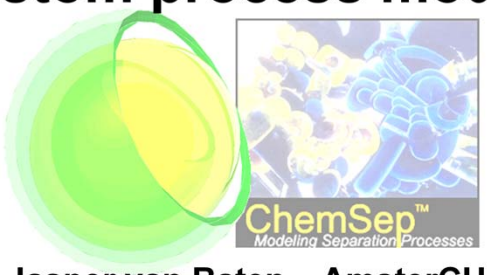
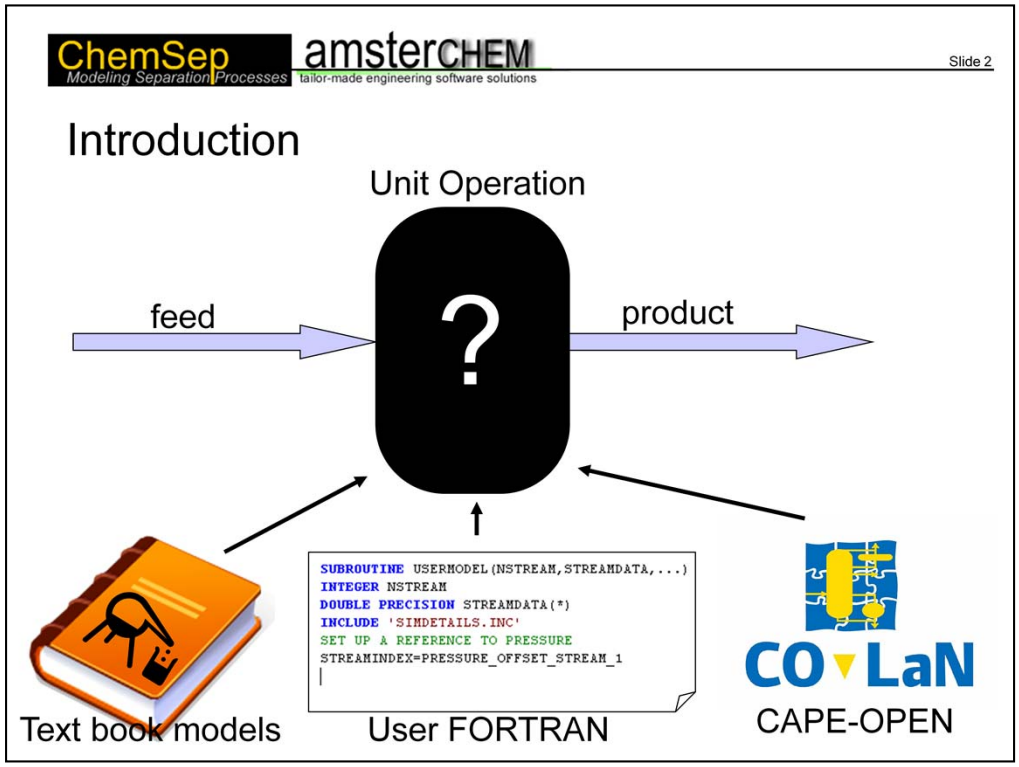
ChemSep Modeling Separation Processes amsterCHEM tailor-made engineering software solutions Slide 1

ChemSep, COCO and other modeling tools for versatility in custom process modeling



Jasper van Baten – AmsterCHEM
Ross Taylor – Clarkson University & Chemsep
Harry Kooijman – Clarkson University & Chemsep

Good morning; my name is Jasper van Baten. I am an independent software consultant from AmsterCHEM, and the principle author of the COCO steady state flowsheet simulation software. COCO is shipped with the separation process software ChemSep, by Ross Taylor and Harry Kooijman; some of you may be familiair with this software. Today's presentation will be about flexibility of custom modelling in steady state flowsheeting. Such flexibility is handy for implementation of custom models in research, but the focus today is somewhat more on teaching purposes.



Steady state flowsheets are used for process design and process analysis. Several commercial packages are available. All of them operate on the same principles. A complete process flowsheet, or a part thereof, is represented by a graph in which the streams are drawn between blocks that represent process equipment, such as reactors, separation columns, pumps, etc. The principle is illustrated here. The unit operation is connected to one or more feed streams, and one or more product streams. The feed streams may come from upstream unit operations, and the products may be feeds to upstream unit operations.

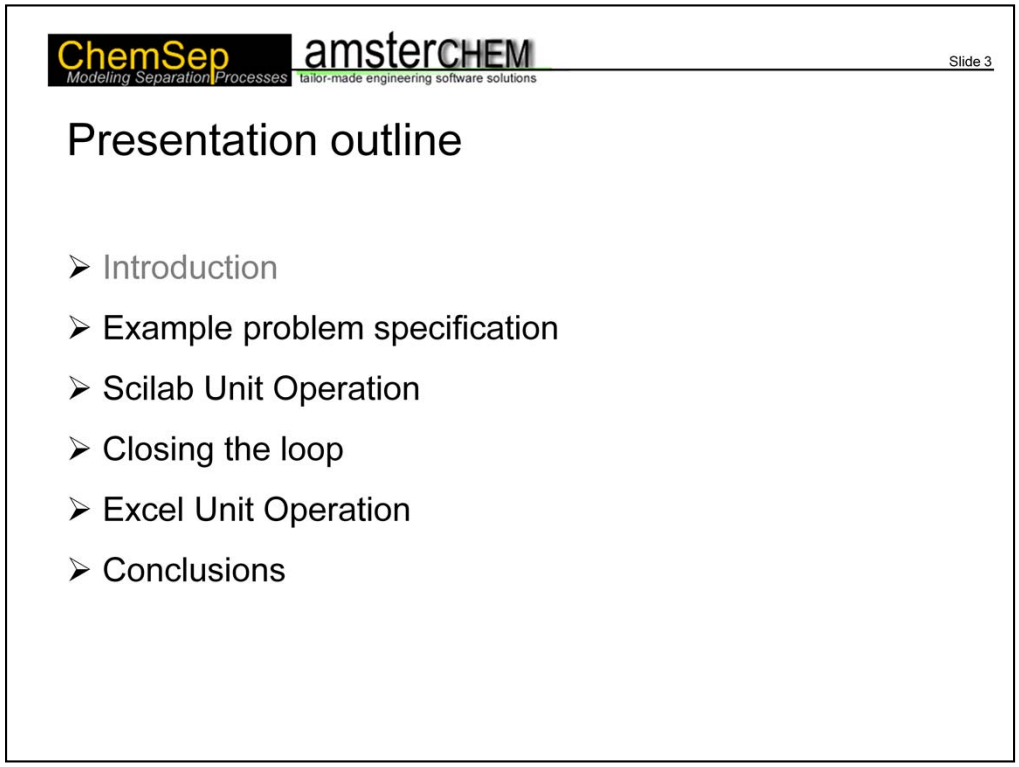
Each simulation environment comes with a number of pre-defined unit operations; each simulation environment will have its own models for compressors, reactors, flash drums, and what not. To the end-user, what happens inside such a unit operation is not always clear; the unit operation operates as a black-box. Even if well-documented, the end-user can generally not modify the equations that are solved. In addition, it is impossible to capture all possible models for process equipment in standard models, and even extensive sets of standard models have their limitations.

