# DWSIM - Open Source Chemical Process Simulator
# User Guide

**Version 8.8.3**

November 2024

## License

DWSIM is released under the GNU General Public License (GPL) version 3.

## Contact Information

**Developer: Daniel Wagner Oliveira de Medeiros**

Offical Website: `https://dwsim.org`

Source Code: `http://github.com/DanWBR/dwsim`

Support: `https://github.com/DanWBR/dwsim/discussions`

# Contents

# Part I.

# Introduction

This document gives a detailed description about how to setup, run, modify and view results of a basic process simulation in DWSIM. The document is organized according to the sequence of execution of a simulation. Each step/task is explained with the help of images and descriptions of the associated windows.

DWSIM has two Graphical User Interfaces: Classic UI and Cross-Platform UI.

- The Classic UI is based on the **Windows Forms** graphical class library (link), which has been used since the initial versions of DWSIM. The Windows Forms graphical library was created for Windows applications. A port of this library exists for Linux, but it has some issues and, since it mimics the Windows look-and-feel, it doesn't look "native" on systems other than Windows:

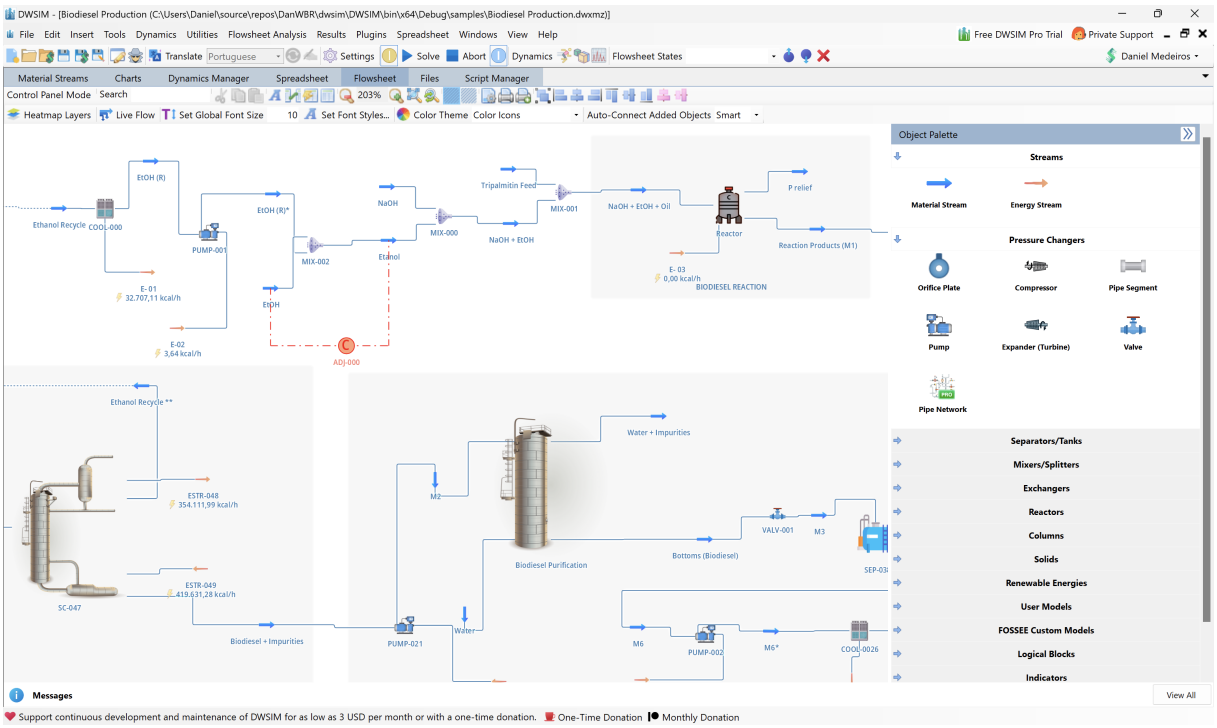



Figure 1: Classic UI on Windows 10 + .NET Framework 4.8.

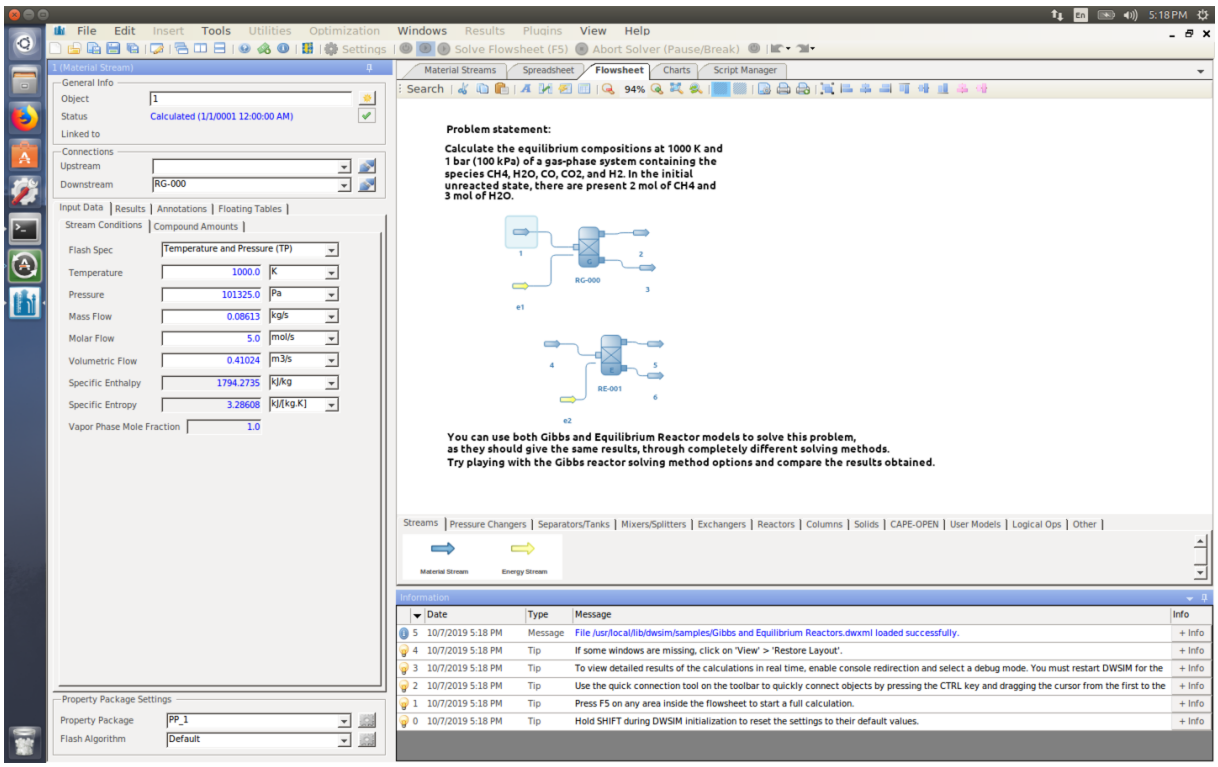



Figure 2: Classic UI on Ubuntu Linux 16.04 + Mono 5.10.

➙ The Cross-Platform UI is based on a library called **Eto.Forms** (link), which is a cross-platform graphical class library, supporting Windows, Linux, macOS and some mobile systems. This UI was created from scratch using the C# language, and looks native when executed on systems other than Windows, since it runs under the *GTK* (link) backend on Linux and *Cocoa* (link) on macOS. On Windows, the Cross-Platform UI runs under the Windows Presentation Foundation backend, though it is recommended to use the Classic UI on Windows systems, since it is more stable and reliable due to the fact that it has been in development for much more time.

