Using Chemsep, COCO and other modeling tools for versatility in custom process modeling

Jasper van Baten – AmsterCHEM (jasper@amsterchem.com)
Ross Taylor – Clarkson University (taylor@clarkson.edu)& Chemsep
Harry Kooijman – Clarkson University (kooijman@clarkson.edu) & Chemsep

Summary

A steady state flowsheeting environment is presented in which out-of-the-box unit operation models can be combined with formula based custom models; the formulas can be presented in Excel, Matlab or Scilab format. The interaction between all simulation software components is based on the CAPE-OPEN standard specifications, which makes that all software components can be reused in other simulation environments. Similarly, software components (such as thermodynamic servers and unit operations) of other simulation environments can be included in the simulation.

The combination of out-of-the-box unit operation models and custom unit operation models makes the presented framework suitable for research as well as teaching purposes.

Introduction

Main stream commercial steady state flowsheet based process simulation applications are available for university courses for attractive prices. The models provided by most simulation environments are often black-box and the model equations may not be readily accessible to the end-user. For many simulation purposes this suffices. However, for the purpose of teaching chemical engineering and for the purpose of research, it may be prudent to allow custom modeling of particular equipment in an easy and transparent way, using generic modeling tools available to students, teachers and researchers.

Although simulation environments generally provide a wide selection of models, it is impossible for any simulation environment to provide all models that the end-user may want to use in a particular context. For example, heat exchangers are often modeled by a counter- or co-current approximation, which does not describe in detail what happens in the most common heat exchanger equipment, the shell-and-tube heat exchanger. In chemical engineering courses plug flow models are often compared to tanks-in-series models. The former is generally provided; the latter is not. Or – the example used in this presentation – a simple model for a membrane separator. Due to the diversity in available data on membrane separations, it is not easy to provide a generic membrane separation model. Other examples of simple equipment that cannot easily be described with generic models are easily provided, all one needs to do is open a text book on chemical engineering and look at the versatility of process equipment provided[1].

Most simulation environments allow for custom models. Often, this involves writing a FORTRAN routine that is suitable only for the simulation environment at hand, i.e. a specialized interface routine only to be used in the simulation environment it was created for. Such an approach requires programming skills on the end-user, which in the case of chemical engineering teaching courses may be a large threshold to use such custom models. In addition, to the use the same custom model in a different simulation environment, the programming exercise needs to be repeated.

Here, we present the COCO[3] simulation environment that is available free-of-charge. For the separation columns, we present ChemSep[4]; the LITE version of which is available free-of-charge (the full version of ChemSep has been available free-of-charge for educational

use from the CACHE corporation for more than 15 years now). The combination of COCO and ChemSep has been presented earlier[5-11], and we refer to these earlier publications and presentations for a full introduction to using COCO with ChemSep. In the current presentation, we will focus on the inclusion of custom models using generic modeling tools like Scilab[12], Matlab[13] and Excel[14]. All these software components are interacting via CAPE-OPEN[15] interfaces: an open standard that enables different chemical engineering software libraries and packages to interoperate for the purpose of solving flowsheets; i.e. reactors, distillation columns, and other unit operations with interconnecting material, energy, and information streams. CAPE-OPEN thermodynamic and unit operation interfaces have been well established, validated and demonstrated[8, 16-22].

The COCO steady state flowsheeting suite is entirely based on this open standard; all thermodynamic and unit operations models can be re-used in other simulation environments that are CAPE-OPEN compliant. Similarly, all third party unit operation models and thermodynamic servers that are CAPE-OPEN compliant can be used in COCO. As the COCO suite does not include models for distillation, absorption, and extraction columns, it is bundled with the ChemSep LITE column simulator.