Especially in teaching this is not uncommon. For example, typically during a chemical engineering course, the parallel between a plug flow reactor and tanks-in-series are drawn. The plug flow reactor will typically be part of the standard set of models, the tanks-in-series model is generally not. Simple heat exchanger models are generally available, but it is hard to come up with a standard model for the most common heat exchanger of the shell-and-tube type. All one needs to do to realize

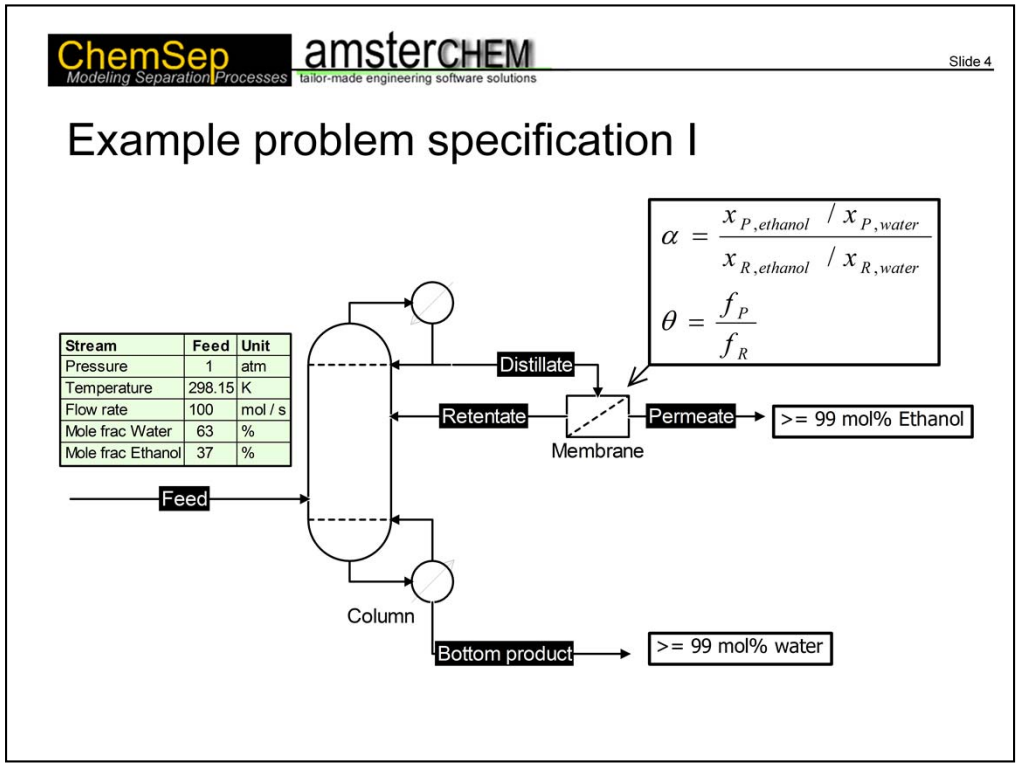
that the possibility of different models is almost endless, is to open a chemical engineering text book and look at the variety of models and equipment presented. In this presentation, we will use a simple membrane separator for example. Standard models are generally not available in simulation software, because of the lack of standard representations of available data on performance of such devices.

So – for teaching purposes it would be handy to have a convenient way to enter text book equations that describe process equipment. All simulation environments are of course equipped with a mechanism to enter custom models. The classic approach is that the user has to write some FORTRAN code that is linked along with the simulator code. Writing such code requires programming skills, and is typically specific for the simulation environment at hand. If you want to use the same model in a different simulation environment, you will need to go through the exercise of writing such code again; not very re-usable and not very user friendly.

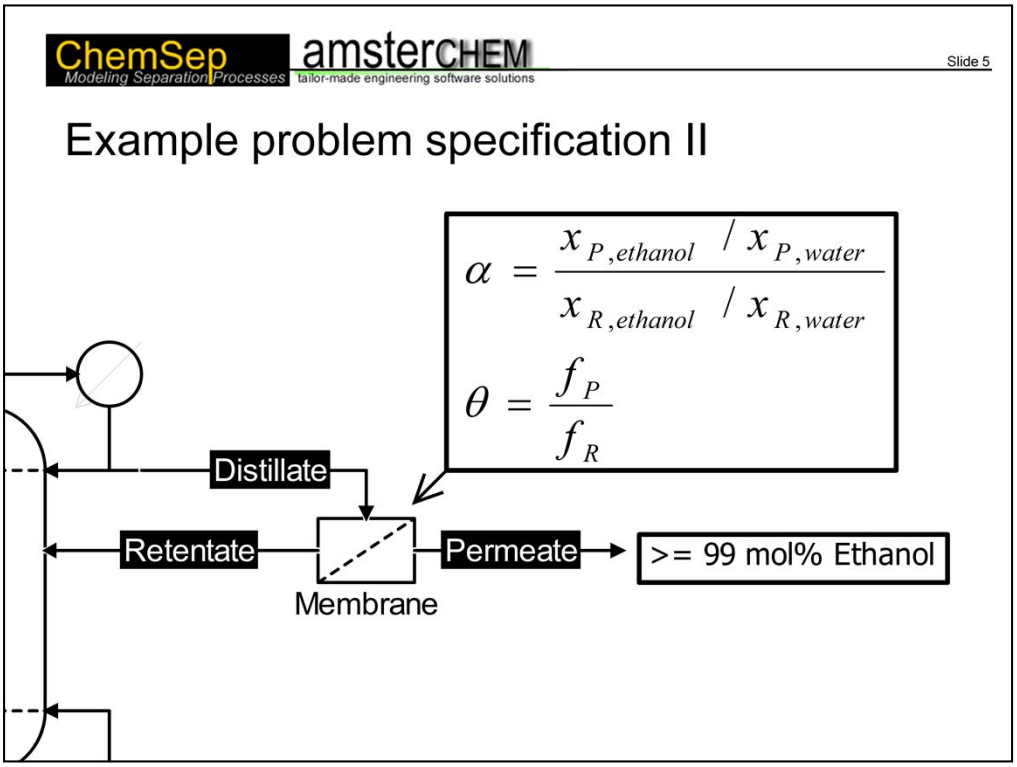
The re-usability can however be resolved. An open software standard exists that allows for exchange of chemical engineering and process models and thermodynamic servers. It is called CAPE-OPEN. Models that adhere to the CAPE-OPEN standard can be re-used in different simulation environments. Today we will present three CAPE-OPEN based custom modeling solutions that the end-user may be more familiar with. One can enter the model equations in Microsoft Excel, in Matlab, or – in case a Matlab license is not available – in Scilab, which is similar to Matlab. We will present these solutions using a simple step-by-step example. The simulation environment we use is COCO, and for separations we will depend on ChemSep.



