I&EC research
Industrial & Engineering Chemistry Research

Article

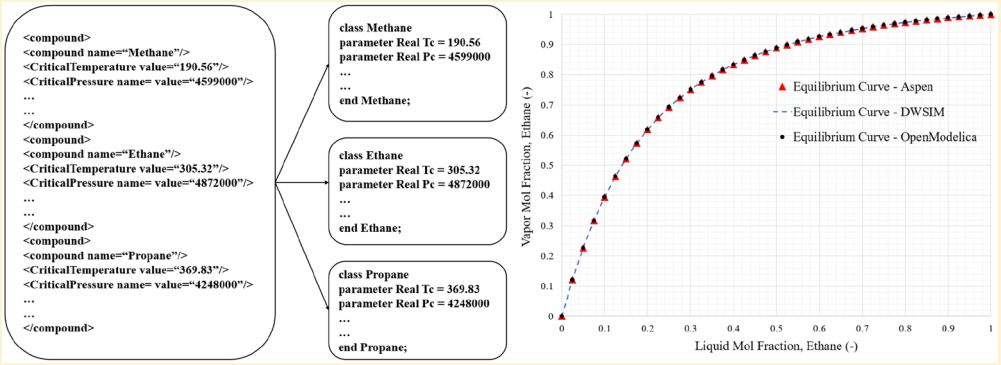
Cite This: Ind. Eng. Chem. Res. 2019, 58, 7551−7560

pubs.acs.org/IECR

Implementation of a Property Database and Thermodynamic Calculations in OpenModelica for Chemical Process Simulation

Rahul Jain,[†] Priyam Nayak,[†] Rahul A. S,[†] Pravin Dalve,[†] Kannan M. Moudgalya,*[,†] P. R. Naren,[‡] Daniel Wagner,[¶] and Peter Fritzson[§]

[†]Department of Chemical Engineering, IIT Bombay, Mumbai 400 076, India
[‡]School of Chemical & Biotechnology, SASTRA Deemed University, Thanjavur, India
[¶]Independent Researcher, Manaus, Brazil
[§]Department of Computer and Information Science, Linköping University, Linköping Sweden



ABSTRACT: An attempt has been made to enhance the thermodynamic capability of the general purpose modeling and simulation environment OpenModelica. The property database ChemSep and the thermodynamic algorithms of DWSIM are made available in OpenModelica. The following three approaches, listed in the order of increasing computational efficiency, are attempted in this work: Python-C API, socket programming, and a native port. The most efficient method of *native port* is adopted to make available NRTL, Peng−Robinson, UNIFAC, and UNIQUAC algorithms in OpenModelica. Through several examples, OpenModelica results are compared with Aspen Plus, indicating a good match in all cases. This work is released as an open source to enhance the collaboration among chemical engineers.

1. INTRODUCTION

This article has its origins to an employment generation project at IIT Bombay for chemical engineers. There are 350 institutions in India that produce a total of ∼20 000 chemical engineers every year. Among them, those who find jobs mainly do so in the information technology (IT) field. With the impending automation, low-level IT jobs are expected to decrease substantially.[1] Since this is expected to affect mainly the jobs performed by non-IT specialists, employment avenues available for chemical engineers are expected to shrink further.

On the other hand, there are many small- and medium-sized chemical processing plants that do not get any engineering input. The flowsheeting project[2] aims to train chemical engineers in a big way, using the open source process simulator DWSIM,[3] and make them suitable for employment by the chemical industry. Approximately 100 flowsheets have been completed by students of many engineering colleges in different parts of India. About 50 more flowsheets are in progress. This pool of chemical engineers with simulation skills is now available for employment and to tackle research challenges in the general area of process simulation.

Most of the small- and medium-scale chemical companies operate in batch mode, to simulate which dynamic simulation capability is required in the simulator. Although there are a few established dynamic process simulators,[4,5] they are proprietary and are not available free of cost. Country-wide education projects, such as the one reported here, cannot be undertaken with commercial software. As a matter of fact, students who contributed to the flowsheeting projects have been trained using Spoken Tutorials[6] on DWSIM.[7]

OpenModelica[8] is an open source platform for dynamic simulation. It implements the Modelica language.[9] It is well developed: the default installation comes with more than 50 libraries from different domains, for example. OpenModelica code is available for solved examples of more than 20 standard chemical engineering books,[10] crowd-sourced through students who have been trained with Spoken Tutorials.[11] OpenModelica implements the equation oriented methodology, which is ideal for the maintenance of models and methods.[12]

Special Issue: Vinay Juvekar Festschrift

Received: October 17, 2018
Revised: February 15, 2019
Accepted: February 20, 2019
Published: February 20, 2019

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

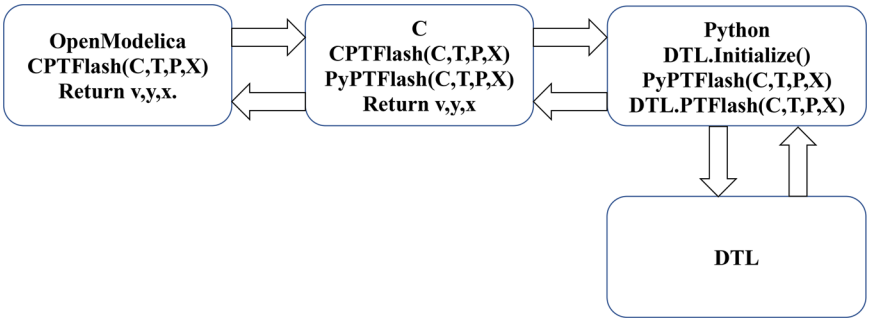


Figure 1. Structure of the Python-C API approach.

The main shortcoming of OpenModelica, with respect to process simulation, is that the chemical engineering support is minimal, in terms of property databases, thermodynamic calculations, and a model library. The current work focuses on the first two of this list. Sandrock and de Vaal[13] were the early authors who attempted to provide thermodynamic capability in OpenModelica. A method to connect to external thermodynamic calculations from Modelica was demonstrated by Windahl et al.[14]

Windahl et al.[14] indicated that it would be desirable to perform the thermodynamic calculations inside Modelica itself, from a calculation efficiency point of view. This is an important observation, as thermodynamic calculations are known to form a major portion of the simulation time. Jain et al.[15] implemented Raoult's law computations inside OpenModelica and demonstrated that it resulted in much more efficient computations. Their work is improved and expanded in this Article. We have used the thermodynamic algorithms of DWSIM[3] and the ChemSep database[16] in this work.

