Development of a Thermodynamic Engine in OpenModelica

Rahul Jain[1] Kannan M. Moudgalya[1] Peter Fritzson[2] Adrian Pop[2]

[1]Dept. of Chemical Engineering, Indian Institute of Technology Bombay, India,
rahjain1@gmail.com,kannan@iitb.ac.in
[2]Dept. Computer and Information Sciences, Linköping University, Sweden,
{peter.fritzson,adrian.pop}@liu.se

Abstract

OpenModelica, an open source equation oriented modeling environment for steady state and dynamic simulation, lacks good chemical engineering support. This problem is addressed by making available in different ways the thermodynamic library Chemsep that comes with DWSIM, an open source sequential modular steady state simulator. Only slow speeds could be achieved through a Python-C API based interface connecting OpenModelica with the thermodynamic library. A socket programming based interface helps achieve faster speeds. Best results have been achieved by porting the thermodynamic library and the calculation routines to OpenModelica, due to two reasons: (1) thermodynamic equations are solved simultaneously with mass and energy balances (2) overheads in calling the external routines of DWSIM are eliminated. Performances of the above mentioned three approaches have been validated with steady state and dynamic simulations. Benzene - toluene separation, methanol - ethanol - water distillation, and steam distillation of an n-octane - n-decane mixture, have been carried out through these simulations. This work makes available a powerful simulation platform to the chemical engineering.
Keywords: OpenModelica, DWSIM, Chemsep, thermodynamics, modeling, simulation, chemical engineering, Python-C API, socket programming, media

Abbreviations

API	Application programming interface
csv	Comma separated values
dll	Dynamic link library
DTL	DWSIM thermodynamics library
EOS	Equation of state
VLE	Vapor liquid equilibrium

1 Introduction

Modelica (Modelica Association, 2000) is a powerful modelling language and OpenModelica (Fritzson, 2014) is its open source implementation. In OpenModelica has an excellent interface to build models and to perform simulations. As it implements an equation oriented solution approach, models and solution methods are maintainable (Piela et al., 1992). Many engineering domains have used OpenModelica.

Unfortunately, OpenModelica does not have a library of chemical engineering models and a thermodynamic database. As a result, it is not yet of much use to the chemical engineering community. If we can add a CAPE Open thermodynamic database to OpenModelica, it can immensely increase the utility for chemical engineers.

DWSIM is a state of the art open source steady state process simulator (Medeiros, 2015). It comes with two CAPE Open thermodynamic databases, Chemsep (Kooijman and Taylor, 2001) and the native one. In this work, we describe the different methods to make the DWSIM chemical engineering library available for OpenModelica.

This paper is organized as following. We explain the Python-C interfacing approach to call the DWSIM's thermodynamic database from OpenModelica. We then explain how to instead use socket programming to connect the two simulators. The final part is devoted to the porting of thermodynamics in native mode on to OpenModelica. We conclude with a comparison of the three approaches.

2 Importing the Thermodynamic engine of DWSIM in OpenModelica

As DWSIM is based on sequential modular solution techniques (Westerberg et al., 1979), it is more suitable to solve *analysis* type of problems. One will have to resort to iterations to *design* systems, which may involve finding the value of some parameters in the output stream or in the equipment, or the building block. It is also difficult to carry out dynamic simulation in DWSIM. DWSIM has a strong thermodynamic engine. DWSIM also has a standalone thermodynamic library (DTL) which can be used externally.

The weakness of DWSIM is the strong point of OpenModelica: it is equation oriented and capable of handling unsteady state equations. Similarly, the weakness of OpenModelica is the strength of DWSIM: thermodynamic database and routines. As they complement each other, there is a good case to integrate OpenModelica with DTL.

2.1 Python-C API approach to Integrate OpenModelica with DTL

DTL consists of a file with an extension .dll (dynamic link library) which is written in VB.NET in windows en-

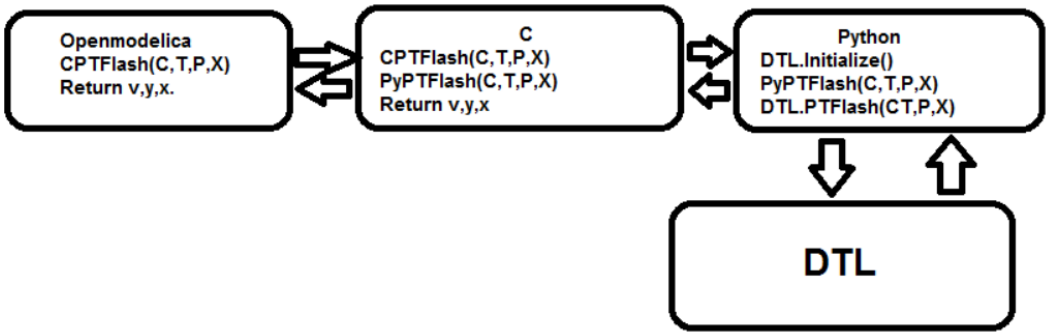


Figure 1. Structure of Python-C API approach

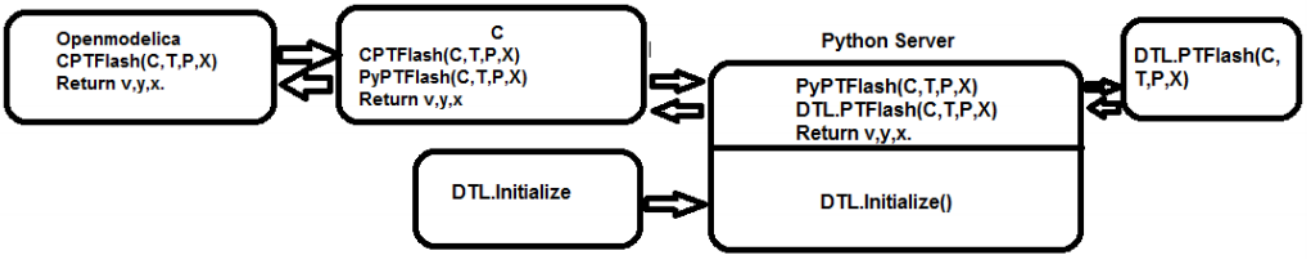


Figure 2. Structure of Python-C socket approach

vironment. This file is COM (component object model) enabled, which means that any programming language which supports COM can import this library and access the built-in thermodynamic subroutines. OpenModelica is written in C in Linux environment and it is not straight forward to call programs written in VB.NET.