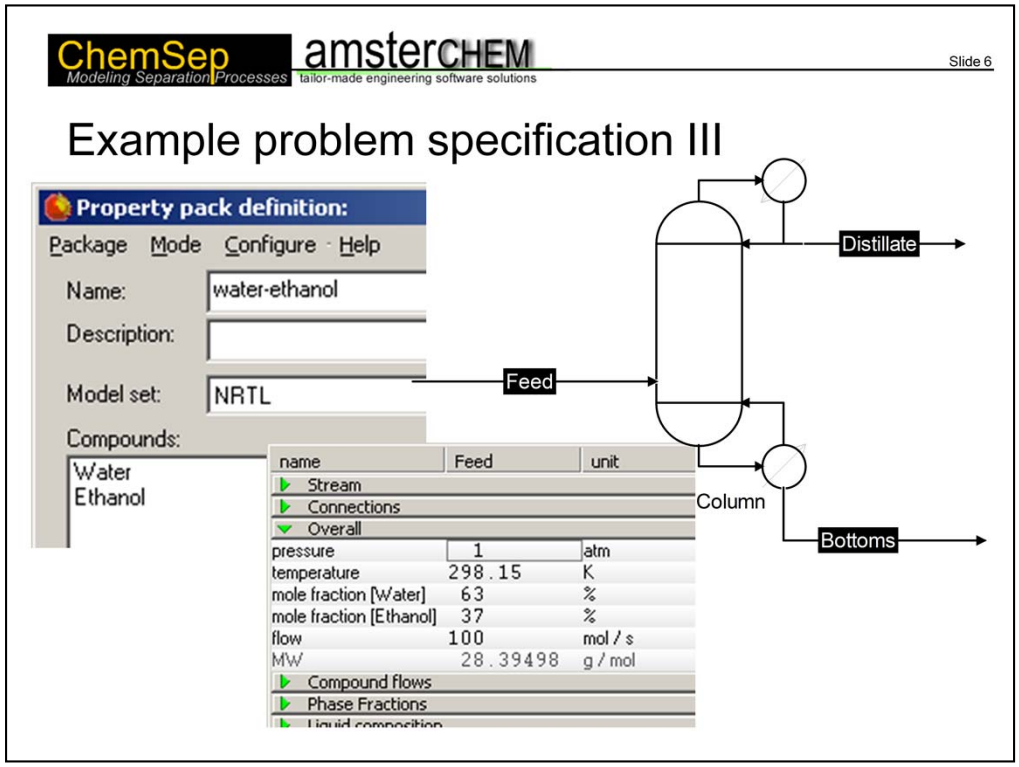
Thus, after this short introduction, the presenation will proceed as follows. First the example problem will be introduced. Then we show how to tackle the problem by typing the equations in Scilab script; we will quickly show how to incorporate this into the flowsheet and that it actually leads to a solution. Next, we will show how to do it using Excel instead of Scilab. The Matlab solution will not be shown; it is very similar to the Scilab solution.



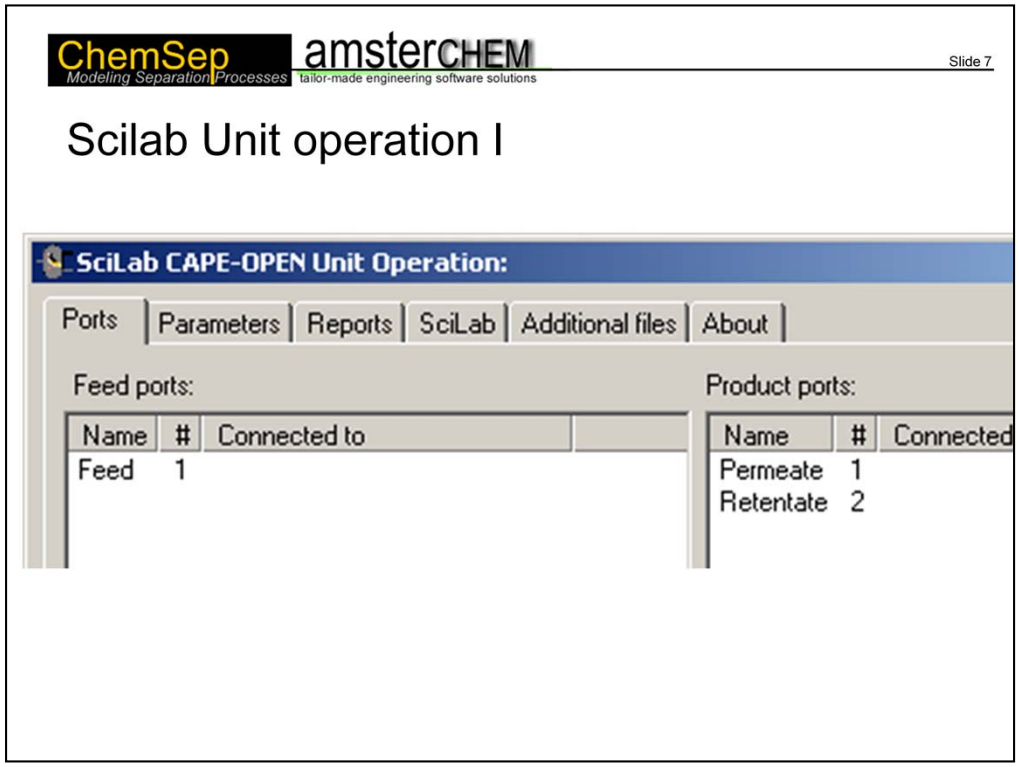
The problem to be solved for this example involves the separation of water and ethanol. At about 90% ethanol, this system has an azeotrope, so direct separation using a single distillation column is not possible. We can however obtain the desired separation specifications if we take the columns top-product and finish its separation using a membrane that will selectively allow methanol to pass. The nearly pure ethanol is then the final product, and the mixture of water and ethanol will be fed back to the distillation column.



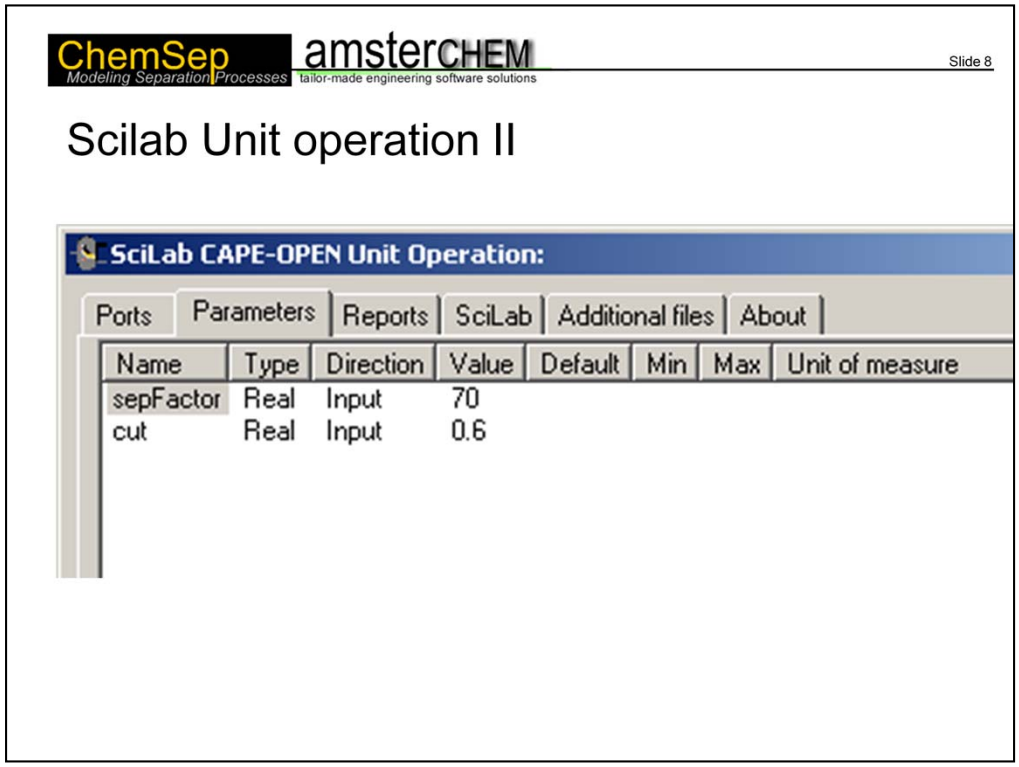
The membrane model is very simple: the separation is defined by two parameters: the separation factor alpha, which defines the product purity, and the membrane cut theta, which defines the amount of flow passing through the membrane. This model has been selected for its simplicity so that the demonstration fits in the scope of this presentation. This simplicity does not reflect on the possibility of the modelling tools; the equations there are not limited by complexity, as long as you can solve them with the tools at hand. For Excel for example this means that as long as you can get the Excel solver to solve your equations, you can make the equations as complex as you want.



