Summer Fellowship Report

On

Modeling of Unit Operations and Cost Estimator in OpenModelica

Submitted by

Akshay Kumar Mehta

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

June 12, 2020

Acknowledgment

First and foremost, I would like to thank Prof. Kannan Moudgalya for establishing this fellowship, which I believe has been an excellent introduction to me on the open source software and technologies. His suggestions have helped me a lot to improve the quality of the project.

I would also like to thank my mentors Mr. Priyam Nayak and Mr. Rahul AS for providing valuable insight and expertise, as well as assisting me in overcoming the several difficulties I faced during the course of this project. Their advice on modeling and simulation of unit operations significantly improved the accuracy of the results. I would never have been able to finish the project without their support.

I would also like to show gratitude to my fellow interns and peers, who were constantly there to clarify my doubts and recommend improvements to the project. Their help and support was of immense value to me.

Contents

Chapter 1

Introduction

“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 efficient C code. The generated C code is combined with a library of utility functions, a run-time library, and a numerical Differential-Algebraic 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.

Chapter 2

Background

OMEdit, the integrated Graphical User Interface (GUI) of OpenModelica consists of libraries across various domains such as Electrical, Thermal, Math, Mechanics, etc. that can be used to develop models of a physical system or process. However, OMEdit does not contain libraries for modelling of chemical process systems. This limits the use of OpenModelica for modelling chemical process systems. Hence, it becomes necessary to build models or packages that aids in modelling of chemical process systems. An essential requirement is the availability of thermodynamic packages to estimate fluid properties. The main aim of this work is to develop standard thermodynamic packages in OpenModelica. In this work, a thermodynamic package, namely" Extended UNIQUAC developed in OpenModelica. In order to develop thermodynamic packages in OpenModelica, properties such as critical temperature, critical pressure, acentric factor, molecular weight, etc. of the components that constitutes the system are required. For now, we are not using the electrolytes present in DWSIM but relying only on the current database of OpenModelica. The developed thermodynamic packages can be used to calculate fugacity coefficient and activity coefficient of solutions. This package was designed for the use of electrolytic solutions which involve water as well as some limited number of ionic compounds.

Chapter 3

Beggs & Brill Correlation

3.1 Description

Many correlations have been defined for flows of fluids and to derive about the various losses it bores under different conditions of flow. These take in account hydrostatic frictional fluid losses. These relations have been defined for all kinds of phases of fluids. While single phase fluid flow calculations are easier to calculate, presence of multi phase complicates the calculations. These take oil and water together as single pseudo liquid alongside gas as a separate phase altogether. The Beggs and Brill Correlation was different from most of the published correlations that could mostly work well with only vertical flow. This relation handles well fully the flow in all of the directions. It was formulated with experiments using a 1' or 1/2 ' pipe that had the option to be placed at any angle from horizontal. This different experiment conditions made it suitable for all types of flow directions and regimes. It has hydrostatic and frictional losses on the base of its working model. unlike other relations it requires the flow pattern must be determined. When used for the single-phase fluid, it generates results similar to Fanning liquid correlation

3.2 Modeling

Total pressure gradient is defined by the following relation.

$$\frac{dP}{dZ} = \left[\left(\frac{dP}{dZ}\right)_{Fric} + \left(\frac{dP}{dZ}\right)_{Elev.}\right] / \left(1 - E_k\right)$$

Where,
(dP/dZ) Fric. = pressure gradient due to friction
(dP/dZ) Ele. = hydro-static pressure difference

Ek = estimates pressure loss due to acceleration.

3.2.1 Flow Pattern Map

A flow regime is identified based on the Froude number of the mixture (F_{rm}) and input liquid content (no slip liquid holdup C_L).

Froude number of the mixture is given by

$$F_{rm} = \frac{V_m^2}{g.D}$$

Where,
V_m is mixture velocity,
D is pipe inside diameter
g is gravitational constant.

And, no slip liquid holdup C_L is given by

$$C_L = \frac{Q_L}{Q_V + Q_L}$$

where,
Q_L is liquid volumetric flow
Q_V is Vapour/Gas volumetric flow.

The transition lines for correlation are defined as
$L_1 = 316 * C_L^{0.302}$
$L_2 = 0.0009252 * C_L^{-2.4684}$
$L_3 = 0.1 * C_L^{-1.4516}$
$L_4 = 0.5 * C_L^{-6.738}$

Conditions for different transition lines are:

1. Segregated Flow
   $C_L < 0.01$ and $F_{rm} < L_1$
   Or, $C_L >= 0.01$ and $F_{rm} < L_2$

2. Intermittent Flow
   $0.01 <= C_L <0.4$ and $L3 < F_{rm} <= L1$
   Or, $C_L >= 0.4$ and $L3 < F_{rm} <= L4$

3. Distributed Flow
   $C_L < 0.4$ and $F_{rm} >= L4$
   Or, $C_L >=0.4$ and $F_{rm} > L4$

4. Transition Flow
   $L2 < F_{rm} < L3$

3.2.2 Liquid Holdup, $E_L(\Theta)$

Once flow pattern has been determined, liquid holdup for horizontal flow $E_L(\theta)$ is calculated.

$$E_L(\theta) = B(\theta) * E_L(0)$$

Here, $B(\theta)$ is obtained as:

$$B(\Theta) = 1 + \beta sin(1.8\theta) - (\frac{1}{3})sin^3(1.8\theta))$$

where, θ is the angle of inclination of pipe with horizontal and β is Correction factor for different rigmes.
Liquid velocity number, N_{LV} is given by:

$$N_{LV} = 0.1938 * V_{SL} * \left(\frac{\rho_L}{g * \sigma}\right)^{\frac{1}{4}}$$

Where, V_{SL} is no slip liquid velocity, ρ_L is liquid density, g is gravitational constant and σ is surface tension.
$E_L(0)$ and β(uphill) for different flow patterns are as follows:

1. Segregated Flow

$$E_L(0) = \frac{0.98 * C_L^{0.4846}}{F_{rm}^{0.0868}}$$

$$\beta = (1 - C_L)ln\left(\frac{0.011 * N_{LV}^{3.539}}{C_L^{3.768} * F_{rm}^{1.614}}\right)$$

2. Intermittent Flow

$$E_L(0) = \frac{0.845 * C_L^{0.5351}}{F_{rm}^{0.0173}}$$

$$\beta = (1 - C_L)ln\left(\frac{2.96 * C_L^{0.305} * F_{rm}^{0.0978}}{N_{LV}^{0.4473}}\right)$$

3. Distributed Flow

$$E_L(0) = \frac{1.065 * C_L^{0.5824}}{F_{rm}^{0.0609}}$$

$$\beta = 0$$

4. Transition Flow
   For transitional flow $E_L(\theta)$ can be calculated as follows:

   $$E_L(\Theta)_{transition} = A * E_L(\Theta)_{segregated} + B * E_L(\Theta)_{intermittent}$$

   Where, constants A and B are as following :

   $$A = \frac{L_3 - F_{rm}}{L_3 - L_2}$$

   $$B = 1 - A$$

β(downhill) for all regime is given by :

$$\beta = (1 - C_L)ln\left(\frac{4.7 * N_{LV}^{0.1244}}{C_L^{0.3692} * F_{rm}^{0.5056}}\right)$$

3.2.3 Mixture Density, ρ_m

Using the value of Liquid Hold up, mixture density is calculated.

$$\rho_m = \rho_L * E_L(\Theta) + \rho_g * (1 - E_L(\Theta))$$

3.2.4 Hydrostatic Head

Pressure change due to the hydrostatic head of the pipe is given by:

$$\left(\frac{dP}{dZ}\right)_{Elev.} = \frac{\rho_m * g * sin(\Theta)}{144 * g_c}$$

3.2.5 No Slip Friction Factor, f$_{NS}$

$$y = \frac{C_L}{E_\Theta^2}$$

Value of S is governed by following conditions-
if 1 < y < 1.2

$$S = ln(2.2 * y - 1.2)$$

Otherwise,

$$S = \frac{ln(y)}{-0.0583 + 3.182 * ln(y) - 0.8725 * ln(y)^2 + 0.01853 * ln(y)^4}$$

Calculate no slip Reynolds's number using no slip mixture density and viscosity.

$$Re_{NS} = \frac{\rho_{NS} * V_m * D}{\mu_{NS}}$$

Using Colebrook-White equation no slip friction factor f$_{NS}$ is then calculated
Friction factor f$_{TP}$ is given by

$$f_{TP} = f_{NS} * e^S$$

3.2.6 Pressure loss due to Friction Factor

Using the calculated value of f$_{TP}$, Pressure loss due to frictional factor is

$$\left(\frac{dP}{dZ}\right)_{Fric} = \frac{2 * f_{TP} * V_m^2 * \rho_{NS}}{144 * g_C * D}$$

3.2.7 Pressure loss due to acceleration

$$E_K = \frac{\rho_m * V_m * V_{sg}}{g_C * P}$$

Where, V_{Sg} is no slip gas velocity.

3.3 Results

The OpenModelica code for flat model of Beggs & Brill was tested at varying Volume and length and the results for the same is reported in table.