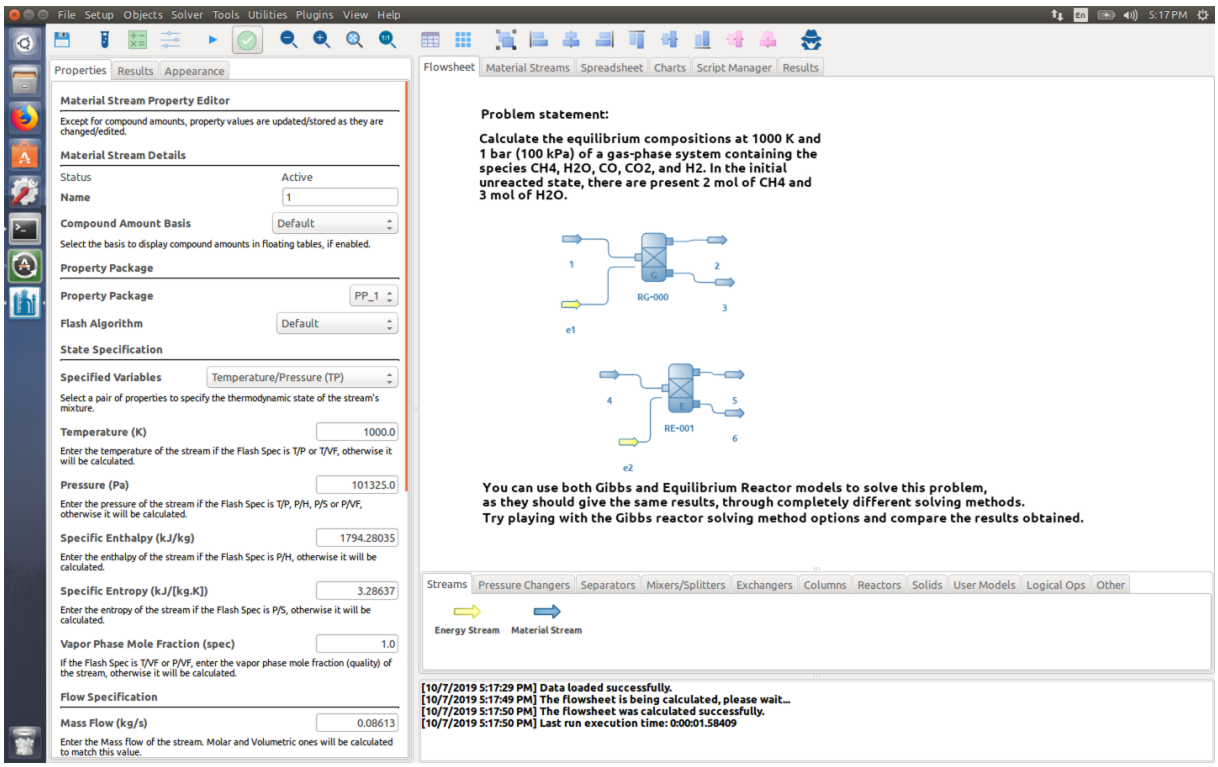



Figure 3: Cross-Platform UI on Linux 16.04 + Mono 5.10 (GTK backend).

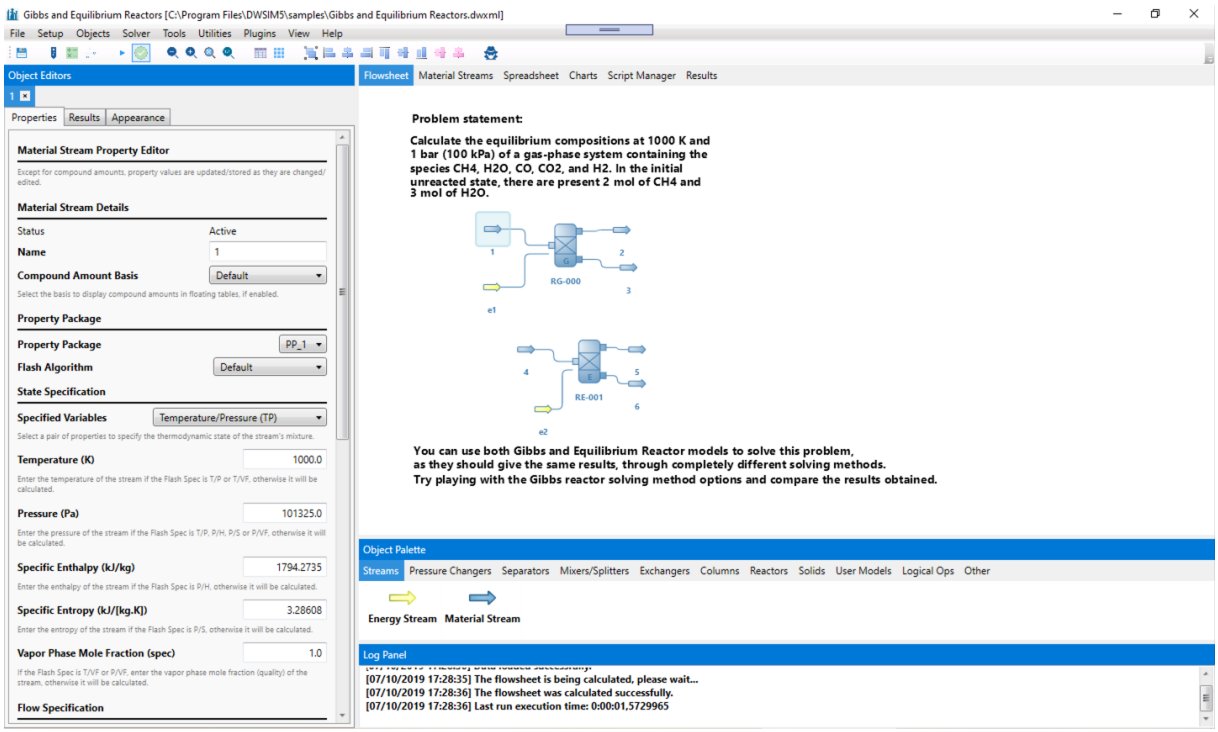



Figure 4: Cross-Platform UI on Windows 10 (WPF backend).

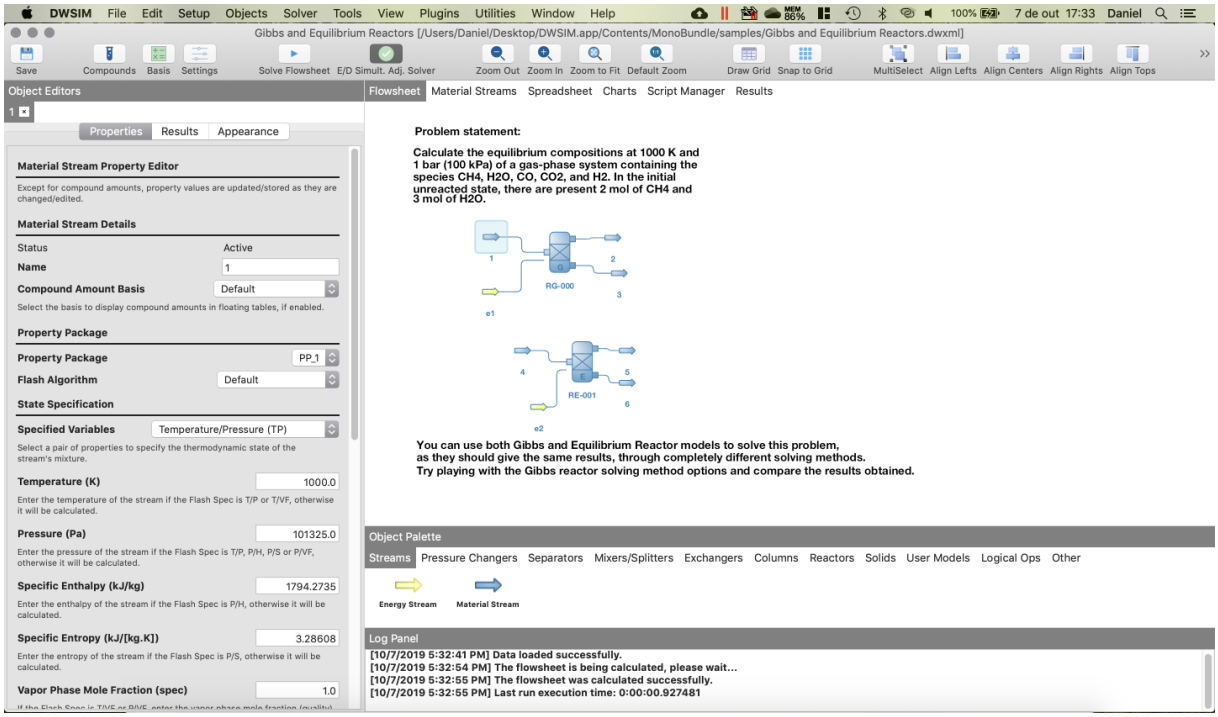



Figure 5: Cross-Platform UI on macOS Mojave (Cocoa backend).