We used Python as the glue language to call the COM enabled objects of DWSIM from C routines of OpenModelica. This was achieved through the package *win32com* of Python. This allowed us to access the DTL library and all the thermodynamic routines available in DWSIM from OpenModelica. Figure 1 describes the flow of the approach, which are further described below.

- DTL routines are imported to Python first through a package named win32com.client. This package allows Python to call routines from a dll file registered in the windows registry. Once the dll file is dispatched through win32com.client, Python has access to all the COM enabled functions of the dll.

- Now Python functions can send input parameters to DTL routines, get the required thermodynamic properties calculated and receive them. As results of calculations are available, these Python functions can be considered to behave similar to DTL routines.

- These Python functions are now called by C through Python-C API. In computer programming, an API (Application Programming Interface) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. This API is responsible for converting C variables to Python and vice versa.

- Finally as OpenModelica is compatible with C, the inputs are then sent to C functions through OpenModelica external C functions, which in turn calls the Python functions, which in turn calls DTL routines.

2.2 Client-Server (sockets) approach to integrate OpenModelica with DTL

Client-Server or socket approach (Rhodes and Goerzen, 2010) is another approach through which the integration is possible. Figure 2 describes the data flow of the approach, which are further explained below.

- In this approach also, firstly the the DTL routines are called in Python with the help of win32com.client package.

- Similar to Python-C approach, functions are written in Python which calls DTL to calculate various physical properties.

- Now a Python server which consists of all the above functions is created. This server waits for a C client to establish connection, receive inputs from it and send the calculated values back to the client.

- For every calculation (e.g. vapor pressure, equilibrium constant, etc.), a Python server is established.

DOI 10.3384/ecp1713289

Table 1. Thermodynamic routines and the procedures to call them

Thermodynamic Prop.	Thermodynamic Func.	Arguments
Vapor Pressure	VapPres	Comp,T
Enthalpy	Ent	Comp., T,P
Liquid Density	LiqDen	Comp,T
Vapor Density	VapDen	Comp,T
Pres. Temp. Flash	PTFlash	Comp,Z,T,P,Model
Pres. Volume. Flash	PVFlash	Comp,Z,V,P,Model
Pres. Enth. Flash	PHFlash	Comp,Z,H,P,Model
Liquid Viscosity	LiqVis	Comp,T
Vapor Viscosity	VapVis	Comp,T
Surface tension	SurfTen	Comp,T

- Clients are coded in C which establishes connections with the Python servers and send and receive data from them.
- Once the connection is established, a Python server receives the data from C client, contacts DTL, calculates the required property as asked by the C client and sends it back to the client.
- Finally, these C clients are called by OpenModelica external C functions giving the required inputs to the client which in turn contacts the Python servers for calculations. The C clients receive calculated values from Python servers and transfer them to OpenModelica.

2.3 Comparison of the two approaches

In this section, we compare the two approaches presented above. In both approaches, before any routine in DTL is called, one has to carry out initialization. This *Initialize* routine loads all the compounds and their properties from the database, which is a time consuming operation. In the Python-C API approach, this initialization is done every time a call is made from OpenModelica to DTL. On the other hand, this has to be done only once in the Client Server approach. As a result, the latter is far more efficient than the former.

Whenever an API is used in any program it makes it slow as there is a lot of conversions involved, such as data type conversions. As the Python-C API approach is based on API it is slow.

To verify the speeds of two approaches, we use the thermodynamic calculations presented next. Table 1 lists the thermodynamic functions and their arguments that we have implemented in OpenModelica to receive values from DTL. These functions can be used directly in any simulation. When using the socket approach, the Python server should be up and running during the execution of the simulation. We now present two case studies that helped compare the two approaches.

- **Steady State Flash Separator**
  An equimolar mixture of Benzene and Toluene was flashed in a flash separator. The thermodynamic package used was Raoult's law. All the pure component and mixture properties were imported from DTL. To test the capability of the integration methods, the composition of the resulting vapor stream was specified, and the temperature at which this composition was attained was left unknown. It was observed that the Python-C API approach took 30 seconds to solve the system, whereas the Client-Server method took less than 1 second to simulate.

- **Dynamic Flash**
  A dynamic flash was simulated with the feed as equimolar mixture of benzene and toluene. The thermodynamic package used was again Raoult's Law. It was assumed that the output liquid stream was at the same composition and temperature, as the holdup inside the flash separator. Heat supplied to flash separator was kept constant. The set of equations involved were mass balance, energy balance and equilibrium equations. The mass and energy balance were differential equations. It was observed that the Python-C API approach took 30 minutes to solve, whereas the Client-Server approach took 4 minutes to solve the system.

The above two examples and others that we have not reported here show that the Client-Server approach is more efficient than the Python-C API approach.

3 Development of a native thermodynamic engine in OpenModelica

A thermodynamic engine consists of the following three components: Compound database, thermodynamic functions and phase equilibria models. In this section, we describe how we have developed a native thermodynamic engine in OpenModelica.

3.1 Development of Compound Database

A Compound database is a comprehensive database of physical and chemical properties of all compounds. It also includes constants for calculating various temperature or pressure dependent properties like vapor pressure, enthalpy, viscosity, etc.

We first describe the Chemsep (Kooijman and Taylor, 2001) database that we ported to OpenModelica. Chemsep is an open source database, written in **xml** format. It has over six hundred compounds with a comprehensive set of thermodynamic properties of each compound. It also has an extensive database of binary interaction parameters for thermodynamic packages like NRTL (Renon and Prausnitz, 1968), Peng Robinson (Peng and Robinson, 1976), UNIQUAC, SRK (Soave, 1972), etc. Most of the thermodynamic properties are calculated by empirical equations that are functions of temperature or pressure. Chemsep database includes the constants which are used in these equations. Therefore, Chemsep database is

Table 2. Independent thermodynamic properties and OpenModelica routines to call them.