This paper is organized as follows: Section 2 presents two different methods of accessing the ChemSep database from OpenModelica and compares their performance. In Section 3, a native thermodynamic port in OpenModelica is presented and its superiority over other methods is demonstrated. Section 4 demonstrates with examples an implementation of NRTL,[17] Peng–Robinson,[18] UNIFAC,[19] and UNIQUAC[20] in OpenModelica and compares the results with Aspen Plus.[21] The last section is devoted to conclusions and future work.

2. TWO METHODS OF USING THE THERMODYNAMIC ENGINE OF DWSIM IN OPENMODELICA

The objective of this work is to make available thermodynamics in OpenModelica. DWSIM's thermodynamic capability is chosen for this purpose, because of the following reasons:

(1) DWSIM already has a property database ChemSep, which is CAPE Open compliant. DWSIM is also constantly adding more thermodynamic databases, such as CoolProp, Electrolytes, and Biodiesel, and the online Korean database.
(2) DWSIM already has a reliable package, DWSIM Thermodynamic Library (DTL), that can be accessed independently by any program. DTL performs thermodynamic calculations and returns results.
(3) All of the new steady-state simulation results, before being accepted into OpenModelica, can be validated with those of DWSIM. This makes it easy for crowdsourcing.

In this section, two approaches for integrating DTL with OpenModelica are presented: (1) a Python-C API and (2) a Client-Server approach.

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 a Windows environment. This file is COM (component object model)-enabled, and therefore, any programming language that supports COM can import this library and access the built-in thermodynamic subroutines. OpenModelica is written in C language in a Linux environment, and it is not an easy task to call programs written in VB.NET.

Python is used as the glue language to call the COM-enabled objects of DWSIM from C routines of OpenModelica. This is achieved through the package *win32com* of Python. It allows one to access the DTL library and all the thermodynamic routines available in DWSIM from OpenModelica. Figure 1 describes the flow of the approach, which is explained now.

- 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 of 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, since OpenModelica is compatible with C, the inputs are then sent to C functions through OpenModelica external C functions. Subsequently, this invokes the Python functions, which, in turn, call DTL routines.

This Python-C API approach presented in this section turned out to be extremely inefficient, as explained next. In the next section, we present a better way to interface OpenModelica with DTL.

2.2. Client-Server (Socket Programming) Approach To Integrate OpenModelica with DTL.

Client-Server or socket programming approach[22] is another method through which the integration is possible. Figure 2 describes the data flow of the approach, which is explained next.

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

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

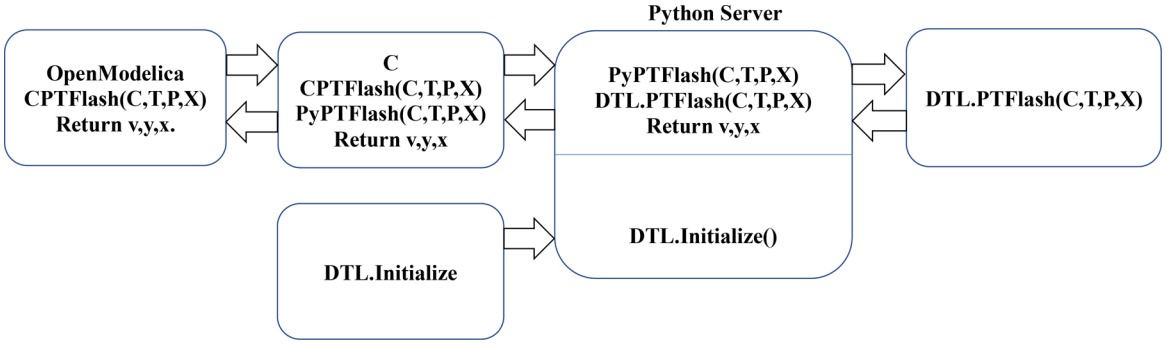


Figure 2. Structure of the Python-C socket programming approach.

- Similar to the Python-C approach, functions are written in Python, which calls DTL to calculate various physical properties.
- Now, a Python server, which consists of all of 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.
- Clients are coded in C language, which establishes connections with the Python servers and sends and receives data from them.
- Once the connection is established, a Python server receives the data from the 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, the two approaches presented above are compared. In both approaches, before any routine in DTL is called, one must perform 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. Whenever an API is used in any program, it makes it slow, since there is a lot of conversions involved, such as data-type conversions. As the Python-C API approach is based on the API, it is slow. On the other hand, this must be done only once in the Client Server approach. As a result, the Client-Server approach is far more efficient than the former.

To verify the speeds of two approaches, the thermodynamic calculations presented next are used. Table 1 lists the thermodynamic functions and their arguments that are implemented in OpenModelica to receive values from DTL. These functions can be used directly in any simulation. When using the socket programming approach, the Python server should be up and running during the execution of the simulation.

We compared the speeds of the two approaches with a steady-state flash separator example. An equimolar mixture of benzene and toluene was flashed in a flash separator. Raoult's law was used for thermodynamic calculations. 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, but its temperature was left unknown. It was observed that the Python-C API approach required 30 s to solve the system, whereas the Client-Server method required <1 s to simulate. A similar high performance of the Client-Server approach was noticed in many other examples as well.

Table 1. Thermodynamic Routines and the Procedures Used To Call Them

thermodynamic properties	thermodynamic functions	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
liquid viscosity	LiqVis	Comp, T
vapor viscosity	VapVis	Comp, T
surface tension	SurfTen	Comp, T

Although the Client-Server approach is more efficient, it is still not good enough, especially for dynamic simulations. This is the motivation to explore the possibility of making a native port of ChemSep into OpenModelica. This is the topic of discussion in the next section.

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, steps undertaken to build a native thermodynamic engine in OpenModelica are explained.

3.1. Development of Compound Database. A compound database is a comprehensive database of physical and chemical properties of all compounds. It also includes parameters for calculating various temperature or pressure-dependent properties such as vapor pressure, enthalpy, viscosity, etc.