# Part II.

# Classic User Interface (Classic UI)

## 1. Welcome Screen

When DWSIM is opened, the welcome screen is shown (Figure 40):

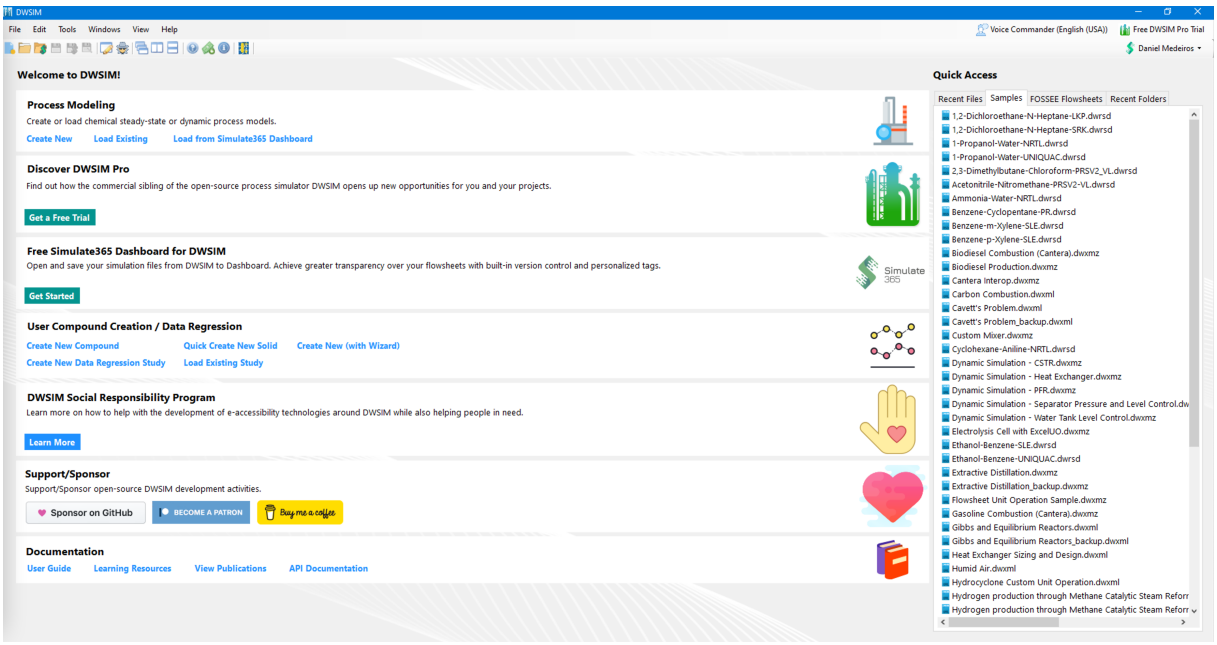



Figure 6: DWSIM's welcome screen.

The welcome screen provides the user with shortcuts to open existing simulations, create new ones, create new compound creator and data regression cases and open the samples folder.

The following items are displayed on DWSIM's main window:

- ***Menu bar***, with buttons to open/save/create simulations, component creator and data regression cases, configure the active simulation, general preferences, launch tools, configure the child windows view mode, etc.;
- ***Button strip,*** to open, save and create new steady-state simulations, component creator and data regression cases.

There are various ways to access the most commonly operations with simulation files and component creator/data regression cases - open, save and create. In the next sections you will be guided through some necessary steps to create and configure a steady-state simulation, a compound creator and/or a data regression case.

# 2. Simulation

## 2.1. User Interface

The "Create a new steady-state simulation" button in the welcome window can be used to create a new simulation. After the simulation is created, the **configuration window** (Figure 40) is shown. The simulation configuration interface consists in a tabbed window:

- ***Compounds*** - Add or remove compounds to/from the simulation and petroleum fractions (pseudo-compounents) utilities.
- ***Basis*** - Property Package configuration, phase equilibrium flash algorithm selection and other advanced thermodynamic model settings.
- ***System of Units*** - Management of Systems of Units.
- ***Behavior*** - Options to control certain behaviors of the flowsheeting environment.
- ***Object Properties*** - Definition of objects properties to be shown on flowsheet floating tables.
- ***Information*** - Simulation info (title, author and description), number formatting and password settings.

## 2.2. Configuration

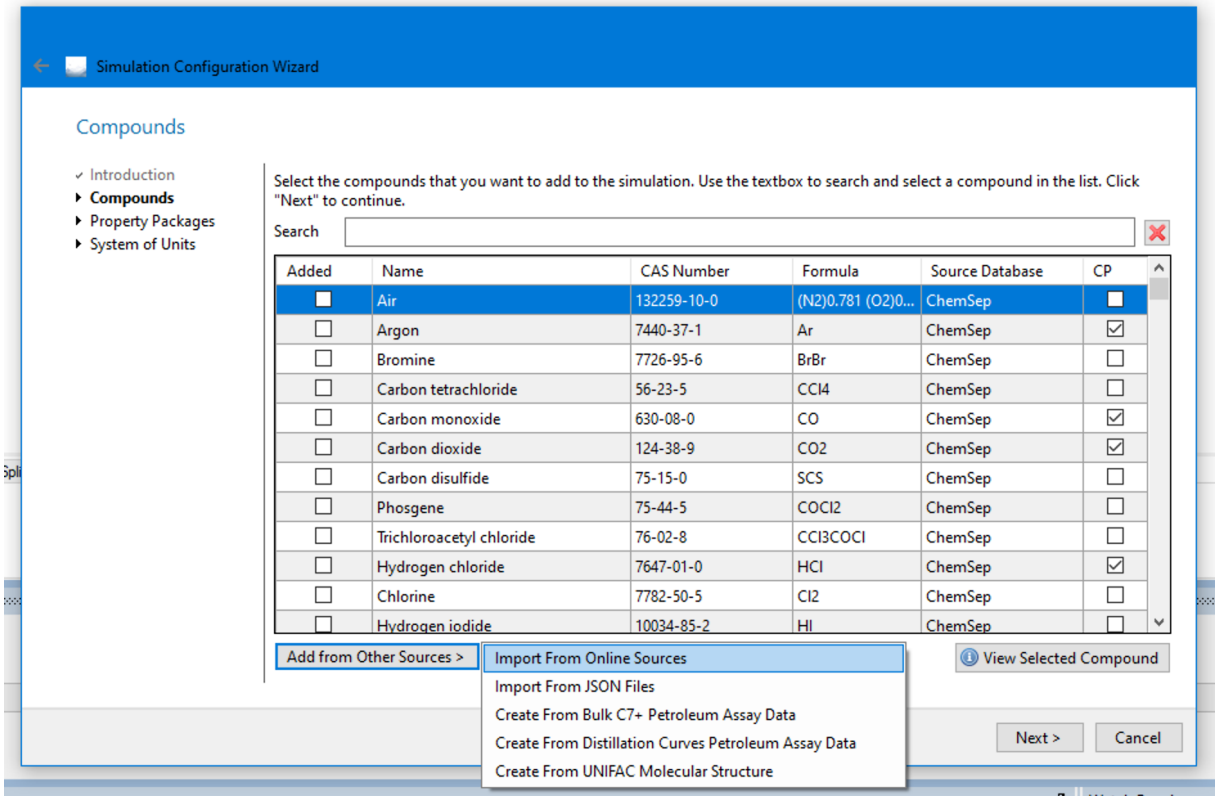



Figure 7: Simulation Configuration Wizard.