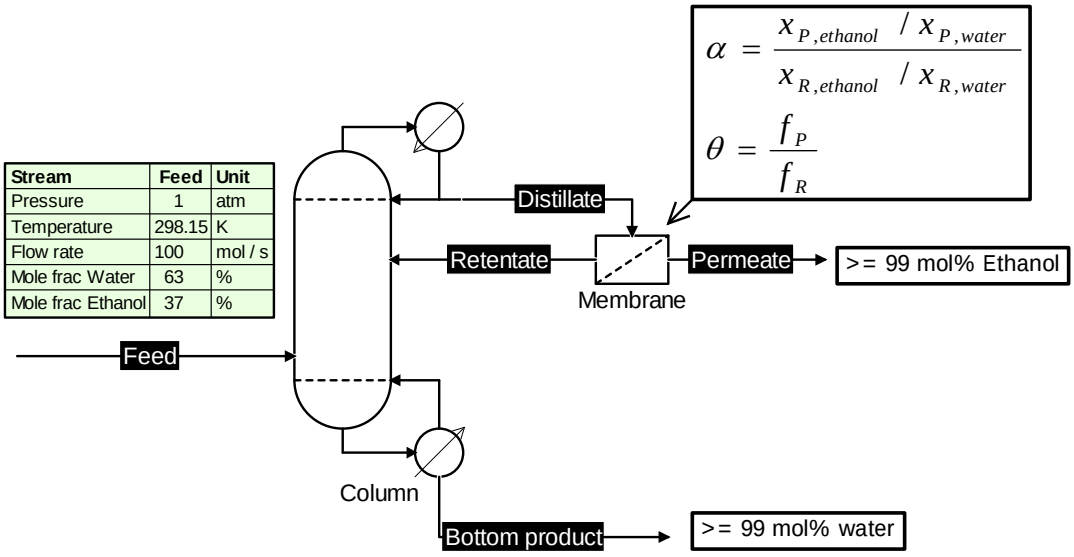


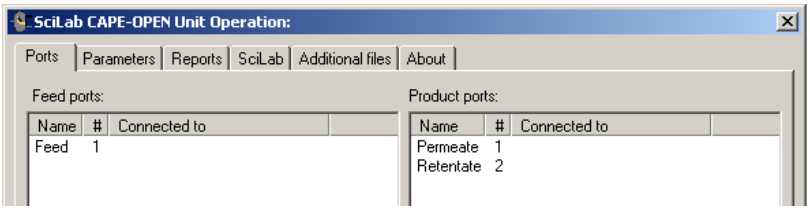
Figure 1: Test case for separation using a distillation column combined with a membrane unit with feed conditions and desired separation targets. The feed goes to a distillation column operating at 1 atmosphere with a partial reboiler and a total condenser.

We will demonstrate how to set up custom models using these modeling tools with a simple test case; this test case has been chosen for its simplicity of demonstration. This simplicity does not reflect on the possibilities of the modeling tools shown here, but is chosen merely for demonstrating the possibilities without getting lost in modeling details. The test case has been taken from a textbook[23]; the goal being to separate water and ethanol. The water-ethanol system has an azeotrope at about 90 mol% ethanol, and direct separation is not possible using a single distillation column. A combined distillation with membrane separation is used as shown in Figure 1. To get started, we start COFE, add a new TEA thermo package called “water-ethanol”, add compounds Water and Ethanol, and select the default NTRL model set. We define and specify the feed stream. We then insert a ChemSep distillation column, and accept all its defaults for “normal distillation” at its creation.

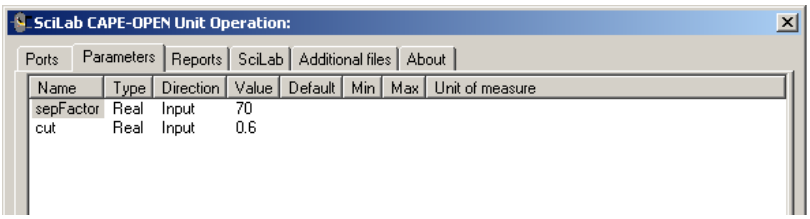
The membrane model is defined by a separation factor α and a permeate cut θ as shown in Figure 1, where $x_{P,ethanol}$ and $x_{R,ethanol}$ are the ethanol mole fractions in permeate and retentate, $x_{P,water}$ and $x_{R,water}$ are the water mole fractions in permeate and retentate and f_P and f_R are the total permeate and retentate flows. Separation factor α and a permeate cut θ are model parameters, with default values of 70 and 0.6 respectively.