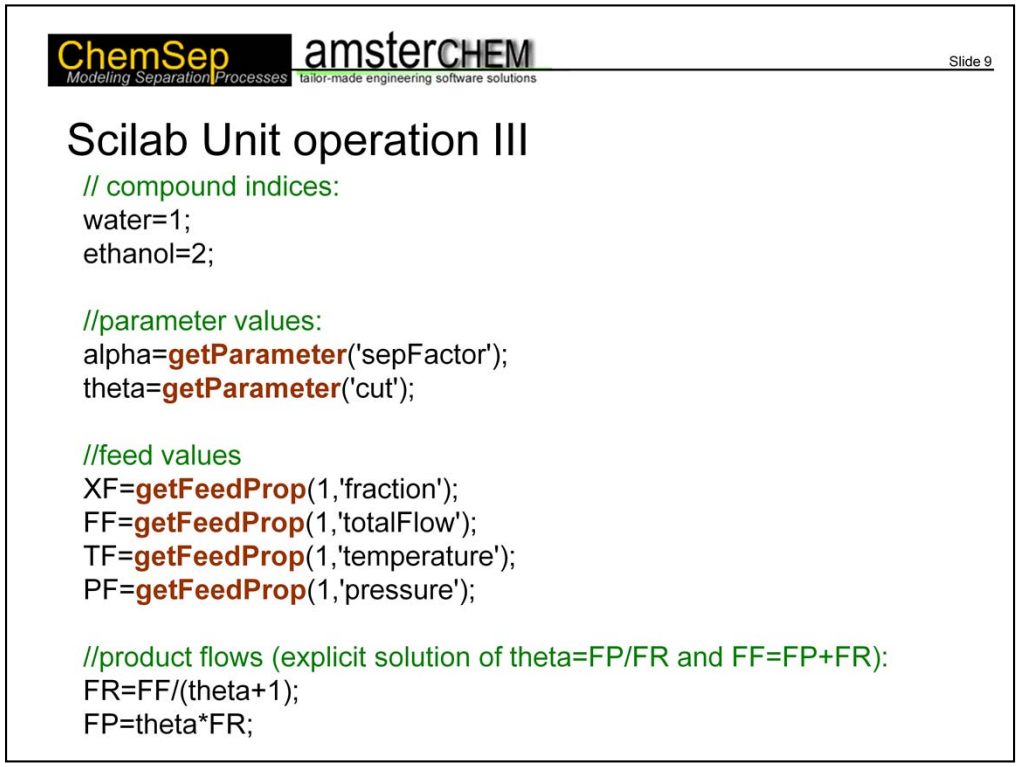
Also the setup of the problem we keep simple. We start the COFE flowsheet simulator, and start a new flowsheet document. To insert thermo, we create a new property package, add the compounds water and ethanol, and choose the default NRTL model set. We insert the feed into the flowsheet, and specify the feed conditions. We insert a new ChemSep column, and accept all its defaults for simple distillation. We connect the feed to the column and we insert the distillate and bottom product streams that we also connect to the column.



For our custom model with Scilab, we enter a new Scilab Unit Operation. Scilab is a generic script based modelling tool, much like Matlab. It knows nothing of chemical engineering per se, so some functionality has been added to make it operate as a unit operation. This functionality includes the ability to access thermodynamic calculations (property calculations, phase equilibrium calculations). But also we need to tell it something about the ports and parameters we want to expose. Ports are the connection points for streams; only after creation of ports will the user be able to connect streams to the Scilab Unit Operation. A GUI is provided for this, so we double click to edit the unit. On the Ports tab, we create 1 feed named Feed, and 2 products, named Permeate and Retentate. We see that the product streams have indices 1 and 2, which we can use from the Scilab script to access the products.



Our model also has two parameters: the separation factor and the permeate cut, as shown on a previous slide. These we can enter on the Parameters tab.



All that remains to be done is enter the script that calculates the unit. The model has a simple analytic solution, but we will not use it here. The reason we are not using it is because we want to demonstrate that we can use the solvers built-in to the generic modelling tools (in this case Scilab) to solve equations for us; also integration tools are generally available.

The script is directly entered into the Scilab Unit Operation, and can be tested at will – presuming that the feeds and products to the units have been connected. We have created the ports, so we can connect the streams at this moment, and we proceed to enter the script.

For convenience, we define the indices to water and ethanol. We can do this in a more dynamic way by asking for the list of compounds, but for demonstration purposes this will suffice.

We then initialize the script by assigning the feed values and parameter values to local variables.

The solution for the product flows is explicit, and requires solving the equation for permeation cut theta, as shown here in the bottom.

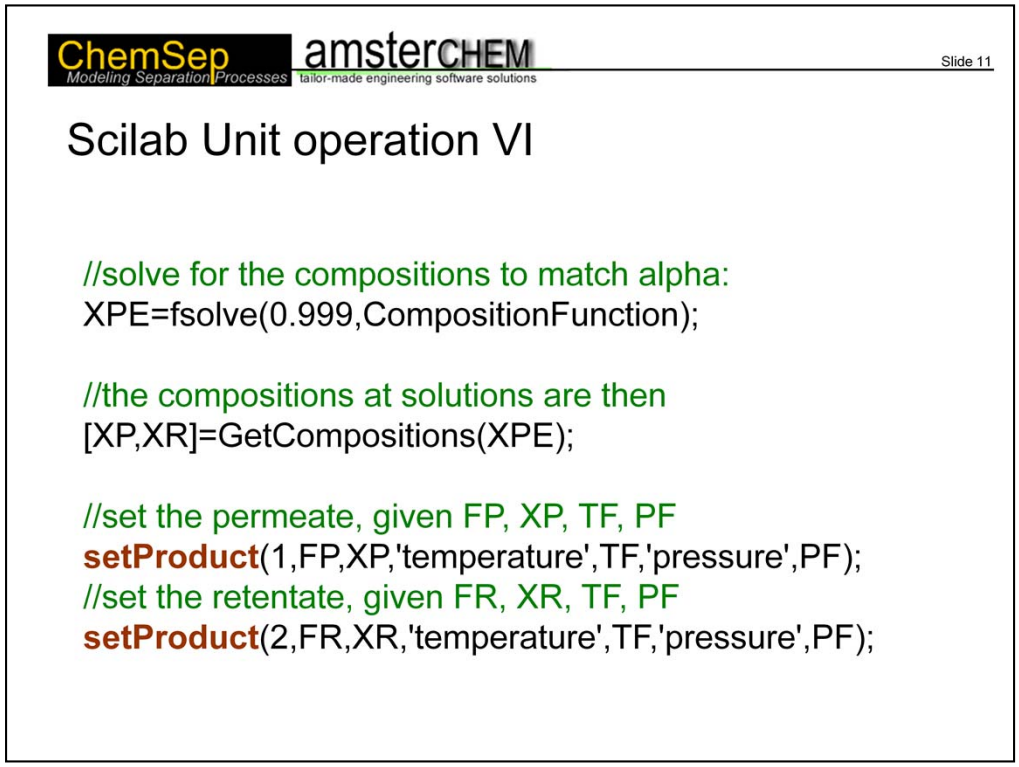
Scilab Unit operation IV

```
//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
```

We then define two functions; the first one, GetCompositions, calculates the compositions of the permeate and retentate products, given the ethanol composition in the permeate, which we will use as an iteration variable. This function is based on the fact that the mole fractions of both products add up to unity. We can then use either the ethanol or water component balance to close the equations; the ethanol balance is used here.