Since DWSIM 3.3, a new Simulation Configuration Wizard (Figure 40) is opened as soon as a new simulation is created, and will display the interfaces described in the following sections in a more streamlined way. The older simulation configuration window can be accessed anytime during the simulation or through a button located in the first page of the config wizard.

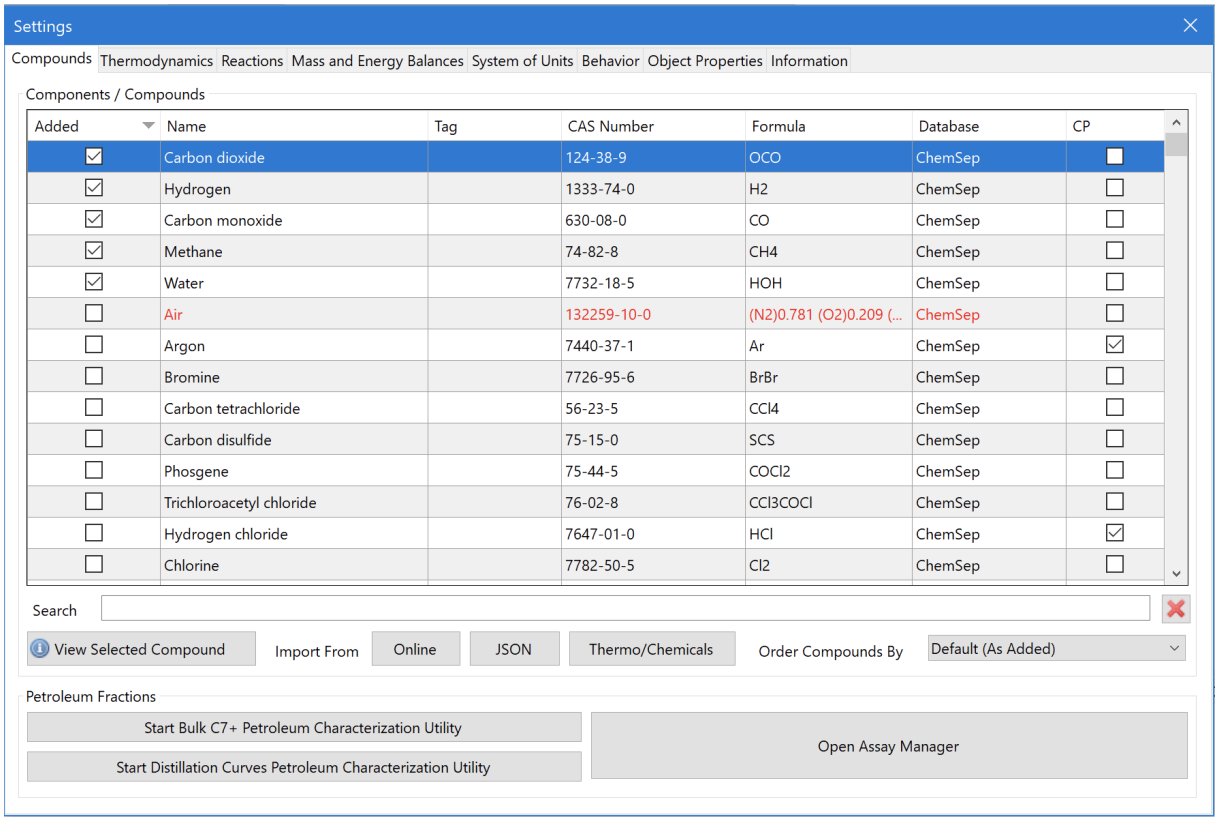



Figure 8: Simulation Configuration window.

The simulation configuration window (Figure 40) is the interface where all the functions for configuration and personalization of a simulation in DWSIM can be found. In this window, the user can manage the simulation components, the property package (thermodynamic model), configure the reactions environment, units system and number format, among other options.

**The simulation configuration window can be accessed anytime when a simulation is opened in DWSIM. The changes made through it have immediate effect on the simulation.**

### 2.2.1. Components/Compounds

There are two essential information required by DWSIM in order to correctly start a simulation. The first refers to the available ***components*** (or ***compounds***). DWSIM comes with six default compound databases (DWSIM, ChemSep, Biodiesel, CoolProp, ChEDL and Electrolytes), with a total of more than 1500 compounds available for your simulation.

To add a compound to the simulation, select it from the list on the left and click on **Add >**. To remove an added compound, select it on the right-hand list and click **< Remove**. To view the data from a compound from on a list, click on the appropriate **View Data** button.

DWSIM also features full compound data importing from **Online Sources** or from **JSON files**, using the appropriate buttons on the Simulation Configuration Wizard or on the Simulation Settings panel. If you manage to find a compound from these sources with a minimum set of data, they can be added directly to the simulation without further action.

**JSON files** are exported from the Compound Creator utility or from the Pure Compound Property Viewer tool.

### 2.2.2. Basis

**2.2.2.1. Property Packages** The Property Package consists in a set of methods and models for the calculation of physical and chemical properties of material streams in the simulation. It is composed of a thermodynamic model - an equation of state or a hybrid model - and methods for property calculation, like the surface tension of the liquid phase. The figure 40 shows the interface for configuration of the property package.

DWSIM allows multiple Property Packages to be added to a single simulation. The Property Packages can be associated to any unit operation and material stream on a individual basis. Each property package has its own settings, even if two or more packages are of the same type.

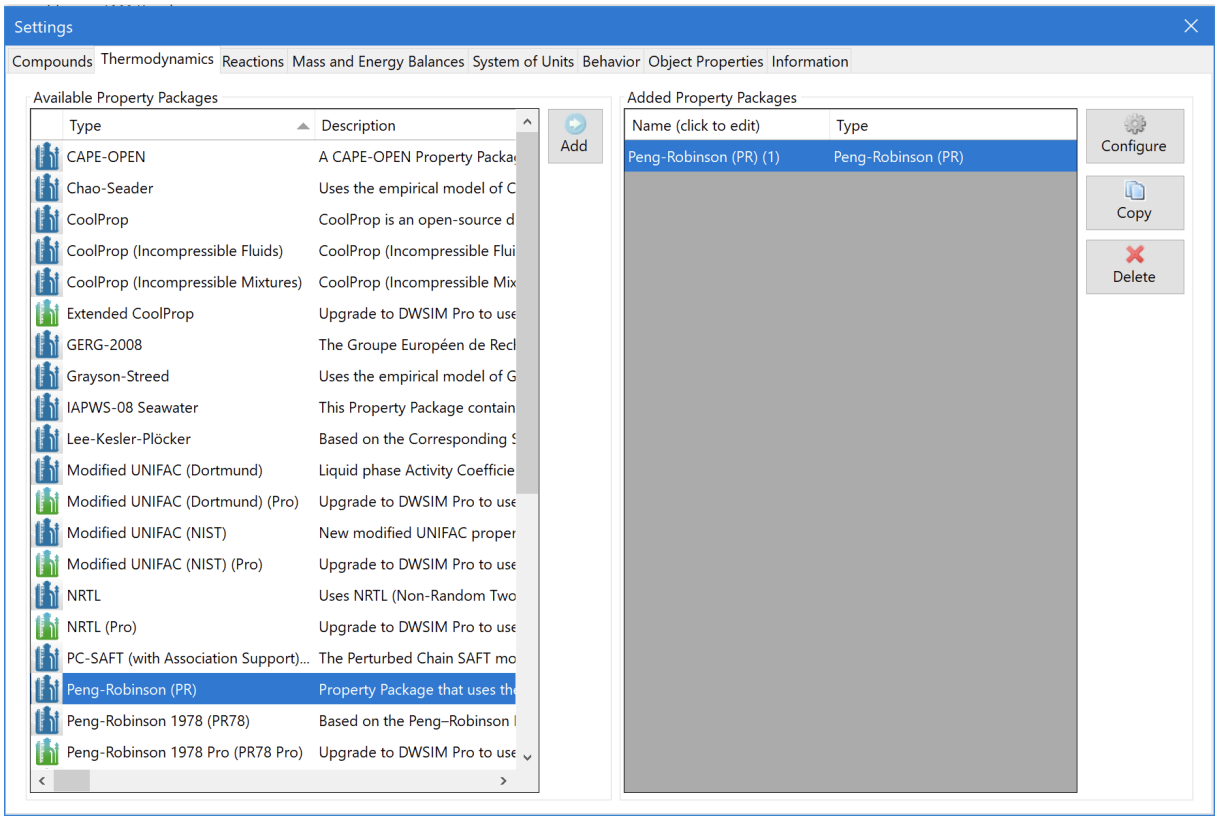



Figure 9: Property Package configuration interface.