Input Parameters
Compound : {Ethylene oxide, Water, Ethylene Glycol }
Mole Fraction : {0.2, 0.8, 0.0}
Temperature : 390 K
Pressure : 100000 Pa
Reactor Volume : 350 m^3
Reactor Length : 1 m

PFR Output		
Parameters	DWSIM Inspector manager	OpenModelica
Pressure Drop (Pa)	0.0201223287719767	0.0201224

Table 3.1: Comparison of Pressure Drop in PFR Ethylene oxide-Water-Ethylene Glycol system

Chapter 4

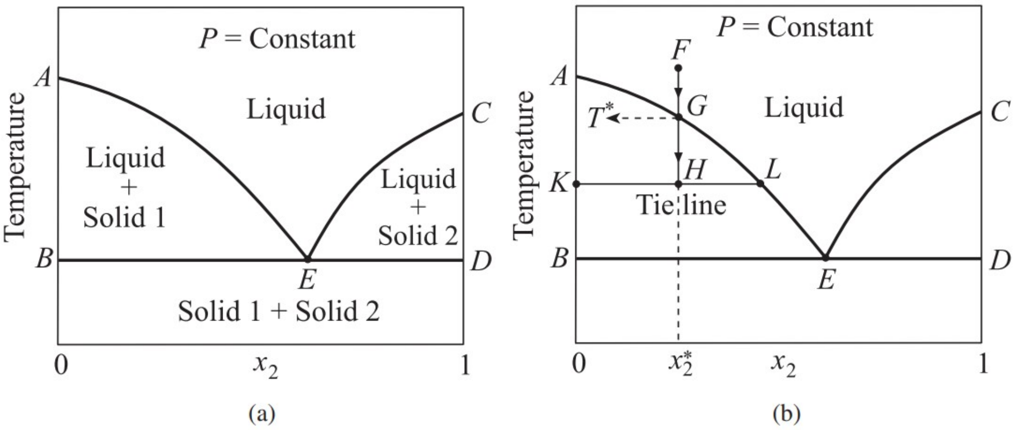
Solid-Liquid Equilibrium Routine

4.1 Description

The Solid-liquid equilibrium data are obtained experimentally by cooling down a liquid mixture of known composition and continuously noting down the temperature as a function of time. The formation of a solid phase is represented by a break point in the curve. The temperature at this mentioned point is the solid-liquid equilibrium temperature, and the composition of the liquid mixture at this point gives the solubility of solute in the solvent. SLE data are generally represented as phase diagrams with temperature versus solubility at axis under isobaric conditions.

Among the phase diagrams that exhibit eutectic behaviour, the simplest one is shown in Figure 4.1a. The temperatures at points A and C represent the freezing (or melting) temperatures of pure 1 and pure 2, respectively. Only homogeneous liquid phase exists over the curve AEC. The curve AE represents the liquid curve on which pure solid 1 is in equilibrium with the liquid mixture. Similarly, the curve CE is the liquid curve on which pure solid 2 is in equilibrium with the liquid mixture. In other words, while the curve AE can be regarded as the solubility curve of solid 1 in solvent 2, the curve CE represents the solubility curve of solid 2 in solvent 1. Therefore, liquid curves are determined from the solubility equation, Eqn (4.2). The curves AE and CE intersect at the point E, that is referred to as the eutectic point. At the eutectic point, the liquid phase is in equilibrium with both the solid phases. Since the compositions of two solids are the same, no separation is possible at the eutectic point. The corresponding eutectic temperature is indicative of the minimum freezing (or melting) temperature of the mixture. We must keep in mind that the eutectic point is analogous to the azeotropic point in vapor-liquid equilibrium. The curve BED represents the solidus curve below which

Figure 4.1: Solid-liquid phase diagram for a binary mixture.



(a) (b)

non homogeneous two-phase solid consisting of pure 1 and pure 2 exists. .

4.2 Modeling

Let substance i be present as a pure solid and also as a component in a liquid solution at constant temperature and pressure . At the triple point, the equation for mutual solubility is calculated by the well-known equation:

$$ln(\gamma_i x_i) = \frac{\Delta H_i^{fus}(T_t)}{RT_m}\left(1 - \frac{T_m}{T}\right) - \frac{1}{RT}\int_{T_m}^{T} \Delta C_{p_i} dT + \frac{1}{R}\int_{T_m}^{T} \frac{\Delta C_{p_i}}{T} dT \quad (4.1)$$

Equation 4.1 can be used in two ways alternatively : When it is applied to a solid, it gives the solubility of solid in a solvent. On the other hand, if it is applied to a solvent, it gives its freezing point depression.

When Δ C_{pi} is close to zero, the above equation become

$$ln(\gamma_i x_i) = \frac{\Delta H_i^{fus}}{RT_m}\left(1 - \frac{T_m}{T}\right) \quad (4.2)$$

If we notice the properties of a mixture considered at point F as shown in Fig. 4.1b. The liquid mixture is cooled at constant pressure, pure solid 1 forms when the temperature drops to T^* (point G) and the relationship between T^* and x_2^* is given by Eqn:

$$ln(\gamma_1(1 - x_2^*)) = \frac{\Delta H_1^{fus}}{RT_{m1}}\left(1 - \frac{T_{m1}}{T^*}\right)$$

4.3 Results

The OpenModelica code for the solid material stream was tested for various component systems at varying temperature and pressure conditions and the results for the same is reported in table.

Input Parameters
Compound : {Acetic Acid, Propionic Acid, Water}
Mole Fraction : {0.2, 0.4, 0.4}
Temperature : 212 K
Pressure : 101325 Pa

Output Stream		
Parameters	DWSIM	OpenModelica
Mole Fraction		
Liquid Mole Fraction	{0.16751307, 0.3794566, 0.45303026}	{0.167513, 0.379457, 0.45303}
Solid Mole Fraction	{0.44504444, 0.55495556, 0}	{0.445044, 0.554956, 0}
Mole Flow (mol/s)	9.76496	9.76496
Liquid Phase Mole Flow (mol/s)	8.62191	8.62191
Solid Phase Mole Flow (mol/s)	1.14305	1.14305
Mass Flow (kg/s)	0.476998	0.476998
Liquid Phase Mass Flow (kg/s)	0.399458	0.399458
Solid Phase Mass Flow (kg/s)	0.0775404	0.0775404

Table 4.1: Comparison of Solid-Liquid Equilibrium data for Acetic acid- Propionic Acid -Water system at a temperature of 212 k

Input Parameters

Compound : {Water, Acetic Acid, Glycerol}
Mole Fraction : {0.2, 0.5, 0.3}
Temperature : 233.53 K
Pressure : 102000 Pa

Output Stream		
Parameters	DWSIM	OpenModelica
Mole Fraction		
Liquid Mole Fraction	{0.53591238, 0.30938375, 0.15470387}	{0.535912, 0.309384, 0.154704}
Solid Mole Fraction	{0, 0.61349165, 0.38650835 }	{0, 0.613492, 0.386508}
Mole Flow (mol/s)	179.57094	179.571
Liquid Phase Mole Flow	67.015039	67.015
Solid Phase Mole Flow	112.55591	112.556
Mass Flow (kg/s)	11	11
Liquid Phase Mass Flow	2.8468547	2.84686
Solid Phase Mass Flow	8.1531453	8.15315

Table 4.2: Comparison of Solid-Liquid Equilibrium data for Water-Acetic acid-Glycerol system at a temperature of 233.53 k

Chapter 5

Solid Separator

5.1 Description

A solids separator as the name suggests, separates solids from mixtures of solids and liquids. It is a device that bifurcates the feed stream into two separate flows: one with a solids content containing lower amount of liquid and a liquid stream with a lower content of solid. Depending on the type of separator used, the lower-solids content stream is referred to as effluent, separated liquids, or liquor. On similar lines, the higher-solids content is called sludge, grit, separated solids, or cake. To keep things easy to comprehend, we shall call the stream entering a solid separator; Feed, the low solids stream leaving the separator liquor, and the high solids stream cake. There are several methods used for the separation of solids. Some of these includes Settling, screening, Filtering, Centrifugal force, Flotation, etc.

5.2 Modeling

5.2.1 Mass and Volume Relationships

A solids separator divides the feed stream into two product streams, and no material is lost in the process. Flow in must equal flow out. Mathematically, this is written as:

$$Q_F = Q_C + Q_L \tag{5.1}$$

Where, Q_F is the Feed Flow, Q_C is the flow of cake and Q_I is liquor flow.

By the law of conservation of mass, The mass or amount of material flowing into a separator must also be equal to the mass or amount of fluid that is leaving the separator.

$$M_F = M_C + M_L \tag{5.2}$$

Where, M_F is the mass flow rate of feed , M_C is the mass flow rate of cake and M_I is liquor mass flow rate

A simple equation clearly depicts the ties between mass and volumetric flow rates together. The mass of material in a given volume is equal to its concentration times the volume. In terms of flow, the mass flow rate of any material in a stream is the concentration of that material times the liquid flow rate.

$$M = QC \tag{5.3}$$

where, C is the concentration.
Combining equation 5.1 and 5.3 we get

$$Q_F C_F = Q_C C_C + Q_L C_L \tag{5.4}$$

Where C_F is concentration of Feed, C_C is concentration of cake and C_L is concentration of liquor

5.2.2 Solids Separation Efficiency

Manufacturers often give the efficiency of their equipment using the difference between Feed and liquor solids concentration. This can further be defined as fraction of solids remaining in the liquor, or more precisely, fraction passing through the separator:

$$FractionPassing = \frac{C_L}{C_F} \tag{5.5}$$

Turning this around, we can define the separation efficiency as :

$$separationefficiency = \frac{C_C}{C_F} = \frac{C_F - C_L}{C_F} \tag{5.6}$$

5.3 Results

The OpenModelica code for the Solid Separator was tested for various component systems at varying temperature and pressure conditions and the results for the same is reported in table.

Input Parameters
Compound : {Acetic Acid, Propionic Acid, Water}
Mole Fraction : {0.2, 0.4, 0.4}
Temperature : 212 K
Pressure : 101325 Pa
Liquid separation efficiency : 100%
Solid separation efficiency : 100%

Output Stream 1		
Parameters	DWSIM	OpenModelica
Mole Fraction		
Liquid Mole Fraction	{0.16751307, 0.3794566, 0.45303026}	{0.167513, 0.379457, 0.45303}
Solid Mole Fraction	{0, 0, 0}	{0, 0, 0}
Mole Flow		
Liquid Mole Flow (mol/s)	88.2943	88.2943
Solid Mole Flow (mol/s)	0	0
Mass Flow		
Liquid Mass Flow (kg/s)	4.09072	4.09072
Solid Mass Flow (kg/s)	0	0

Output Stream 2		
Parameters	DWSIM	OpenModelica
Mole Fraction		
Liquid Mole Fraction	{0, 0, 0}	{0, 0, 0}
Solid Mole Fraction	{0.44504444, 0.55495556, 0}	{0.445044, 0.554956, 0}
Mole Flow		
Liquid Mole Flow (mol/s)	0	0
Solid Mole Flow (mol/s)	11.7057	11.7057
Mass Flow		
Liquid Mass Flow (kg/s)	0	0
Solid Mass Flow (kg/s)	0.794067	0.794067

Table 5.1: Comparison of output streams from solid separator at an efficiency of 100%

Chapter 6

Filter

6.1 Description

Filtration may be defined as that unit operation in which the insoluble solid component of a solid-liquid suspension is separated from the liquid component by passing the latter through a porous membrane or septum which retains the solid particles on its upstream surface, or within its structure, or both. A broader definition could be that the fluid may be a liquid or a gas; the valuable stream from the filter may be the fluid, or the solids, or both. The solid – liquid suspension is known as the feed slurry or 'prefilt', the liquid component that passes through the membrane is called the filtrate and the membrane itself is referred to as the filter medium. The separated solids are known as the filter cake, once they form a detectable layer covering the upstream surface of the medium. Often the feed is modified in some way by pre-treatment to increase the filtration rate, as by heating, re-crystallizing, or adding a "filter aid" such as cellulose or diatomaceous earth.