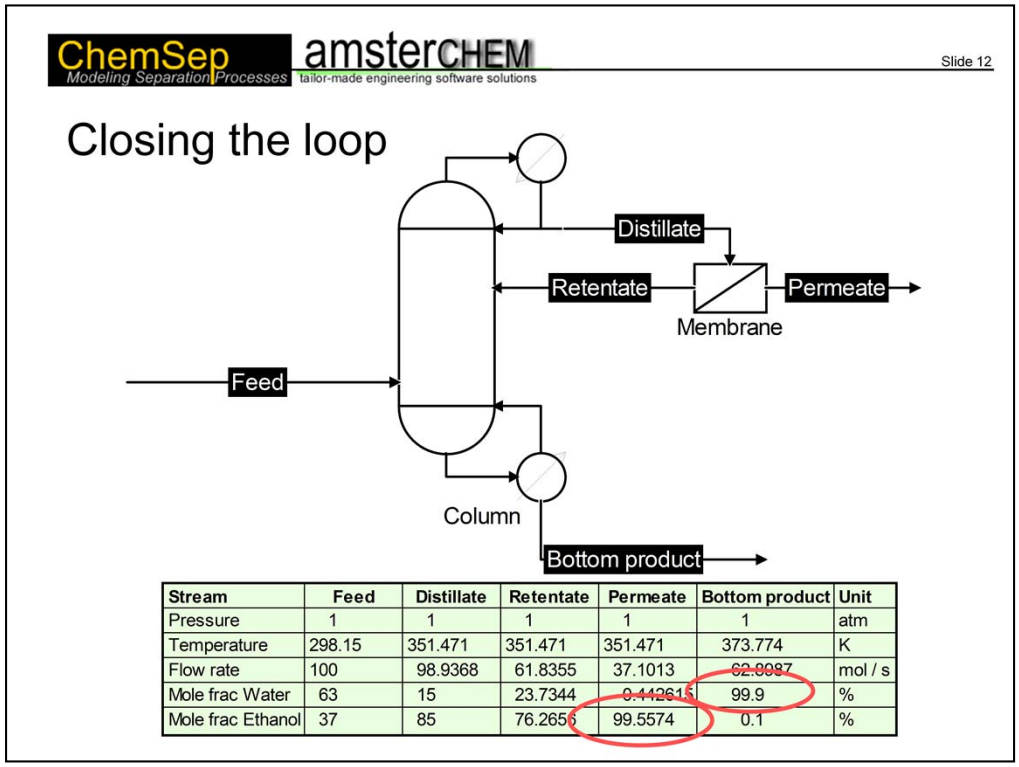
The second function, CompositionFunction, is the equation we solve: we iterate over the ethanol composition in the permeate; the other compositions follow from the GetCompositions function. This function returns the actual value of the separation factor minus the specified value. Hence, the problem is solved in for a value of permeate ethanol mole fraction XPE for which this function returns zero.



To finish of, we use the Scilab function fsolve to solve the CompositionFunction defined on the previous slide to be zero, by varying the permeate ethanol mole fraction with an initial guess of 0.999. The solution value of the permeate ethanol mole fraction gives us all compositions, given the GetCompositions function on the previous slide. We then define the products, using temperature, pressure and the calculated flows and compositions.

We always need to specify flows and compositions for a product. Instead of pressure and temperature, we may however choose other variables to fix the phase equililbrium; for example pressure and enthalpy is a common combination for adiabatic units or units with a specified heat duty.

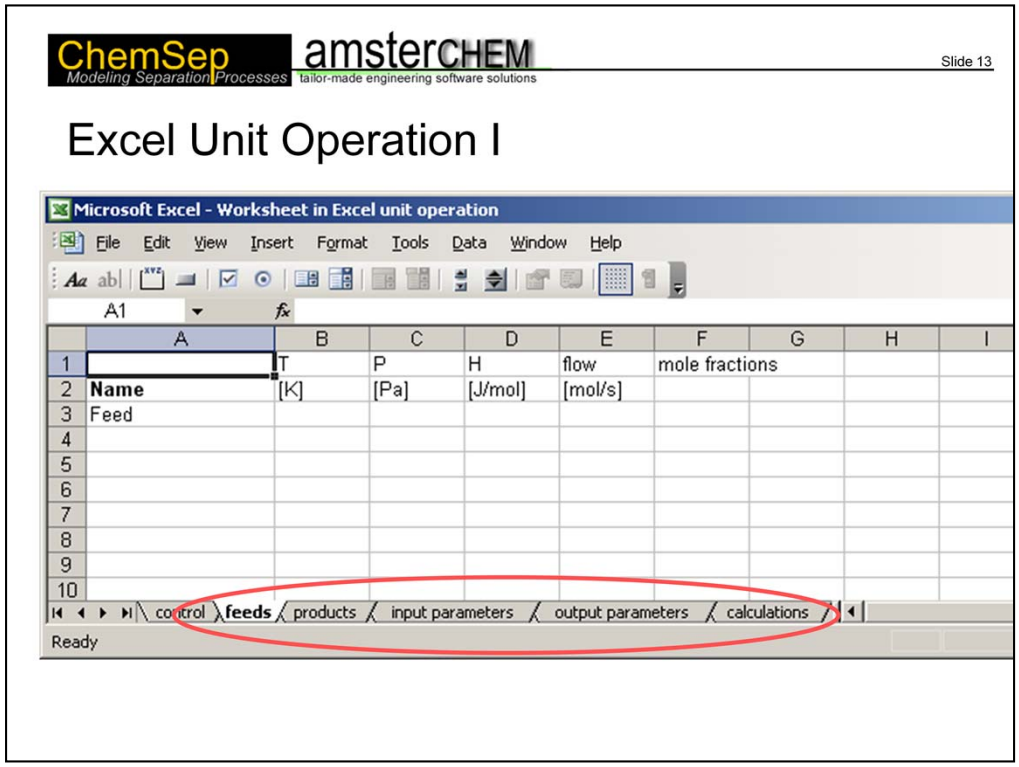
This particular example does not use any thermodynamic calculations. Such calculations are however available, and the extended abstract provides a simple example. The extended abstract is on the proceedings CD, on the cocosimulator web site, or available on request.