The Scilab Unit Operation and the Matlab Unit Operation

To create the simple membrane model using the Scilab scripting language, we start by inserting a Scilab Unit Operation[24]. The first thing we do is define the ports; the feed, the retentate and the permeate products:



We notice that the ports have received indices; these we will use in the model script to access the values associated with the ports. We then proceed to define the parameters for our model:



The parameters are in this case dimensionless; dimensions and bounds for parameters are optional. For our simple model, the definition of the parameters, the component balances and the product composition summation equations can be solved explicitly. This is not a requirement however of the modeling tools; all modeling tools that are used come with built-in solvers. To demonstrate their use, we will only use a partly analytic solution. The flows are solved explicitly. To solve the compositions, we will solve a one dimensional problem given by: find $x_{P,ethanol}$ so that the target value of separation factor α is satisfied. Given $x_{P,ethanol}$, the other compositions follow from the component balances and summation as shown by the `GetCompositions` function in the Scilab script for solving the unit:

```
// compound indices:
water=1;
ethanol=2;

//parameter values:
alpha=getParameter('sepFactor');
theta=getParameter('cut');

//feed values
XF=getFeedProp(1,'fraction');
FF=getFeedProp(1,'totalFlow');
TF=getFeedProp(1,'temperature');
PF=getFeedProp(1,'pressure');
```

```
//product flows (explicit solution of theta=FP/FR and FF=FP+FR):
FR=FF/(theta+1);
FP=theta*FR;

//function to calculate compositions from XPE:
function [XP,XR]=GetCompositions(XPE)
 XP=[1-XPE,XPE];
 XRE=(FF*XF(ethanol)-FP*XPE)/FR;
 XR=[1-XRE,XRE];
endfunction

//function to solve compositions
function errorValue=CompositionFunction(XPE)
 //get the compositions given XPE:
 [XP,XR]=GetCompositions(XPE);
 //return the error in alpha
 errorValue= (XP(ethanol)/XP(water))/(XR(ethanol)/XR(water))-alpha;
endfunction

//solve for the compositions to match alpha:
XPE=fsolve(0.999,CompositionFunction);
//the compositions at solutions are then
[XP,XR]=GetCompositions(XPE);

//set the permeate, given FP, XP, TF, PF
setProduct(1,FP,XP,'temperature',TF,'pressure',PF);

//set the retentate, given FR, XR, TF, PF
setProduct(2,FR,XR,'temperature',TF,'pressure',PF);
```

The script obtain the inputs, calculates the product flow rates explicitly, defines a function that gives the product compositions as a function of $x_{P,ethanol}$, defines a function that gives the error in α as a function of $x_{P,ethanol}$, and solves for the error to be zero. The product streams are defined by temperature and pressure of the feed in combination with the calculated flows and compositions. Programming is largely unnecessary; we mostly type the functions and ask the Scilab program to solve them. As pointed out earlier, the iterative solution here is for demonstration purposes, the problem has a simple explicit solution. We could have typed that instead. The `CompositionFunction` has multiple solutions; undesired solutions lie at negative $x_{P,ethanol}$; we choose the initial guess of $x_{P,ethanol}$ at a value of 0.999 to avoid the undesired negative solution.

This example does not use any thermodynamics from the simulation environment; all thermodynamic property calculations and phase equilibrium calculations are however available and can be used in unit operation models. Appendix A provides an example.