Thermodynamic Property	Calling procedure
Critical Temperature	Compound.Tc
Critical Pressure	Compound.Pc
Critical Volume	Compound.Vc
Boiling point	Compound.Tb
Melting point	Compound.Tm
Molecular weight	Compound.MW
Acentric Factor	Compound.AF
Triple Point	Compound.TT
Solubility parameter	Compound.SP
Dipole moment	Compound.DP
Heat of formation	Compound.HOF
Absolute enthalpy	Compound.ABSENT

a comprehensive database that can be used to built a powerful and robust thermodynamic engine.

We now explain how we ported Chemsep to OpenModelica. First, the xml data have to be rewritten in a form understandable by OpenModelica. Therefore, each compound (including all its thermodynamic properties) is replicated as a single class in OpenModelica, as shown in Figure 3. The properties are given abbreviations (as shown in Table 2) so that they can be called conveniently. The conversion from xml to OpenModelica classes is carried out by developing a Python script which automates this process, thus making it fast and robust.

Now, one can extract any independent property of a compound by the **.** (dot) operator, followed by the property relevant abbreviation. For example the critical temperature (Tc) of methane can be accessed by **Methane.Tc**. Similarly, all properties of any compound can be accessed in the same way as shown in table 2.

3.2 Development of Thermodynamic Functions

Thermodynamic properties are generally calculated through empirical equations that include constants, whose values are provided by the compound database as explained above, and independent variables, such as temperature, pressure and composition. These properties are written in the form of functions in OpenModelica. Arguments to these functions are the independent variables mentioned above, and the coefficients of respective compounds whose properties have to be calculated. These coefficients can be accessed by instantiating the base compound class. The functions then return the calculated property. For example, the vapor pressure of methane at 300 K can be calculated by first instantiating the base Methane class (**parameter Methane methane**) and then calling **Pvap(methane.VP, 300)**. Where Pvap is a generic function to calculate the vapor pressure of any compound at any given temperature. The whole process is shown in Figure 4. Similarly, all other thermodynamic properties can be calculated using their respective functions as shown in Table 3.

Table 3. Dependent thermodynamic properties and OpenModelica functions to call them.

Thermodynamic Property	Calling procedure
Liquid density	LiqDen(Compound name,temp)
Vapor pressure	VP(Compound name,temp)
Heat of Vaporization	HOV(Compound name,temp)
Liquid heat capacity	LiqCp(Compound name,temp)
Liquid viscosity	LiqVis(Compound name,temp)
Vapor viscosity	VapVis(Compound name,temp)
Liquid thermal conductivity	LiqK(Compound name,temp)
Vapor thermal conductivity	VapK(Compound name,temp)

3.3 Development of Phase Equilibria models

Phase equilibria models consist of modelling equations for Vapor Liquid Equilibrium (VLE) models like Peng Robinson, NRTL, UNIQUAC, etc. These models are used to predict the behavior of various systems.

In a mixture of phases that are in an equilibrium, the component fugacities are the same in all phases (Smith et al., 2005), that is :

$$f_i^L = f_i^V \tag{1}$$

where f_i^L and f_i^V are the liquid and vapor phase fugacities of the ith component respectively. The fugacity of a component in a mixture depends on temperature, pressure and composition. In order to relate fi^V with temperature, pressure and molar fraction, we define the fugacity coefficient,

$$\Phi_i = \frac{f_i^V}{y_i P^*} \tag{2}$$

where Φ_i is the fugacity coefficient and P^* is the pressure of the system, which can be calculated from PVT data, commonly obtained from an equation of state (EOS). For a mixture of ideal gases, $\Phi_i = 1$. The fugacity of component i in the liquid phase is related to the composition of that phase by the activity coefficient γ_i, which by itself is related to x_i and standard-state fugacity f_i^0 by

$$\gamma_i = \frac{f_i^L}{x_i f_i^0} \tag{3}$$

The standard state fugacity f_i^0 is the fugacity of the ith component at the system temperature, i.e. mixture, and in an arbitrary pressure and composition. Here, the standard-state fugacity of each component is considered to be equal to pure liquid i at the system temperature and pressure. An Equation of State is used to calculate equilibria. The fugacity of the ith component in the liquid phase is calculated by