A description of ChemSep[16] database is required to understand the method used to port it into OpenModelica. ChemSep is an open source database, written in xml format. It has over 600 compounds with a comprehensive set of thermodynamic properties of each compound. It also has an extensive database of binary interaction parameters to implement algorithms like NRTL,[17] Peng–Robinson,[18] UNIQUAC,[20] and SRK.[23] Most of the thermodynamic properties are calculated by empirical equations that are functions of temperature or pressure. The ChemSep database includes parameters that are used in these equations. Therefore, ChemSep database is a comprehensive database that can be used to build a powerful and robust thermodynamic engine.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

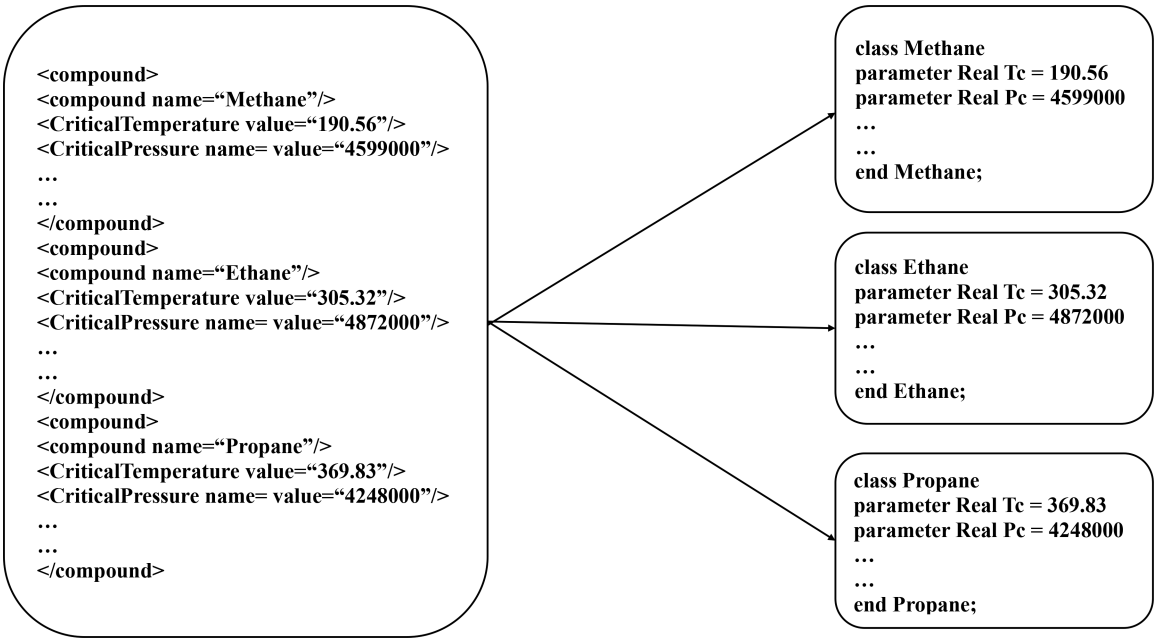


Figure 3. Porting ChemSep database in OpenModelica.

The steps to port ChemSep to OpenModelica are explained now. First, the xml data must be rewritten in a form compatible with 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 performed by developing a Python script that automates this process, thus making it fast and robust.

Table 2. Independent Thermodynamic Properties and OpenModelica Routines Used 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

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 (T_c) 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.

The efficiency of this method is demonstrated now with an example. An equimolar mixture of benzene and toluene was fed into a flash. It was assumed that the output liquid stream was at the same composition and temperature, as the holdup inside the flash separator. The heat supplied to the 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. Raoult's law was used for property prediction.

With the native port of thermodynamics explained in this section, OpenModelica required less than 10 s to solve this problem. In contrast, the Client-Server approach took 4 min to solve the system. The least efficient method of the Python-C API approach took 30 min to solve this problem. As a matter of fact, it is not at all fair to compare the speeds of the native port with other methods. This forms the motivation to build a solution based on native port of properties and algorithms, which is the topic of discussion in the next section.

4. DEVELOPMENT OF THERMODYNAMIC AND TRANSPORT CORRELATIONS IN OPENMODELICA

As discussed in the earlier section, the built-in thermodynamic engine is found to be the best, compared to other methods. Hence, this approach is adopted for further development of a thermodynamic package and transport correlations. For the sake of brevity, a detailed discussion on equations pertaining to thermodynamic models is not included and can be found in any standard thermodynamic documentation.[17−20] Suffice it to say that pertinent equations of thermodynamic models are directly coded in OpenModelica.

Four of the most popular phase equilibria models are implemented in this work: NRTL,[17] Peng−Robinson,[18] UNIFAC,[19] and UNIQUAC.[20] While Peng−Robinson is an equation of state (EoS) model, all of the rest are activity coefficient-based models. This section describes the different thermodynamic functions that are developed. Later, these functions are used to compute phase equilibrium data. The developed functions are simulated for typical cases and compared to arrive at appropriate inferences.

4.1. Development of Transport Correlations. Transport correlations are necessary to calculate thermophysical properties such as density, thermal conductivity, and viscosity. These properties find relevance in the design of chemical engineering equipment, such as distillation columns, heat exchangers, absorption columns, etc.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

Table 3. Dependent Transport Properties and OpenModelica Functions Used To Call Them

transport property	calling procedure
liquid density	LiqDen(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)

Table 4. Dependent Thermodynamic Properties and OpenModelica Functions Used To Call Them

thermodynamic property	calling procedure
vapor pressure	Pvap(Compound name, temp)
heat of vaporization	HOV(Compound name, temp)
ideal enthalpy	Hid(Compound name, temp)
heat capacity	Cp(Compound name, temp)

Transport properties are generally calculated through empirical equations that are dependent on parameters, whose values are provided by the compound database, and independent variables, such as temperature, pressure, and composition. These are passed as arguments and transport properties determined through OpenModelica functions, which return the calculated property. For example, the liquid density of methane at 300 K can be calculated by first instantiating the base Methane class (parameter Methane methane) and then calling LiqDen(methane.rho, 300). Here, LiqDen is a generic function that is used to calculate the liquid density of any compound at any given temperature. Similarly, all other transport properties can be calculated using the respective function as shown in Table 3.

4.2. Development of Thermodynamic Functions. Thermodynamic functions are necessary to calculate thermophysical properties, such as heat capacity, vapor pressure, and enthalpy. These properties are required in the algorithms that calculate activity coefficients, fugacity coefficients, and *k*-values, which are also implemented in the thermodynamic property package.

Thermodynamic properties are also calculated in a manner similar to the way transport properties are calculated, as explained in Section 4.1. 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). Here, Pvap is a generic function used to calculate the vapor pressure of any compound at any given temperature. Similarly, all other thermodynamic properties can be calculated using the respective function as shown in Table 4.

Also, functions are developed to store the binary interaction parameters (BIPs) for the above-mentioned activity coefficient models as well as the EOS model. These BIPs are imported in OpenModelica from the ChemSep database, where they are stored in a .dat file. To port the BIPs to OpenModelica, the dat file format from ChemSep database is converted to the .csv format, which is convenient for processing by Python. The .csv file is then converted to an OpenModelica function by a Python script, which converts the compound and the BIPs as an array. BIPs can be accessed as follows: ThermodynamicFunctions.BIP Model(Number of components, Compound name).