Stream	Feed	Distillate	Retentate	Permeate	Bottom product	Unit
Pressure	1	1	1	1	1	atm
Temperature	298.15	351.471	351.471	351.471	373.774	K
Flow rate	100	98.9368	61.8355	37.1013	62.8987	mol / s
Mole frac Water	63	15	23.7344	0.442615	99.9	%
Mole frac Ethanol	37	85	76.2656	99.5574	0.1	%

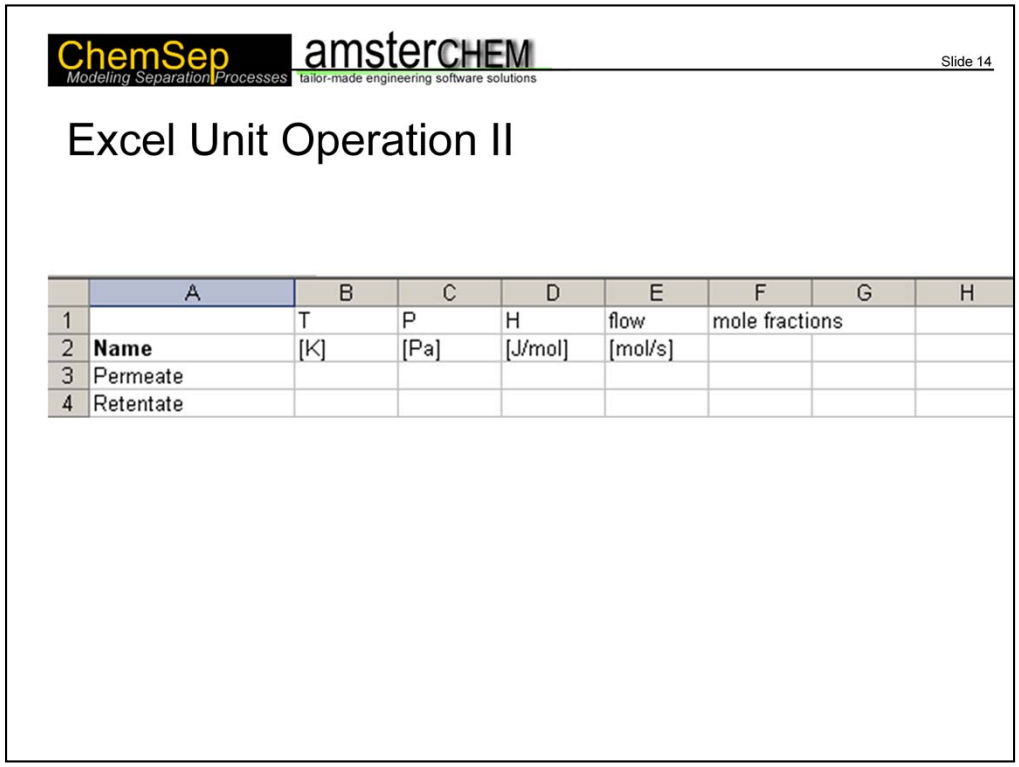
We have seen that we can enter our equations without any significant programming. Two scripted functions have been provided, in order to use the Scilab solver. As mentioned, the problem has a simple analytic solution; the script would have fitted on a single slide if we would have used that.

With the ports, the parameters and the script defined, our unit is complete. To meet the bottom product specification, we change the number of stages in the column to 20, move the feed stage to stage 17. We can run the test problem, and inspect the concentrations of the Retentate stream. We see that stage 6 is a good place to feed the retentate back into the column. We solve the flowsheet, and notice that we easily have obtained all our problem target specifications.



Even though in Scilab, we could pretty much enter the equation in a functional form and be done, some users may prefer to enter the equations in Excel. The principles that hold are similar. We remove the Scilab unit operation from our example flowsheet, and replace it by a new empty Excel Unit Operation. If we edit this, Excel appears. A number of worksheets are present in the Excel workbook that pops up. The meanings of these slides are specific to the unit operation, and all we need to do is fill them out. We start on the feeds sheet by defining our Feed. It has the name; Feed.

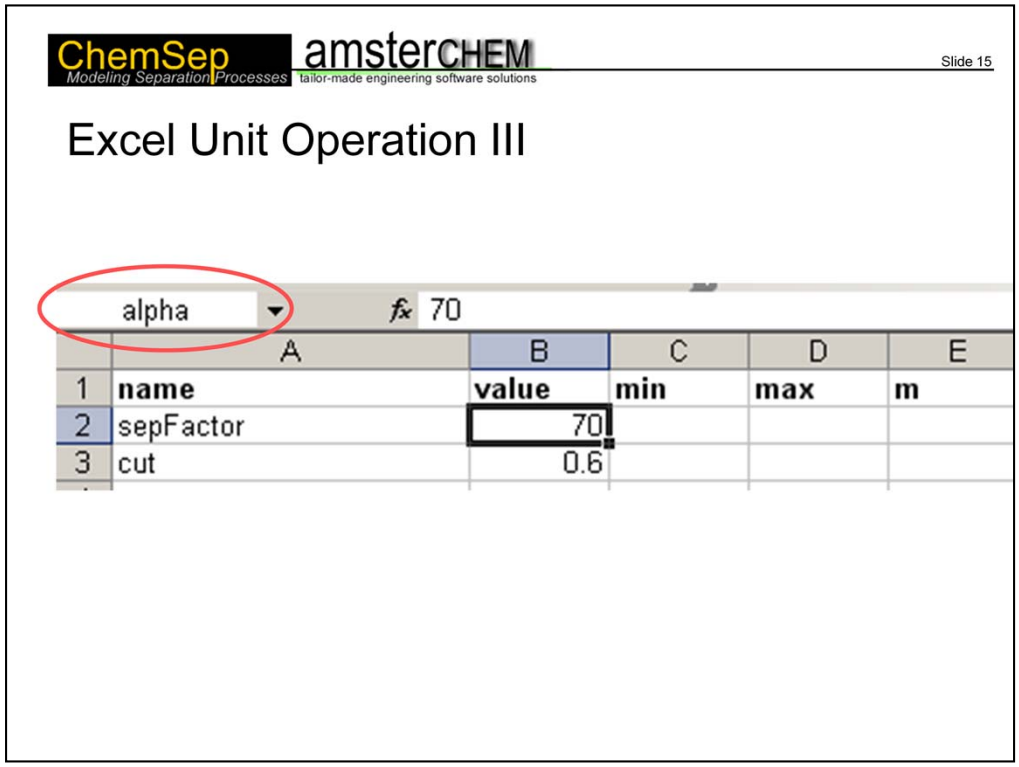
As soon as we connect the feed, the values in the feed stream will appear in this sheet.



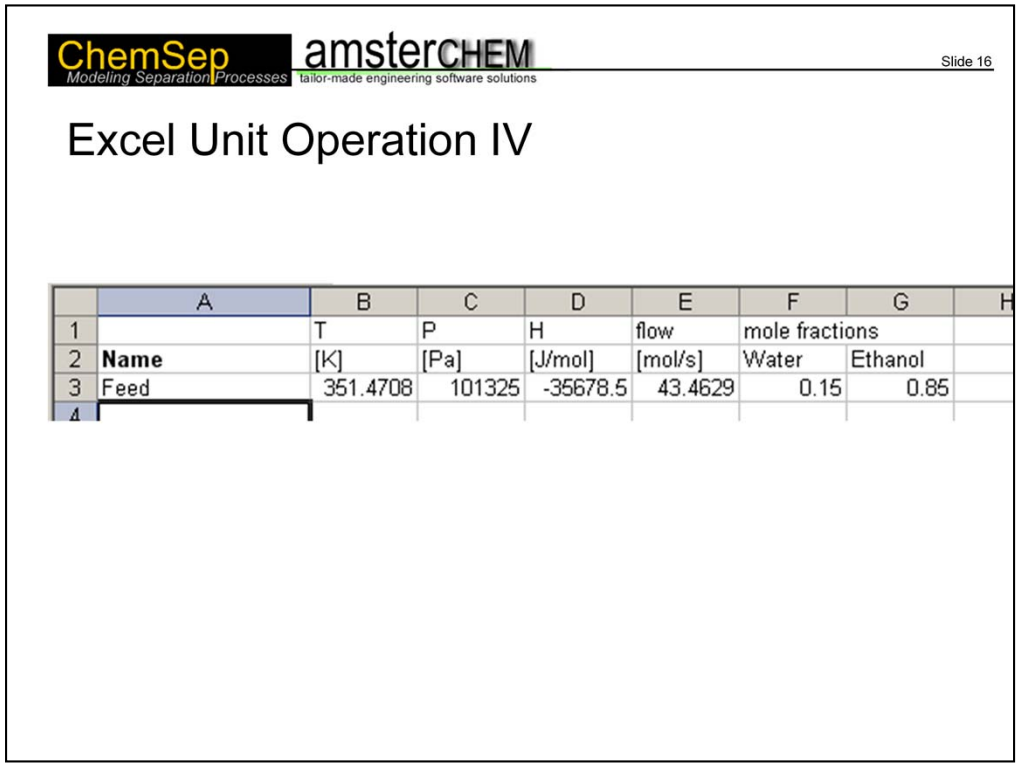
	A	B	C	D	E	F	G	H
1		T	P	H	flow	mole fractions		
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]			
3	Permeate							
4	Retentate							