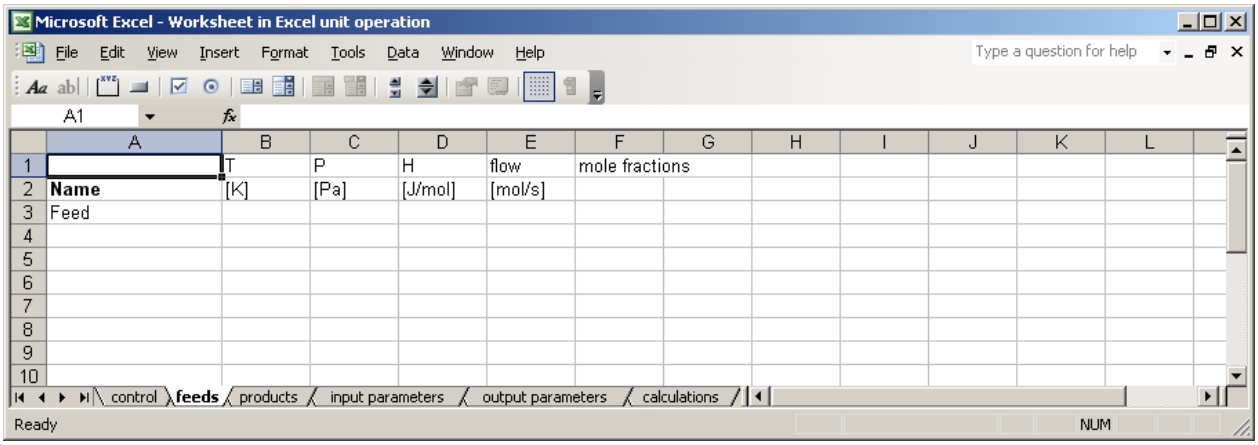
The Matlab Unit Operation[25] is based on the same principles as the Scilab Unit Operation and very similar to the Scilab Unit Operation. The example is readily transferred to the Matlab Unit Operation; the Matlab version is not shown here.

To complete the flowsheet, we edit the ChemSep column, set the number of stages to 20, move the feed to stage 17. We set the bottom specification to 99.9 mol% water; this is tighter than required by the problem specification. For the top specification, we use 85% ethanol, to stay below the azeotrope. Running the simulation without the recycle gives us an idea of the retentate composition; we check the column profiles to see where the tray compositions are close to this: stage 6. Here we create a second feed to the column and connect the retentate stream. We solve the problem, and our target specifications have been satisfied.

The Excel Unit Operation

Even though programming is largely unnecessary for the Scilab implementation of the membrane unit as shown above, some users may be more comfortable with entering the formulas in a spread sheet manner. This is possible, using the Excel Unit Operation[26]. We take the flowsheet created above as a starting point, and remove the Scilab unit operation. We

insert a new empty Excel Unit Operation. Editing the unit opens Excel; ports and parameters are simply entered on dedicated pre-existing Excel sheets. On the *feeds* sheet, we enter our feed:



	A	B	C	D	E	F	G	H	I	J	K	L
1		T	P	H	flow	mole fractions						
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]							
3	Feed											
4												
5												
6												
7												
8												
9												
10												

We create the Permeate and Retentate on the *products* sheet:

	A	B	C	D	E	F	G	H
1		T	P	H	flow	mole fractions		
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]			
3	Permeate							
4	Retentate							

Here, we need to provide values for the flow and compositions for each of the products, and two out of three of T, P and H. The parameters are entered on the *input parameters* sheet:

alpha 70

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**name**	**value**	**min**	**max**	**m**	**kg**	**s**	**A**	**K**	**mole**	**cd**	**rad**	**relative flag**
2	sepFactor	70											
3	cut	0.6											

Initial values for the parameters are entered; the user can modify these from the simulation environment, and the new values will automatically appear in the sheet. Hence, we can refer to the values in these cells in our formulas; for easy reference, the names *alpha* and *theta* have been assigned to the cells. If we close Excel, connect the ports and re-edit the unit, the feed values are present, which makes entering the formulas easier:

	A	B	C	D	E	F	G	H
1		T	P	H	flow	mole fractions		
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]	Water	Ethanol	
3	Feed	351.4708	101325	-35678.5	43.4629	0.15	0.85	
4								

The formulas we enter on a separate *calculations* sheet:

	A	B	C	D
1	F retentate	=feeds!E3/(theta+1)	mol/s	
2	F permeate	=theta*B1	mol/s	
3				
4	X p ethanol guess	0.999	mol/mol	Initial guess for X p ethanol
5	X p ethanol	0.999	mol/mol	Will be solved to match alpha
6	X p water	=1-B5	mol/mol	
7	X r ethanol	=(feeds!E3*feeds!G3-B2*B5)/B1	mol/mol	
8	X r water	=1-B7	mol/mol	
9				
10	alpha	=(B5/B6)/(B7/B8)		
11	error	=B10-alpha		