4.3. Methods of Comparison. As outlined earlier, the efficiency and accuracy of developed thermodynamic models in OpenModelica is tested by simulating the model for different

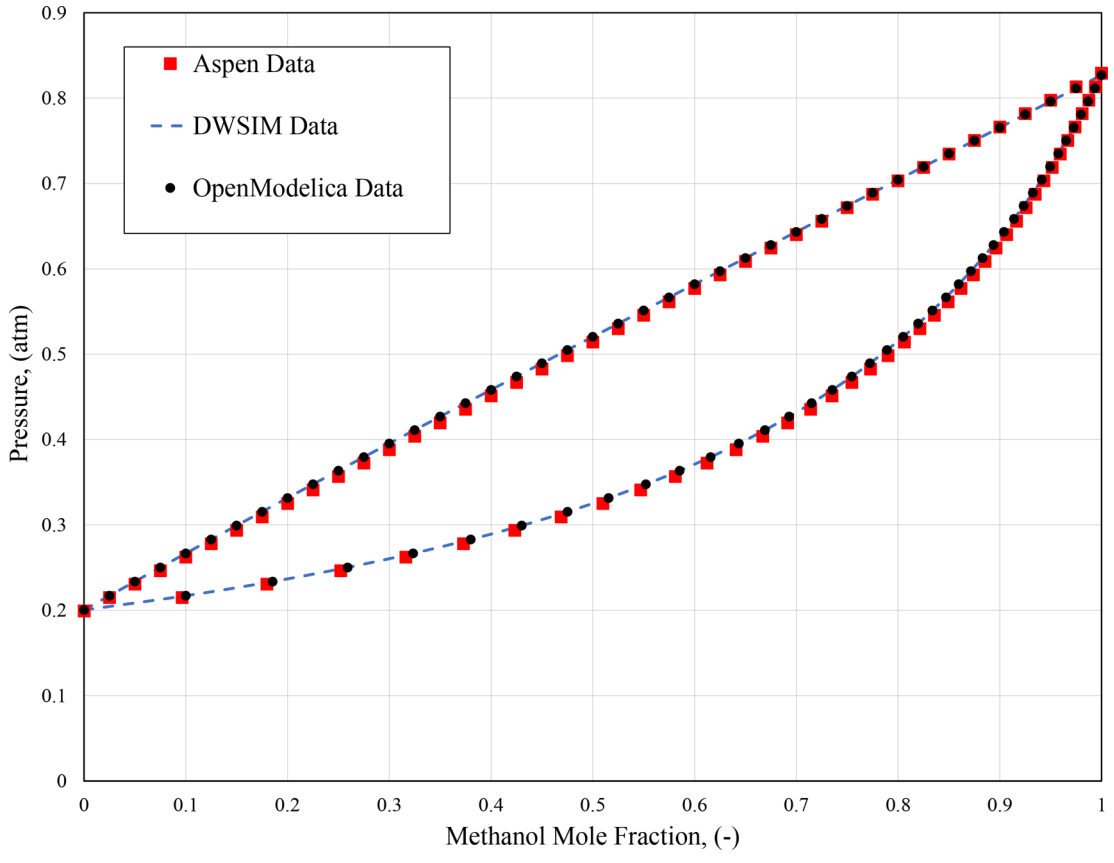


Figure 4. Comparison of the Pxy curve for the methanol/1-propanol system at 333 K using the NRTL model.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

Table 5. Problem Statement and Results of Design Problem Simulated in OpenModelica (Using the Native Port of NRTL), DWSIM, and Aspen Plus[a]

	Problem Statement—NRTL		
property	feed	vapor	liquid
molar flow rate (mol/s)	100		
temperature (K)	366	366	366
pressure (atm)			
component mole fraction			
methanol	0.25		
ethanol	0.25	0.26	
water	0.25		
acetic acid	0.25		
	Results		
output parameter	OpenModelica DWSIM	Aspen Plus	
pressure (atm)	0.9288	0.9277	
liquid molar flow rate (mol/s)	7.9256	6.8691	
component mole fraction in liquid phase			
methanol	0.0928	0.1085	
ethanol	0.1340	0.1144	
water	0.2506	0.2446	
acetic acid	0.5226	0.5325	
vapor molar flow rate (mol/s)	92.0744	93.1309	
component mole fraction in vapor phase			
methanol	0.2636	0.2604	
water	0.2500	0.2504	
acetic acid	0.2265	0.2292	

[a]OpenModelica and DWSIM give identical results. They differ slightly from Aspen Plus.

test conditions. These are described in the following sections. Note that the choice of thermodynamic model for a given system is not the focus of the present work. Hence, it is presumed that, for a given system, appropriate thermodynamic model will be used by the user. The work only illustrates the efficiency of model built in OpenModelica and not on the choice of appropriate model for a given system. For every thermodynamic model to be discussed in this section, we will compare the results from OpenModelica implementation with those of Aspen Plus, using two metrics: binary flash and steady-state flash.

With regard to the binary flash, the flash results of a binary system at different operating conditions are used to construct a binary phase envelope.[24] These envelopes are also generated using DWSIM and Aspen Plus, and the results compared. As the OpenModelica implementation is based on DWSIM (both property database and models), the results are identical. Differences between OpenModelica and Aspen Plus results are highlighted.

With regard to steady-state flash, a model for the steady-state flash column is built in OpenModelica and used with the thermodynamic library to simulate the model for a specified set of conditions. The flash column is also simulated using DWSIM.

The flash column model is simulated in design mode, wherein different types of output conditions (flow rate, compositions, temperature, and pressure) are specified, and the system is solved. OpenModelica being an equation-oriented simulator, solves such systems easily. In sequential modular process simulators, such design simulations are to be performed by trial and error.

Detailed problem statement along with the results comparison for different thermodynamic packages have been discussed in Sections 4.4—4.7.