Similarly, on the products sheet, we define our product ports, they are named Permeate and Retentate.

The values in this sheet we need to provide by entering our calculations. We will provide a formula for temperature, pressure, flows and mole fractions of both the product streams.



On the parameters sheet, we define the parameters for separation factor and permate cut. For use in equations, I have added a name to each of the values; we can refer to the separation factor as alpha, and to the permeate cut as theta. If the flowsheet user changes the parameter values, the values in this sheet will automatically be updated.



After connecting the ports, the feed sheet looks like these: not only are the compounds known at this point, but also all feed values have been filled out. We can refer to these values in our calculations.

ChemSep Modeling Separation Processes amsterCHEM tailor-made engineering software solutions Slide 17

Excel Unit Operation V

	A	B	C	
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 t
5	X p ethanol	0.999	mol/mol	Will be solve
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		

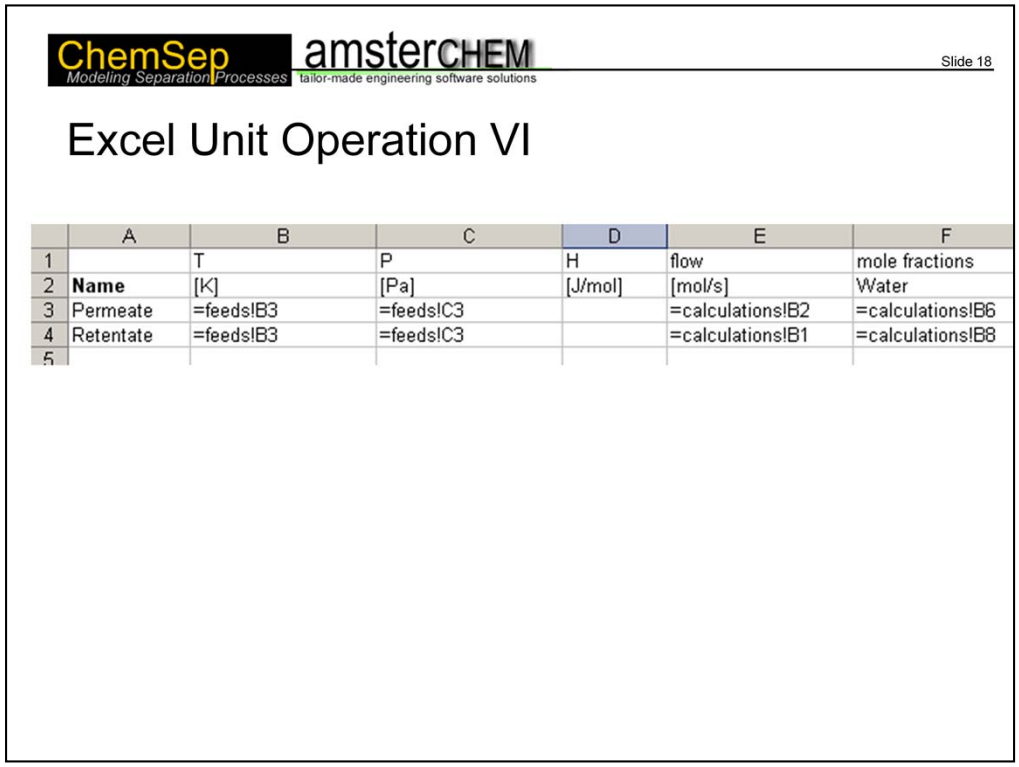
The calculations are analogous to what we had done in Scilab. The flows are calculated explicitly in the rows 1 and 2.

In row 4 we define the initial guess for our solver problem. The actual value being solved for is in row 5. Rows 6, 7 and 8 list the remainder of the product compositions as a function of the iteration variable.

Based on these values, we can calculate the separation factor, as on row 10. The error is given by calculated and specified separation factor, and we set up the Excel solver to change the iteration variable on row 5 for the error in row 11 to be zero.

We can save the solver scenario, and on the control sheet we can tell the Excel Unit Operation to use this solver scenario when solving the unit; we can also state that the initial guess should be taken from row 4.

For each Excel Unit Operation, multiple solver scenarios can be specified that way, and they will be solved in the order in which they are specified.



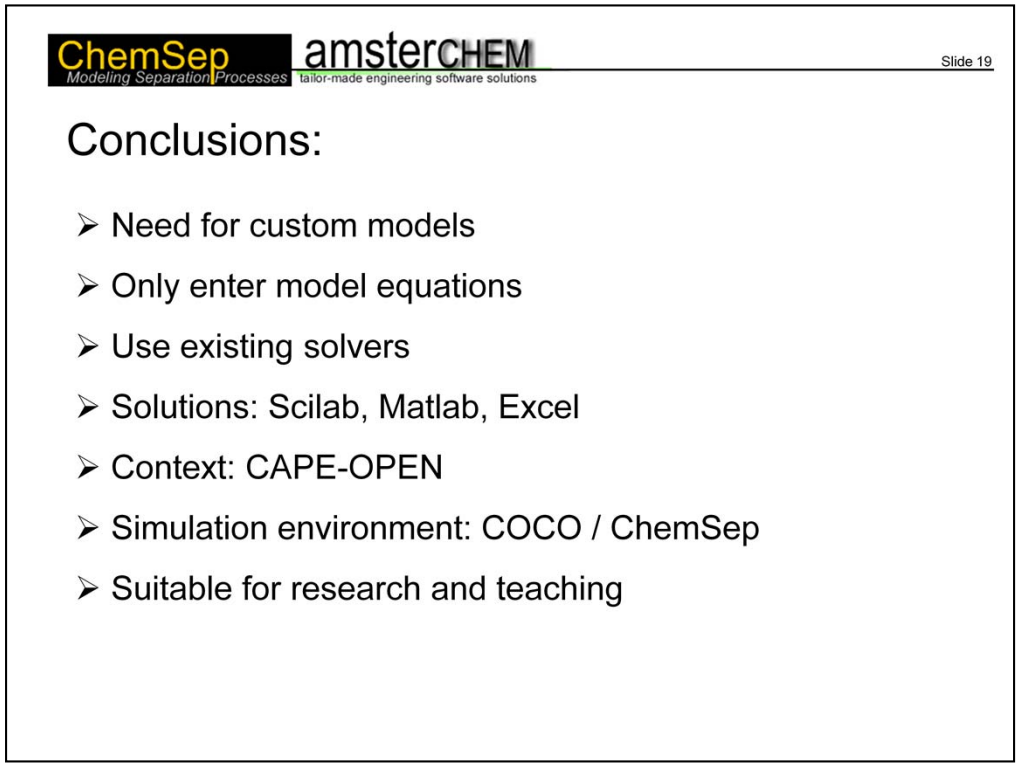
	A	B	C	D	E	F
1		T	P	H	flow	mole fractions
2	**Name**	[K]	[Pa]	[J/mol]	[mol/s]	Water
3	Permeate	=feeds!B3	=feeds!C3		=calculations!B2	=calculations!B6
4	Retentate	=feeds!B3	=feeds!C3		=calculations!B1	=calculations!B8
5						

Now all that remains, is to refer to the solution on the products page.