We now use the Excel solver to solve B11 for a value of zero by modifying B5. We save the solve scenario, and on the *control* sheet, we indicate that this solver scenario must be used during the calculation, using an initial guess specified in B4 for cell B5; to accomplish this, we assign a name to the saved solver scenario region, the initial guess ad the target cell for the initial guess. These names are specified in the *control* sheet; multiple sequential solver scenarios can be specified. Notice that in this simple problem, the initial guess is a constant; a formula can be entered here as well. On the *products* sheet, we set references to the solution:

	A	B	C	D	E	F	G
1		T	P	H	flow	mole fractions	
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]	Water	Ethanol
3	Permeate	=feeds!B3	=feeds!C3		=calculations!B2	=calculations!B6	=calculations!B5
4	Retentate	=feeds!B3	=feeds!C3		=calculations!B1	=calculations!B8	=calculations!B7
5							

and our problem specification is complete.

Conclusions

It has been demonstrated how custom models can be incorporated into simulation applications using generic modeling tools such as Scilab, Matlab and Excel, via the CAPE-OPEN interface standards. Together with powerful readily available unit operation implementations such as ChemSep and the unit operations that come with the simulation environment, a highly flexible modeling environment is presented. Formula based custom modeling such as in Scilab, Matlab or Excel, where thermodynamics of the simulation environment are readily available and equation solvers are integrated, allows easy implementation of text-book models and custom models. Each of the units comes with an integrated help and formula reference. Both research and chemical engineering courses may benefit from this setup.

References

1. Coulson, J.M., Richardson, J.F., Sinnot, R.K., (1991), “Chemical Engineering”. Volume 6, Design. Pergamon, Oxford, England.
2. http://www.aspentech.com/
3. COCO is freely available from http://www.cocosimulator.org/
4. http://www.chemsep.org/
5. Kooijman, H., Taylor, R., and van Baten, J.M., 2007. The ChemSep/COCO Casebook: Air Separation Unit, CACHE News, Winter.
6. van Baten, J.M., Kooijman, H. and Taylor, R., 2007. Flowsheeting for Free with COCO, CACHE News, Winter.
7. Taylor, R., Kooijman, H, van Baten, J.M., 2006. CAPE-OPEN Overhead in Distillation Modeling, AIChE presentation by Ross Taylor, San Francisco, Nov

8. Baur, R., van Baten, J., Kooijman, H. and Drewitz, W., 2006, “Cross Platform Chemical Processes”. NPT procestechnologie, September
9. van Baten, J.M., 2007. An introduction to COCO. 4th US CAPE-OPEN conference, AIChE annual meeting, Saltlake City, November.
10. Kooijman, H and Taylor, R, 2008. Chapter 3: CAPE-OPEN Flowsheet Simulations with ChemSep, part of The ChemSep book, 2nd edition.
11. Taylor, R. and van Baten, J.M., 2008. Introduction to flowsheeting with COCO and ChemSep. Available from: http://www.cocosimulator.org/down.php?dl=FlowsheetingWithCOCOandChemsep_Notes.pdf
12. Scilab is available free-of-charge from http://www.scilab.org/
13. Matlab is trademarked and licensed by Mathworks: http://www.mathworks.com/
14. Excel is trademarked and licensed by Microsoft: http://www.microsoft.com/
15. The CAPE-OPEN Laboratories Network, CO-LaN, http://www.colan.org/
16. Barrett, William M. and Yang, Jun (2005), “Development of a chemical process modeling environment based on CAPE-OPEN interface standards and the Microsoft .NET framework”. *Computers and Chemical Engineering* 30, pp. 191-201
17. Fermeglia, M., Longo, G. and Toma, L., 2009, “Computer Aided Design for Sustainable Industrial Processes: Specific Tools and Applications”. AIChE Journal, Vol. 55, No. 4, 1065.
18. Barrett, W., Yang J. and Strunjas, S., 2007, “Final Report For Verification Of The Metal Finishing Facility Pollution Prevention Tool”. EPA report, 2007.
19. Lang, Y., Malacina, A., Biegler, L., Munteanu, S., Madsen and J., Zitney, S., 2009, “Reduced Order Model Based on Principal Component Analysis for Process Simulation and Optimization”. Energy & Fuels 23, 1695
20. Morales-Rodríguez R., Gani, R., d´Echelotte, S., Vacher A, and Baudouin, O., 2008, “Use of CAPE-OPEN standards in the interoperability between modelling tools (MoT) and process simulators (Simulis® Thermodynamics and ProSimPlus)”. Chemical Engineering Research and Design, 86, 823
21. Fermeglia, M., Longo, G., and Toma, L., 2008, “COWAR: A CAPE OPEN Software Module for the Evaluation of Process Sustainability”. Environmental Progress 27, No.3
22. Breil, M., Kontogeorgis, G., von Solms, N. and Stenby, E., 2007, “CAPE-OPEN: An International Standard for Process Simulations”. Chemical Engineering, Dec., 53
23. Wankat, P.C., 2006. Separation Process Engineering (2nd edition). Prentice Hall, New Jersey, United States.
24. Scilab CAPE-OPEN Unit Operation, available from http://www.amsterchem.com/scilabunitop.html
25. Matlab CAPE-OPEN Unit Operation, available from http://www.amsterchem.com/matlabunitop.html
26. Excel CAPE-OPEN Unit Operation, available from http://www.amsterchem.com/excelunitop.html