Filter, must provide a support for the filter medium, a space for the accumulation of the solids, channels for the introduction of the feed. A means of inducing the flow of filtrate through the filter and medium must also be provided.

6.2 Modeling

6.2.1 Constant Pressure

The total pressure drop is sum of the pressure drop across cake and filter medium.

$$\Delta P = \Delta P_c + \Delta P_m \tag{6.1}$$

Where, ΔP is overall pressure, ΔP_c is pressure drop over cake and ΔP_m is pressure drop over medium.
Specific Cake Resistance is given by

$$\alpha = \frac{\Delta P_c}{\mu v m_c} \tag{6.2}$$

and, Filter Medium Resistance is given by

$$R_m = \frac{\Delta P_m}{\mu v} \tag{6.3}$$

From equation 6.1 and 6.2 , the total pressure drop comes out to be

$$\Delta P = \frac{\mu \alpha v m_c}{A} + R_m \mu v \tag{6.4}$$

where,

$$V = \frac{1}{A}\frac{dv}{dt}$$

and mass of cake (m_c) = concentration of cake per unit filterate(c) x volume of filterate(v)
Equation 6.4 is further reduced to

$$\Delta P = \frac{\mu}{A}\frac{dv}{dt}\left(\frac{cV\alpha}{V} + R_m\right)$$

6.2.2 Results

The OpenModelica code for the Filter was tested for various component systems at varying temperature and pressure conditions and the results for the same is reported in table.

Input Parameters
Compound : {Water, Calcium Carbonate}
Mole Fraction : {0.95926159, 0.040738415}
Temperature : 293.15 K
Pressure : 101325 Pa
Filter Cycle : 300 s
Submerged Area Fraction f : 0.3
Specific Cake Resistance α : 2.9E+10 m/Kg
Filter Medium Resistance Rm : 0 m^{-1}
Cake Relative Humidity R_h : 50 %
Liquid Density ρ : 1024.76 Kg/m^3

Liquid Viscosity μ : 0.0009887 Pa.s
Pressure Drop ΔP : 67702.8 Pa

Output Stream 1		
Parameters	DWSIM	OpenModelica
Mole Fraction	{0.97257458, 0.02742542}	{0.972575, 0.0274254}
Mole Flow (mol/s)	26.2972	26.297108
Mass Flow (kg/s)	0.532942	0.5329405

Output Stream 2		
Parameters	DWSIM	OpenModelica
Mole Fraction	{0.91202177, 0.087978232}	{0.912022, 0.0879782}
Mole Flow (mol/s)	7.41098	7.410953
Mass Flow (kg/s)	0.187022	0.187022

Table 6.1: Comparison of output streams from Filter

Chapter 7

Analysis of Cost Estimation

7.1 Description

7.1.1 Capital Cost Estimation

Before setting up an industrial plant and putting it into operation, a large wholesome lump of money must be gathered to purchase and install the necessary machinery and equipment. Land must be acquired, service facilities must be fetched, and the plant must be set up completely with all piping, controls, and services. In addition to these investments in fixed capital, certain amount of funds are required for the daily expenses involved in the plant operation which is refered to as the working capital.

The sum of the fixed-capital investment and the working capital is together known as the total capital investment (TCI). The fixed-capital portion may be further subdivided into manufacturing fixed-capital investment, also known as direct cost, and non manufacturing fixed-capital investment, also known as indirect cost.

Fixed-Capital Investment

Manufacturing fixed-capital investment represents the capital which is deemed of utmost importance for the installation of process equipment along with all components that are needed for complete process operation. Expenses for site preparation, piping, instruments, insulation, foundations, and auxiliary facilities are a few examples that can be classified under the manufacturing fixed-capital investment.

Working Capital

The working capital for any industrial plant is decided by taking many factors in consideration. The total amount of money is a sum of all of what is invested in raw materials, supplies of stock; finished products ready to be sold in stock and semi finished products that are still in the process. It is also about all the accounts receivable; the cash kept in hand for monthly payment of operating expenses(such as salaries, wages, and raw material purchases; accounts payable; and taxes payable.)

The ratio of working capital to total capital investment varies with different companies, but most chemical plants use an initial working capital amounting to 10 to 20 percent of the total capital investment. This percentage may increase to as much as 50 percent or more for companies producing products of seasonal demand, because of the large inventories which must be maintained for appreciable period

7.1.2 Operating Cost Estimation

The third major component of any economic analysis is the total product cost. This cost is based on the sum total of costs that are required for operating the plant, selling the products, recovering the capital investment, and contributing to corporate functions such as management and research and development. These costs are generally divided into two categories: manufacturing costs and general expenses. Manufacturing costs are also referred to as operating or production costs. The manufacturing costs is factor dependent upon variable, fixed, and overhead costs.

Total product costs are calculated on one of three basis as daily basis, unit of product basis, or annual basis. Annual cost is the best alternative for the purpose of economic analysis. Moreover, the other advantages of annual estimates are:

1. smooth out the effect of seasonal variations,
2. include plant on-stream time or equipment operation,
3. permit more rapid calculation of operating costs at less than full capacity, and
4. provide a convenient way of considering large expenses that occur infrequently such as annual planned maintenance shutdowns.

7.2 Modeling

7.2.1 Capital Cost Estimation

Percentage of Delivered-Equipment Cost

The method for estimating the fixed-capital and total capital investment requires determination of the delivered equipment cost. The total direct plant cost are estimated as percentages of the delivered-equipment cost. The additional components of the capital investment are based on average percentages of the total direct plant cost, total direct and indirect plant costs, or total capital investment. This is summarized in the following cost equation.

$$C_n = \sum (E + f_1E + f_2E + f_3E + f_4E + f_5E + ... + f_nE \tag{7.1}$$

where f_1, f_2, f_3, f_4 ... are multiplying factors for piping,electrical, indirect costs,etc

7.2.2 Operating Cost Estimation

Simplified Model For Estimating Operating costs

A cost model can be derived based on the breakdown of costs in different categories as shown in Table 6.2 The TOC can be expressed as the sum of MC and general expenses (SARE)

$$TOC = MC + SARE \tag{7.2}$$

The first term is calculated as the contribution of DC, FC and plant overhead (OVHD) as

$$MC = DC + FC + OVHD \tag{7.3}$$

Item	Fluids	Fluids–Solids	Solids
1.Direct costs			
F_1 – purchase cost of basic equipment	1.0	1.0	1.0
F_2 – equipment installation	0.47	0.39	0.45
F_3 – instrumentation and control	0.36	0.26	0.18
F_4 – piping	0.68	0.31	0.16
F_5 – electrical	0.11	0.10	0.10
F_6 – building, including services	0.18	0.29	0.25
F_7 – yard improvement	0.10	0.12	0.15
F_8 – service and utilities facilities	0.70	0.55	0.40
Total direct costs	3.60*PCE	3.02*PCE	2.69*PCE
2.Indirect costs			
F_9 – engineering and supervision	0.33	0.32	0.33
F_{10} – construction expenses	0.41	0.34	0.39
F_{11} – legal expenses	0.04	0.04	0.04
F_{12} – contractor fee	0.22	0.19	0.17
F_{13} – contingency	0.44	0.37	0.35
Total indirect costs	1.44*PCE	1.26*PCE	1.28*PCE
Fixed capital	5.04*PCE	4.28*PCE	3.97*PCE
Working capital (15% total capital)	0.89	0.75	0.70
Total capital investment	5.93*PCE	5.03*PCE	4.67*PCE

Table 7.1: Ratio Factors for Estimating Fixed-Capital Investment Based on Purchasing Cost of Basic Equipment PCE

Costs	Typical Values
1.Direct costs (DC)	
Raw materials	From material balance (10–80%)TOC
Operating labour (OL)	Fm manpower allocation (10–20% TOC)
Direct supervisory and clerical labour	10–20% OL
Utilities	From energy balance (10–20% TOC)
Maintenance and repairs (MR)	2–10% FC
Operating supplies	1% FC
Laboratory charges	10–20% OL
Patents and royalties	0–6% TOC
Total DC	60–66% TOC
2.Fixed charges (FC)	
Local taxes	1–4% FC
Insurance	0.5–1% FC
Rent	8–12% from rented land and buildings
Financing (interest)	0–10% from TCI
Total FC	10–20% TOC
3. Plant overhead (OVHD)	50–70% from operating labour or 5–15% TOC
Total manufacturing costs = DC + FC + OVHD	80–90% TOC
General expenses (SARE)	
Administrative costs	20% OL
Distribution and marketing costs	20% OL
Research and development	2–20% TOC
Total General expenses (SARE)	10–20% TOC or 2.5% from Revenue
Total operating costs= MC + SARE	TOC

Table 7.2: Estimation of the Total Operating Costs

7.3 Results

The OpenModelica code for the Cost Estimator was tested for an example adapted from a research paper and the results for the same is reported in table.

Input Parameter : Fluid Plant Capital Cost	
Parameters	Fraction of purchased equipment
Direct Cost	
Purchased Equipment Cost	$ 368014
Purchased Equipment Installation	20%
Instrumentation and control	15%
Piping	25%
Electrical	8%
Building, including services	10%
Yard improvement	5%
Service and utilities facilities	50%
Indirect Cost	
Engineering and supervision	10%
Construction expenses	20%
Legal expenses	4%
Contractor fee	15%
Contingency	22%

Output : Fluid Plant Capital Cost	
Parameters	Calculated Values (in $)
Working Capital Investment	327532
Fixed Capital Investment	1118800
Total capital investment	1446300

Table 7.3: Estimation of the total capital investment of the wastewater treatment plant.