If the selected property package has any editable property, the "Configure" button becomes clickable and the user can click on it to show the property package configuration window.

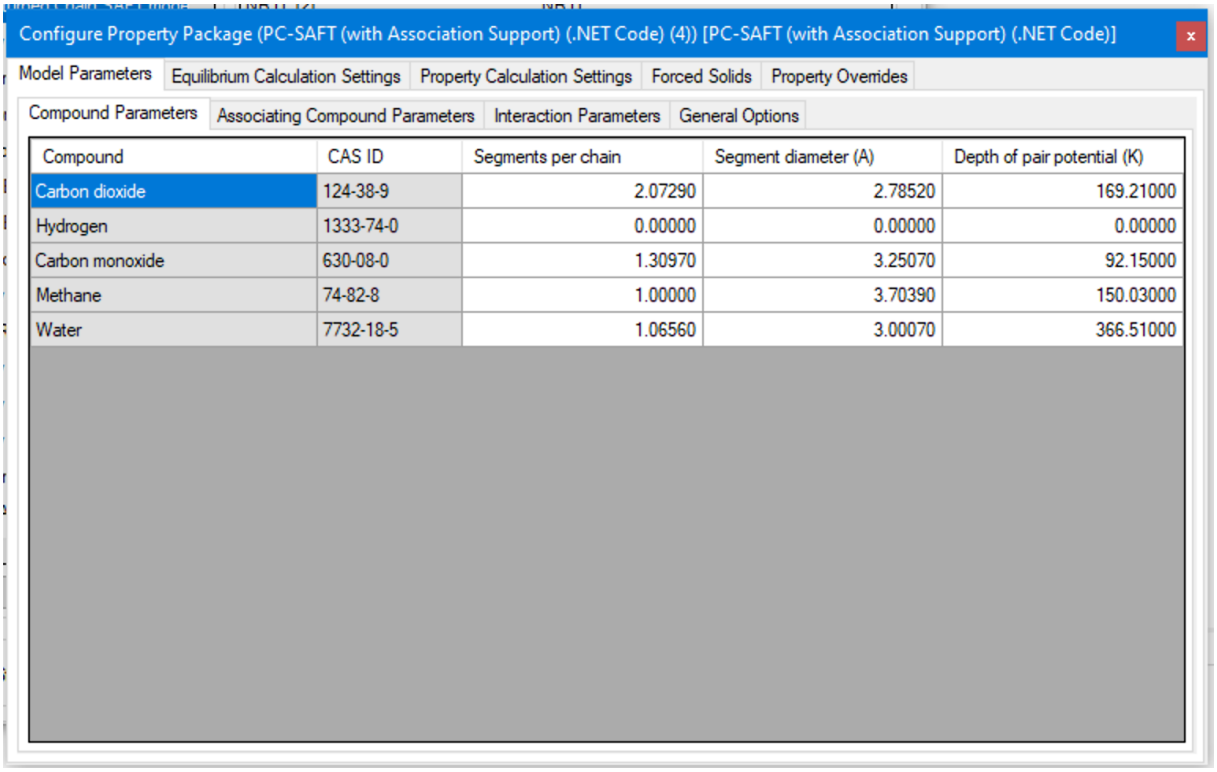



Figure 10: Property package configuration window (1).

**Property Package configuration options** Some Property Packages have extra configuration options in order to allow a deeper control of the thermodynamic calculations for the user. They are:

➞ *Use Peneloux Volume Translation correction (PR/SRK EOS only)*

This option is available for PR and SRK Property Packages. It enables correction of EOS-calculated densities by the inclusion of a correction factor named *volume translation coefficient*. This option will be effective only if the EOS is selected as the calculation method for Liquid Density.

➞ *Ignore maximum salinity limit (IAPWS-08 Seawater Property Package only)*

Ignores the maximum supported salinity value (0.12 kg/kg) for calculations and doesn't display any warnings. Use 0 to disable, 1 to enable this option. If enabled, the calculated salinity will be send directly to the property calculation routines without further check. If disabled, the maximum value of 0.12 will be used if the calculated salinity is higher, and a warning message will be displayed in the flowsheet log window.

➞ *Calculate Bubble and Dew points at stream conditions*

Check this box if you want the DWSIM to calculate bubble and dew points at conditions specified on each material stream. The calculated values will be shown only if the stream is at VLE equilibrium. The calculations are not exactly fast, so use this option with caution and only if needed.