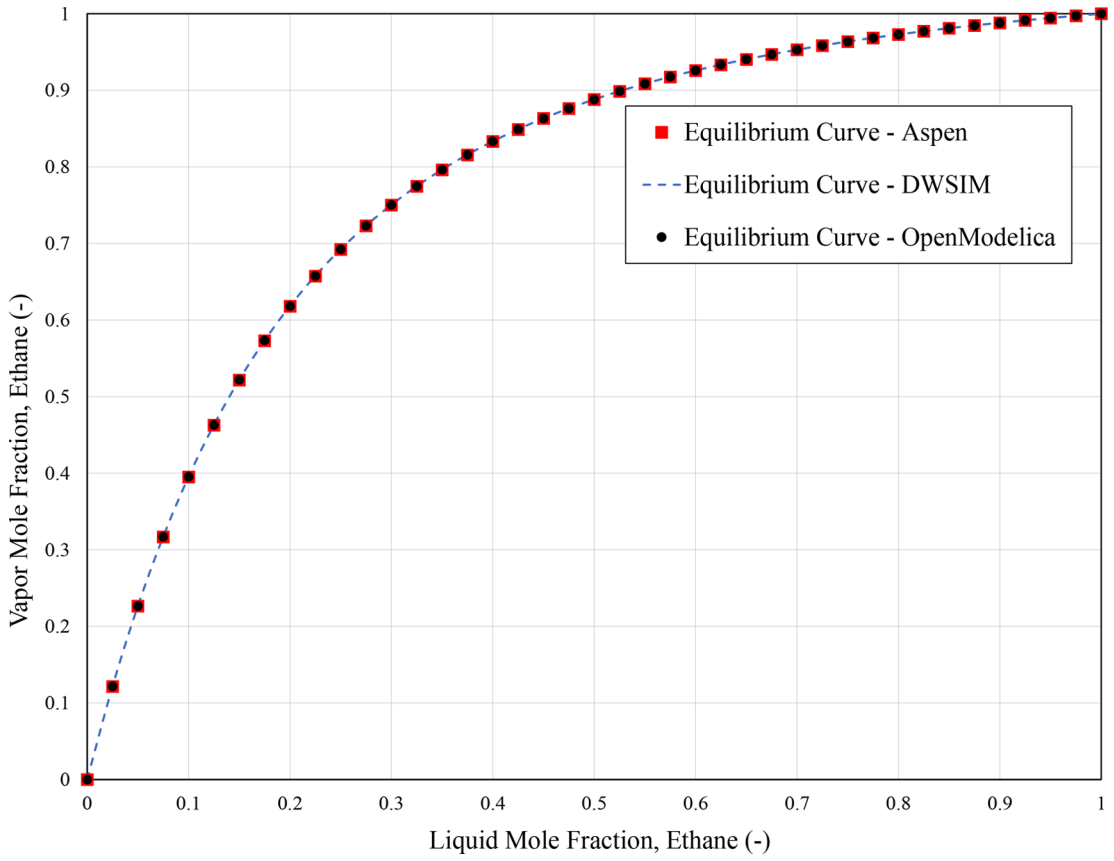


Figure 5. Comparison of the equilibrium curve for the ethane—propane system at 2 atm, using the Peng—Robinson model.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551—7560

4.4. Development of NRTL Model. The Non-Random Two-Liquid (NRTL) model is an activity coefficient model that correlates the activity coefficient of the compound with its mole fraction in the liquid phase. In this work, the NRTL model equations are coded in OpenModelica, and the results are compared with DWSIM and Aspen Plus, using the methods of comparison presented in Section 4.3.

For a binary component system consisting of methanol and 1-propanol, the bubble point curve and the dew point curve, at a constant temperature of 333 K, have been generated using OpenModelica and Aspen Plus, and the results plotted in Figure 4 for comparison. As calculation methods and data of OpenModelica and DWSIM are identical, the results are also identical. They also match the results of Aspen Plus very well.

The second method of comparison is now performed through a steady-state flash problem, specified at the top of Table 5. The four-component system consists of methanol, ethanol, water, and acetic acid. It is solved using NRTL. The flow rate and the composition of the feed is specified. Temperature is kept constant at 366 K. It is desired to calculate all other variables for a given vapor phase mole fraction of ethanol. In other words, all the blanks in this table are unspecified. At the bottom of the same table, the results are presented. OpenModelica and DWSIM give identical results. Although they compare reasonably well with the results of Aspen Plus, they are not exact.

The reason for this difference is now explained. Interaction parameters of the water-acetic acid binary pair is missing in DWSIM. Older versions of DWSIM used to take missing parameters as 0. The latest version of DWSIM (Version 5.6 Update 11 onwards) warns a user when any interaction parameter is missing. It is then possible to use some other prediction method, as explained in Section 4.6.

4.5. Development of the Peng−Robinson Model. The Peng−Robinson EoS is one of the most popular EOS models for hydrocarbon systems. In this work, the Peng−Robinson EoS model equations were coded in OpenModelica, and the results are compared with DWSIM and Aspen Plus using the methods of comparison presented in Section 4.3.

For a binary hydrocarbon system of ethane−propane, equilibrium curves at a constant pressure of 2 atm have been generated using OpenModelica and Aspen Plus, and the results are plotted in Figure 5. Since the calculation methods and data of OpenModelica and DWSIM are identical, the results are also identical. They also match the results of Aspen Plus.

The second method of comparison is now performed through a steady-state flash problem, specified at the top of Table 6. The five-component system consists of ethane, propane, cyclopentane, cyclohexane, and *n*-hexane. It is solved the using Peng−Robinson EoS. The flow rate and the composition of the feed is specified. Temperature is kept constant at 280 K. It is desired to calculate all other variables for a given liquid phase mole fraction of *n*-hexane. In other words, all the blanks in this table are unspecified. At the bottom of the same table, the results are presented. OpenModelica and DWSIM give identical results. They also match the results of Aspen Plus.

4.6. Development of UNIFAC Model. UNIFAC is a semi-empirical activity coefficient model used to predict nonelectrolyte activity in nonideal mixtures. In this work, the UNIFAC model equations are coded in OpenModelica and results are compared with DWSIM and Aspen Plus using the modes of comparison as discussed in Section 4.3.

For a binary system of acetic acid−methyl ethyl ketone (MEK), the activity coefficient of acetic acid, at constant temperature of 375 K, has been generated using OpenModelica and Aspen Plus, and all of the results plotted in Figure 6 for comparison. As calculation methods and data of OpenModelica and DWSIM are identical, the results are also identical. They also match the results of Aspen Plus very well.

Table 6. Problem Statement and Results of Design Problem Simulated in OpenModelica (Using the Native Port of Peng−Robinson), DWSIM, and Aspen Plus[a]

	Problem Statement—Peng−Robinson		
property	feed	vapor	liquid
cyclohexane	0.1		
molar flow rate (mol/s)	100		
temperature (K)	280	280	280
pressure (atm)			
component mole fraction			
ethane	0.3		
propane	0.3		
cyclopentane	0.2		
n-hexane	0.1		0.2
	Results		
output parameter	OpenModelica DWSIM		Aspen Plus
pressure (atm)	2.281		2.304
liquid molar flow rate (mol/s)	47.6391		47.6757
component mole fraction in liquid phase			
propane	0.1660		0.1674
ethane	0.0523		0.0503
cyclopentane	0.3784		0.3788
cyclohexane	0.2034		0.2035
vapor molar flow rate (mol/s)	52.3609		52.3243
component mole fraction in vapor phase			
propane	0.4219		0.4208
ethane	0.5254		0.5275
cyclopentane	0.0377		0.0371
cyclohexane	0.0060		0.0057
n-hexane	0.0090		0.0089

[a]OpenModelica and DWSIM given identical results. They differ slightly from Aspen Plus.