Input Parameter : Operating Cost	
Parameters	Typical Values
Calculation Method	(1)Default litrature value
Raw Material	\$0
Utilities	\$275721.60
Operating Labour	\$201600
Fixed capital Investment	\$1977305
Revenue	\$3988407.60

Output : Operating Cost	
Parameters	Calculated Values (in \$)
Total Fixed Cost	59319.1
General Expenses	99710.2
Manufacturing Cost	1.05E+06
Plant Overhead Cost	212729
Variable Cost	785158
Total Operating Cost	1.15E+06

Table 7.4: Estimation of the Operating Cost of a plant.

Chapter 8

Conclusions

This work summarizes the effort undertaken to add a few missing unit operations and Correlations to the existing chemical library called OMChemSim in OpenModelica.

In chapter 3, Calculation routine for pressure drop in PFR is developed. Pressure drop in PFR is mainly dependent on the famous correlation called Beggs & Brill. This correlation is capable of handling all flow directions encountered in oil and gas operations, namely uphill, downhill, horizontal, inclined and vertical flow for two phase fluid. First, phase is checked and depending on the phase single or multi, respective function is called. For multi phase pressure drop, a flow regime is identified and based on the flow regime pressure drop parameters are calculated.

In chapter 4, the objective was to create a routine that will calculate the solid-equilibrium data for a mixture .This routine tries to find the compositions of liquid and solid phases at equilibrium. During the convergence process, solubility is checked for compounds in the liquid phase. Using the solubility solid-liquid phase mole fraction and Liquid-solid phase composition is calculated.

In chapter 5, solid separator is developed to handle solid and liquid separation. The input parameter for solid separator is basically liquid and solid efficiency. Solids Separation Efficiency defines the amount of solids in the liquid stream. 100% efficiency means no solids in the liquid stream. The generated results were compared with that of DWSIM. In all cases for solid separator, the results were nearly same for DWSIM and OpenModelica.

In chapter 6, filter was developed for the separation of solid which retains the Solid and allows the liquid to pass through. The filter allows us to choose our calculation mode namely design or simulation. If design is selected , DWSIM will calculate the filter area given the Total pressure drop. If simulation is selected, vice versa action is taken. The generated results were compared with that of DWSIM. In all cases for filter the results were nearly same for DWSIM and OpenModelica.

Finally in chapter 7, two cost calculation model are developed. The first model handles the capital cost estimation which calculates total capital cost required if we feed total equipment cost. The model contains default percentage, user defined percentage and absolute value calculation method , giving freedom to the user to choose their own parameters. Similarly, the second model handles total operating cost based on the previously calculated total capital cost.

Reference

1. Beggs & Brill Correlation. URL: `https://cheguide.com/beggs_brill.html`
2. Ismail Tosun, in The Thermodynamics of Phase and Reaction Equilibria, 2013
3. Solids separation in manure handling systems
4. Timmerhaus, K. D., Peters, M. S. & West R. E. (2003). Plant Design and Economics for Chemical Engineers, 5th ed. New York: McGraw-Hill.
5. Mexandre C. Dimian. Integrated Design and Simulation of Chemical Processes, Volume 13
6. Development of a Technology for Treating Wastewater Contaminated with Nitric Acid
7. GitHub - Source Code DWSIM

Chapter 9

OpenModelica Code

9.1 Beggs & Brill Correlation

```
function BeggsBrill
 input Real muv,qv,rhov;
  input Real mul,ql,rhol;
  input Real D,k,L,deltaz,surft;
  output Real dP[3];

  protected
  Real q,mu,V,NRe,fric;

  //protected
      Real dm;
      Real lm;
      Real deltazm;
      Real thetam;
      Real rhovm;
      Real rholm;
      Real surftm;
      Real Aream;
      Real qvm;
      Real qlm;
      Real Vm;
      Real Clm;
      Real L1m;
      Real L2m;
      Real L3m;
      Real L4m;
      Real Frm;
      Real ElThetam;
      Real Betam;
      Real Vslm;
      Real Nvlm;
      Real Bthetam;
      Real ElTheta1m;
      Real Rhom;
      Real dp_hhm;
```

```
   Real Ym;
   Real Sm;
   Real rho_nsm;
   Real mu_nsm;
   Real NRe_nsm;
   Real f_nsm;
   Real f_tpm;
   Real dp_frm;
   Real Constant;
algorithm




if qv == 0.0 then
 // Reynold's Number Calculation
          q := ql*(1.15741e-5);  //Unit Conversion of Volumetric Flow Rate (m3/d -m3/s)
          mu := (0.001 )* mul;      //Unit Conversion of Viscosity
          V := q / (Modelica.Constants.pi * D ^ 2 / 4);
          NRe := rhol * D * V / mu;

          if NRe > 3250 then
           fric := (1/((-2)*log(((k/D)/3.7065) - ((5.0452*(log((((k/D)^1.1096)/(2.8257)) + ((7.149/ NRe) ^ 0.8961))/log(10)))/NRe)) /log(10)))^2;

          else
          fric := 64 / NRe;
          end if ;
  // Delta P Friction - dPl
  dP[1] := fric * (L / D) * (V ^ 2 / 2) * rhol;
  // Delta P Elevation - dPh
  dP[2] := rhol * 9.8 *Modelica.Math.sin(Modelica.Math.asin(deltaz / L)) * L;
   // Delta P Total - dPt
  dP[3] := dP[1] + dP[2];



elseif ql == 0.0 then
 // Reynold's Number Calculation
          q  := qv*(1.15741e-5) ;   //Unit Conversion of Volumetric Flow Rate (m3/d -m3/s)
          mu := (0.001 )* muv;      //Unit Conversion of Viscosity
          V := q / (Modelica.Constants.pi * D ^ 2 / 4);
          NRe := rhov * D * V / mu;

          if NRe > 3250 then
           fric := (1/((-2)*log(((k/D)/3.7065) - ((5.0452*(log((((k/D)^1.1096)/(2.8257)) + ((7.149/ NRe) ^ 0.8961))/log(10)))/NRe)) /log(10)))^2;

          else
          fric := 64 / NRe;
          end if ;
```

```
      // Delta P Friction - dPl
      dP[1] := fric * (L / D) * (V ^ 2 / 2) * rhov;
      // Delta P Elevation - dPh
      dP[2] := rhov * 9.8 *Modelica.Math.sin(Modelica.Math.asin(deltaz / L)) * L;
       // Delta P Total - dPt
      dP[3] := dP[1] + dP[2];
    else

      //Unit Conversion: Diameter m to ft
        dm := D * 3.28084;

        //Unit Conversion: Flow Rate of Liquid and Gas m3/d to ft3/s
        qvm := qv/ 24 * 0.00980963;
        qlm := ql/ 24 * 0.00980963;

        //Unit Conversion: Density of Liquid and Gas kg/m3 to lb/ft3
        rhovm := rhov * 0.062428;
        rholm := rhol * 0.062428;

        //Unit Conversion: Surface Tension N/m to dyn/cm
        surftm := surft * 1000;

        //Unit Conversion: Length and Height m to ft
        lm := L * 3.28084;
        deltazm := deltaz * 3.28084;

        //Calculation of Angle of Inclination
        thetam := (atan(deltazm/(lm^2-deltazm^2)^0.5) * 180 )/Modelica.Constants.pi;
        //Calculation of Area
        Aream  := ( Modelica.Constants.pi*(dm^2))/4;
    //Delta P calculation as a function of Hydrostatic
        Vm := (qvm + qlm)/Aream;
        Clm := qlm/(qvm +qlm);
    //The transition lines
         L1m := 316 * Clm ^ 0.302;
         L2m := 0.0009252*(Clm^(-2.4684));
         L3m := 0.1*(Clm^(-1.4516));
         L4m := 0.5*(Clm^(-6.738));




         Frm := (Vm^2)/(32.2*dm);
    //Froude number of the mixture
    //Condition to check the Flow Regime
      if Clm < 0.01 and Frm < L1m then
        Constant := 1 "Flow Regime : Segregated";
      elseif Clm >= 0.01 and Frm < L2m then
        Constant := 1 "Flow Regime : Segregated";
      elseif Clm >= 0.01 and Clm < 0.4 and Frm > L3m and Frm <= L1m then
        Constant := 2 "Flow Regime : Intermittent";
      elseif Clm >= 0.4 and Frm > L3m and Frm <= L4m then
        Constant := 2 "Flow Regime : Intermittent";
      elseif Clm < 0.4 and Frm >= L1m then
```

```
        Constant := 3 "Flow Regime : Distributed";
      elseif Clm >= 0.4 and Frm > L4m then
        Constant := 3 "Flow Regime : Distributed";
      elseif Clm >= 0.01 and Frm > L2m and Frm < L3m then
        Constant := 4 "Flow Regime : Transition";
      else
        Constant := 0;
      end if "Flow Regime : None";
    //ELTheta for Seleceted Regime
      if Constant == 1 then
        ElThetam := 0.98 * Clm ^ 0.4846 / Frm ^ 0.0868;
      elseif Constant == 2 then
        ElThetam := 0.845 * Clm ^ 0.5351 / Frm ^ 0.0173;
      elseif Constant == 3 then
        ElThetam := 1.065 * Clm ^ 0.5824 / Frm ^ 0.0609;
      elseif Constant == 4 then
        ElThetam := (L3m - Frm) / (L3m - L2m) * 0.98 * Clm ^ 0.4846 / Frm ^ 0.0868 + (1 - (L3m - Frm) / (L3m - L2m)) * 0.845 * Clm ^ 0.5351 / Frm ^ 0.0173;
      else
        ElThetam := 0;
      end if;


        Vslm := qlm /Aream;
        Nvlm := 1.938*Vslm*(rholm/(32.2*surftm))^0.25;

        if deltazm > 0 then

              if Constant == 1 then
               Betam := (1 - Clm)*log(0.011*(Nvlm^3.539)/((Clm^3.768)*(Frm^1.614)));
              elseif Constant == 2 then
                Betam := (1 - Clm) *log(2.96 * (Clm^0.305) * (Frm^0.0978) / (Nvlm^0.4473));
              elseif Constant == 3 then
                Betam := 0;
              else Betam := 0;
              end if;

        else  Betam := (1 - Clm) *log(4.7*(Nvlm^0.1244)/((Clm^0.3692)*(Frm^0.5056)));
        end if;



              Bthetam := 1+ Betam*(sin(1.8*thetam) - 0.3333*(sin(1.8*thetam))^3);

              ElTheta1m := Bthetam*ElThetam;

              Rhom := (rholm*ElTheta1m) + (rhovm*(1-ElTheta1m)); //Density of Mixture

              //calculation of delta P as a function of hydrostatic load in lbf / ft2
```

```
           dp_hhm := Rhom*deltazm;


  //Delta P calculation as a function of friction

           Ym := log(Clm/(ElTheta1m^2));

           if Ym > 1 and Ym < 1.2 then
             Sm := log(2.2*exp(Ym) - 1.2);
           else
             Sm := Ym /((-0.0523) + (3.182*Ym) - (0.8725*((Ym^2))) + (0.01853*(Ym^4)));
           end if;

           //No slip Reynold's number using no slip mixture density and viscosity
           rho_nsm  := Clm*rholm + (1 - Clm)*rhovm;
           mu_nsm   := Clm*mul  + (1 - Clm)*muv;
           NRe_nsm  := (rho_nsm * Vm *dm )/ (mu_nsm * 0.00067197);

           //calculation of the friction factor
           if NRe_nsm  > 3250 then
             f_nsm  := (1/((-2)*log((((k*3.2808)/dm)/3.7065) - ((5.0452*(log(((((k*3.2808)/dm)^1.1096)/(2.8257)) + ((7.149/NRe_nsm) ^ 0.8961))/log(10)))/ NRe_nsm)) /log(10)))^2;

           else
             f_nsm  := 64 / NRe_nsm;
           end if ;

           f_tpm  := f_nsm*exp(Sm);

           //delta P due to friction in lbf/ft2
           dp_frm  := (f_tpm * ((Vm^2) *0.5)* rho_nsm * lm) / (32.2 * dm);

  //Total Pressure

        //Delta P friction (in Pa)
           dP[1]  := dp_frm * 47.88;

           //Delta P elevation (in Pa)
           dP[2]  := dp_hhm*47.88;

           //Delta P total (in Pa)
           dP[3]  := dP[1] + dP[2];
   end if;

end BeggsBrill;
```