$$\gamma_i = \frac{f_i^L}{x_i P^*} \tag{4}$$

DOI 10.3384/ecp1713289

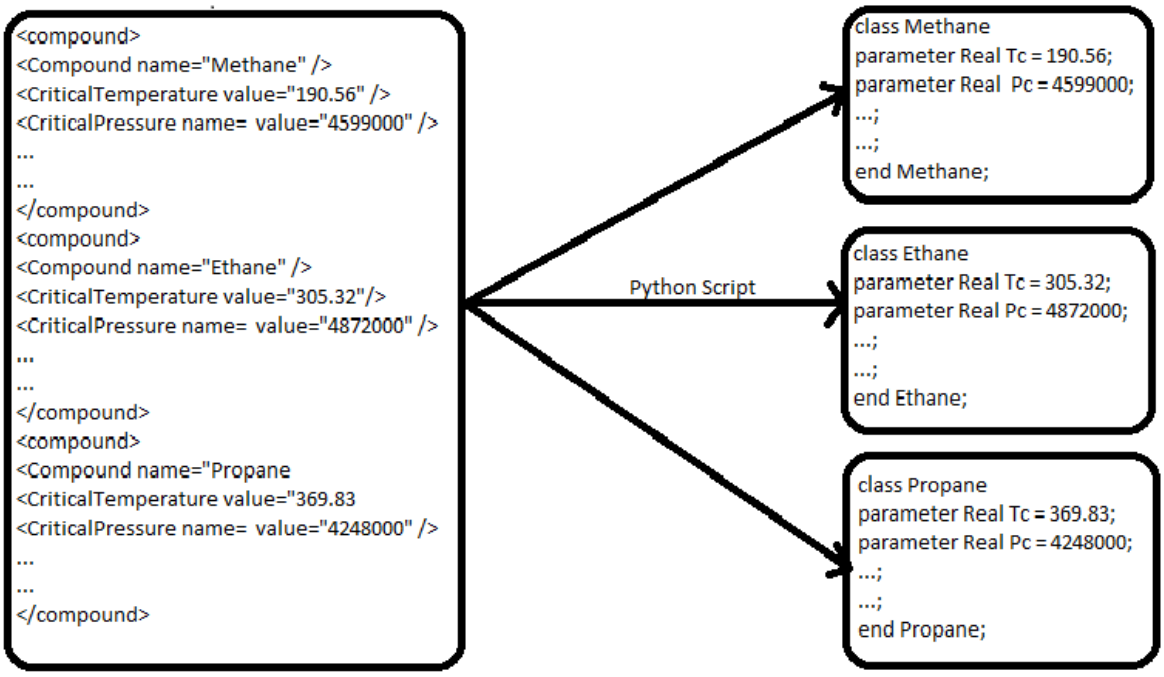


Figure 3. Porting Chemsep database in OpenModelica

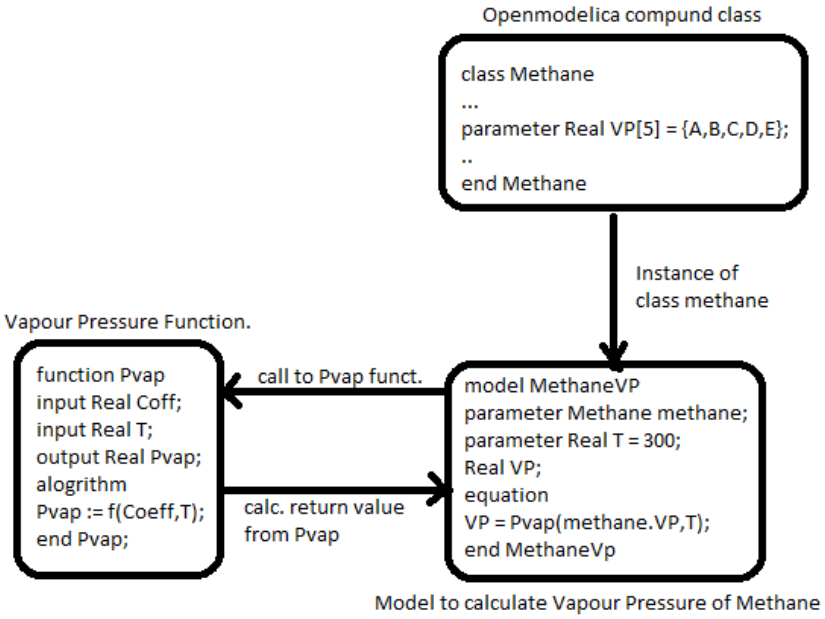


Figure 4. Using built in thermodynamic functions (Pvap)

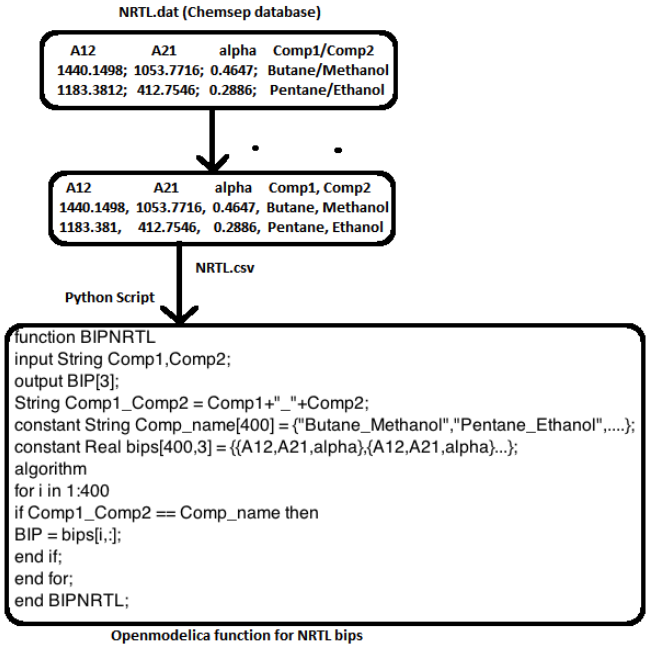


Figure 5. Porting Chemsep's binary interaction parameters in OpenModelica

with the fugacity coefficient Φ_i calculated by the EOS, just as it is done for the vapor phase.

We have implemented the following four phase equilibria models: Peng Robinson, SRK, NRTL and UNIQUAC. Peng Robinson and SRK are the most abundantly used EOS models, whereas NRTL and UNIQUAC find a wide variety of applications where activity coefficient models are required (Medeiros, 2015).

The binary interaction parameters for each of the EOS and activity coefficient models have been extracted from Chemsep database where they are stored in a .dat file. The following procedure is used to port all the binary interaction parameters to OpenModelica.

- First, the dat file is converted to a csv file, which is easier to process by Python.
- This csv file is then converted to an OpenModelica function by a Python script which converts the compound and the binary interaction parameters as an array.

Figure 5 demonstrates the above process for NRTL activity coefficient model. This code is automatically generated by the Python script. Line 6 of this code has been shortened for convenience. The actual code has 400 triplets of real numbers on the right hand side of line 6.

Figure 6 shows the NRTL model. The model acquires the required binary interaction parameters from the BIPNRTL function. The model incorporates the equilibrium relation described in equation 4. This model can now be directly extended into any model which requires calculation of phase equilibrium. All other phase equilibria models have been modeled similarly.

```
model NRTL
parameter Real BIP[3] = BIPNRTL(Comp.name); //Binary interaction parameters
Real P, T (start = 373, min = 350, max = 380); //Pressure and Temperature
Real x[2](each start = 0.5, each min = 0.0001, each max = 1); //liquid compositions
Real y[2](each start = 0.5, each min = 0.0001, each max = 1); //vapor compositions
Real gamma[2](each start = 1 ); //activity coefficients
equation
gamma = f(x,y,P,T,BIP); //equilibrium equations relating gamma,x,y,P,T,BIP
end NRTL;
```

Figure 6. NRTL model as written in OpenModelica 1.11.0

4 VLE curve (Txy) for a binary system through the UNIQUAC model

In this section, we explain the procedure to generate the VLE curve for a binary system and demonstrate it with results from an ethanol-water system.

We will first explain the procedure to generate the bubble point curve. Suppose γ_1 and γ_2 are the activity coefficients, y_1 and y_2 are vapor phase compositions, x_1 and x_2 are liquid phase compositions, and $Pvap_1$, $Pvap_2$ are corresponding vapor pressures, of components 1 and 2, respectively. Then, the following equations are used to generate the bubble point curve.

$$y_1.P = \gamma_1.x_1.Pvap_1 \tag{5}$$

$$y_2.P = \gamma_2.x_2.Pvap_2 \tag{6}$$

This is known as the modified Raoult's law.

