I&EC research
Industrial & Engineering Chemistry Research

Article

Cite This: Ind. Eng. Chem. Res. 2019, 58, 11164−11174

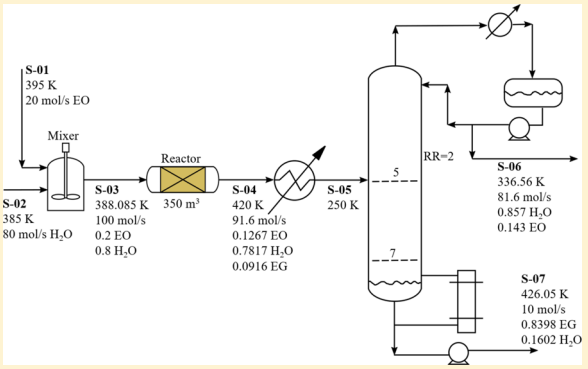
pubs.acs.org/IECR

Chemical Process Simulation Using OpenModelica

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

[†]Department of Chemical Engineering, IIT Bombay, Mumbai 400 076, India
[‡]School of Chemical & Biotechnology, SASTRA Deemed University, Thanjavur 613 401, India
[¶]Linkoping University, 581 83 Linköping, Sweden
[§]Independent Researcher, Manaus 69053-020, Brazil

ABSTRACT: The equation-oriented general-purpose simulator OpenModelica provides a convenient, extendible modeling environment, with capabilities such as an easy switch from steady-state to dynamic simulations. This work reports the creation of a library of steady-state models of unit operations using OpenModelica. The use of this library is demonstrated through a few representative flowsheets, and the results are compared with the steady-state simulators Aspen Plus and DWSIM. Being open-source and supported by a large community of developers across the world, OpenModelica provides a convenient platform to train a large number of chemical engineers to increase collaborative research and employment.



1. INTRODUCTION

Material and energy-balance computation of process plants helps in auditing of resource utilization. This in turn results not only in cost and energy savings but also helps in decreasing the load on the environment. If only the thousands of small- and medium-scale chemical plants carry out just material and energy-balance calculations, there can be substantial savings and also less damage to the environment. This is especially true in developing countries, which also suffer from a lack of quality work force. Making available affordable simulators and training a large number of engineers capable of using the simulators is one way to address this issue.

Modern technology is one way to address the issue raised above. Millennials are comfortable with the social media. They are not scared of working with like-minded people at distributed locations. New technologies have also shown that it is possible for experts to share their knowledge with thousands who are hungry for it. Projects like Wikipedia have shown that it is possible to produce useful content through collaborative crowdsourcing. The fact that this can work in the field of chemical engineering is already demonstrated by the collaborative work on the open-source steady-state process simulator DWSIM,[1] which has helped produce more than 100 flowsheets.[2]

In the current work, an effort to extend the flowsheeting capability of the general-purpose object-oriented simulation environment OpenModelica[3] is described. OpenModelica is based on the Modelica language,[4,5] which enforces equation-oriented simulation strategy.[6] It incorporates many of the recommendations of Piela et al.[7] In particular, it helps maintain models and solvers. OpenModelica is also suitable for studying dynamics, required in the analysis of continuous processes, and also batch processes that are prevalent in thousands of small chemical plants. As it is an open-source software, OpenModelica can be used by large numbers of students and small- and medium-scale chemical companies.

This paper is organized as follows. Section 2 explains some features of OpenModelica that make it a suitable candidate for process simulation. A brief outline of models of unit operations created in OpenModelica is presented in Section 3. A few typical flowsheets are also solved in that section, with a comparison of results with Aspen Plus. Section 4 is devoted to conclusions and future work.

2. OPENMODELICA FOR PROCESS SIMULATION

As a first step in building research communities in the area of simulation, the authors undertook an exercise of crowdsourcing steady-state flowsheets. This has resulted in more than 100 DWSIM flowsheets and a conference.[8] Each of these flowsheets solves a chemical process using DWSIM and compares the results with the literature or a commercial process simulator. All the flowsheets are released under the Creative Commons Attribution Share Alike (CC-BY-SA) license.[2]

The focus of the work in this article is to make available a versatile process-simulation environment that is capable of carrying out both steady-state simulations and dynamic simulations

Special Issue: Sirish Shah Festschrift

Received: January 7, 2019
Revised: May 15, 2019
Accepted: May 20, 2019
Published: May 22, 2019

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

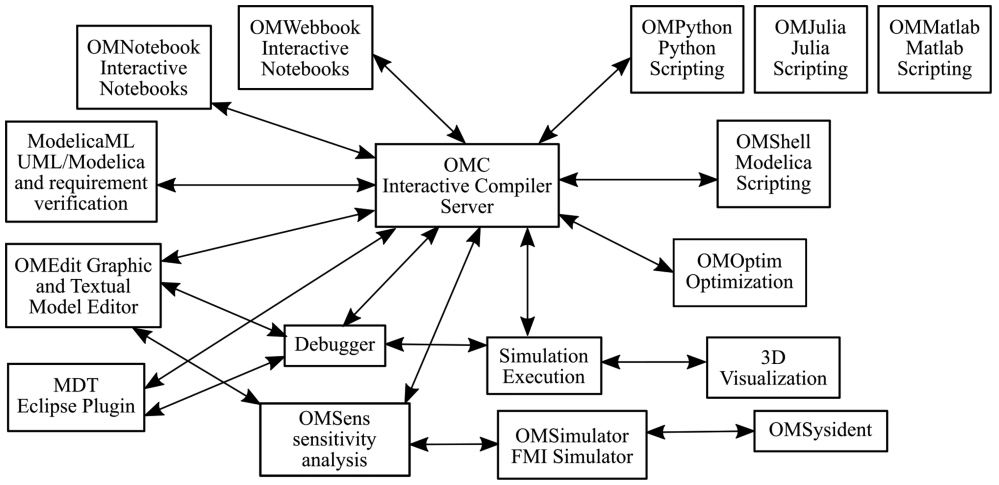


Figure 1. Schematic of the OpenModelica Modeling and Simulation Environment.

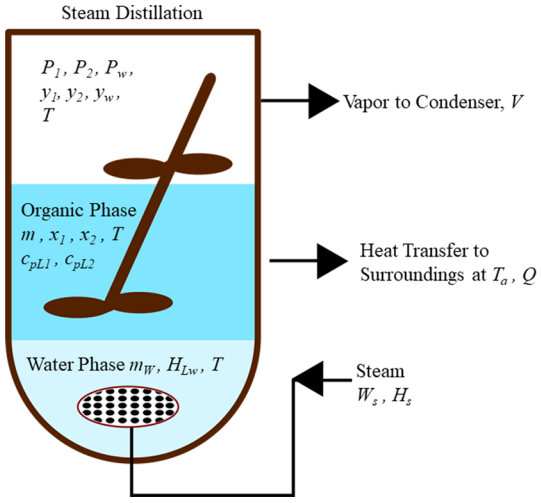


Figure 2. Schematic of the steam-distillation apparatus.

```
model SemiBatch_Distill
extends Thermo_functions;
  parameter Real UA(unit="J/s-K") = 1.05;
  parameter Real Ta(unit = "K") = 298.15;
  parameter Real T0(unit = "K")= 273.15;
  parameter Real Ts(unit = "K") = 372.35;
  parameter Real Ws(unit="kmol/s") = 3.85e-5;
  parameter Real P(unit="Pa") = 9.839E4;
  Real Q(unit="J/s"),T(start=298.15,unit="K"), Tc(unit="degree C");
  Real Mw(start=0,unit="kmol"), M(start=0.015,unit="kmol");
  Real y1, y2, yw, x1(start=0.725), x2;
  Real fT, V(unit="kmol/s"), E, t1(unit="min");

  equation
  Tc = T-273.15;
  t1 = time/60;
  der(Mw) = Ws-V*yw;
  Q = UA*(T-Ta);
  y1 = P1*x1/P;
  y2 = P2*x2/P;
  yw = Pw/P;
  x2 = 1-x1;
  der(M) = -V*(y1+y2);
  if fT>=0 then
     der(T) = (Ws*(Hs-Hlw)-Q)/(Mw*CpLw+M*(x1*CpL1+x2*CpL2));
     fT = 1-(y1+y2+yw);
     V = 0;
     E = 0;
     der(x1) = 0;
  else
     der(T) = 1000*E;
     fT = -1;
     V = (Ws*(Hs-Hlw)+Q)/(Hv-(Hlw*yw+(y1*Hl1+y2*Hl2)));
     E = 1-(y1+y2+yw);
     der(x1) = (((-V * y1) - x1 * (-V * (y1 + y2))) / M);
  end if;
end SemiBatch_Distill;
```