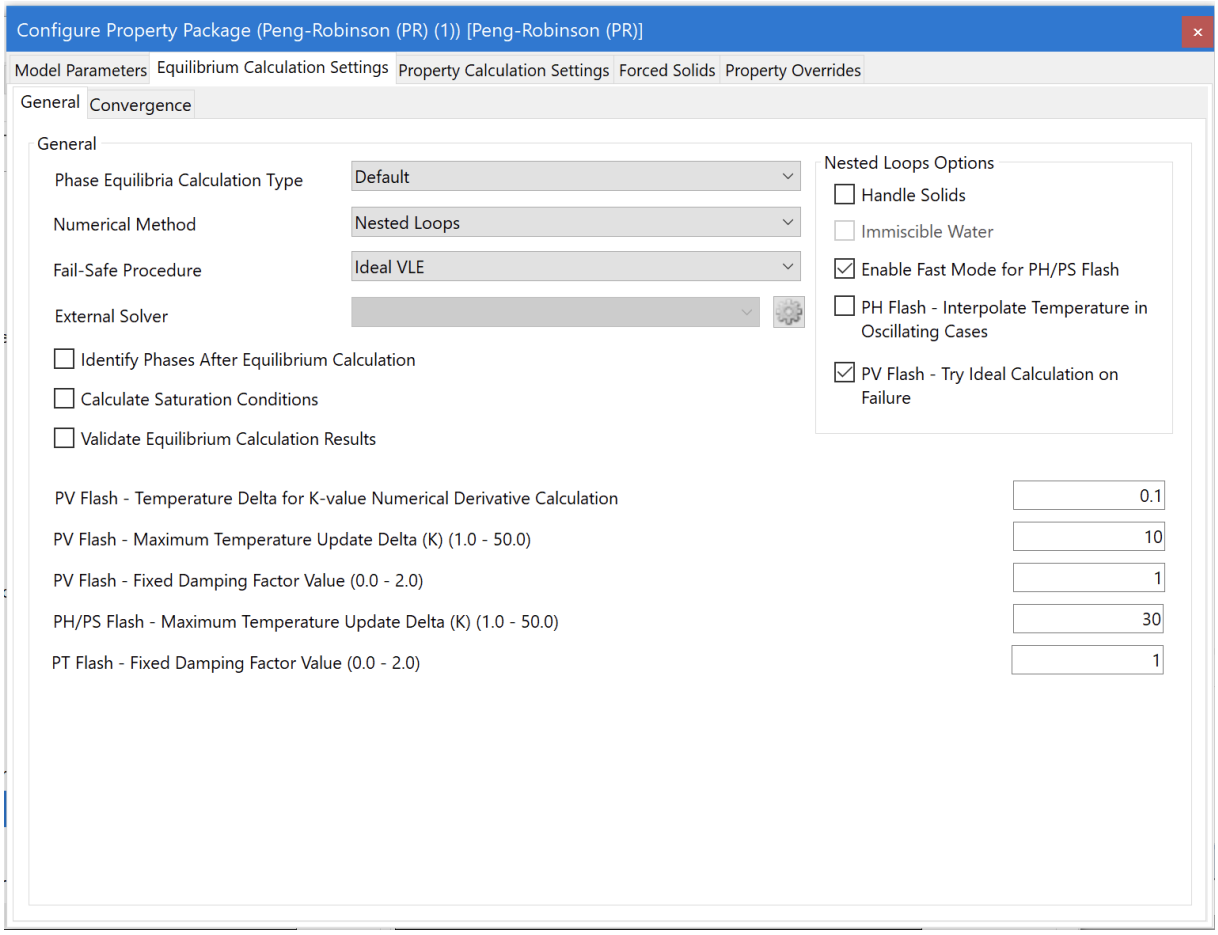



Figure 11: Property Package Equilibrium Calculation Settings.

**Property Package Flash (Equilibrium) Calculation Settings**

➙ ***Phase Equilibria Calculation Type***

The default calculation type considers one vapor and two liquid phases. Check the "Handle Solids" box to include the solid phase in the Default equilibrium calculation mode.

Change this setting to a different value if the default setting gives you convergence errors in the simulation.

➙ ***Numerical Method***

You can choose from three different approaches to calculate phase equilibria: Nested Loops (default), Inside-Out and Gibbs Minimization. If an external optimizer is available, you can select one from the External Solver drop down list to use when in Gibbs Minimization mode.

➙ ***Fail-Safe Procedure***

Select a fail-safe calculation mode if the main phase equilibria calculation fails. You can select one of the following options:

1. Rigorous VLE: does a VLE calculation using the currently selected Property Package.
2. Ideal VLE: does a VLE calculation using the Raoult's Law Property Package.

3. Do Not Calculate Equilibrium: doesn't perform any equilibrium calculation.

4. Throw error/exception: this was the default behavior on older DWSIM versions.

➙ ***Force Pressure-Enthalpy (PH) Flash calculations***

If enabled, all requests by unit operations for Pressure-Temperature Flashes will be replaced by Pressure-Enthalpy ones.

➙ ***Validate Equilibrium Flash Calculation Results***

If enabled, DWSIM will check the mixture Gibbs energy before and after the equilibrium flash calculation. If the gibbs energy *increases* after the calculation (it should always *decrease* when there is a phase split), an error message will be shown and the flowsheet calculation will be aborted.

➙ ***Apply a Phase Identification Algorithm after Equilibrium Calculations***

Check this to apply an identification algorithm to each phase after the equilibrium calculation is finished. This can be useful for supercritical compounds which behave as liquid at high pressures and temperatures, or special mixtures which exhibit LLE behavior at low temperatures, incorrectly identified as VLE by the flash algorithms.

Visit DWSIM's wiki for more information about the phase identification algorithm.

**Forced Solids** Use the Forced Solids option to define the compounds which will always be in the Solid Phase.

**Property Overrides** Since DWSIM Version 5.1, you can override the calculated phase properties through Python scripts. This can be useful if the calculated property is far from the expected value, or if you need to include advanced mixing rules when calculating mixed phase properties.

For more information, go to the Overriding Calculated Properties page on the Wiki.

**2.2.2.2. Property Package Selection Guide** Most thermodynamic models have binary interaction parameters which are fitted to match experimental data. Always check if the selected thermodynamic model has interaction parameters for the compounds in the simulation, if required. To view the list of IPs, open the Property Package Configuration Window and go to the Interaction Parameters tab.

Whenever possible, one should either use experimental data to check the predicted properties, or to use these data to fit suitable thermodynamic models. DWSIM has a tool to regress experimental data and calculate binary interaction parameters for various thermodynamic models.

**Non-polar gases at low pressures (< 10 atm)** Use the Raoult's Law Property Package. It assumes that both phases (gas and liquid) are ideal.

**Non-polar gases at high pressures (> 10 atm)** Use one of the Equation of State models like Peng-Robinson, Soave-Redlich-Kwong and PRSV2.

**Polar gases at high pressures (> 10 atm)** Use the PRSV2 Property Package. Check if it has the required parameters for your system as DWSIM lacks many parameters for this model. If it doesn't, fallback to an EOS model like PR or SRK.

**Systems with high Hydrogen content** You can use the Chao-Seader, Grayson-Streed or Lee-Kesler-Plöcker model. The LKP model is very slow but can be more reliable depending on the system. The LKP model is very sensitive to the interaction parameter values being used.

**Air Separation** / **Refrigeration systems** Use the CoolProp Property Package.

**Steam/Water simulations** Use the Steam Tables Property Package.

**Polar chemicals** Use one of the activity coefficient models like NRTL or UNIQUAC. If no interaction parameters are available for your system, you can fallback to one of the UNIFAC-type models. Modified UNIFAC (NIST) is recommended.

**Salt/Water systems** Use the Seawater Property Package.

### 2.2.3. Systems of Units

Three basic units systems are present in DWSIM: ***SI System*** (selected by default), ***CGS System*** and ***English (Imperial) System***. The simulation's units system can be viewed/modified in the "Units System" section of the "Options" tab in the simulation configuration window (Figure 40).

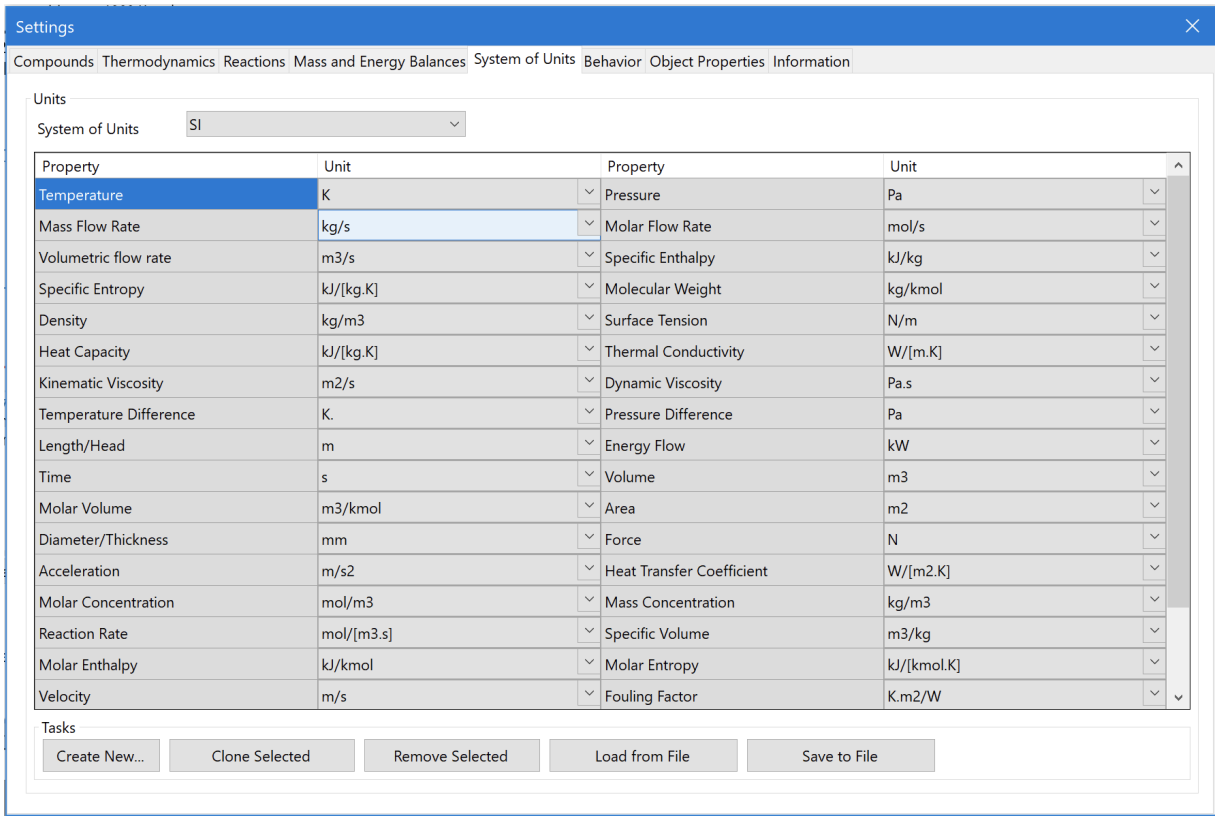



Figure 12: System of Units configuration interface.