The second method of comparison is now performed through a steady-state flash problem, specified at the top of Table 7. The system consists of MEK acetone and water. It is solved using UNIFAC. 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 a given vapor phase flow rate. In other words, all the blanks in this table are unspecified. At the bottom of the same table, the results are presented. OpenModelica and DWSIM give identical results. They closely match the results of Aspen Plus as well. UNIFAC is an alternative to NRTL for systems characterized by activity coefficients. In Section 4.4, it was shown that the predictions of DWSIM/OpenModelica differed from that of Aspen Plus for the methanol, ethanol, water and acetic acid system. Such a problem doesn't arise when UNIFAC is used: as UNIFAC is a completely predictive estimation method, it doesn't require any model parameters, and the results are identical to that of Aspen Plus. Nevertheless, use of NRTL results in faster calculations in general, and hence should be used whenever interaction parameters are available.

4.7. Development of UNIQUAC Model. UNIQUAC is an activity coefficient model used to describe the phase equilibrium. In this work, the UNIQUAC model equations are coded in

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

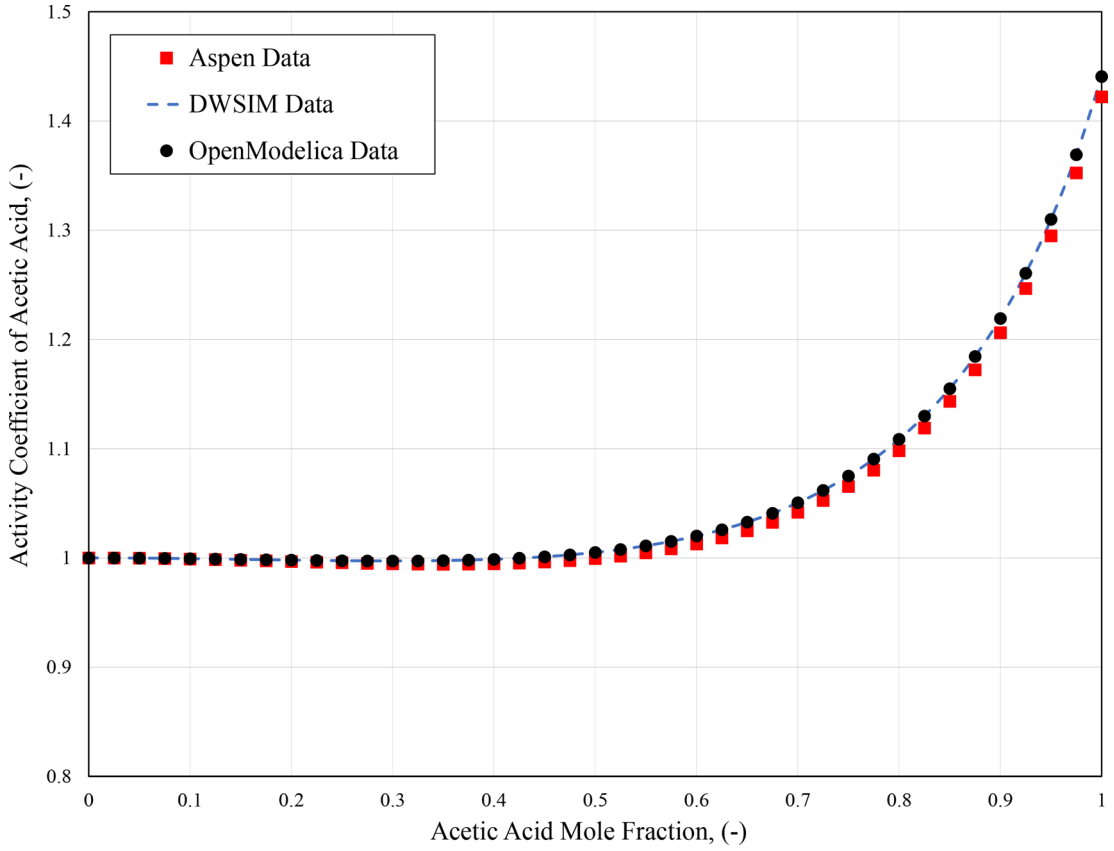


Figure 6. Activity coefficient of acetic acid−MEK binary component system at 375 K using a UNIFAC model.

Table 7. Problem Statement and Results of a Design Problem Simulated in OpenModelica (Using the Native Port of UNIFAC), DWSIM, and Aspen Plus[a]

	Problem Statement−UNIFAC		
property	feed	vapor	liquid
molar flow rate (mol/s)	100	71	
temperature (K)			
pressure (atm)	1	1	1
component mole fraction			
methyl ethyl ketone	0.2		
acetone	0.3		
water	0.5		
	Results		
output parameter	OpenModelica DWSIM	Aspen Plus	
temperature (K)	344.97	345.37	
liquid molar flow rate (mol/s)	29	29	
component mole fraction in liquid phase			
methyl ethyl ketone	0.0231	0.0277	
acetone	0.0357	0.0421	
water	0.9412	0.9302	
component mole fraction in vapor phase			
methyl ethyl ketone	0.2722	0.2704	
acetone	0.4080	0.4053	
water	0.3198	0.3243	

[a]They differ only slightly from Aspen Plus.

OpenModelica, and results are compared with Aspen Plus using the two methods of comparison presented in Section 4.3.

For a binary azeotropic system of ethanol−water, the bubble point curve and the dew point curve, at constant pressure of 1 atm, have been generated using OpenModelica and Aspen Plus, and all the results plotted in Figure 7 for comparison. Since the calculation methods and data of OpenModelica are identical, the results are also identical. They also match the results of Aspen Plus reasonably well.

The second method of comparison is now performed through a steady-state flash problem, specified at the top of Table 8. The system consists of ethanol, 1-propanol and water. It is solved using UNIQUAC. 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 a given liquid phase flow rate. In other words, all the blanks in this Table are unspecified. At the bottom of the same Table, the results are presented. OpenModelica and DWSIM give identical results. They also match the results of Aspen Plus.

Finally, to address the question of how the results vary if a different thermodynamic correlation is used, the example of this section is solved again using UNIFAC, and the results are presented in Table 9. One can see that these results are not much different from those of Table 8.