9.2 SLE Routine

```
within Simulator.Streams;

 model SolidmaterialStream
```

```
 extends Simulator.Files.Icons.SolidMaterialStream;

  import data = Simulator.Files.ChemsepDatabase;
  parameter Integer Nc  "Number of components";
  parameter Real gma = 1;
  parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "Component instances array";
  Real dCp;
  Real xliq "Final estimates for liquid phase mole fraction";
  Real MaxAct[Nc];
  Real xmax[Nc]"Maximum solubilities (mole fractions)";
  Real SumMaxX;
  Real Vx[Nc];
  Real xnl[Nc];
  Real SF;
  Real SLP  ;
  Real SLP2;
  Real Diff[Nc];
  Real xsolid "Final estimates for solid phase mole fraction";
  Real xnliq[Nc] "Liquid phase mole composition";
  Real xnsol[Nc]"Solid phase mole composition";
  Real T(unit = "K") "Temperature";
  Real P(unit = "Pa") "Pressure";
  Real x_pc[3, Nc] "Component mole fraction in phase";
  Real F_p[3](unit = "mol/s")"Total molar flow in phase";
  Real F_pL[Nc](unit = "mol/s")  "Mole flow rate of Liquid composition";
  Real F_pS[Nc](unit = "mol/s")  "Mole flow rate of Solid composition";
  Real Fm_p[3] (unit = "kg/s") "Total mass flow in phase";
  Real Fm_pL[Nc](unit = "kg/s")  "Mass flow rate of Liquid composition";
  Real Fm_pS[Nc](unit = "kg/s")  "Mass flow rate of Solid composition" ;
  Real x1_PC[Nc];

 Simulator.Files.Interfaces.solidConn In(Nc = Nc) annotation(
    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)));
 Simulator.Files.Interfaces.solidConn Out(Nc = Nc) annotation(
    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)));
   equation

  //Connector equations
  In.P = P;
  In.T = T;
  In.F = F_p[1];
  In.x_pc = x_pc;
  In.xsolid = xsolid;
  Out.P = P;
  Out.T = T;
  Out.F = F_p[1];
  Out.x_pc = x_pc;
  Out.xsolid = xsolid;


//
```

```
==========================================================================
   F_p[2]  =  xliq * F_p[1];
   F_p[3]  =  xsolid * F_p[1];

   for i in 1:Nc loop
   F_pL[i] = F_p[2]*xnliq[i];
   F_pS[i] = F_p[3]*xnsol[i];

   Fm_pL[i] = (F_pL[i]*C[i].MW)/1000; // L mass flow rate
   Fm_pS[i] = (F_pS[i]*C[i].MW)/1000;//
   end for;

   Fm_p[2] = sum(Fm_pL[i] for i in 1:Nc);
   Fm_p[3] = sum(Fm_pS[i] for i in 1:Nc);
   Fm_p[1] = Fm_p[2] + Fm_p[3];

   x_pc[1, :] = x1_PC;  // mixture
   x_pc[2,:] = xnliq; //liquid
   x_pc[3,:] = xnsol; //Solid
//==========================================================================
  dCp = 0;

  for i in 1:Nc loop
  MaxAct[i] = exp((((-C[i].HFMP/1000))/ (8.31446 * T)) * (1 - T / C[i].Tm) - (dCp/ (8.31446 * T)*(T - C[i].Tm)) +(dCp/ 8.31446)*log(C[i].Tm / T));
  end for;


  for i in 1:Nc loop
  xmax[i] = MaxAct[i]/gma;
  end for;

  SumMaxX = sum(xmax[i] for i in 1:Nc );


    for i in 1:Nc loop
    if  x1_PC[i] > xmax[i] then
       Vx[i] = xmax[i];        //Component fraction above max solubility. fix fraction to max solubility
    else Vx[i] = 0;
    end if;
    end for;

    for i in 1:Nc loop
    if  x1_PC[i] <= xmax[i] then
       xnl[i] =  x1_PC[i];        //Component fraction below max solubility. -> put to liquid completely
    else xnl[i] = 0;
    end if;
    end for;


```

```

SLP = sum(xnl[i] for i in 1:Nc ); //Sum mole fractions of components in liquid phase
SF  = sum(Vx[i] for i in 1:Nc );  //Sum mole fractions of components fixed by max solubility


 if SumMaxX < 1 then
 xliq = 0 ;
 else  xliq = SLP / (1 - SF);     //Current estimates for liquid phase mole fraction
end if;
 xsolid = 1- xliq;


SLP2 = sum(Diff[i] for i in 1:Nc );

 for i in 1:Nc loop
 if  x1_PC[i] > xmax[i] then
        Diff[i] =  x1_PC[i] - (xmax[i]*xliq);
 else Diff[i] = 0;
 end if;
end for;


 if SumMaxX < 1 then    //Solid
        for i in 1:Nc loop
        xnliq[i] = 0;
        xnsol[i] =  x1_PC[i];
        end for;


 elseif SumMaxX > 1 and xsolid == 1 then    //Solid
        for i in 1:Nc loop
        xnliq[i] = 0;
        xnsol[i] =  x1_PC[i];

        end for;

 elseif xliq > 0 and xliq < 1 and SumMaxX > 1 then    //Solid-Liquid
        for i in 1:Nc loop
        xnliq[i] = ((1-SF)/(SLP/(xnl[i]))) + Vx[i];
        xnsol[i] = ((1)/(SLP2/(Diff[i])));
        end for;

 elseif xliq >= 1 then    //Liquid
        for i in 1:Nc loop
        xnliq[i] = ((1-SF)/(SLP/(xnl[i]))) + Vx[i];
        xnsol[i] = 0;
        end for;

 else   for i in 1:Nc loop //vapour
        xnliq[i] = 0;
        xnsol[i] = 0;
        end for;
end if;
```

```

end SolidmaterialStream;
```