Figure 3. Model of the semibatch reactor.

for the process industry. It should be open-source so as to benefit large numbers of researchers and small-scale manufacturing units. OpenModelica[3] is a candidate simulation environment for the above task. If successful, an attempt will be made to migrate at least some of the above-mentioned DWSIM flowsheets to OpenModelica and then explore introducing dynamic models in them.

This section summarizes the capabilities of OpenModelica from the point of view of relevance to process simulation.

2.1. Motivation To Use Modelica and OpenModelica. As mentioned earlier, the main focus of this work is to create a chemical engineering library in Modelica and to demonstrate its use by creating reasonable flowsheets. One important reason for this attempt is that Modelica is suitable for both steady-state and dynamic simulations.

The other reason to pursue Modelica is that it enables the separation of concerns, as propounded by Piela et al.,[7] through a declarative modeling framework, combined with the equation oriented solution technique.[6] Here, separation refers to keeping modeling and solutions separate from each other. This approach allows modeling teams to articulate their models well, without worrying about the solution technique. At the solution stage, one can use the appropriate solvers, without worrying about the underlying modeling assumptions. This separation of concerns makes maintenance of models and solvers more tractable compared with in the systems wherein modeling and solutions are intertwined.

The Modelica modeling language is a nonproprietary, object-oriented, equation-based language for modeling physical systems consisting of components from different disciplines.[4,5] The Modelica Association ensures that the Modelica language is maintained and constantly improved. The Modelica standard library contains 1600 model components and 1350 functions from many domains. The Modelica language has been used in industry since 2000. There have been more than 10 biennial Modelica Conferences, attended by a large number of users of the Modelica language. There are about 10 commercial implementations of the Modelica language, which shows that the Modelica approach is popular in industry.

OpenModelica is an open-source implementation of the Modelica language[3] by a consortium of more than 50 members. It has a good set of solvers for different kinds of mathematical systems: algebraic, differential, and differential−algebraic. It comes with an OMEdit graphical interface, interactive DrModelica course material, and a cloud version (namely, OMWebbook). There are

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

```
model Thermo_functions

Real P1(unit="Pa"),P2(unit="Pa"),Pw(unit="Pa");
Real CpL1(unit="J/kmolK"),CpL2(unit="J/kmolK"),CpLw(unit="J/kmolK");
Real Hs(unit="J/kmol"), Hl1(unit="J/kmol"), Hl2(unit="J/kmol"), Hlw(unit="J/kmol")
    ;
Real Hv1(unit="J/kmol"), Hv2(unit="J/kmol"), Hvw(unit="J/kmol"), Hv(unit="J/kmol")
    , Hl(unit="J/kmol");

equation
  P1 = exp(96.084-7900.2/T-11.003*log(T)+7.1802E-6*T^2);
  P2 = exp(112.73-9749.6/T-13.245*log(T)+7.1266E-6*T^2);
  Pw = exp(73.649-7258.2/T-7.3037*log(T)+4.1653E-6*T^2);
  CpL1 = 0-186.63*T+0.95981*T^2+224830;
  CpL2 = 0-197.91*T+1.0737*T^2+278620;
  CpLw = 276370-2090.1*T+8.125*T^2-0.014116*T^3+9.3701E-6*T^4;
  Hs = 33363*Ts+26790*2610.5*(1/tanh(2610.5/Ts)^2)+8896*1169*tanh(1169/Ts)-4.471E7
      ;
  Hl1 = 224830*(T-T0)-186.63*(T^2-T0^2)/2+0.95891*(T^3-T0^3)/3;
  Hl2 = 278620*(T-T0)-197.91*(T^2-T0^2)/2+1.0737*(T^3-T0^3)/3;
  Hlw = 276370*(T-T0)-2090.1*(T^2-T0^2)/2+8.125*(T^3-T0^3)/3-0.014114*(T^4-T0^4)
      /4+9.3701E-6*(T^5-T0^5)/5;
  Hv1 = 135540*T+443100*1635.6*(1/(tanh(1635.6/T)))-305400*746.4*(tanh(746.4/T))
      -4.928E8;
  Hv2 = 167200*T+535300*1614.1*(1/(tanh(1614.1/T)))-378200*742*(tanh(742/T))-5.791
      E8;
  Hvw = 33363*T+26790*2610.5*(1/(tanh(2610.5/T))^2)+8896*1169*(tanh(1169/T))-4.471
      E7;
  Hv = yw+Hvw+y1*Hv1+y2*Hv2;
  Hl = x1*Hl1+x2*Hl2;
end Thermo_functions;
```

Figure 4. Property correlations for the semibatch reactor, as given in Shacham et al.[14]

also many self-teaching Spoken Tutorials and workshop campaigns to teach OpenModelica.[9,10]

OpenModelica comes with a very useful debugger. It can provide model-level debugging, indicating at which model equation the problem has occurred, and it can provide the incidence matrix of the system that is being solved. At compile time, it can point out whether the number of variables is correctly, over, or under specified.

OpenModelica Compiler (OMC) translates Modelica to C code, with a symbol table having definitions of variables, functions, and classes. These definitions can be predefined, user-defined, or taken from libraries. A Modelica interpreter for interactive usage and constant expression evaluation is also a part of the OMC. The OMC produces executable code incorporating required numerical solvers. What is shipped as OpenModelica is presented schematically in Figure 1.[11]

OpenModelica comes with 75 libraries in diverse fields, such as hydraulics, power-system simulation, motorcycle dynamics, servomechanisms, and thermal power, with about 1000 models. There are also efforts to make it suitable for handling large systems upward of 750 000 equations.[12]The objective of the current work is to include models of unit operations in OpenModelica and make it useful for process simulation. Property data and thermodynamic correlations have already been ported to OpenModelica by the authors.[13]

Some features of OpenModelica that are useful for carrying out simulations will be discussed next.

2.2. Semibatch Steam Distillation of a Binary Organic Mixture. The capability of OpenModelica to simulate the semibatch steam distillation of a binary mixture consisting of *n*-octane (compound 1) and *n*-decane (compound 2) is demonstrated in this section.[14]

This semibatch system has two phases of operation, namely, heating and distillation phases. In the heating phase, the binary mixture is taken into a vessel, and steam is bubbled into it continuously until the boiling point is reached (see Figure 2).