5. CONCLUSIONS AND FUTURE WORK

This work summarizes an effort undertaken to make available general purpose modeling and simulation environment OpenModelica for chemical engineers. The native port of thermodynamic algorithms and property database of DWSIM and ChemSep into OpenModelica is the first step in this direction.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

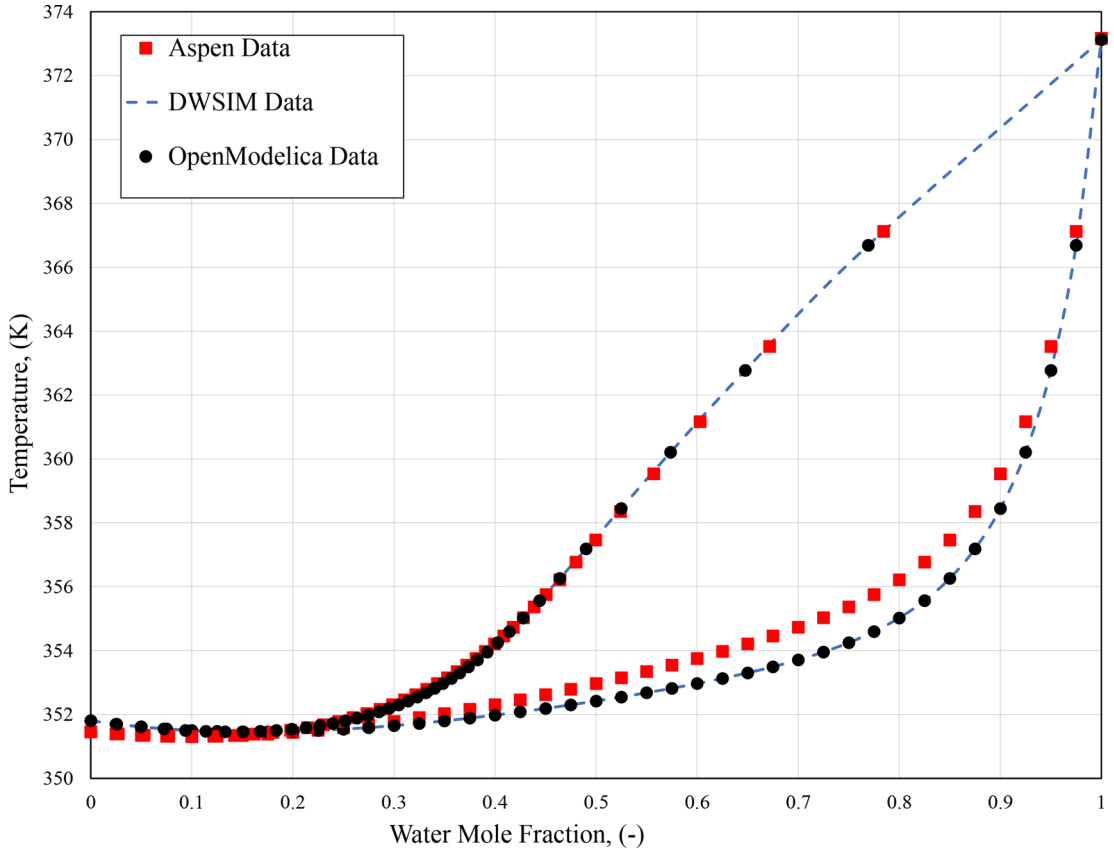


Figure 7. Comparison of Txy plot for ethanol−water binary component system at 1 atm using the UNIQUAC model.

Table 8. Problem Statement and Results of a Design Problem Simulated in OpenModelica (Using the Native Port of UNIQUAC), DWSIM and Aspen Plus[a]

	Problem Statement−UNIQUAC		
property	feed	vapor	liquid
flow rate (mol/s)	100		26
temperature (K)			
pressure (atm)	1	1	1
component mole fraction			
ethanol	0.3		
1-propanol	0.4		
water	0.3		

	Results	
output parameter	OpenModelica DWSIM	Aspen Plus
temperature (K)	358.899	358.878
component mole fraction in liquid phase		
ethanol	0.2444	0.2346
1-propanol	0.5443	0.5263
water	0.2113	0.2391
vapor molar flow rate (mol/s)	74	74
component mole fraction in vapor phase		
ethanol	0.3195	0.3230
1-propanol	0.3493	0.3556
water	0.3312	0.3214

[a]OpenModelica and DWSIM given identical results. They differ greatly from Aspen Plus.

This method of making thermodynamics is arrived at after attempting two other methods (Python-C API, socket programming), which have been found to be less efficient.

Table 9. Problem Reported in Table 8 Solved Again with the Native Port of UNIFAC in OpenModelica[a]

	Results	
output parameter	OpenModelica DWSIM	Aspen Plus
temperature (K)	359.246	359.086
component mole fraction in liquid phase		
ethanol	0.2377	0.2372
1-propanol	0.5173	0.5164
water	0.2450	0.2464
vapor molar flow rate (mol/s)	74	74
component mole fraction in vapor phase		
ethanol	0.3219	0.3221
1-propanol	0.3588	0.3591
water	0.3193	0.3188

[a]OpenModelica and DWSIM give identical results. They differ only slightly from Aspen Plus.

The capability of the native port of thermodynamics is demonstrated through several examples. These results are also compared with those of DWSIM and Aspen Plus. As the data and the methods used in OpenModelica are the same as in DWSIM, the results are identical. In all four cases, these results are in good agreement with Aspen Plus. Although the applicability of different correlations for a specific problem is demonstrated, the question of appropriateness of a correlation for a given chemical mixture is not addressed.

This work would not have been possible had DWSIM and ChemSep not been open source. Details of thermodynamic calculations in DWSIM are available for everyone, coding which, in OpenModelica and matching, the results has raised the

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560

competence of some authors of this article. It also helped identify a few minor errors in DWSIM and correct them. Releasing the fruits of this work as open source may help several learners in a similar way. This, in turn, will help improve the code/example base available for these open source systems. All the code developed in this work is available to the public.[25]

The authors hope to add more thermodynamic algorithms, such as SRK, Lee-kesler-Plocker, and Chao-Seeder, in the near future. It is also proposed to start the task of augmenting the property database through crowdsourcing, using DWSIM and migrating it to OpenModelica. Most users will find it easier to access DWSIM, as the steady-state simulator has a less steep learning curve, compared to OpenModelica.