9.3 Solid Separator

```
within Simulator.UnitOperations;

model SolidSeparator
  extends Simulator.Files.Icons.SolidSeparator;
  import data = Simulator.Files.ChemsepDatabase;
  parameter Integer Nc "Number of components" annotation(
    Dialog(tab = "Separator Specifications", group = "Component Parameters"));
  parameter Real LiquidSeparationEfficiency = 100 ;
  parameter Real SolidSeparationEfficiency =  100 ;
  parameter Simulator.Files.ChemsepDatabase.GeneralProperties C[Nc] "Component instances array" annotation(
    Dialog(tab = "Separator Specifications", group = "Component Parameters"));
  Real P(unit = "Pa")  "Inlet stream pressure";
  Real T(unit = "K")   "Inlet stream temperature";
  Real x_pc[3,Nc]     "Inlet stream component mole fraction";
  Real Fm_poutS;
  Real Fm_poutLV;
  Real Pout_s[2](unit = "Pa") "Outlet stream pressure";
  Real Tout_s[2](unit = "K") "Outlet stream temperature";
  Real sse;
  Real lse;
  Real B[Nc];
  Real liqMassFraction[Nc];
  Real liqMoleFraction[Nc];
  Real liqMassFlow[Nc];
  Real liqMolarFlow[Nc];
  Real SumMW1;
  Real SumMW2;
  Real D1[Nc];
  Real D2[Nc];
  Real SolMassFraction[Nc];
  Real SolMoleFraction[Nc];
  Real SolMassFlow[Nc];
  Real SolMolarFlow[Nc];
  Real F_p[3]   (unit = "mol/s") "Total molar flow in phase";
  Real F_pL[Nc](unit = "mol/s") "Mole flow rate of Liquid composition";
  Real F_pS[Nc](unit = "mol/s") "Mole flow rate of Solid composition";
  Real Fm_p[3]  (unit = "kg/s")  "Total mass flow in phase";
  Real Fm_pL[Nc](unit = "kg/s")"Mass flow rate of Liquid composition";
  Real Fm_pS[Nc](unit = "kg/s")"Mass flow rate of Solid composition";
  Real xsolid  "Solid phase mole fraction";
  //Real F_outs;
  //Real F_outl;



  Simulator.Files.Interfaces.solidConn In(Nc = Nc) annotation(
```

```
    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)));
  Simulator.Files.Interfaces.solidConn Out1(Nc = Nc) annotation(
    Placement(visible = true, transformation(origin = {100, 70}, extent = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin = {100, 80}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Simulator.Files.Interfaces.solidConn Out2(Nc = Nc) annotation(
    Placement(visible = true, transformation(origin = {100, -70}, extent = {{-10, -10}, {10, 10}}, rotation = 0), iconTransformation(origin = {100, -80}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));


equation
// Connector equation
  In.P = P ;
  In.T = T;
  In.F = F_p[1] ;
  In.x_pc[:] = x_pc[:];
  In.xsolid = xsolid ;
  Out1.P = Pout_s[1];
  Out1.T = Tout_s[1];
  Out1.F = sum(liqMolarFlow);
  Out1.x_pc[1, :] = liqMoleFraction;
  Out2.F = sum(SolMolarFlow);
  Out2.x_pc[1, :] = SolMoleFraction;
  Out2.P = Pout_s[2];
  Out2.T = Tout_s[2];




  //mole flow rate of liquid and solid phase
   F_p[2]  =  (1-xsolid) * F_p[1];
   F_p[3]  =  xsolid * F_p[1];

   //mass flow rate of liquid and solid phase
   Fm_p[2]  =  sum(Fm_pL[i] for i in 1:Nc);
   Fm_p[3]  =  sum(Fm_pS[i] for i in 1:Nc);
   Fm_p[1] = Fm_p[2] + Fm_p[3];

   //mole flow rate of liquid and solid composition at equillibrium
   for i in 1:Nc loop
   F_pL[i] = F_p[2]* x_pc[2,i];
   F_pS[i] = F_p[3]* x_pc[3,i];

   //mass flow rate of liquid and solid composition at equillibrium
   Fm_pL[i] = (F_pL[i]*C[i].MW)/1000;
   Fm_pS[i] = (F_pS[i]*C[i].MW)/1000;
   end for;


   //Pressure and temperature equations
   Pout_s[1] = P;
   Pout_s[2] = P;
```

```
    Tout_s[1] = T;
    Tout_s[2] = T;

    sse = SolidSeparationEfficiency / 100;
    lse = LiquidSeparationEfficiency / 100;

    Fm_poutS  = (sse * Fm_p[3]) + ((1 - lse) * Fm_p[2]);
    Fm_poutLV = (1 - sse) * Fm_p[3] + (lse * Fm_p[2]);

//=========================Output1 (UP)==========================================

     for i in 1:Nc loop
     B[i] = (1-sse) * Fm_pS[i];
     liqMassFlow[i] = B[i] + lse*( Fm_pL[i]);    //Mass Flow
     liqMassFraction[i] = liqMassFlow[i] / Fm_poutLV;          //mass fraction
     end for;

     for i in 1:Nc loop
     D1[i]=  liqMassFraction[i]/C[i].MW;
     end for;

     SumMW1 = sum(D1[i] for i in 1:Nc );

     for i in 1:Nc loop
     liqMoleFraction[i] = (liqMassFraction[i]/C[i].MW)/(SumMW1); // Mole fraction
     end for;

     for i in 1:Nc loop
     liqMolarFlow[i] = (liqMassFlow[i]/C[i].MW)*1000; //Molar Flow
     end for;

//=========================Output2 (down)=======================================

     for i in 1:Nc loop
     SolMassFlow[i] = sse * Fm_pS[i] + (1-lse)*( Fm_pL[i]);   //Mass Flow
     SolMassFraction[i] = SolMassFlow[i] / Fm_poutS;
     //mass fraction
     end for;

     for i in 1:Nc loop
     D2[i]=  SolMassFraction[i]/C[i].MW;
     end for;

     SumMW2 = sum(D2[i] for i in 1:Nc );

     for i in 1:Nc loop
     SolMoleFraction[i] = (SolMassFraction[i]/C[i].MW)/(SumMW2);      // Mole fraction
     end for;

     for i in 1:Nc loop
     SolMolarFlow[i] = (SolMassFlow[i]/C[i].MW)*1000;                 //Molar
```

```
        Flow
    end for;




end SolidSeparator;
```

9.4 Filter

```
model Filter

// Compound : { Water, Calcium Carbonate}
 parameter Integer Nc = 2 ”Number of components” ;
 parameter Real MW[Nc] = {18.0153, 100.087}; //Molecular weight of components


 parameter Real tc = 300 ”filter cycle”;
 parameter Real f = 0.3  ”Submerged Area Fraction”;
 parameter Real alpha = 2.9E+10 ”Specific Cake Resistance”;
 parameter Real Rm = 0 ”Filter Medium Resistance”;
 parameter Real Rh = 50 ”Cake Relative Humidity”;
 parameter Real rho = 1024.76 ”Liquid Density”;
 parameter Real mu = 0.0009887 ”Liquid Density”;
 parameter Real Pressuredrop = 67702.8 ”pressure drop”;
 parameter Real Area = 3.26889;
 parameter Real CalcMode = 1 ; // 1 for Design   2 for Simulation

 parameter  Real P = 101325 ”Inlet stream pressure”;
 parameter  Real T = 293.15 ”Inlet stream temperature”;
 parameter  Real F_P[3] = {33.7081, 30.9112, 2.79682}; // Molar flow rate of inlet stream
 //parameter  Real x_PC[1, Nc] = {0.95926159, 0.040738415}; // Mole fraction of inlet stream
 parameter  Real x_pL[Nc] = {0.97257458, 0.02742542}; // Mole fraction of Liquid composition
 parameter  Real x_pS[Nc] = {0.8121229, 0.1878771}; // Mole fraction of Solid composition
 parameter  Real Fv_p[1] = {0.000630902};           //Volumetric flow rate of mixture

//Output stream 1
 Real liqMassFraction[Nc] ;
 Real liqMoleFraction[Nc];
 Real liqMassFlow[Nc];
 Real liqMolarFlow[Nc];


 Real SumMW1;
 Real SumMW2;
 Real D1[Nc];
 Real D2[Nc];

```

```

 //Output stream 2
 Real SolMassFraction[Nc];
 Real SolMoleFraction[Nc];
 Real SolMassFlow[Nc];
 Real SolMolarFlow[Nc];


 Real Pout_s[2] "Outlet stream pressure";
 Real Tout_s[2] "Outlet stream temperature";
 Real n;
 Real mf_mc;
 Real crh;
 Real cf;
 Real frh;
 Real c;
 Real At "Total filter area";
 Real dp;
 Real F_PL[Nc];
 Real F_PS[Nc];
 Real Fm_P[3];
 Real Fm_PL[Nc];
 Real Fm_PS[Nc];
 Real Fm_psout;
 Real Fm_plout;

equation

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

   //mass flow rate of liquid and solid phase
   Fm_P[2]  =  sum(Fm_PL[i] for i in 1:Nc);
   Fm_P[3]  =  sum(Fm_PS[i] for i in 1:Nc);
   Fm_P[1] = Fm_P[2] + Fm_P[3];

  //mole flow rate of liquid and solid composition at equillibrium
   for i in 1:Nc loop
   F_PL[i] = F_P[2]* x_pL[i] ;
   F_PS[i] = F_P[3]* x_pS[i];

   //mass flow rate of liquid and solid composition at equillibrium
   Fm_PL[i] = (F_PL[i]* MW[i])/1000;
   Fm_PS[i] = (F_PS[i]* MW[i])/1000;
   end for;

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


// Pressure and temperature equations
   Pout_s[1] = P - dp;
   Pout_s[2] = P - dp;
   Tout_s[1] = T;
   Tout_s[2] = T;


```

```

     n = 1 /tc;
     mf_mc = 100 / (100 - Rh);
     crh = Rh / 100;

     cf =  Fm_P[3] / Fv_p[1];
     frh = Fm_P[2]/ (Fm_P[3] + Fm_P[2]);

     c = cf / (1 - (mf_mc - 1) * (cf / rho));


    if CalcMode == 1 then
    //Calculation Mode Design
    At = Fm_P[3] * alpha / ((2 * c * alpha * Pressuredrop * f * n / mu + (n * Rm) ^ 2) ^ 0.5 - n * Rm);

    else At = Area;
    end if;

    if CalcMode == 2 then
    //Calculation Mode Simulation
    dp = ((n * Rm) ^ 2 + (n * Rm + Wsin * alpha / Area) ^ 2) / (2 * c * alpha * f * n / mu);
    else dp = Pressuredrop;
    end if;

  Fm_psout =  Fm_P[3] / (1 - crh);
  Fm_plout =  Fm_P[1] - Fm_psout;

  //=====================Outlet stream 1=====================

      for i in 1:Nc loop
      liqMassFlow[i]= (Fm_PL[i]* Fm_plout)/ Fm_P[2];             //Mass Flow
      liqMassFraction[i] = liqMassFlow[i] / Fm_plout;         //mass fraction
      end for;

      for i in 1:Nc loop
      D1[i]=  liqMassFraction[i]/MW[i];
      end for;

      SumMW1 = sum(D1[i] for i in 1:Nc );

      for i in 1:Nc loop
      liqMoleFraction[i] = (liqMassFraction[i]/MW[i])/(SumMW1); // Mole fraction
      end for;

      for i in 1:Nc loop
      liqMolarFlow[i] = (liqMassFlow[i]/MW[i])*1000; //Molar Flow
      end for;

  //=========================== Output2 (down) ===========================================

      for i in 1:Nc loop
```

```
    SolMassFlow[i] = Fm_PL[i]*(Fm_P[2]- Fm_plout)/Fm_P[2] + Fm_PS[i];   // Mass Flow
    SolMassFraction[i] = SolMassFlow[i] / Fm_psout;                    // mass fraction
    end for;

    for i in 1:Nc loop
    D2[i]=  SolMassFraction[i]/MW[i];
    end for;

    SumMW2 = sum(D2[i] for i in 1:Nc );

    for i in 1:Nc loop
    SolMoleFraction[i] = (SolMassFraction[i]/MW[i])/(SumMW2);     // Mole fraction
    end for;

    for i in 1:Nc loop
    SolMolarFlow[i] = (SolMassFlow[i]/MW[i])*1000;               //Molar Flow
    end for;


end Filter;
```