There are buttons available on this interface to create custom units systems and save/load them. It is worth remembering that the units systems can also be modified at any time during the simulation - the changes are applied immediately.

### 2.2.4. Behavior

#### 2.2.4.1. Behavior options for Flowsheet Objects

➞ ***Skip Equilibrium Calculation in Well-Defined Streams***

Tells the Flowsheet Solver to avoid doing unnecessary work and skip equilibrium calculations in Material Streams connected to specific ports like Separator Vessel and Distillation Column outlets.

➞ ***Force Material Stream Phase***

You can override the equilibrium phase for **all** Material Streams in the flowsheet by setting this property property to the desired value (*Vapor*, *Liquid* or *Solid*. The default value is *Do Not Force*.

When this property is set to one of the phase names (*Vapor*, *Liquid* or *Solid*), the equilibrium calculation for all Material Streams is bypassed and all compounds are put into the selected phase with the same composition as the mixture.

➞ ***Force object calculation even when input parameters don't change***

This is the main feature of the Smart Object Solver added in v8.4. You can turn it off by unchecking the corresponding box.

➞ ***Specification Blocks calculation mode***

You can define how and when the specification blocks are calculated in the flowsheet.

#### 2.2.4.2. Number Formatting

➞ ***Numerical Values Formatting Scheme:*** select the formatting for general numbers.

➞ ***Stream Composition Formatting Scheme:*** select the formatting for stream compositions.

#### 2.2.4.3. General

➞ ***Enable Undo/Redo:*** allows the flowsheet to quickly return to a previous state.

➞ ***Include flowsheet messages when saving file:*** for debugging purposes, the log messages are added to the flowsheet file by default.

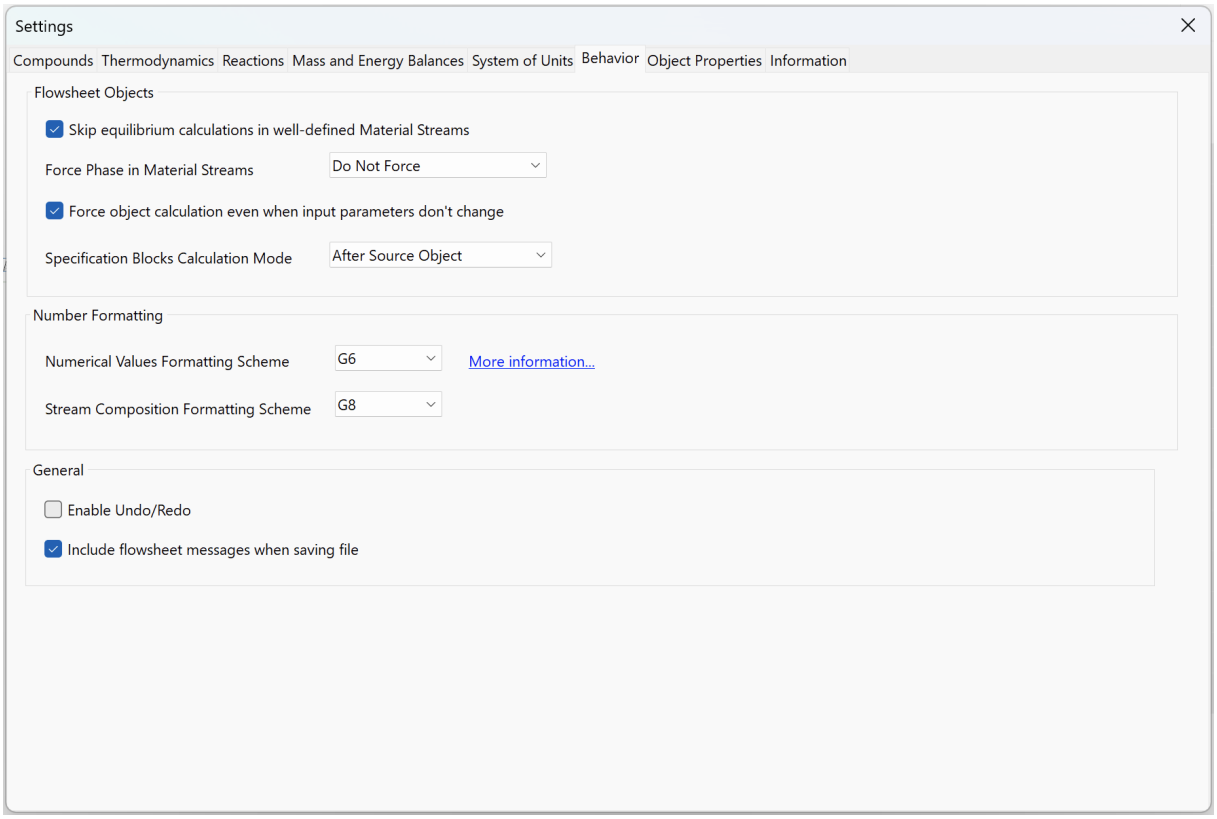



Figure 13: Behavior settings interface.

### 2.2.5. Information

In the "Description" group box it is possible to edit some information about the active simulation (title, author and description). You can also define a password to prevent the simulation of being opened by anyone, but this feature only works with the Compressed XML simulation file format (*.dwxmz).