Adding the above two equations and equating the vapor phase mole fractions to one ($y_1 + y_2 = 1$), we get

$$P = \gamma_2.x_2.Pvap_2 + \gamma_1.x_1.Pvap_1 \tag{7}$$

Here γ_1 and γ_2 are complex nonlinear functions of temperature and liquid compositions and $Pvap_1$ and $Pvap_2$ are functions of temperature.

The pressure is kept constant at 1 atm. The value of x_1 is varied from 0 to 1 with an interval of 0.1 and for each value of x_1, the corresponding value of temperature is calculated by equation 7. This is known as the bubble point.

Now we explain how the dew point curve is generated. We once again use the modified Raoult's law for this purpose. Manipulating the equations 5 and 6 and putting $x_1 + x_2 = 1$ we get

$$\frac{y_1}{\gamma_1.Pvap_1} + \frac{y_2}{\gamma_2.Pvap_2} = 1 \tag{8}$$

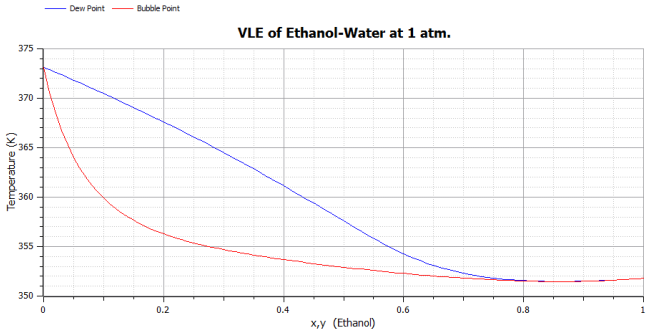
The pressure is kept constant at 1 atm. The value of y_1 is varied from 0 to 1 with an interval of 0.1 and for each value of y_1 the corresponding value of temperature if calculated by equation 8.

Figure 7 describes the implementation of the bubble point model in OpenModelica. The dew point model have also has been modeled similarly. As shown all the three parts of the thermodynamic engine, namely, compound

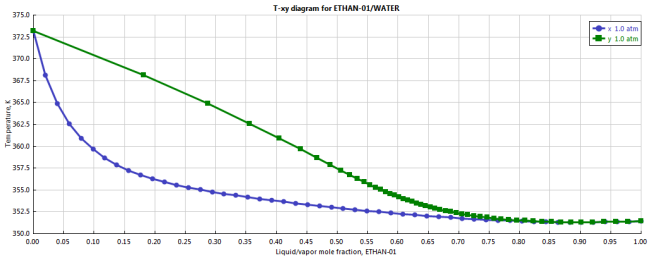
DOI 10.3384/ecp1713289

```
model bubblepnt
parameter Ethanol ethanol; // compound properties from database
parameter Water water; // compound properties from database
parameter Real z[2] = {0.5, 0.5};
parameter Real compound[2] = {ethanol,water}
extends UNIQUAC(Comp = compound, P = 101325); //Phase equilibria model
Real  Psat[2];
algorithm
Psat:= Thermodynamic_Functions.Psat(Compound.VP, T); //Thermodynamic functions
equation
x[1] = time;
x[2] = 1 - x[1];
P = sum(gamma[:] .* x[:] .* Psat[:]);
y[1] = 0;
y[2] = 0;
end bubblepnt;
```

Figure 7. Bubble point model as written in OpenModelica 1.11.0



(a) Results from OpenModelica 1.11.0



(b) Results from Aspen Plus 8.1

Figure 8. Comparison of T-xy curve for ethanol water system using UNIQUAC VLE model

database, thermodynamic functions and phase equilibria models, have been incorporated in the model.

Using the above procedure, we have calculated the bubble point curve and the dew point curve for the ethanol(1)-water(2) system, and presented them in Figure 8(a). One can see it to be identical to the figure generated by Aspen Plus (Aspentech, 2017), presented in Figure 8(b).

Reliable azeotropic data source by American Chemical Society (Gmehling et al., 1995) says that for ethanol-water system, at 1 atm, the azeotropic composition and temperatures are 0.96 mole fraction ethanol and 351.4 K, respectively. These values are also in agreement with the OpenModelica results.

The same simulation when carried out with the imported DWSIM's thermodynamic engine in OpenModelica resulted in an execution time of about 20 minutes, whereas for the built in thermodynamic engine, the execution time was 0.58s.

5 Steady State Flash

Now that thermodynamics is available in OpenModelica, we simulate a steady state flash of a methanol, ethanol, water system, using NRTL. To check the design efficiency of the developed thermodynamic engine in OpenModelica, the output composition of the vapor product is specified, while the temperature at which this desired composition of vapor is attained is left unspecified. Thus, it is a design problem. To carry out this simulation in DWSIM, we have to use the *adjust* operation that uses a trial and error method.

We now explain the problem we propose to solve. The flow rate and the composition of the feed is specified. Pressure is kept constant at 1 atm. It is desired to calculate all other variables for three different values of methanol mole fraction, x_1. In other words, vapor compositions of ethanol and water, temperature of all streams and all flow rates need to be calculated. The schematic of the problem statement is presented in Figure 9.

The input stream enters at 1 atm and its temperature is to be determined according to the specified input composition. The simulation is run for three different desired vapor compositions of methanol. The minimum and maximum temperatures were taken to be boiling points of pure methanol and water and the initial guess for temperature is taken to be average of these two boiling points. The following equations describe the model.

Mass balance:

$$z_i F = x_i L + y_i V \tag{9}$$

$$F = L + V \tag{10}$$

Equilibrium equation:

$$y_i = K_i x_i \tag{11}$$

Summation Equation:

$$\sum_{i=1}^{2} y_i = 1 \tag{12}$$

Here, F, L, V are the feed, liquid, and vapor flow rates, respectively, in kmol/hr and z_i, x_i, y_i are the feed, liquid, and vapor compositions respectively. K_i is the equilibrium constant. UNIQUAC activity coefficient model is used as the phase equilibria model.

Figure 10 depicts the example as developed in OpenModelica. The type *compound* in the fifth line is a general class used to represent the compound class.

Results of these calculations have been presented in Table 4. One can see that these results are consistent with the general requirement that higher the mole fraction of the least volatile component, lower the temperature. All calculations got completed in OpenModelica in less than a second.

