loop
141     Uass:=Ucalc;
142     HTA:=Q/(LMTDf*Uass);
143     Nt:=integer(HTA/(3.14*Do*L/1000));
144     //Tube Side Coeff
145     TCA:=3.14*(Di*Di/1000000)/4;
146     TubesPerPass:=integer(Nt/NP);
147     TFA:=TubesPerPass*TCA;
148     if Case=="Cold in Tube" then
149       Tube_MassVel:=ColdFm/TFA;
150       Tube_Re:=Tube_MassVel*Di/(VisCold*1000);
151       Tube_Pr:=(Cpc*1000)*VisCold/Cold_K;
152       if Tube_Re<2000 then
153         hi:=(1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Cold_K/(Di/1000);
154       elseif Tube_Re>10000 then
155         hi:=(0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Cold_K/(Di/1000);
156       else
157         if((1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Cold_K/(Di/1000))
              <((0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Cold_K/(Di/1000)) then
158           hi:=(1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Cold_K/(Di
                /1000);
159         else
160           hi:=(0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Cold_K/(Di/1000);
161         end if;
162       end if;
163     else
164       Tube_MassVel:=HotFm/TFA;
165       Tube_Re:=Tube_MassVel*Di/(VisHot*1000);
166       Tube_Pr:=(Cph*1000)*VisHot/Hot_K;
```

```
167         if Tube_Re<2000 then
168           hi:=(1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Hot_K/(Di/1000);
169         elseif Tube_Re>10000 then
170           hi:=(0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Hot_K/(Di/1000);
171         else
172           if((1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Hot_K/(Di/1000))
                  <((0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Hot_K/(Di/1000)) then
173               hi:=(1.86*((Tube_Re*Tube_Pr*(Di/1000)/L)^0.33))*Hot_K/(Di/1000)
                      ;
174           else
175               hi:=(0.023*(Tube_Re^0.8)*(Tube_Pr^0.33))*Hot_K/(Di/1000);
176           end if;
177         end if;
178       end if;
179
180       Bundle_Dia:= Do*((Nt/0.249)^(1/2.207));
181       Shell_Dia:=Modelica.Math.Vectors.interpolate(Shell_Dia_x,Shell_Dia_y,(
              Bundle_Dia/1000))+Bundle_Dia;
182       BaffleSpacing:=Shell_Dia/5;
183       TubePitch:=1.25*Do;
184       CrossFlowArea:=((TubePitch-Do)*Shell_Dia*BaffleSpacing)/(TubePitch
              *1000000);
185       Equ_Dia:=1.1*((TubePitch^2)-(0.917*(Do^2)))/Do;
186       if Case=="Cold in Tube" then
187         Shell_MassVel:=HotFm/CrossFlowArea;
188         Shell_Re:=Shell_MassVel*Equ_Dia/(VisHot*1000);
189         Shell_Pr:=(Cph*1000)*VisHot/Hot_K;
190         jh:=Modelica.Math.Vectors.interpolate(jh_x,jh_y,Shell_Re);
191         hs:=Hot_K*jh*Shell_Re*(Shell_Pr^0.33)/(Equ_Dia/1000);
192       else
193         Shell_MassVel:=ColdFm/CrossFlowArea;
194         Shell_Re:=Shell_MassVel*Equ_Dia/(VisCold*1000);
195         Shell_Pr:=(Cpc*1000)*VisCold/Cold_K;
196         jh:=Modelica.Math.Vectors.interpolate(jh_x,jh_y,Shell_Re);
197         hs:=Cold_K*jh*Shell_Re*(Shell_Pr^0.33)/(Equ_Dia/1000);
198       end if;
199       Ucalc:=1/((1/hs)+(1/Fouling_hot)+(Do*log(Do/Di)/(2*Kw*1000))+((Do/Di)
              *((1/Fouling_cold)+(1/hi))));
200     end while;
201
202
203
204   equation
205     jft=Modelica.Math.Vectors.interpolate(jft_x,jft_y,Tube_Re);
206     jst=Modelica.Math.Vectors.interpolate(jst_x,jst_y,Shell_Re);
207     if Case=="Cold in Tube" then
208       Pt=NP*((8*jft*(L*1000/Di))+2.5)*((ColdDens*((Tube_MassVel/ColdDens)^2))
              /2);
209       Ps=8*jst*(Shell_Dia/Equ_Dia)*(L*1000/BaffleSpacing)*((HotDens*((
              Shell_MassVel/HotDens)^2))/2);
210     else
211       Pt=NP*((8*jft*(L*1000/Di))+2.5)*((HotDens*((Tube_MassVel/HotDens)^2))
              /2);
212       Ps=8*jst*(Shell_Dia/Equ_Dia)*(L*1000/BaffleSpacing)*((ColdDens*((
              Shell_MassVel/ColdDens )^2))/2);
213     end if;
214   end he2;
```