The system is modeled by the following equations, wherein subscripts 1 and 2 refer to the two organic compounds, w refers to water, s refers to steam, and a refers to ambient. We begin with the mass balance:

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

The energy-balance equation is given by

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

$$Q = UA(T - T_a)$$

Equilibrium relations are given by Raoult's law:

$$y_1 = \frac{x_1 P_1}{P}$$

$$y_2 = \frac{x_2 P_2}{P}$$

For an explanation of variables, the reader is referred to Shacham et al.[14]

The heating phase continues until the sum of vapor mole fractions becomes 1 (i.e., $f(T) = 1 - (y_1 + y_2 + y_w) = 0$). After this point, the distillation phase starts. Let V be the flow rate of vapor that includes organic compounds and water. The model equations are given next:

$$\frac{dm_w}{dt} = W_s - Vy_w$$

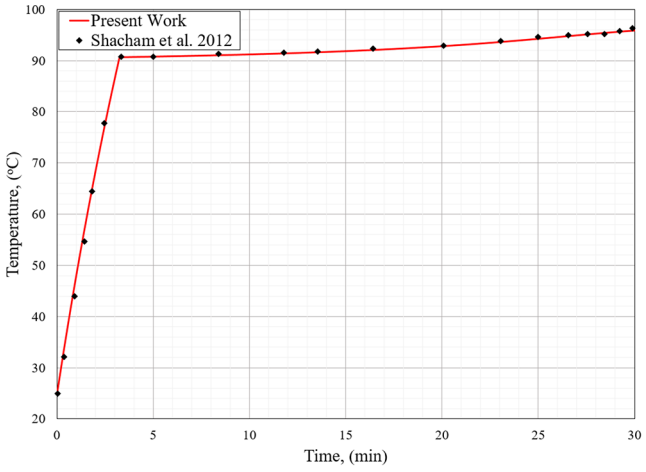
$$\frac{d(mx_1)}{dt} = -Vy_1$$

$$\frac{d(mx_2)}{dt} = -Vy_2$$

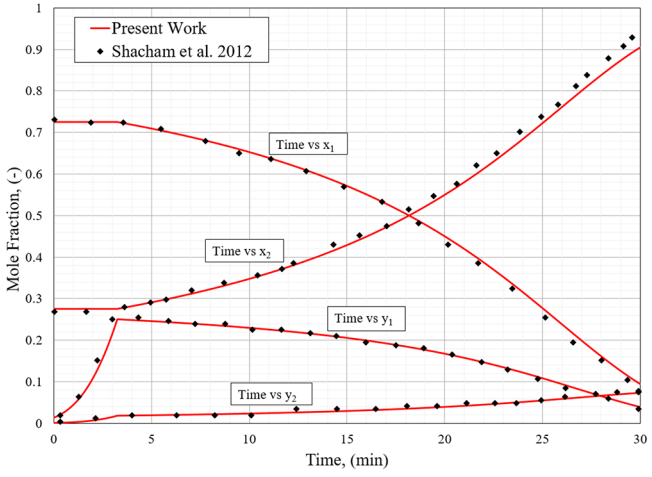
From the energy balance, the vapor flow rate is determined thus:

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

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174



(a) Comparison of temperature profiles



(b) Comparison of composition profiles[14]

Figure 5. (a) Temperature and (b) composition change during semibatch steam distillation: this work vs that of Shacham at al.[14]

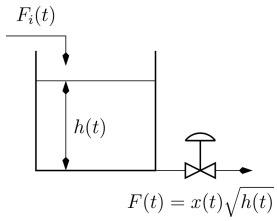


Figure 6. Simple tank.

The reader is referred to Shacham et al.[14] for a more detailed explanation.

An implementation of this system in OpenModelica is given in Figure 3. Required property values as suggested by Shacham et al.[14] are calculated in the code given in Figure 4. Note that both of these are presented as models. The former *extends* the latter. This results in equations given in both getting appended and being solved simultaneously, thereby enabling a true equation-oriented-solution approach. It is a practice in OpenModelica to enclose the models discussed here within a package construct, as shown below, with the order in which the two models appear being immaterial:

Once a package is invoked in OpenModelica, all models present within it get instantiated. When the model in Figure 3 is

```
package SemiBatch

  model Thermo_functions
  // code given in Fig. 4
  end Thermo_functions;

  model SemiBatch_Distill
  // code given in Fig. 3
  end SemiBatch_Distill;

end SemiBatch;
```

invoked with a simulation time of 30 min, the profiles presented below are obtained.

The code given in Figure 3 gives the values of various variables used in this specific simulation. These are identical to the ones used by Shacham et al.[14] When the simulation starts, `fT = 0`, and hence the heating-phase model that starts in line 25 of Figure 3 gets invoked. This variable subsequently gets redefined in line 26. When distillation starts, this variable becomes 0, and the distillation phase begins. In this declarative framework, it is easy to see the correspondence between the code and the model equations presented earlier.

```
1  model Tank_Model
2
3  parameter Real Fi(unit="m3/s") = 16 "Input flowrate to the tank";
4  parameter Real A(unit="m2") = 3 "Area of the tank";
5  parameter Real x(unit="(m^2.5)/s") = 8 "Constant";
6  parameter Real hinit(unit="m") = 3 "Initial level in the tank";
7  Real h(unit="m") "Tank level";
8  Real F(unit="m3/s") "Output flowrate from the tank";
9  parameter Boolean dynamic = true "true(1) indicates dynamic state";
10
11 initial equation
12   if dynamic == true then
13     h = hinit;
14   end if;
15
16 equation
17   if dynamic == true then
18     A*der(h) = Fi-F;
19   else
20     Fi-F = 0;
21   end if;
22   F = x*sqrt(h);
23
24 end Tank_Model;
```

Figure 7. OpenModelica code required to model the system given in Figure 6.

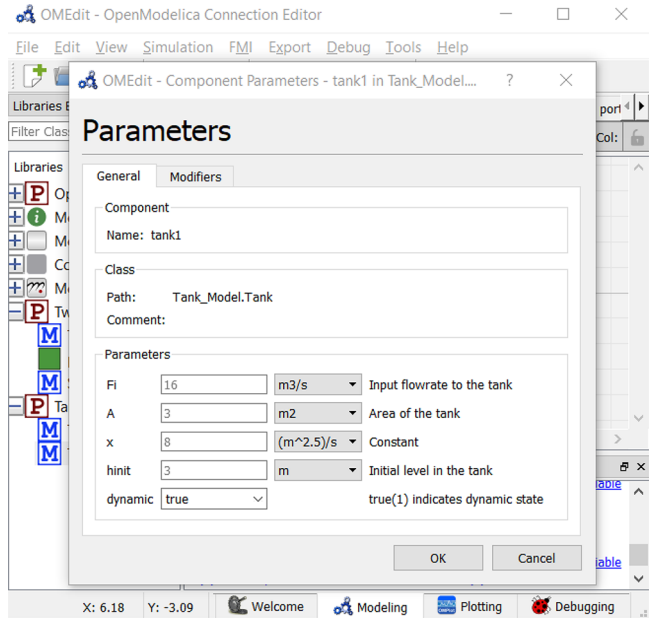


Figure 8. Selecting the Boolean parameter dynamic at compile time.

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

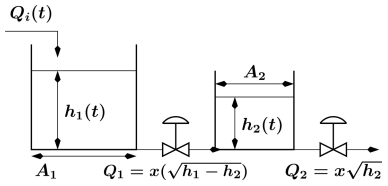


Figure 9. Two tanks in series.

```
1   model Tank
2     Real Fi(unit="m3/s") "Input flowrate to the tank";
3     parameter Real x(unit="(m^2.5)/s") "Constant";
4     parameter Real A(unit="m2") "Area of the tank";
5     Real F(unit="m3/s") "Output flowrate from the tank";
6     Real h(unit="m") "Tank level";
7     Real delH(unit="m");
8     parameter Real hinit(unit="m") "Initial level in the tank";
9     parameter Boolean dynamic "true(1) indicates dynamic state";
10  Two_Tank.port inlet;
11  Two_Tank.port outlet;
12    initial equation
13    if dynamic ==  true then
14    h = hinit;
15    end if;
16  equation
17    inlet.f = Fi;
18    outlet.f = F;
19    if dynamic == true then
20      A * der(h) = Fi - F;
21    else
22      Fi - F = 0;
23    end if;
24    F = x*sqrt(delH);
25  end Tank;
26
27  connector port
28  Real f;
29  end port;
```

Figure 10. Building block required to model the system given in Figure 9. It is arrived at by including connection details and a variable for $h_2 - h_1$ in the code of Figure 7.