9.5 Analysis of Cost Estimation

```
within Simulator;

package CostEstimator
  extends Modelica.Icons.VariantsPackage;

  package CapitalCostEstimator
    extends Modelica.Icons.UtilitiesPackage;

    package SolidFluidPlant
      extends Simulator.Files.Icons.SolidFluidPlant;

      model SolidFluidPlantCE
        parameter Real CalcMode " 1 = Default litrature value
                                2 = Percentage based
                                3 = Absolute Value";
        parameter Real E "Purchased equipment (delivered)";
        //Direct Cost Parameters
        parameter Real A1 = 39 "Purchased equipment installation";
        parameter Real A2 = 26 "Instrumentation and controls (installed)";
        parameter Real A3 = 31 "Piping (installed)";
        parameter Real A4 = 10 "Electrical systems (installed)";
        parameter Real A5 = 29 "Buildings (including services)";
        parameter Real A6 = 12 "Yard improvements";
        parameter Real A7 = 55 "Service facilities (installed)";
        //Indirect Cost Parameters
        parameter Real A8 = 32 "Engineering and supervision";
```

```
  parameter Real A9 = 34 "Construction expenses";
  parameter Real A10 = 4 "Legal expenses";
  parameter Real A11 = 19 "Contractor's fee";
  parameter Real A12 = 37 "Contingency";
  parameter Real W = 75 " Working capital";
  Real C[3];
  Real TF(unit = "$") "Fixed-Capital Investment";
  Real TW(unit = "$") "Working capital";
  Real TC(unit = "$") "Total capital investment";
equation
  if CalcMode == 1 then
    C = DefaultPercentage(E);
    TF = C[1];
    TW = C[2];
    TC = C[3];
  elseif CalcMode == 2 then
    C = UserDefinedPercentage(E, A1, A2, A3, A4, A5, A6, A7, A8,
        A9, A10, A11, A12, W);
    TF = C[1];
    TW = C[2];
    TC = C[3];
  else
    C = UserDefinedAbsolute(E, A1, A2, A3, A4, A5, A6, A7, A8, A9,
        A10, A11, A12, W);
    TF = C[1];
    TW = C[2];
    TC = C[3];
  end if;
end SolidFluidPlantCE;

function DefaultPercentage
  input Real E;
  Real f[12], I, D;
  output Real C[3];
algorithm
  f[1] := 0.39 * E;
  f[2] := 0.26 * E;
  f[3] := 0.31 * E;
  f[4] := 0.10 * E;
  f[5] := 0.29 * E;
  f[6] := 0.12 * E;
  f[7] := 0.55 * E;
  D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
  f[8] := 0.32 * E;
  f[9] := 0.34 * E;
  f[10] := 0.04 * E;
  f[11] := 0.19 * E;
  f[12] := 0.37 * E;
  I := f[8] + f[9] + f[10] + f[11] + f[12];
  C[1] := I + D;
  C[2] := 0.75 * E;
  C[3] := C[1] + C[2];
end DefaultPercentage;

function UserDefinedPercentage
  input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12,
```

```
         W;
      Real f[12], I, D;
      output Real C[3];
    algorithm
      f[1] := A1 * 0.01 * E;
      f[2] := A2 * 0.01 * E;
      f[3] := A3 * 0.01 * E;
      f[4] := A4 * 0.01 * E;
      f[5] := A5 * 0.01 * E;
      f[6] := A6 * 0.01 * E;
      f[7] := A7 * 0.01 * E;
      D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
      f[8] := A8 * 0.01 * E;
      f[9] := A9 * 0.01 * E;
      f[10] := A10 * 0.01 * E;
      f[11] := A11 * 0.01 * E;
      f[12] := A12 * 0.01 * E;
      I := f[8] + f[9] + f[10] + f[11] + f[12];
      C[1] := I + D;
      C[2] := W * 0.01 * E;
      C[3] := C[1] + C[2];
    end UserDefinedPercentage;

    function UserDefinedAbsolute
      input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12,
         W;
      Real I, D;
      output Real C[3];
    algorithm
      D := E + A1 + A2 + A3 + A4 + A5 + A6 + A7;
      I := A8 + A9 + A10 + A11 + A12;
      C[1] := I + D;
      C[2] := W * 0.01 * E;
      C[3] := C[1] + C[2];
    end UserDefinedAbsolute;
  end SolidFluidPlant;

  package FluidPlant
    extends Simulator.Files.Icons.FluidPlant;

    function DefaultPercentage
      input Real E;
      Real f[12], I, D;
      output Real C[3];
    algorithm
      f[1] := 0.47 * E;
      f[2] := 0.36 * E;
      f[3] := 0.68 * E;
      f[4] := 0.11 * E;
      f[5] := 0.18 * E;
      f[6] := 0.10 * E;
      f[7] := 0.70 * E;
      D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
      f[8] := 0.33 * E;
      f[9] := 0.41 * E;
      f[10] := 0.04 * E;
```

```
    f[11] := 0.22 * E;
    f[12] := 0.44 * E;
    I := f[8] + f[9] + f[10] + f[11] + f[12];
    C[1] := I + D;
    C[2] := 0.89 * E;
    C[3] := C[1] + C[2];
  end DefaultPercentage;

  function UserDefinedPercentage
    input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, W;
    Real f[12], I, D;
    output Real C[3];
  algorithm
    f[1] := A1 * 0.01 * E;
    f[2] := A2 * 0.01 * E;
    f[3] := A3 * 0.01 * E;
    f[4] := A4 * 0.01 * E;
    f[5] := A5 * 0.01 * E;
    f[6] := A6 * 0.01 * E;
    f[7] := A7 * 0.01 * E;
    D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
    f[8] := A8 * 0.01 * E;
    f[9] := A9 * 0.01 * E;
    f[10] := A10 * 0.01 * E;
    f[11] := A11 * 0.01 * E;
    f[12] := A12 * 0.01 * E;
    I := f[8] + f[9] + f[10] + f[11] + f[12];
    C[1] := I + D;
    C[2] := W * 0.01 * E;
    C[3] := C[1] + C[2];
  end UserDefinedPercentage;

  function UserDefinedAbsolute
    input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, W;
    Real I, D;
    output Real C[3];
  algorithm
    D := E + A1 + A2 + A3 + A4 + A5 + A6 + A7;
    I := A8 + A9 + A10 + A11 + A12;
    C[1] := I + D;
    C[2] := W * 0.01 * E;
    C[3] := C[1] + C[2];
  end UserDefinedAbsolute;

  model FluidPlantCE
    parameter Real CalcMode " 1 = Default litrature value
                              2 = Percentage based
                              3 = Absolute Value";
    parameter Real E "Purchased equipment";
    //Direct Cost Parameters
    parameter Real A1 = 47 "Purchased equipment installation";
    parameter Real A2 = 36 "Instrumentation and controls (installed)";
    parameter Real A3 = 68 "Piping (installed)";
```

```
      parameter Real A4 = 11 "Electrical systems (installed)";
      parameter Real A5 = 18 "Buildings (including services)";
      parameter Real A6 = 10 "Yard improvements";
      parameter Real A7 = 70 "Service facilities (installed)";
      //Indirect Cost Parameters
      parameter Real A8 = 33 "Engineering and supervision";
      parameter Real A9 = 41 "Construction expenses";
      parameter Real A10 = 4 "Legal expenses";
      parameter Real A11 = 22 "Contractor's fee";
      parameter Real A12 = 44 "Contingency";
      parameter Real W = 89 "Working capita percentage";
      Real C[3];
      Real TF(unit = "$") "Fixed-Capital Investment";
      Real TW(unit = "$") "Working capital";
      Real TC(unit = "$") "Total capital investment";
    equation
      if CalcMode == 1 then
        C = DefaultPercentage(E);
        TF = C[1];
        TW = C[2];
        TC = C[3];
      elseif CalcMode == 2 then
        C = UserDefinedPercentage(E, A1, A2, A3, A4, A5, A6, A7, A8,
            A9, A10, A11, A12, W);
        TF = C[1];
        TW = C[2];
        TC = C[3];
      else
        C = UserDefinedAbsolute(E, A1, A2, A3, A4, A5, A6, A7, A8, A9,
            A10, A11, A12, W);
        TF = C[1];
        TW = C[2];
        TC = C[3];
      end if;
    end FluidPlantCE;
  end FluidPlant;

  package SolidPlant
    extends Simulator.Files.Icons.SolidPlant;

    function DefaultPercentage
      input Real E;
      Real f[12], I, D;
      output Real C[3];
    algorithm
      f[1] := 0.45 * E;
      f[2] := 0.18 * E;
      f[3] := 0.16 * E;
      f[4] := 0.10 * E;
      f[5] := 0.25 * E;
      f[6] := 0.15 * E;
      f[7] := 0.40 * E;
      D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
      f[8] := 0.33 * E;
      f[9] := 0.39 * E;
      f[10] := 0.04 * E;
```

```
    f[11] := 0.17 * E;
    f[12] := 0.35 * E;
    I := f[8] + f[9] + f[10] + f[11] + f[12];
    C[1] := I + D;
    C[2] := 0.70 * E;
    C[3] := C[1] + C[2];
  end DefaultPercentage;

  function UserDefinedPercentage
    input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, W;
    Real f[12], I, D;
    output Real C[3];
  algorithm
    f[1] := A1 * 0.01 * E;
    f[2] := A2 * 0.01 * E;
    f[3] := A3 * 0.01 * E;
    f[4] := A4 * 0.01 * E;
    f[5] := A5 * 0.01 * E;
    f[6] := A6 * 0.01 * E;
    f[7] := A7 * 0.01 * E;
    D := E + f[1] + f[2] + f[3] + f[4] + f[5] + f[6] + f[7];
    f[8] := A8 * 0.01 * E;
    f[9] := A9 * 0.01 * E;
    f[10] := A10 * 0.01 * E;
    f[11] := A11 * 0.01 * E;
    f[12] := A12 * 0.01 * E;
    I := f[8] + f[9] + f[10] + f[11] + f[12];
    C[1] := I + D;
    C[2] := W * 0.01 * E;
    C[3] := C[1] + C[2];
  end UserDefinedPercentage;

  function UserDefinedAbsolute
    input Real E, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, W;
    Real I, D;
    output Real C[3];
  algorithm
    D := E + A1 + A2 + A3 + A4 + A5 + A6 + A7;
    I := A8 + A9 + A10 + A11 + A12;
    C[1] := I + D;
    C[2] := W * 0.01 * E;
    C[3] := C[1] + C[2];
  end UserDefinedAbsolute;

  model SolidPlantCE