```
within Simulator.UnitOperations;

package PIPE
  extends Modelica.Icons.Package;
  model Pipe
        //extends Simulator.Files.Icons.Pipe;
        //extends Simulator.Files.Models.Flash;
        parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "
            Component instances array" annotation(
        Dialog(tab = "Component Specifications", group = "Component
            Parameters"));
        parameter Integer Nc "Number of components" annotation(
        Dialog(tab = "Component Specifications", group = "Component
            Parameters"));

      extends GuessModels.InitialGuess;
      parameter Real Di=0.055;
      parameter Real Li=5;
      parameter Real Hi=0;
      parameter Real Ki=0.000045;
      parameter Integer inc=5;
      parameter Real Qi=0;
      Real Tf,Pf,Hf,Xf[3,Nc],Ff,xvapf,sf;



      //
          ==========================================================================

      Simulator.Files.Interfaces.matConn In(Nc = Nc) annotation(
        Placement(visible = true, transformation(origin = {-100, 0}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {-90, 10}, extent = {{-6, -6}, {6, 6}}, rotation = 0))
            );
      Simulator.Files.Interfaces.matConn Out(Nc = Nc) annotation(
        Placement(visible = true, transformation(origin = {100, 0}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {85, 9}, extent = {{-7, -7}, {7, 7}}, rotation = 0)));
      Simulator.Files.Interfaces.enConn En annotation(
        Placement(visible = true, transformation(origin = {0, -100}, extent
            = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
            origin = {-1, -1}, extent = {{-7, -7}, {7, 7}}, rotation = 0)))
            ;
      //
          ==========================================================================


    equation

      En.Q = Qi;

      for i in 1:inc loop
       if i==1 then
          connect(In,increment[1].In);
       else
          connect(increment[i-1].Out,increment[i].In);
       end if;
      end for;


      increment[inc].Out.T=Tf;
      increment[inc].Out.P=Pf;
      increment[inc].Out.H=Hf;
      increment[inc].Out.x_pc=Xf;
```

```
50        increment[inc].Out.F=Ff;
51        increment[inc].Out.xvap=xvapf;
52        increment[inc].Out.S=sf;
53
54        //=====================
55        Out.T=Tf;
56        Out.P=Pf;
57        Out.x_pc[1,:]=Xf[1,:];
58        Out.F=Ff;
59      end Pipe;
60
61    model Increments
62      extends Simulator.Files.Models.Flash;
63        parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc];
64        parameter Integer Nc;
65        Real Fmin[3](each unit = "Kg/s") "Inlet mass Flow of each phase";
66        Real X_cin[3,Nc];
67        Real Fin(unit = "mol/s", min = 0) "Inlet stream molar flow rate";
68        Real Pin(unit = "Pa", min = 0) "Inlet stream pressure";
69        Real Tin(unit = "K", min = 0) "Inlet stream temperature";
70        Real Hin(unit = "kJ/kmol") "inlet stream molar enthalpy";
71        Real xvapin(unit = "-", min = 0, max = 1) "Inlet stream vapor phase
              mole fraction";
72        Real Sin(unit = "kJ/[kmol.K]") "Inlet stream molar entropy";
73        Real x_c[Nc](each unit = "-", each min = 0, each max = 1) "Component
              mole fraction";
74        Real Fout(unit = "mol/s", min = 0) "outlet stream molar flow rate";
75        Real Pout(unit = "Pa", min = 0) "Outlet stream pressure";
76        Real Tout(unit = "K", min = 0) "Outlet stream temperature";
77        Real Tdel(unit = "K") "Temperature Increase";
78        Real Hout(unit = "kJ/kmol") "outlet mixture molar enthalpy";
79        Real Q(unit = "W");
80        Real D;
81        Real L;
82        Real H;
83        Real K;
84        Real Mol_wt[Nc];
85        Real In_rhol_C[Nc];
86        Real In_Visl_C[Nc];
87        Real In_Visv_C[Nc];
88        Real In_rhol;
89        Real In_rhov;
90        Real In_visl;
91        Real In_visv;
92        Real In_Ql;
93        Real In_Qv;
94        //=============================================================
95        Simulator.Files.Interfaces.matConn In(Nc = Nc) annotation(
96          Placement(visible = true, transformation(origin = {-100, 0}, extent
                = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
              origin = {-90, 10}, extent = {{-6, -6}, {6, 6}}, rotation = 0))
              );
97        Simulator.Files.Interfaces.matConn Out(Nc = Nc) annotation(
98          Placement(visible = true, transformation(origin = {100, 0}, extent
              = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(
              origin = {85, 9}, extent = {{-7, -7}, {7, 7}}, rotation = 0)));
99
100       //=============================================================
101     equation
102     //=====================
103       In.P = Pin;
104       In.T = Tin;
105       In.F = Fin;
106       In.H = Hin;
```