Plots in Figure 5 show that the results of OpenModelica are in agreement with that of Shacham et al.[14] Temperature increases during the heating phase, becoming constant during the distillation period when the bubble point is reached. The liquid-phase composition is constant during the heating phase as there is no vapor formation.

In order to briefly explain the way one models in OpenModelica, a distilled version of the required code is presented here, using the correlations given by Shacham et al.[14] In general, one makes use of property databases, such as Chemsep, and thermodynamic correlations, as explained by the authors.[13] To complete the picture, a knocked-down version of the code, with the minimal required property database and thermodynamic calculations, is given in the Appendix. Although it may appear imposing, it is to be noted that one has to write the code given in lines 105−145 only, as all other calculations will be taken care of by the freely accessible thermodynamic libraries.[15]

2.3. Switching between Steady-State and Dynamic Simulations. As explained earlier, the objective of this work is to create a simulation environment suitable for steady-state and dynamic simulation of chemical processes. In general, only the hold-up equations need to be changed when one goes from steady-state to dynamic models. A modeling environment that provides this capability will reduce the efforts required to do these simulations. The fact that OpenModelica meets this requirement through the parameter statement is the topic of discussion in this section.

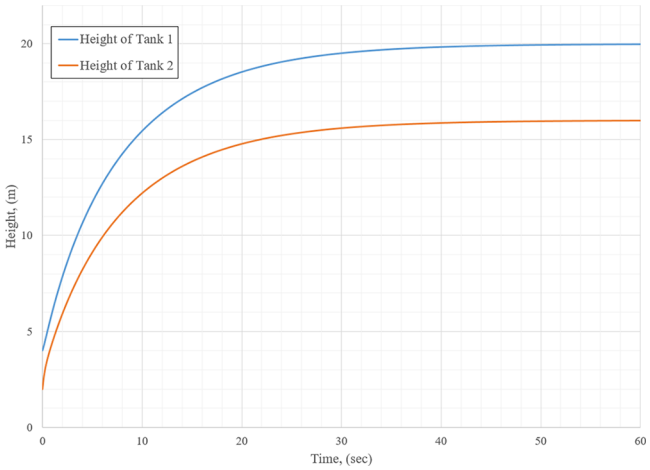


Figure 12. Evolution of the heights of the two-tank system described in Figure 9.

Consider the simple tank shown in Figure 6, wherein a fluid of constant density flows in and flows out into a tank of uniform cross-section A. The outflow rate is proportional to the square root of the height of the liquid in the tank. It is desired to solve this system and determine the value of $h(t)$.

The dynamics of this system are described by the following equations:

$$A\frac{dh(t)}{dt} = F_i(t) - F(t)$$

$$F(t) = x(t)\sqrt{h(t)}$$

The steady-state model of this system is given by

$$0 = F_i(t) - F(t)$$

$$F(t) = x(t)\sqrt{h(t)}$$

The second equation is identical in both models.

This problem is coded in OpenModelica as given in Figure 7. The code is self-explanatory. The following values were chosen with appropriate units: $F_i = 16$, $A = 3$, and $x = 8$.

When this code is executed through the OpenModelica GUI, an interface, as given in Figure 8, appears. Depending on the value assigned to the Boolean parameter `dynamic`, either the steady-state model is solved or the dynamic model is integrated. Assigning a value to this parameter is done at the compile

```
1   model System
2   Tank Tank1(A = 3, dynamic = true, hinit = 4, x = 8, F(start=0.01), delH(start=1)
      );
3   Tank Tank2(A = 1, dynamic = true, hinit = 2, x = 4, F(start=0.01), delH(start=1)
      );
4   equation
5   connect(Tank1.outlet,Tank2.inlet);
6   Tank1.Fi = 16;
7   Tank1.delH = Tank1.h - Tank2.h;
8   Tank2.delH = Tank2.h;
9   end System;
```

Figure 11. Code that, along with the code given in Figure 10, helps model the system given in Figure 9.

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

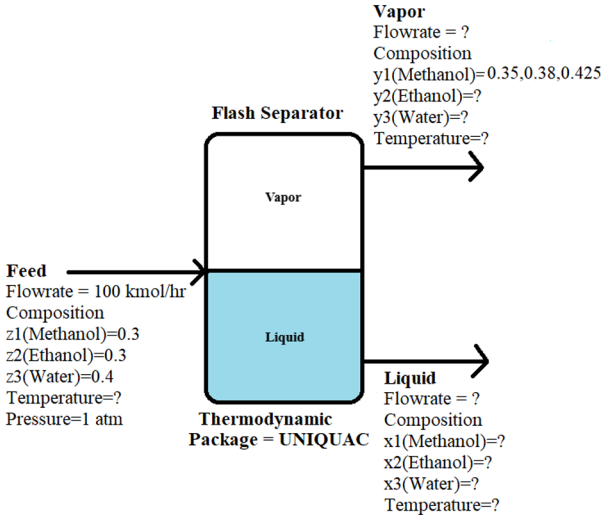


Figure 13. Model with problem statement for steady-state flash of methanol−ethanol−water. The temperatures of the feed and the flash drum are to be determined. The vapor-phase methanol mole fraction is specified. All other product compositions are to be calculated.

time. Thus, the conditional statement involving this parameter is evaluated only once, as opposed to the block `fT` occurring in lines 23−35 of Figure 3 with implications on the execution time.

The ability of switching the model outside the model definition may help extend OpenModelica to carry out batch process simulation, a topic currently investigated.

2.4. Connection of Building Blocks. Connecting unit models in OpenModelica is extremely simple. In this section, two of the tanks described previously are connected and simulated. A schematic of this arrangement is given in Figure 9

The building block of this system is the tank discussed earlier, with a modification to account for the fact that we would want to connect it with the outside world. A model of the code is given in Figure 10. The required two-tank system is obtained by instantiating this model twice and connecting them, as shown in Figure 11. The resulting profiles are given in Figure 12.

3. FLOWSHEETING IN OPENMODELICA

This section begins with a description of a library of unit operations created for OpenModelica. The use of this library is demonstrated with the help of a few representative flowsheets, and the results are compared with those from DWSIM and Aspen Plus. All the flowsheets presented in this section are available to the community.[16]

Table 1. Results of Simulations in OpenModelica and in DWSIM[a]

OpenModelica		
desired vapor composition (methanol)	temperature (K)	liquid composition
0.35	351.21	0.1985
0.38	350.28	0.2354
0.425	349.24	0.274
DWSIM		
desired vapor composition (methanol)	temperature (K)	liquid composition
0.35	351.26	0.199
0.38	350.211	0.234
0.425	349.12	0.279

[a]The two sets match. The flash temperatures for different vapor mole fractions are listed.