The pressure and temperature for the products directly refer to the feed sheet.

The flows and compositions of the products refer to the solution of the equations we have just entered.

This is all that is required in Excel to define our unit operation.



We have identified the need for custom unit operation models in flowsheeting; a text book model has been picked for demonstration purposes with the focus on teaching.

We do not want to be bothered by programming; essentially we want to only enter the model equations.

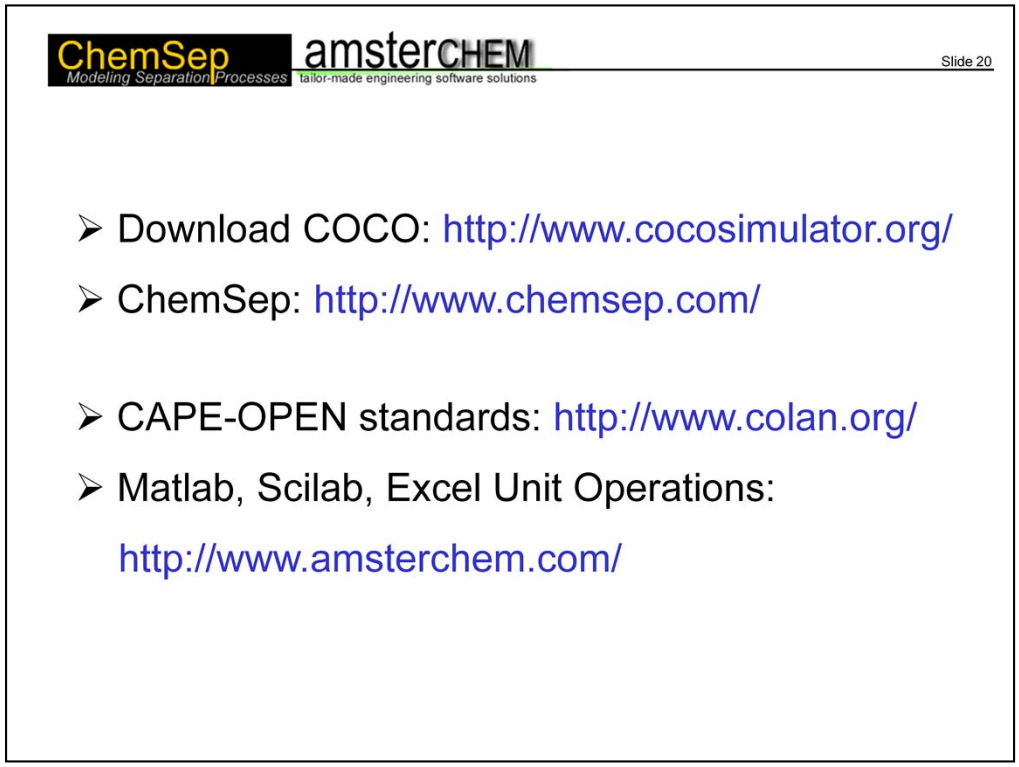
We do not want to write our own solvers; we want to re-use the solvers at hand.

We have presented a solution methodology that uses Scilab. The Matlab methodology is not presented here, as to avoid repetition. The Scilab example translated 1:1 to the Matlab environment. We can also use Excel, which has a bit of a different approach to enter the equations.

The solution approaches are implemented in a CAPE-OPEN context, and can therefore be reused in simulation environments with CAPE-OPEN support.

We have shown the application of the presented solution methodologies in the COCO simulator, using ChemSep to model the distillation column.

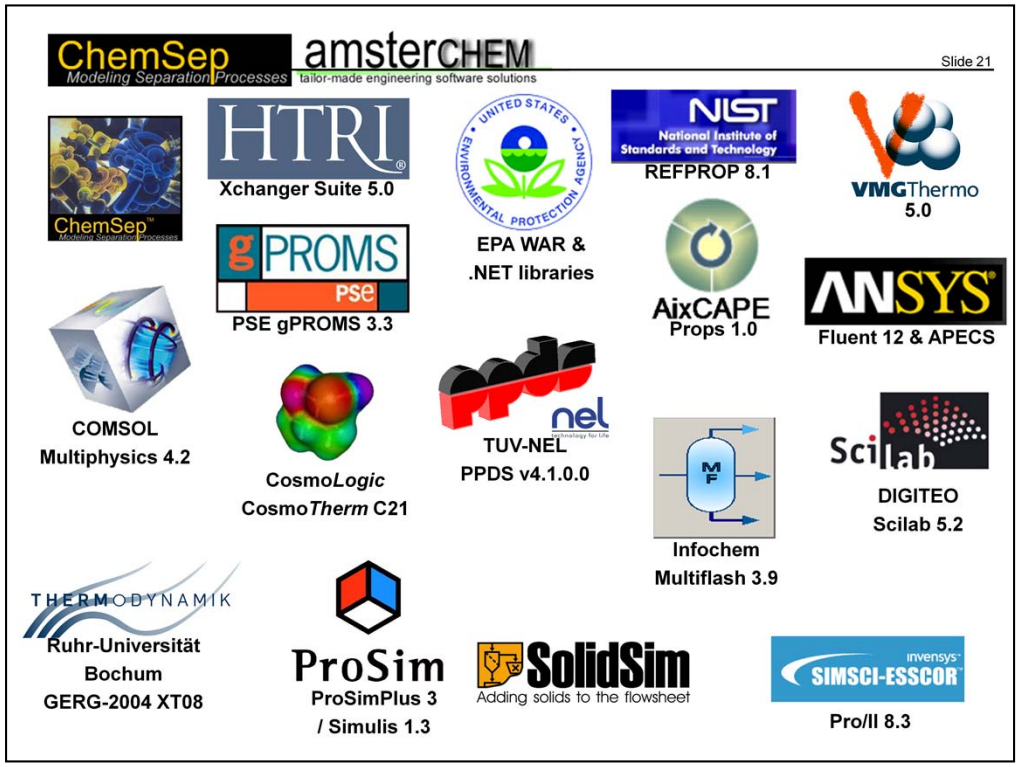
We believe that the demonstrated approach is very suitable for teaching purposes and for research.



COCO is freely available for download from cocosimulator.org. This includes the free LITE version of ChemSep. The commercial version has been avaiable to CACHE members for more than 15 years already. It is available from chemsep.com.

The CAPE-OPEN standards are free and open, and available from the CAPE-OPEN Laboratories Network: colan.org.

The Matlab, Scilab and Excel Unit Operations presented here are free for non-commercial use, and available from Amsterchem.com.



Finally, Jasper van Baten would like to thank all companies that generously provided software and licenses to AmsterCHEM for CAPE-OPEN development and testing.