```
107         In.S = Sin;
108         In.x_pc[1 , :] = x_c[:];
109         In.xvap = xvapin;
110
111         //=================================================
112         X_cin=In.x_pc;
113         Fin = Fout;
114         Hout = Hin + (Q/Fin);
115         Tin + Tdel = Tout;
116       for i in 1:Nc loop
117         Mol_wt[i]=C[i].MW;
118       end for;
119       Fmin[1]=(Fin*(Mol_wt*X_cin[1,:]))/1000;
120       Fmin[2]=(Fin*(1-xvapin)*(Mol_wt*X_cin[2,:]))/1000;
121       Fmin[3]=(Fin*(xvapin)*(Mol_wt*X_cin[3,:]))/1000;
122       for i in 1:Nc loop
123          In_rhol_C[i]=(Simulator.Files.ThermodynamicFunctions.Dens(C[i].
               LiqDen,C[i].Tc,Tin,Pin))*C[i].MW/1000;
124       end for;
125       In_rhol=In_rhol_C*X_cin[2,:];
126       for i in 1:Nc loop
127         In_Visl_C[i]=Simulator.Files.TransportProperties.LiqVis(C[i].
               LiqVis,Tin);
128       end for;
129       for i in 1:Nc loop
130         In_Visv_C[i]=Simulator.Files.TransportProperties.VapVisc(C[i].
               VapVis,Tin);
131       end for;
132       In_visl=In_Visl_C*X_cin[2,:];
133       In_visv=In_Visv_C*X_cin[3,:];
134       In_rhov=(Pin*(Mol_wt*X_cin[3,:]))/(8314.7295*Tin);
135       if In_rhov==0 then
136         In_Qv=0;
137       else
138         In_Qv=Fmin[3]/In_rhov;
139       end if;
140       if In_rhol==0 then
141         In_Ql=0;
142       else
143         In_Ql=Fmin[2]/In_rhol;
144       end if;
145       Pout=Pin-Simulator.Files.TransportProperties.LockhartMartinelli(
              D,L,H,K,In_Qv,In_Ql,In_visl,In_visv,In_rhov,In_rhol,0.02);
146
147
148
149         Fin = F_p[1];
150         Pout = P;
151         Hout=H_p[1];
152         x_c[:] = x_pc[1, :];
153
154
155         Tout=T;
156         Out.xvap=xvap;
157         Out.x_pc[2,:]=x_pc[2,:];
158         Out.x_pc[3,:]=x_pc[3,:];
159         Out.S = S_p[1];
160         Out.P = Pout;
161         Out.T = Tout;
162         Out.F = Fout;
163         Out.H = Hout;
164         Out.x_pc[1, :] = x_c[:];
165       end Increments;
166   end PIPE;
```