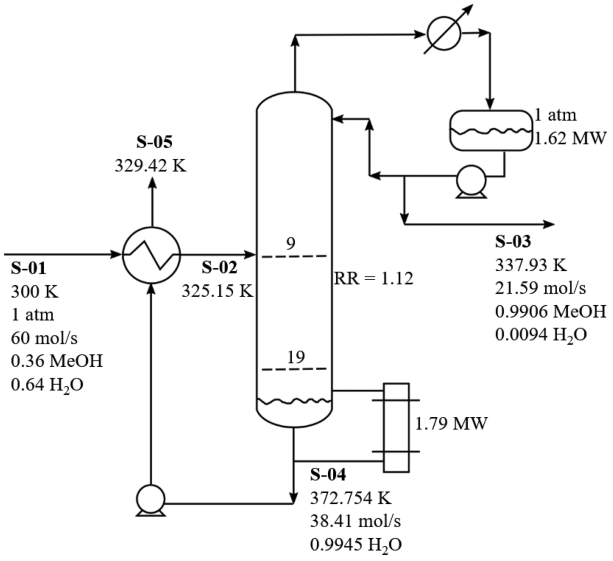


Figure 15. Process flowsheet for methanol−water distillation with a preheater.

```
1  model flash
2  parameter Chemsep_Database.Methanol methanol;
3  parameter Chemsep_Database.Ethanol ethanol;
4  parameter Chemsep_Database.Water water;
5  extends Thermodynamic_Packages.UNIQUAC(NOC=3,Comp = {methanol,ethanol,water},P
       =101325);
6  Real FF(min=0),VV(min=0),LL(min=0),k[3](each start=1),Psat[3];
7  parameter Real z[3] = {0.3,0.3,0.4};
8  equation
9  FF = 100;
10 y[1] = 0.425;
11 z[:].*FF - (x[:].*LL + y[:].*VV) = {0,0,0};
12 y[:] = k[:].*x[:];
13 for i in 1:NOC loop
14 Psat[i] = Thermodynamic_Functions.Psat(Comp[i].VP, T);
15 end for;
16 k[:] = (gamma[:].*Psat[:])./P;
17 sum(x[:]) = 1;
18 sum(y[:]) = 1;
19 end flash;
```

Figure 14. Flash model as written in OpenModelica.

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

Table 2. Streamwise Results of Methanol−Water Distillation in OpenModelica, DWSIM, and Aspen Plus

	S-01			S-02			S-03		
Parameter	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature (K)	300	300	300	325.15	325.15	325.15	337.934	337.934	337.934
Pressure (Pa)	101325	101325	101325	101325	101325	101325	101325	101325	101325
Mass Flow Rate (kg/s)	1.383	1.383	1.383	1.383	1.383	1.383	0.6889	0.6889	0.6889
Molar Flow Rate (mol/s)	60	60	60	60	60	60	21.59	21.589	21.59
Mole Fraction									
Methanol	0.36	0.36	0.36	0.36	0.36	0.36	0.9906	0.9906	0.9906
Water	0.64	0.64	0.64	0.64	0.64	0.64	0.0094	0.0094	0.0094

	S-04			S-05		
Parameter	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature (K)	372.754	372.754	371.691	329.419	329.421	329.4
Pressure (Pa)	101325	101325	101325	101325	101325	101325
Mass Flow Rate (kg/s)	0.694	0.694	0.673	0.694	0.694	0.694
Molar Flow Rate (mol/s)	38.41	38.41	38.41	38.41	38.41	38.41
Mole Fraction						
Methanol	0.0045	0.0045	0.0045	0.0045	0.0045	0.0045
Water	0.9945	0.9945	0.9945	0.9945	0.9945	0.9945

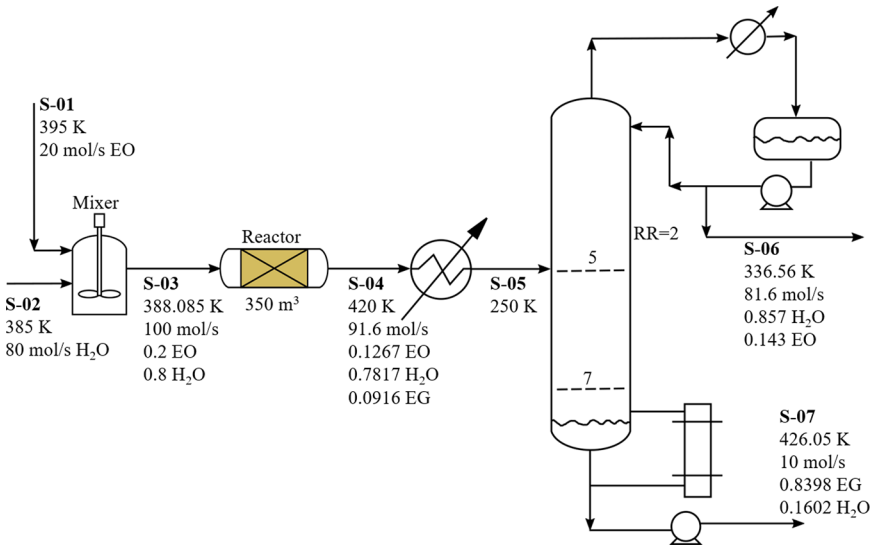


Figure 16. Process flowsheet for the production of ethylene glycol.

3.1. Chemical Engineering Models in OpenModelica. Although OpenModelica comes with a large collection of models from many domains, a chemical-engineering library is nonexistent, with the exception of refrigeration systems. An attempt is made in this work to overcome this shortcoming. As mentioned earlier, property databases and thermodynamic correlations have already been made available in OpenModelica.[13]

Material streams and unit operations are two types of models created in this work and released to the community.[15] Material-stream models do the separation into different phases and connect them to appropriate ports of unit operations. For example, in two-phase streams, the mixture is flashed in the material-stream models. Different types of flash are possible, depending on the specification.

The important part of this modelica library is the Unit Operations package. At present, models of the following building blocks are available: (1) Mixer; (2) Heater/Cooler; (3) Flash; (4) Shortcut Column; (5) Rigorous Column: Distillation Column and Absorption Column; (6) Reactors: Plug Flow, Continuously Stirred Tank and Conversion; (7) Heat Exchanger: Counter Current and Co-Current; (8) Compound Separator; (9) Valve; (10) Splitter; and (11) Adiabatic Expander/Compressor.

The input and output variables specific to the unit operation are defined in the respective unit operation models developed in OpenModelica. The unit operation library essentially has all equations that describe the process model for every unit operation. Additionally, the connector equations that are used to connect the inlet and outlet ports of the unit operations with its variables are also defined.

A few sample flowsheets will be presented next.

3.2. Design Problem of a Steady-State Flash. The steady-state flash simulation of a methanol−ethanol−water system is presented in this section. The output composition of the vapor product is fixed, whereas the operating temperature of the flash column is a free variable. The feed temperature is also not specified, whereas the flow rate and the composition are fixed. Pressure is kept constant at 1 atm. Because the objective is to determine the temperature at which the flash column operates and at which the feed enters, it is a design problem (see Figure 13).