    parameter Real CalcMode " 1 = Default litrature value
                     2 = Percentage based
                     3 = Absolute Value";
    parameter Real E "Purchased equipment (delivered)";
    //Direct Cost Parameters
    parameter Real A1 = 45 "Purchased equipment installation";
    parameter Real A2 = 18 "Instrumentation and controls (installed)";
    parameter Real A3 = 16 "Piping (installed)";
```

```
        parameter Real A4 = 10 "Electrical systems (installed)";
        parameter Real A5 = 25 "Buildings (including services)";
        parameter Real A6 = 15 "Yard improvements";
        parameter Real A7 = 40 "Service facilities (installed)";
        //Indirect Cost Parameters
        parameter Real A8 = 33 "Engineering and supervision";
        parameter Real A9 = 39 "Construction expenses";
        parameter Real A10 = 4 "Legal expenses";
        parameter Real A11 = 17 "Contractor's fee";
        parameter Real A12 = 35 "Contingency";
        parameter Real W = 70 " Working Capital";
        Real C[3];
        Real TF(unit = "$") "Fixed-Capital Investment";
        Real TW(unit = "$") "Working capital";
        Real TC(unit = "$") "Total capital investment";
      equation
        if CalcMode == 1 then
          C = DefaultPercentage(E);
          TF = C[1];
          TW = C[2];
          TC = C[3];
        elseif CalcMode == 2 then
          C = UserDefinedPercentage(E, A1, A2, A3, A4, A5, A6, A7, A8,
              A9, A10, A11, A12, W);
          TF = C[1];
          TW = C[2];
          TC = C[3];
        else
          C = UserDefinedAbsolute(E, A1, A2, A3, A4, A5, A6, A7, A8, A9,
              A10, A11, A12, W);
          TF = C[1];
          TW = C[2];
          TC = C[3];
        end if;
      end SolidPlantCE;
    end SolidPlant;
  end CapitalCostEstimator;

  package OperatingCostEstimator
    extends Modelica.Icons.UtilitiesPackage;

    model operatingCost



      parameter Real CalcMode " 1 = Default litrature value
                                 2 = Absolute Value";


      parameter Real RM(unit = "$") "Raw materials";
      parameter Real U(unit = "$") "Utilities";
      parameter Real OL(unit = "$") "Operating labour";
      parameter Real F(unit = "$") "Fixed-Capital Investment";
      parameter Real S(unit = "$") "Revenue";
//
    ================================================================================
```

```
    //Variable Costs Parameters
    parameter Real B1(unit = "$") "Direct supervisory and clerical labor)";
    parameter Real B2(unit = "$") "Maintenance and repairs";
    parameter Real B3(unit = "$") "Operating supplies";
    parameter Real B4(unit = "$") "Laboratory charges";
    parameter Real B5(unit = "$") "Patents and royalties";
    //Fixed Costs Parameters
    parameter Real B6(unit = "$") "Local Taxes/Income Taxes";
    parameter Real B7(unit = "$") "Insurances";
    parameter Real B8(unit = "$") "Financing (interest)";
    //Plant Overheads Parameters
    parameter Real B9(unit = "$") "Plant overheads";
  //General Expenses Parameters
    parameter Real B10(unit = "$")  "Administrative costs";
    parameter Real B11(unit = "$") "Distribution and marketing costs";
    parameter Real B12(unit = "$") "Research and development costs";


    Real P[6];
    Real TV(unit = "$") "Total Variable costs";
    Real TF(unit = "$") "Total Fixed costs";
    Real TP(unit = "$") "Total Plant Overheads Costs";
    Real TMC(unit = "$") "Manufacturing cost";
    Real TGC(unit = "$") "General Expenses";
    Real TPC(unit = "$/Year") "Total product cost";



  equation
   if CalcMode == 1 then
      P = DefaultPercentage(RM, U, OL, F, S);
      TV  = P[1];
      TF  = P[2];
      TP  = P[3];
      TGC = P[4];
      TMC = P[5];
      TPC = P[6];
    else
      P = UserDefinedAbsolute(RM, U, OL, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11, B12);
      TV  = P[1];
      TF  = P[2];
      TP  = P[3];
      TGC = P[4];
      TMC = P[5];
      TPC = P[6];
    end if;

  end operatingCost;

  function DefaultPercentage


    input Real RM, U, OL, F, S;
```

```
      Real R[14];
      output Real C[6];


    algorithm
//Direct Production Costs Calculation
      R[1] := RM;
      R[2] := U;
      R[3] := OL;
      R[4] := 0.2 * OL;
      R[5] := 0.1 * F;
      R[6] := 0.01 * F;
      R[7] := 0.15 * OL;
      R[8] := 0.01 * F;
      C[1] := R[1] +R[2] + R[3] + R[4] + R[5] + R[6] + R[7] + R[8];
//Fixed Costs Calculation
      R[9] := 0.01 * F;
      R[10]  := 0.01 * F;
      R[11] := 0.01 * F;
      C[2] := R[8] + R[10] + R[11];
//Plant Overheads Calculation
      R[12] := 0.81 * OL;
      R[13] := 0.025 * F;
      C[3] := R[12] + R[13];
//General Expenses Calculation
      R[14] := 0.025 * S;
      C[4] := R[14];
//Manufacturing cost Calculation
      C[5] := C[1] + C[2] + C[3];
//Total Product Cost Calculation
      C[6] := C[4] + C[5];
    end DefaultPercentage;

    function UserDefinedAbsolute


      input Real RM, U, OL, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11,
          B12;
      Real R[14] ;
      output Real C[6];


    algorithm
//Direct Production Costs Calculation
      R[1] := RM;
      R[2] := U;
      R[3] := OL;
      R[4] := B1;
      R[5] := B2;
      R[6] := B3;
      R[7] := B4;
      R[8] := B5;
      C[1] := R[1] +R[2] + R[3] + R[4] + R[5] + R[6] + R[7] +R[8];
//Fixed Costs Calculation
      R[9] := B6;
      R[10] := B7;
```

```
      R[11] := B8;
      C[2]  := R[9] + R[10] + R[11];
//Plant Overheads Calculation
      R[12] := B9;
      C[3] := R[12] + R[13];
//General Expenses Calculation
      R[13] := B10;
      R[14] := B11;
      R[15] := B12;
      C[4]  := R[14];
//Manufacturing cost Calculation
      C[5] := C[1] + C[2] + C[3];
//Total Product Cost Calculation
      C[6] := C[4] + C[5];
    end UserDefinedAbsolute;
  end OperatingCostEstimator;

  package Examples "Examples that demonstrate the usage of the Capital
      Cost Estimator packages"
    extends Modelica.Icons.ExamplesPackage;

    package FluidPlantCostEstimation
      extends Modelica.Icons.ExamplesPackage;

      model CostEstimator
        extends
            Simulator.CostEstimator.CapitalCostEstimator.FluidPlant.FluidPlantCE
            ;
      end CostEstimator;

      model Flowsheet
        extends Modelica.Icons.Example;
        //This example is adopted from Development of a Technology for
            Treating Wastewater Contaminated with Nitric Acid
        //https://www.researchgate.net/publication/258391609
            _Development_of_a_Technology_for_Treating_Wastewater_Contaminated_with_Nitric_A
            /figures?lo=1
        //Caculation mode is 2 : User defined percentage values.
        Simulator.CostEstimator.Examples.FluidPlantCostEstimation.CostEstimator
             C1(A1 = 20, A10 = 4, A11 = 15, A12 = 22, A2 = 15, A3 = 25,
            A4 = 8, A5 = 10, A6 = 5, A7 = 50, A8 = 10, A9 = 20, CalcMode
            = 2, E = 368014) annotation(
          Placement(visible = true, transformation(origin = {-14, 10},
              extent = {{-40, -40}, {40, 40}}, rotation = 0)));
      equation

      end Flowsheet;
    end FluidPlantCostEstimation;

    package SolidPlantCostEstimation
      extends Modelica.Icons.ExamplesPackage;

      model CostEstimator
        extends
            Simulator.CostEstimator.CapitalCostEstimator.SolidPlant.SolidPlantCE
            ;
```

```
      end CostEstimator;

      model Flowsheet
        extends Modelica.Icons.Example;
        Simulator.CostEstimator.Examples.SolidPlantCostEstimation.CostEstimator
             C1(CalcMode = 1, E = 500000) annotation(
          Placement(visible = true, transformation(origin = {3, 1},
              extent = {{-43, -43}, {43, 43}}, rotation = 0)));
      equation

      end Flowsheet;
    end SolidPlantCostEstimation;

    package OperatingCostEstimation
      extends Modelica.Icons.ExamplesPackage;
      model OperatingCost
extends Simulator.CostEstimator.OperatingCostEstimator.operatingCost;
      end OperatingCost;

      model Flowsheet
        extends Modelica.Icons.Example;
      //Example Reference:-

      //https://www.researchgate.net/publication/304590984
          _Effect_of_particle_size_on_anaerobic_digestion_of_cow_dung_chicken_manure_pig_man
          /figures?lo=1

      //Table 7 : Summary of annual production of AD technology model


      Simulator.CostEstimator.Examples.OperatingCostEstimation.OperatingCost
           C1(CalcMode = 1, F = 1977305, OL = 201600, RM = 0, S =
          3988407.60, U = 275721.60)  annotation(
          Placement(visible = true, transformation(origin = {-7, 15},
              extent = {{-43, -43}, {43, 43}}, rotation = 0)));
      equation

      end Flowsheet;
    end OperatingCostEstimation;
  end Examples;
end CostEstimator;
```