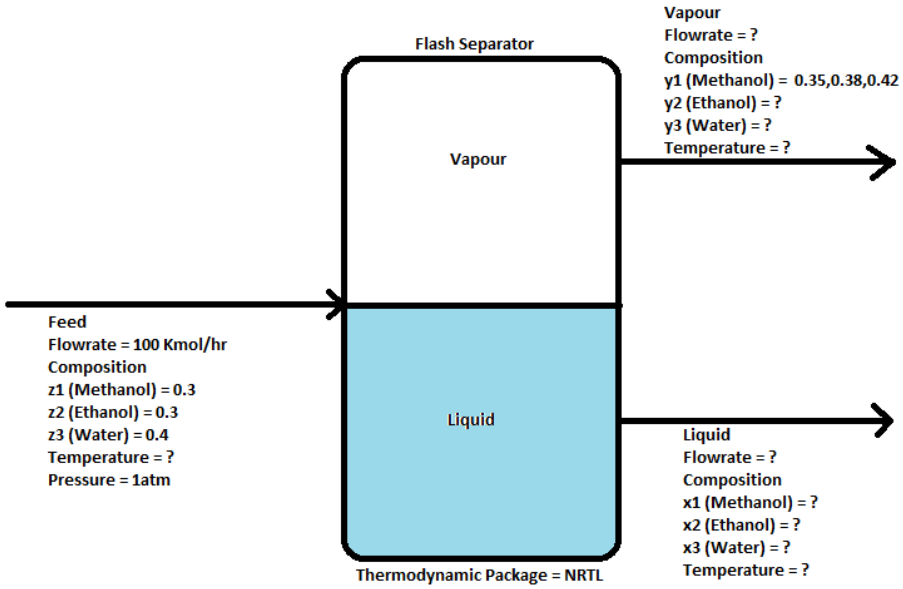


Figure 9. Model with problem statement for steady state flash of Methanol-ethanol-water.

```
model Flash
parameter Methanol methanol;
parameter Ethanol ethanol;
parameter Water water;
parameter compound compounds[3] = {methanol,ethanol,water};
parameter Real z[3] = {0.3,0.3,0.4};
Real x[3], y[3], k[3], T, Psat[3];
Real L,V,F;
extends UNIQUAC{Comp=compounds,P=101325}; //
equation
y[1] = 0.35;
F = 100;
z[:].*F - (x[:].*L+y[:].*V) = 0;
y[:] = k[:].*x[:];
k[:] = {gamma[:].*x[:].*Psat[:])/P;
Psat[:] = Thermodynamic_Functions.Psat(compounds.VP,T);
end Flash;
```

Figure 10. Flash model as written in OpenModelica 1.11.0

Same calculations are repeated in DWSIM using the *adjust* function, by trial and error, and the results are reported in the same Table. One can see the results to be comparable. It took 15 to 20 seconds to do each of these calculations in DWSIM, however.

6 Semi-Batch Steam Distillation of a Binary Organic Mixture

We now illustrate the ease with which dynamic simulation can be carried out in OpenModelica, using the semi-batch steam distillation of a binary mixture. We present the model first and then an example.

Table 4. Results of simulation in OpenModelica using the built-in thermodynamics and in DWSIM

OpenModelica 1.11.0		
Desired Vapor Comp.(Methanol)	Temperature	Liquid Comp.
0.35	351.21	0.1985
0.38	350.28	0.2354
0.425	349.24	0.274
DWSIM 3.4		
0.35	351.26	0.199
0.38	350.211	0.234
0.425	349.12	0.279

6.1 Model of the process

This illustrative example involves semi-batch steam distillation of a binary mixture (n-octane and n-decane). A schematic plot of the steam distillation apparatus is shown in Figure 11. The organic mixture is charged into the still initially, and then steam is bubbled through continuously until the desired degree of separation has been reached. There are two different periods in the operation of the still: the *heating period*, until the boiling point temperature of the organic mixture is reached, and the *distillation period*. A brief description of the mathematical models for the two periods follows (Shacham et al., 2012).

We present the model for the heating period first. A simple mass balance on the water phase yields

$$\frac{dm_w}{dt} = W_s \tag{13}$$

where W_s is the steam flow rate in kmol/s and m_w is the mass of water in the still in kmol. It is assumed that all the steam condenses in the distillation vessel and that the organic phase masses remain constant during the heating period.

DOI 10.3384/ecp1713289

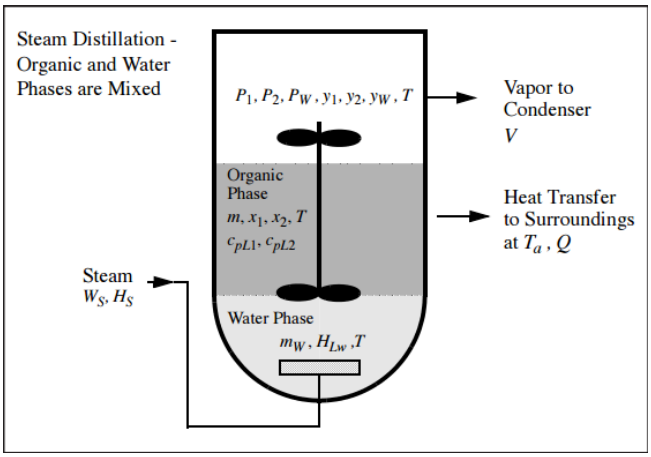


Figure 11. Schematic of steam distillation apparatus (Shacham et al., 2012).

An energy balance on the still provides the equation for the change of the temperature T in °C

$$\frac{dT}{dt} = \frac{W_s(H_s - H_{lw}) - Q}{m_w c_{pLw} + m(x_1 c_{pL1} + x_2 c_{pL2})} \tag{14}$$

where H_s is the enthalpy of the steam in J/kmol, H_{lw} is the enthalpy of liquid water in J/kmol, Q is the rate of heat transfer to the surroundings in J/sec, c_{pLw} is the molar specific heat of the water in J/kmol-K, m is the mass of the organic phase in the still in kmol, x_1 and x_2 are the mole fractions, and c_{pL1} and c_{pL2} are the molar specific heats of organic compounds No. 1 and 2, respectively, in J/kmol-K. The heat transfer rate to the surroundings is calculated from the following equation.

$$Q = UA(T - T_a) \tag{15}$$

where UA is the product of the overall heat transfer coefficient U and the contact area A with the surroundings in J/s-K, T_a is the ambient temperature in K, and T is the temperature of the liquid in the still in K.