It is desired to calculate all other variables for three different values of the methanol mole fraction, y_1. In other words, vapor compositions of ethanol and water, the temperatures of all streams, and all flow rates need to be calculated. The schematic of the problem statement is presented in Figure 13. The UNIQUAC

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

Table 3. Streamwise Results of Ethylene Glycol Production in OpenModelica, DWSIM, and Aspen Plus

	S-01			S-02			S-03			S-04		
	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature (K)	395	395	395	385	385	385	388.085	390.01	390.011	420	420.3	420
Pressure (Pa)	200000	200000	200000	100000	100000	100000	100000	100000	100000	99998	99997.97	100000
Mass Flow Rate (kg/s)	0.88	0.88	0.88	1.44	1.44	1.44	2.32	2.32	2.32	2.32	2.32	2.32
Molar Flow Rate (mol/s)	20	20	20	80	80	80	100	100	100	91.6	91.6	91.837
Mole Fraction												
Ethylene Oxide	1	1	1	0	0	0	0.2	0.2	0.2	0.1267	0.13	0.1289
Water	0	0	0	1	1	1	0.8	0.8	0.8	0.7817	0.78	0.7822
Ethylene Glycol	0	0	0	0	0	0	0	0	0	0.0916	0.09	0.0889

	S-05			S-06			S-07		
	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature (K)	250	250	250	336.562	336.56	336.138	426.05	426.08	422.278
Pressure (Pa)	99998	9999.97	100000	101325	101325	101325	101325	101325	101325
Mass Flow Rate (kg/s)	2.32	2.32	2.32	1.77	1.77	1.782	0.55	0.55	0.54
Molar Flow Rate (mol/s)	91.6	91.6	91.8371	81.6	81.6	81.8371	10	10	10
Mole Fraction									
Ethylene Oxide	0.1267	0.13	0.1289	0.143	0.14	0.1446	0	0	0
Water	0.7817	0.78	0.7822	0.857	0.86	0.8554	0.1602	0.16	0.1837
Ethylene Glycol	0.0916	0.09	0.0889	0	0	0	0.8398	0.84	0.8163

activity coefficient model is used as the phase equilibrium model.

The code in Figure 14 depicts the flash example as developed in OpenModelica. It inherits thermodynamic correlations and implements the well-known flash algorithm. The simulation is run for three different desired vapor compositions of methanol. The minimum and maximum temperatures were taken to be the boiling points of pure methanol and water, and the initial guess for temperature was taken to be the average of these two boiling points.

Results of these calculations are presented in Table 1. One can see that these results are consistent with the general requirement that the higher the mole fraction of the least volatile component, the lower the temperature. The human effort to solve the system remains the same no matter which variable is to be solved. In a sequential modular approach;[6] however, one has to resort to a trial and error method, depending on which variable is the unknown.

3.3. Methanol−Water Distillation. In this section, a methanol−water distillation system with a preheater is studied. The schematic for the flowsheet is displayed in Figure 15. A material stream containing 36% methanol (by mole) and 64% water at 300 K and 1 atm of pressure is sent to a preheater at 60 mol/s. The feed is preheated to 325.15 K before entering the distillation column.

The feed is sent to the distillation column at the ninth stage. The condenser and reboiler pressures are maintained at 1 atm. A reflux ratio of 1.12 is defined as the top specification and a product molar flow rate of 38.4 mol/s is defined as the bottom specification. Methanol with 99% purity is obtained from the top at a temperature of 338 K. Water with 99.45% purity is obtained from the bottom at 372.75 K and is used to preheat the feed.

The streamwise results obtained from the simulated flowsheet are presented in Table 2 and are compared with the DWSIM and Aspen Plus results. It can be observed that the OpenModelica results closely match those of DWSIM and Aspen Plus.

3.4. Ethylene Glycol Production. The next example demonstrates the production of ethylene glycol (EG) from ethylene oxide (EO) and water. The schematic for the flowsheet is displayed in Figure 16. Two streams of pure ethylene oxide at 20 mol/s, 395 K, and pure water at 80 mol/s, 385 K, enter a stream mixer. The mixed output stream is used as the feed to the plug flow reactor. The following reaction takes place in the plug flow reactor:

$$C_2H_4O + H_2O \rightarrow C_2H_6O_2$$

The reaction kinetics is taken as $-r_A = 0.005\ C_{C_2H_4O}$ for the sake of brevity. The reaction is first-order with respect to ethylene oxide.

In the reactor, 42% conversion of ethylene oxide takes place. Ethylene glycol is formed as the product. Ethylene glycol, along with unreacted reactants, are cooled and sent to a distillation column with eight stages for further purification. The feed enters at the fifth stage of the distillation column. The condenser and reboiler pressure are maintained at 1 atm.

A reflux ratio of 2 and a product molar flow rate of 10 mol/s are defined as the top and bottom specifications, respectively; 84% ethylene glycol is obtained as the bottom product.

The streamwise results obtained from the simulated flowsheet are presented in Table 3 and are compared with the DWSIM and Aspen Plus results. It can be observed that the OpenModelica results closely match those of DWSIM and Aspen Plus.

3.5. Esterification of Acetic Acid To Produce Ethyl Acetate. In this section, production of ethyl acetate from acetic acid using an esterification reaction, as given in Figure 17, is studied. Ethanol (EtOH) at 60 mol/s is mixed with acetic acid (HAc) at 40 mol/s and fed to a conversion reactor. They both enter at 300 K and 1 atm of pressure. The following reaction takes place in the conversion reactor:

$$CH_3COOH + C_2H_5OH \rightarrow CH_3COOC_2H_5 + H_2O$$

The outlet temperature of the reactor is maintained at 300 K, and 60% conversion is defined for the acetic acid in the conversion reactor. Ethyl acetate (EtAc) and water are formed as

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

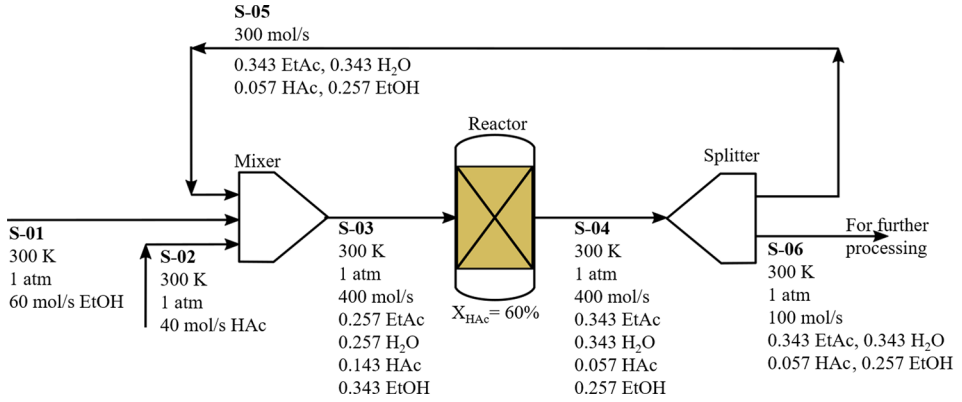


Figure 17. Process flowsheet for acetic acid esterification by ethanol to produce ethyl acetate and water.

Table 4. Streamwise Results of Acetic Acid Esterification by Ethanol in OpenModelica, DWSIM, and Aspen Plus

	S-01			S-02			S-03		
Parameter	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature(K)	300	300	300	300	300	300	300	300	300
Pressure(Pa)	101325	101325	101325	101325	101325	101325	101325	101325	101325
Mass Flow Rate(g/s)	2.764	2.764	2.76414	2.402	2.402	2.402	20.664	20.665	20.665
Molar Flow Rate(mol/s)	60	60	60	40	40	40	400	400	400
Mole Fraction									
Ethyl acetate	0	0	0	0	0	0	0.2571	0.2571	0.2571
Water	0	0	0	0	0	0	0.2571	0.2571	0.2571
Acetic acid	0	0	0	1	1	1	0.1429	0.1429	0.1429
Ethanol	1	1	1	0	0	0	0.3429	0.3429	0.3429
	S-04			S-05			S-06		
Parameter	OM	DWSIM	Aspen	OM	DWSIM	Aspen	OM	DWSIM	Aspen
Temperature (K)	300	300	300	300	300	300	300	300	300
Pressure(Pa)	101325	101325	101325	101325	101325	101325	101325	101325	101325
Mass Flow Rate(g/s)	20.664	20.607	20.665	15.4987	15.4554	15.4987	5.1662	5.1518	5.1662
Molar Flow Rate(mol/s)	400	398.883	400	300	299.162	300	100	99.7207	100
Mole Fraction									
Ethyl acetate	0.3429	0.3428	0.3428	0.3429	0.3428	0.3428	0.3429	0.3428	0.3428
Water	0.3429	0.3428	0.3428	0.3429	0.3428	0.3428	0.3429	0.3428	0.3428
Acetic acid	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571	0.0571
Ethanol	0.2571	0.2573	0.2573	0.2571	0.2573	0.2573	0.2571	0.2573	0.2573

products. Unreacted acetic acid and ethanol are also present in the product stream; 75% of the product stream is recycled back to the mixer, and the rest is sent out for further processing. The NRTL thermodynamic package is used to simulate the flowsheet.