Appendix A: using thermodynamics

To illustrate the use of thermodynamic property calculations with a simple example, we take the equations for the membrane separator, and solve for temperature so that the unit operates adiabatically. This means we have to solve the additional equation:

$$f_F h(T,P,X_F) = f_R h(T,P,X_R) + f_P h(T,P,X_P)$$

where the enthalpy terms [J/mol] are obtained from the simulation environment. A small modification to the presented Scilab script is required:

```
// compound indices:
water=1;
ethanol=2;

//parameter values:
alpha=getParameter('sepFactor');
theta=getParameter('cut');

//feed values
XF=getFeedProp(1,'fraction');
FF=getFeedProp(1,'totalFlow');
TF=getFeedProp(1,'temperature');
PF=getFeedProp(1,'pressure');

//product flows:
FR=FF/(theta+1);
FP=theta*FR;

//function to calculate compositions from XPE:
function [XP,XR]=GetCompositions(XPE)
 XP=[1-XPE,XPE];
 XRE=(FF*XF(ethanol)-FP*XPE)/FR;
 XR=[1-XRE,XRE];
endfunction

//function to solve compositions
function errorValue=CompositionFunction(XPE)
 //get the compositions given XPE:
 [XP,XR]=GetCompositions(XPE);
 //return the error in alpha
 errorValue= (XP(ethanol)/XP(water))/(XR(ethanol)/XR(water))-alpha;
endfunction

//solve for the compositions to match alpha:
XPE=fsolve(0.999,CompositionFunction);
//the compositions at solutions are then
[XP,XR]=GetCompositions(XPE);

//now that we have the composition, solve the heat balance
HF=getEquilibriumProperty('enthalpy',XF,'temperature',TF,'pressure',PF);

//function to solve the heat balance at constant P
function errorValue=HeatBalance(T)
 HR=getEquilibriumProperty('enthalpy',XR,'temperature',T,'pressure',PF);
 HP=getEquilibriumProperty('enthalpy',XP,'temperature',T,'pressure',PF);
 errorValue=HF*FF-HR*FR-HP*FP;
endfunction

//solve temperature to close the heat balance
TOUT=fsolve(TF,HeatBalance);

//set the permeate, given FP, XP, TOUT, PF
setProduct(1,FP,XP,'temperature',TOUT,'pressure',PF);

//set the retentate, given FR, XR, TOUT, PF
setProduct(2,FR,XR,'temperature',TOUT,'pressure',PF);
```

For the enthalpy calculations, we cannot get away with cheaper single phase calculations as we cannot be sure what phase(s) the feed and products are in at given temperature. In fact, we know that the feed stream is at its dew point, so we likely undergo phase changes during the solution of the heat balance. The `getEquilibriumProperty` function combines a phase equilibrium calculation at specified conditions with a property calculation.