Assuming ideal liquid behavior, Raoult's law can be used to calculate the vapor mole fraction of the components in the organic phase

$$y_1 = \frac{x_1 P_1}{P} y_2 = \frac{x_2 P_2}{P} \tag{16}$$

where P is the total pressure in Pa and P_1 and P_2 are the vapor pressures of the organic compounds in Pa. The mole fraction of the water which is immiscible in the organic phase is given by $y_W = P_W/P$. y_1 and y_2 are the vapor phase mole fraction of n-octane and n-decane respectively. The heating period continues until the sum of vapor pressures of the organic compounds and the water is equal to the total pressure. Thus, the bubble point equation to be satisfied can be expressed as

$$f(T) = 1 - (y_1 + y_2 + y_w) = 0 \tag{17}$$

We now present the model for the distillation period. During the distillation period, there is output of water vapor from the still.

$$\frac{dm_w}{dt} = W_s - V y_w \tag{18}$$

where V is the outlet vapor flow rate. Material balances on the two organic compounds yield two additional differential equations

$$\frac{d(mx_1)}{dt} = -V y_1 \frac{d(mx_2)}{dt} = -V y_2 \tag{19}$$

The organic mass in the still at any time is given by: $m = mx_1 + mx_2$. The temperature in the still changes in a manner so that the bubble point equation is satisfied. The energy balance at a particular temperature yields the momentary vapor flow rate

$$V = \frac{W_s(H_S - H_{LW}) - Q}{Hv - [y_w h_{lw} + (y_1 h_{L1} + y_2 h_{L2})]} \tag{20}$$

where H_V is the molar enthalpy of the vapor phase; h_{Lw}, h_{L1}, and h_{L2} are the liquid phase molar enthalpies of water, n-octane and n-decane, respectively. Material balances on the water and organic phases in the still can provide the amount and the mole fractions of the various components in the distillate.

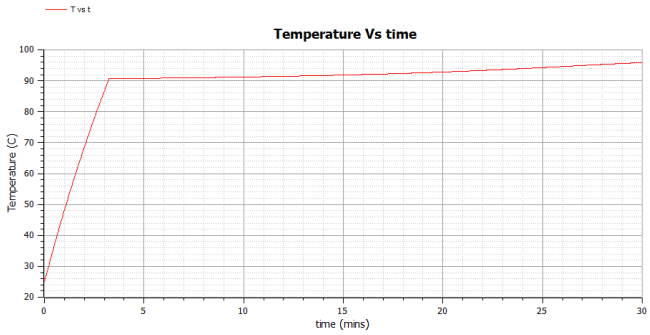
6.2 Example: n-octane, n-decane distillation

Semibatch steam distillation of a mixture containing n-octane (compound 1) and n-decane (compound 2) is to be processed. Initially M = 0.015 kmol of organics with composition x_1 = 0.725 is charged into the still. The initial temperature in the still is T_0 = 25 °C. Starting at time t = 0, steam at a temperature Tsteam = 99.2 °C is bubbled continuously through the organic phase at the rate of M_S = 3.85e-5 kmol/s. All the steam is assumed to condense during the heating period. The ambient temperature is T_E = 25 °C and the heat transfer coefficient between the still and the surrounding is UA = 1.05 J/s-K. The ambient pressure is P = 9.839E+04 Pa.

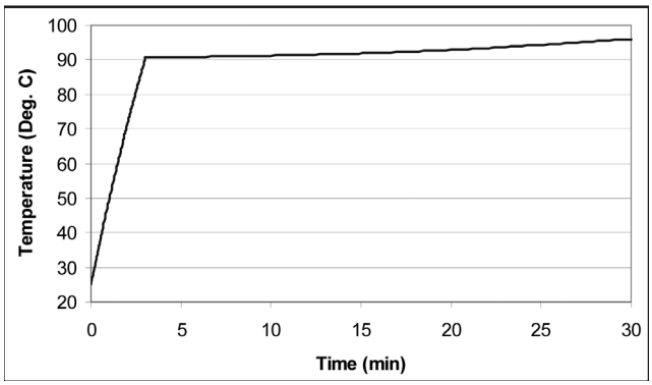
Assumptions: 1) Ideal behavior of all components in pure state or mixture; 2) complete immiscibility of the water and the organic phases; 3) ideal mixing in the boiler; and 4) equilibrium between the organic vapor and its liquid at all times. The standard state for enthalpy calculations pure liquids at 0 °C and 1 atm. can be used.

We have to Calculate and plot the still temperature (T), component mole fractions inside the still (x_1, x_2, y_1, and y_2), and the component mole fractions in the distillate (x_{1dist} and x_{2dist}) using the data and the initial values provided.

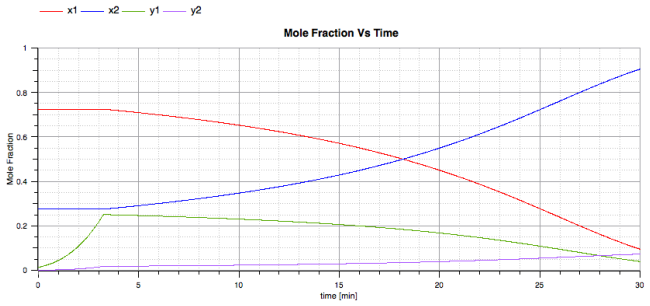
We have to determine the lowest n-octane mole fraction in the feed that can yield a distillate concentration of 90% of n-octane. Compute the percent recovery of n-octane in the distillate as function of its concentration in the feed.