The streamwise results obtained from the simulated flowsheet are presented in Table 4 and are compared with the DWSIM and Aspen Plus results. It can be observed that the OpenModelica results closely match those of DWSIM and Aspen Plus.

4. CONCLUSIONS AND FUTURE WORK

The use of OpenModelica is proposed for chemical-process simulations. Some useful features for this purpose have been identified. These are (a) the short development times required for model development, (b) the reusability of the code, (c) the ability to easily connect different building blocks of a flowsheet, and (d) the ability to switch from a steady-state simulation to a dynamic simulation at the compile time.

A library of streams and unit operations has been created to make OpenModelica suitable for process simulations. Using this library, a few representative flowsheets have been solved and the results validated with those from DWSIM and Aspen Plus. The usefulness of the equation-oriented approach used in OpenModelica is verified with the help of a design problem. Semibatch steam-distillation has also been simulated, and the results are matched with literature findings.

The benefits of using OpenModelica for process simulation are manifold. It can be used to carry out dynamic simulations, and training students to use OpenModelica will raise the overall levels of education, as the use of any equation-oriented simulator requires high skills. Finally, the large number of students proposed to be trained to use OpenModelica will be available as a work force for researchers around the world.

With the above objectives in mind, the authors are embarking on OpenModelica promotion in a big way. In addition to training a large number of people, this will also result in increasing the number of examples worked out in OpenModelica, which in turn will increase the usability of the software itself.

The following activities are proposed to be undertaken in the near future: (1) migrate more DWSIM flowsheets[2] and make them available to the community,[16] (2) create more process models for OpenModelica, (3) make the property data of more chemicals available in OpenModelica, (4) train a large number of students on the use of OpenModelica and offer them to small- and medium-scale industry, and (5) create more collaborative content for OpenModelica through activities such as Textbook Companion.[17]

APPENDIX

The following is the OpenModelica Code for semibatch steam distillation.

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

```
package Steam_Distillation
  package Chemsep_Database
    model General_Properties
      parameter Real Tc, SH, VP[6], LiqCp[6], HOV[6], VapCp[6];
    end General_Properties;

    model Noctane
      extends General_Properties(Tc =568.7, SH =-250256526.702, VP =
          {101,87.46069,-7578.199,-9.657211,5.664818E-06,2}, LiqCp =
          {16,184080,362.58,6.1268,0.015908,-0.000010697}, HOV = {106,6.509104E
          +07,0.906328,-0.61829,0.0251605,0.114898}, VapCp =
          {16,123360,-700.1,13.486,-0.00019118,4.5401E-08});
    end Noctane;

    model Ndecane
      extends General_Properties(Tc =617.7, SH =-300841257.339, VP =
          {101,6.023802,-5713.196,3.410225,-0.000012633,2}, LiqCp =
          {16,160660,291.43,8.5687,0.0098408,-0.0000060811}, HOV = {106,5.7689E
          +07,-1.1412,5.1463,-6.2946,2.6623}, VapCp =
          {16,152020,-697.29,13.714,-0.00021747,4.9426E-08});
    end Ndecane;

    model Water
      extends General_Properties(Tc =647.14, SH =-285790218.187, VP =
          {101,74.55502,-7295.586,-7.442448,0.0000042881,2}, LiqCp =
          {16,75539,-22297,136.02,-0.25622,0.00018273}, HOV = {106,5.964E
          +07,0.86515,-1.1134,0.67764,-0.026925}, VapCp =
          {16,33200,-878.9001,8.436956,0.00207627,-6.467085E-07});
    end Water;
  end Chemsep_Database;

  package Thermodynamic_Packages
    model Raoults_Law
      Real K[NOC];
      equation
        for i in 1:NOC loop
          K[i] = Thermodynamic_Functions.Psat(comp[i].VP, T) / P;
        end for;
    end Raoults_Law;
  end Thermodynamic_Packages;

  package Thermodynamic_Functions
     function Psat
      /*Returns vapor pressure at given temperature*/
      input Real VP[6] "from chemsep database";
      input Real T(unit = "K") "Temperature";
      output Real Vp(unit = "Pa") "Vapor pressure";
    algorithm
      Vp := exp(VP[2] + VP[3] / T + VP[4] * log(T) + VP[5] .* T .^ VP[6]);
    end Psat;

    function LiqCpId
      /*Calculates specific heat of liquid at given Temperature*/
      input Real LiqCp[6] "from chemsep database";
      input Real T(unit = "K") "Temperature";
      output Real Cp(unit = "J/kmol") "Specific heat of liquid";
    algorithm
      Cp := (LiqCp[2] + exp(LiqCp[3] / T + LiqCp[4] + LiqCp[5] * T + LiqCp[6] * T
          ^ 2));
    end LiqCpId;

    function VapCpId
      /*Calculates Vapor Specific Heat*/
      input Real VapCp[6] "from chemsep database";
      input Real T(unit = "K") "Temperature";
      output Real Cp(unit = "J/kmol.K") "specific heat";
    algorithm
      Cp := (VapCp[2] + exp(VapCp[3] / T + VapCp[4] + VapCp[5] * T + VapCp[6] * T
          ^ 2));
    end VapCpId;

    function HV
      /*Returns Heat of Vaporization*/
      input Real HOV[6] "from chemsep database";
      input Real Tc(unit = "K") "Critical Temperature";
      input Real T(unit = "K") "Temperature";
      output Real Hv(unit = "J/kmol") "Heat of Vaporization";
    protected
      Real Tr = T / Tc;
    algorithm
      if T < Tc then
        Hv := HOV[2] * (1 - Tr) ^ (HOV[3] + HOV[4] * Tr + HOV[5] * Tr ^ 2 + HOV[6]
            * Tr ^ 3);
      else
        Hv := 0;
      end if;
    end HV;

    function HLiqId
      /* Calculates Enthalpy of Ideal Liquid*/
      input Real SH(unit = "J/kmol") "from chemsep database std. Heat of formation
          ";
      input Real VapCp[6] "from chemsep database";
      input Real HOV[6] "from chemsep database";
      input Real Tc "critical temp, from chemsep database";
      input Real T(unit = "K") "Temperature";
      output Real Ent(unit = "J/kmol") "Molar Enthalpy";
    algorithm
      Ent := HVapId(SH, VapCp, HOV, Tc, T) - Thermodynamic_Functions.HV(HOV, Tc, T
          );
    end HLiqId;

    function HVapId
      /* Calculates enthalpy of ideal vapor */
      input Real SH(unit = "J/kmol") "from chemsep database std. Heat of formation
          ";
      input Real VapCp[6] "from chemsep database";
      input Real HOV[6] "from chemsep database";
      input Real Tc "critical temp, from chemsep database";
      input Real T(unit = "K") "Temperature";
      output Real Ent(unit = "J/kmol") "Molar Enthalpy";
    protected
      Integer n = 100;
      Real Cp[n - 1];
    algorithm
      for i in 1:n - 1 loop
        Cp[i] := VapCpId(VapCp, 298.15 + i * (T - 298.15) / n);
      end for;
      Ent := SH + (T - 298.15) * (VapCpId(VapCp, T) / 2 + sum(Cp[:]) + VapCpId(
          VapCp, 298.15) / 2) / n;
    end HVapId;
  end Thermodynamic_Functions;

  model Stm_Distill
    //For all array variables, [1]:N-octane, [2]:N-decane, [3]:Water
    extends Thermodynamic_Packages.Raoults_Law;
    parameter Integer NOC = 3;
    parameter Chemsep_Database.Noctane noctane;
    parameter Chemsep_Database.Ndecane ndecane;
    parameter Chemsep_Database.Water water;
    parameter Chemsep_Database.General_Properties comp[NOC] = {noctane, ndecane,
        water};
    parameter Real UA(unit = "J/s-K") = 1.05;
    parameter Real Ta(unit = "K") = 298.15;
    parameter Real P(unit = "Pa") = 9.839E+4;
    parameter Real Ws(unit = "kmol/s") = 3.85E-5;
    Real Mw(start = 0, unit = "kmol"), T(start = 298.15, unit = "K"), Q(start = 0
        unit = "J/s"), x[NOC](start = {0.725, 0.275, 0}), CpL[NOC](each unit = "J
        kmolK"), Hs(unit = "J/kmol"), M(start = 0.015, unit = "kmol"), V(start = 0
        unit = "kmol/s"), y[NOC](each start = 0), Hl[NOC](each unit = "J/kmol"),
        Hvmix(unit = "J/kmol"), Hv[NOC](each unit = "J/kmol");
    Real Tc(unit = "degree C"), t1(unit = "min"), fT;
  equation
    Tc = T - 273.15;
    x[3] = 1;
    Hs = Thermodynamic_Functions.HVapId(water.SH, water.VapCp, water.HOV,
        water.Tc, 372.2);
    t1 = time / 60;
    x[2] = 1 - x[1];
    y = K .* x;
    for i in 1:NOC loop
      CpL[i] = Thermodynamic_Functions.LiqCpId(comp[i].LiqCp, T);
      Hl[i] = Thermodynamic_Functions.HLiqId(comp[i].SH, comp[i].VapCp, comp[i].
          HOV, comp[i].Tc, T);
      Hv[i] = Thermodynamic_Functions.HVapId(comp[i].SH, comp[i].VapCp, comp[i].
          HOV, comp[i].Tc, T);
    end for;
    Hvmix = sum(Hv .* y);
    der(Mw) = Ws - V * y[3];
    Q = UA * (T - Ta);
    der(M) = -V * (y[1] + y[2]);
    fT = 1 - sum(y);
    if abs(fT) < 0.01 then
      der(x[1]) = ((-V * y[1]) - x[1] * (-V * (y[1] + y[2]))) / M;
      V = (Ws * (Hs - Hl[3]) - Q) / (Hvmix - sum(Hl .* y));
      der(T) = 1000 * (1 - sum(y));
    else
      der(x[1]) = 0;
      V = 0;
      der(T) = (Ws * (Hs - Hl[3]) - Q) / (Mw * CpL[3] + M * (x[1] * CpL[1] + x[2]
          * CpL[2]));
    end if;
  end Stm_Distill;

end Steam_Distillation;
```