Work is underway to enhance the process simulation capabilities of OpenModelica. Porting unit operation libraries of DWSIM to OpenModelica is in an advanced stage. Including custom models through Scilab and Python in DWSIM, and migrating them to OpenModelica are proposed to be undertaken in the near future. A large number of students have already been trained in Scilab and Python through Textbook Companion projects.[26,27] This approach should allow many users at various levels of competence to contribute from all over the world and help the system grow.

OpenModelica is a useful platform to undertake research and development activities that are in the general area of equation-oriented simulation approach and dynamic simulation. Some of the activities currently studied are initialization of variables and introduction of dynamics in steady-state models.

DWSIM and OpenModelica, along with property databases, such as ChemSep, have the potential to become a useful open-source platform through which all chemical engineers can collaborate for education and research.

AUTHOR INFORMATION

Corresponding Author

*Tel.: +91 9869 32 6979. E-mail: kannan@iitb.ac.in.

ORCID

Kannan M. Moudgalya: 0000-0003-0847-3825

P. R. Naren: 0000-0002-7589-0728

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors would like to thank the Ministry of Human Resource Development, Government of India, for funding this work through the National Mission on Education through ICT. The authors also thank all creators of open source software in general, and the creators of ChemSep database in particular.

REFERENCES

(1) PWC. Will robots really steal our jobs? An international analysis of the potential long term impact of automation. 2017; available via the Internet at: https://www.pwc.com/hu/hu/kiadvanyok/assets/pdf/impact_of_automation_on_jobs.pdf, last accessed on Oct. 1, 2018.

(2) FOSSEE. DWSIM Flowsheeting Project; available via the Internet at: https://dwsim.fossee.in/flowsheeting-project, last accessed on Oct. 1, 2018.

(3) Medeiros, D. DWSIM technical document; available via the Internet at: http://dwsim.inforside.com.br/, 2018; last accessed on Oct. 1, 2018.

(4) Aspen Technology. Available via the Internet at: https://www.aspentech.com/en/products/pages/aspen-plus-dynamics, last accessed on Oct. 1, 2018.

(5) gPROMS ModelBuilder. Available via the Internet at: https://www.psenterprise.com/products/gproms/modelbuilder, last accessed on Oct. 1, 2018.

(6) Moudgalya, K. M. IT Skills Training through Spoken Tutorials for Education and Employment: Reaching the Unreached. *CEC J. Digital Educ.* **2017**, *1*, 19−62. http://spoken-tutorial.org/media/CEC.pdf.

(7) FOSSEE. DWSIM Spoken Tutorials; available via the Internet at: http://spoken-tutorial.org/tutorial-search/?search_foss=DWSIM&search_language=English, Last accessed on Oct. 1, 2018.

(8) Fritzson, P. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*; Wiley IEEE Press: New York, 2014.

(9) Fritzson, P.; Engelson, V. Modelica—A unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, 1998; pp 67−90.

(10) OpenModelicaTeam. OpenModelica Textbook Companion; available via the Internet at: https://om.fossee.in/textbook-companion/completed-books; last accessed on Oct. 1, 2018.

(11) Moudgalya, K. M.; Nemmaru, B.; Datta, K.; Nayak, P.; Fritzson, P.; Pop, A. Large Scale Training through Spoken Tutorials to Promote OpenModelica. In *Proceedings of the 12th International Modelica Conference*, Prague, 2017; pp 275−284.

(12) Piela, P. C.; Epperly, T.; Westerberg, K.; Westerberg, A. W. ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. *Comput. Chem. Eng.* **1991**, *15*, 53−72.

(13) Sandrock, C.; de Vaal, P. L. Dynamic simulation of Chemical Engineering systems using OpenModelica and CAPE-OPEN. In *19th European Symposium on Computer Aided Chemical Engineering, ESCAPE19*, 2009; pp 859−864.

(14) Windahl, J.; Prölss, K.; Bosmans, M.; Tummescheit, H.; van Es, E.; Sewgobind, A. MultiComponentMultiPhase—A framework for thermodynamic properties in Modelica. In *Proceedings of the 11th International Modelica Conference*, Versailles, 2015; pp 653−662.

(15) Jain, R.; Moudgalya, K. M.; Fritzson, P.; Pop, A. Development of a thermodynamic engine in OpenModelica. In *Proceedings of the 12th International Modelica Conference*, Prague, 2017; pp 89−99.

(16) Kooijman, H. A.; Taylor, R. *The ChemSep book*; Books on Demand: Norderstedt, Germany, 2000; available via the Internet at: http://www.chemsep.com/downloads/docs/book2.pdf, last accessed on Oct. 1, 2018.

(17) Renon, H.; Prausnitz, J. M. Local compositions in thermodynamic excess functions for liquid mixtures. *AIChE J.* **1968**, *14*, 135−144.

(18) Peng, D.-Y.; Robinson, D. B. A New Two-Constant Equation of State. *Ind. Eng. Chem. Fundam.* **1976**, *15*, 59−64.

(19) Fredenslund, A.; Jones, R. L.; Prausnitz, J. M. Group-contribution estimation of activity coefficients in nonideal liquid mixtures. *AIChE J.* **1975**, *21*, 1086−1099.

(20) Abrams, D. S.; Prausnitz, J. M. Statistical thermodynamics of liquid mixtures: a new expression for the excess Gibbs energy of partly or completely miscible systems. *AIChE J.* **1975**, *21*, 116−128.

(21) AspenTech. Aspen Plus; available via the Internet at: https://www.aspentech.com/en/products/engineering/aspen-plus, last accessed on Oct. 1, 2018.

(22) Rhodes, B.; Goerzen, J. Foundations of Python Network Programming, 2010, ISBN 9781430230045; available via the Internet at: https://link.springer.com/book/10.1007%2F978-1-4302-3004-5.

(23) Soave, G. Equilibrium constants from a modified Redlich−Kwong equation of state. *Chem. Eng. Sci.* **1972**, *27*, 1197−1203.

(24) Smith, J. M.; Van Ness, H. C.; Abbott, M. M. *Introduction to Chemical Engineering Thermodynamics*, Vol. *27*; McGraw−Hill Education, 2005; pp 338−377.

(25) FOSSEE. Git Hub link to OpenModelica Thermodynamics; available via the Internet at: https://github.com/FOSSEE/OMChemSim, last accessed on Oct. 1, 2018.

(26) Braatz, R. D. Scilab Textbook Companions. *IEEE Control Syst. Mag.* **2014**, *76*.

(27) PythonTeam. Python Textbook Companion; available via the Internet at: http://tbc-python.fossee.in, last accessed on Oct. 1, 2018.

DOI: 10.1021/acs.iecr.8b05147
Ind. Eng. Chem. Res. 2019, 58, 7551−7560