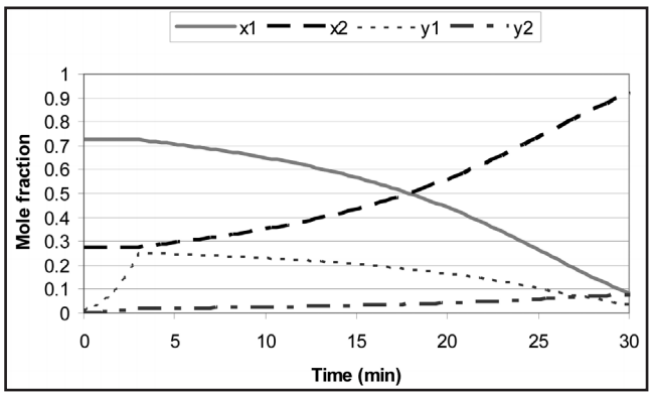
(a) Temperature profiles generated in OpenModelica 1.11.0



(b) Temperature profiles as reported in literature (Shacham et al., 2012)



(c) Profiles of vapor and liquid compositions, as generated in OpenModelica 1.11.0



(d) Profiles of vapor and liquid compositions, as reported in the literature (Shacham et al., 2012)

Figure 12. Comparison of temperature and composition change during semi-batch steam distillation

Vary the feed concentration in the range where the requirement for the n-octane concentration in the distillate is attainable.

Plots in Figure 12 shows that the results of OpenModelica are in agreement with that of (Shacham et al., 2012). It can be observed that the temperature increases during the heating period and then stays constant during the distillation period when the Bubble point is attained. The liquid phase compositions is constant during the heating period as there is no vapor formation.

7 Conclusion and Future Work

In this paper, we have implemented and compared three different ways of making available thermodynamics in OpenModelica. Each of these approaches has been illustrated with simulations of one or more chemical processes. We have found the native port to be the most efficient.

We have compared the results of OpenModelica with those from DWSIM, Aspen Plus, and published literature, and they match quite well in all calculations.

As now OpenModelica has its own thermodynamic engine, a library of steady state chemical process models could be modeled. It may also be possible to build a library of dynamic chemical process models in OpenModelica, to carry out general purpose dynamic simulation.

We hope to explore the possibility of enhancing OMEDIT's (Asgharand et al., 2011) features so that the GUI shall resemble that of established simulators, such as Aspen Plus or DWSIM, for chemical process simulation.

We propose to check the correctness of thermodynamic calculations by solving a large number of already solved flowsheets. Sources for these will include examples from books, journals, reports and sample problems from other process simulators. We hope to present these flowsheets in a way similar to what we have done for DWSIM (DWSIM-Team-FOSSEE-Project, 2017). Usefulness of such an initiative has been articulated in a similar context (Braatz, 2014).

A difficult task we face is with respect to thermodynamics in general and the thermodynamic database, in particular. This facility has to be strengthened by adding information on more chemicals and more thermodynamic calculations. We invite experts in this important area to contribute and to make OpenModelica a much better open source process simulator.

Acknowledgements

This work has been supported by Swedish Vinnova governmental agency and the Indian Department of Science and Technology governmental agency in the Indo-Swedish RTISIM project, and by the National Mission on Education through ICT, Ministry of Human Resource Development, through the FOSSEE project. The OpenModelica development is supported by the Open Source Modelica Consortium.

References

Adeel Asgharand, Sonia Tariq, Mohsen Torabzadeh-Tariand, Peter Fritzson, Adrian Pop, Martin Sjolund, Parham Vasaiely, and Wladimir Schamai. An open source modelica graphic editor integrated with electronic notebooks and interactive simulation. *Proc. of the 8th International Modelica Conference 2011, pp*, pages 739–747, 2011.

Aspentech. Aspen plus. http://www.aspentech.com/products/engineering/aspen-plus/, 2017. Last seen on 1 April 2017.

R. D. Braatz. Scilab textbook companions. *IEEE Control Systems Magazine*, page 76, June 2014.

DWSIM-Team-FOSSEE-Project. Dwsim flowsheets. http://dwsim.fossee.in/flowsheeting-project/completed-flowsheet, 2017. Last seen on 1 Aprtil 2017.

Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Second edition, 2014. ISBN 9781118989166. doi:10.1002/9781118989166.

Jürgen Gmehling, Jochen Menke, Jörg Krafczyk, and Kai Fischer. A data bank for azeotropic data - status and applications. *Fluid Phase Equilibria*, 103(1):51–76, 1995. ISSN 03783812. doi:10.1016/0378-3812(94)02569-M.

H.A. Kooijman and R. Taylor. *The ChemSep Book*. Books on Demand, Norderstedt, Germany, 2001.

Daniel Medeiros. Dwsim technical document. Technical report, 2015. http://dwsim.inforside.com.br/.

Modelica Association. ModelicaTM - A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification. *ReVision*, 2000. ISSN 09284869. doi:10.1016/S0928-4869(97)84257-7.

Ding-Yu Peng and Donald B. Robinson. A New Two-Constant Equation of State. *Industrial & Engineering Chemistry Fundamentals*, 15(1):59–64, 1976. ISSN 0196-4313. doi:10.1021/i160057a011. URL http://pubs.acs.org/doi/abs/10.1021/i160057a011.

Peter Piela, Roy McKelvey, and Arthur Westerberg. An introduction to the ascend modeling system: Its language and interactive environment. *Journal of Management Information Systems*, 9(3):91–121, 1992.

Henri Renon and J. M. Prausnitz. Local compositions in thermodynamic excess functions for liquid mixtures. *AIChE Journal*, 14(1):135–144, 1968. ISSN 15475905. doi:10.1002/aic.690140124.

Brandon Rhodes and John Goerzen. Foundations of Python Network Programming, 2010. ISSN 1098-6596. URL http://www.springerlink.com/index/10.1007/978-1-4302-3004-5{%}5Cnhttp://it-ebooks.info/book/1796/.

M. Shacham, M. B. Cutlip, and M. Elly. Semi-Batch Steam Distillation Of a Binary Organic Mixture: a Demonstration of Advanced Problem-Solving Techniques and Tools. *Chemical Engineering Education*, 46(3):173–181, Summer 2012.

J M Smith, H C Van Ness, and M M Abbott. *Introduction to Chemical Engineering Thermodynamics*, volume 27. McGraw Hill Education, 2005. ISBN 0072402962. doi:10.1021/ed027p584.3.

G. Soave. Equilibrium constants from a modified redlich-kwong equation of state. *Chemical Engineering Science*, 27(6):1197–1203, 1972.

A.W. Westerberg, H.P. Hutchison, R.L. Motard, and P. Winter. *Process Flowsheeting*. Cambridge University Press, Cambridge, 1979.