AUTHOR INFORMATION

Corresponding Author

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

ORCID

Priyam Nayak: 0000-0003-3144-9137

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 FOSSEE project[18] at IIT Bombay.

DEDICATION

Professor Sirish Shah has been passionate about improving the state of chemical-engineering education in developing countries like Bangladesh, Kenya, and India. The authors are happy to contribute this article to the Sirish Shah Festschrift.

REFERENCES

(1) Medeiros, D. *DWSIM—Process Simulation, Modeling and Optimization Technical Manual*. http://dwsim.inforside.com.br/ (accessed Jan 1, 2019).

(2) DWSIM Team. Completed Flowsheets. *DWSIM Website*. https://dwsim.fossee.in/flowsheeting-project/completed-flowsheet (accessed Jan 1, 2019).

(3) Fritzson, P. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, 2nd ed.; Wiley-Blackwell: New York, 2015.

(4) Fritzson, P.; Engelson, V. Modelica - A unified object-oriented language for system modeling and simulation. *European Conference on Object-Oriented Programming*. **1998**, *1445*, 67−90.

(5) Modelica Language. *Modelica Association Website*. https://www.modelica.org/modelicalanguage (accessed Jan 1, 2019).

(6) Westerberg, A. W.; Hutchison, H.; Motard, R. L.; Winter, P. *Process flowsheeting*; Cambridge University Press, 1979.

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174

(7) Piela, P. C.; Epperly, T.; Westerberg, K.; Westerberg, A. W. ASCEND: An objectoriented computer environment for modeling and analysis: The modeling language. *Comput. Chem. Eng.* **1991**, *15*, 53−72.
(8) Moudgalya, K. M., Naren, P. R., Wagner, D., Eds.; *Proceedings of the National Conference on Chemical Process Simulation-2018*, Mumbai, India, Nov 26, 2018; Shroff Publishers, 2018. https://fossee.in/nccps-2018.
(9) Moudgalya, K. M.; Nemmaru, B.; Datta, K.; Nayak, P.; Fritzson, P.; Pop, A. Large Scale Training through Spoken Tutorials to Promote OpenModelica. *Proceedings of the 12th International Modelica Conference*, Prague, Czech Republic, May 15−17, 2017; Linköping University Electronic Press, 2017; pp 275−284.
(10) The OpenModelica Team of FOSSEE. Spoken Tutorials on OpenModelica. *Spoken Tutorial, IIT Bombay*. http://spoken-tutorial.org/tutorial-search/?search_foss=OpenModelica&search_language=English.
(11) Fritzon, P.; Pop, A.; Asghar, A.; Bachmann, B.; Braun, W.; Braun, R.; Buffoni, L.; Casella, F.; Castro, R.; Danós, A.; et al. The OpenModelica Integrated Modeling, Simulation and Optimization Environment. *Proceedings of the 1st American Modelica Conference*, Cambridge, MA, Oct 9−10, 2018; Modelica Association, 2018; pp 206−219.
(12) Pop, A.; Östlund, P.; Casella, F.; Sjölund, M.; Franke, R. A New OpenModelica Compiler High Performance Frontend. *Proceedings of the 13th International Modelica Conference*, Regensburg, Germany, March 4−6, 2019; Modelica Association, 2019; pp 689−698.
(13) Jain, R.; Nayak, P.; A. S., R.; Dalve, P.; Moudgalya, K. M.; Naren, P. R.; Wagner, D.; Fritzson, P. Implementation of a Property Database and Thermodynamic Calculations in OpenModelica for Chemical Process Simulation. *Ind. Eng. Chem. Res.* **2019**, *58*, 7551−7560.
(14) Shacham, M.; Cutlip, M. B.; Elly, M. Semi-Batch Steam Distillation of a Binary Organic Mixture: a Demonstration of Advanced Problem-Solving Techniques and Tools. *Chem. Eng. Educ.* **2012**, *46*, 173−181.
(15) FOSSEE. OMChemSim. *GitHub*. https://github.com/FOSSEE/OMChemSim (accessed Jan 1, 2019).
(16) OpenModelica Team of the FOSSEE Project. Completed Flowsheets. *OpenModelica Website*. https://om.fossee.in/chemical/flowsheeting-project/completed-flowsheet (accessed May 15, 2019).
(17) OpenModelica Team. Completed Books. *OpenModelica Website*. https://om.fossee.in/textbook-companion/completed-books (accessed Jan 1, 2019).
(18) FOSSEE-Team. *Free and Open Source Software in Education (FOSSEE) Website*. https://fossee.in/ (accessed Jan 1, 2019).

DOI: 10.1021/acs.iecr.9b00104
Ind. Eng. Chem. Res. 2019, 58, 11164−11174