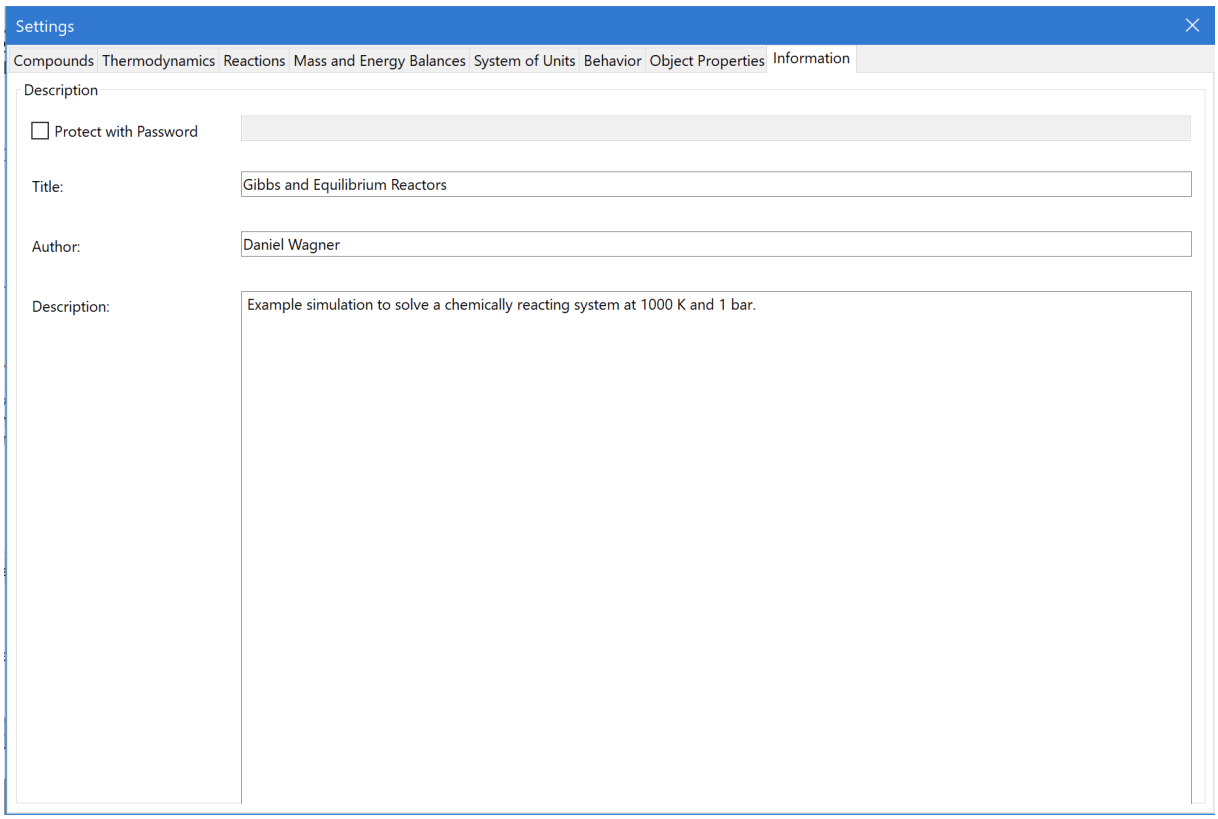



Figure 14: Information settings interface.

### 2.2.6. Property Tables

In the "Property Tables" section you can define which properties are going to be shown for each object type when you hover the mouse over the objects on the flowsheet. The property list is saved in a per-simulation basis.

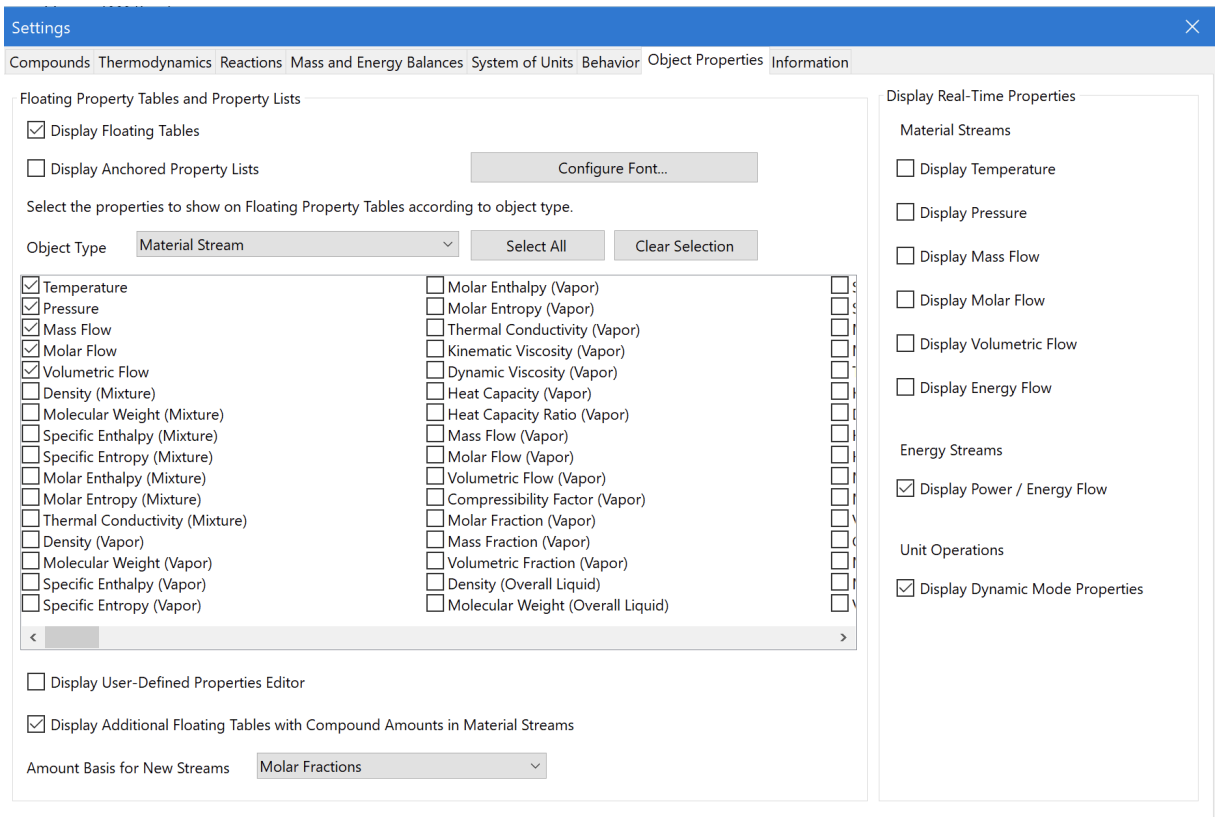



Figure 15: Property Tables settings interface.

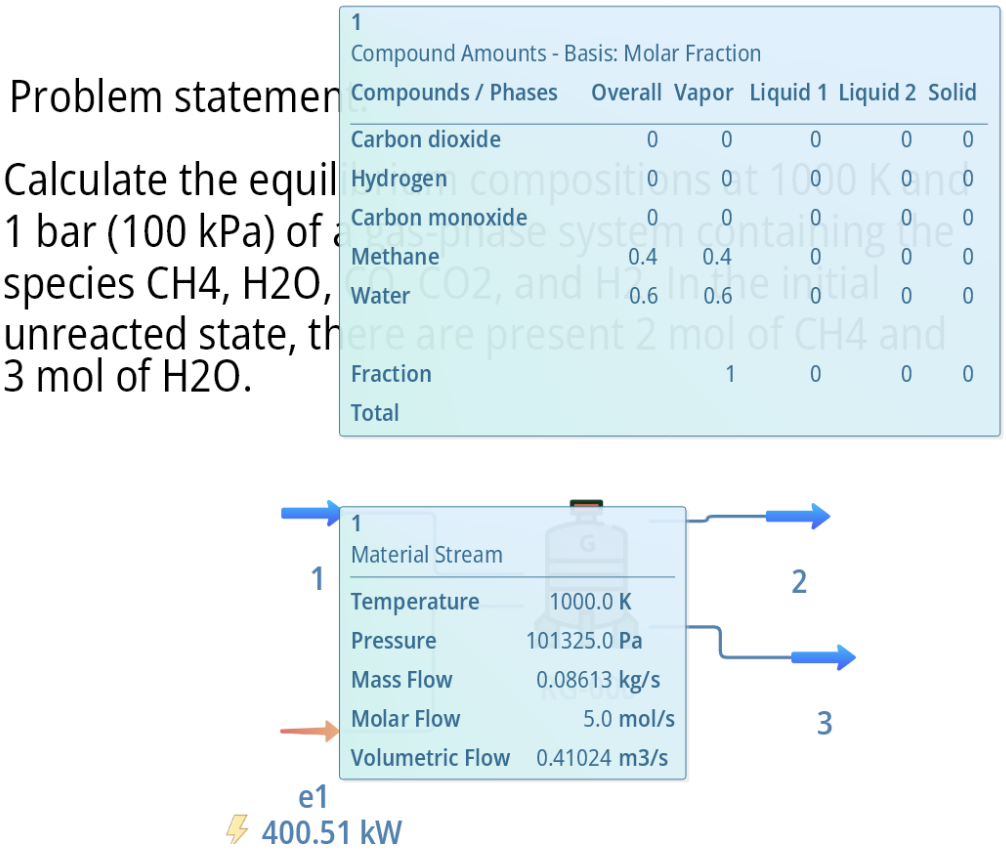



Figure 16: Selected properties on the previous image are shown on the flowsheet for the Material Streams.

## 2.3. Process Modeling (Flowsheeting)

After configuring the simulation, the user is taken to the main simulation window (Figure 40). In this window we can highlight the following areas:

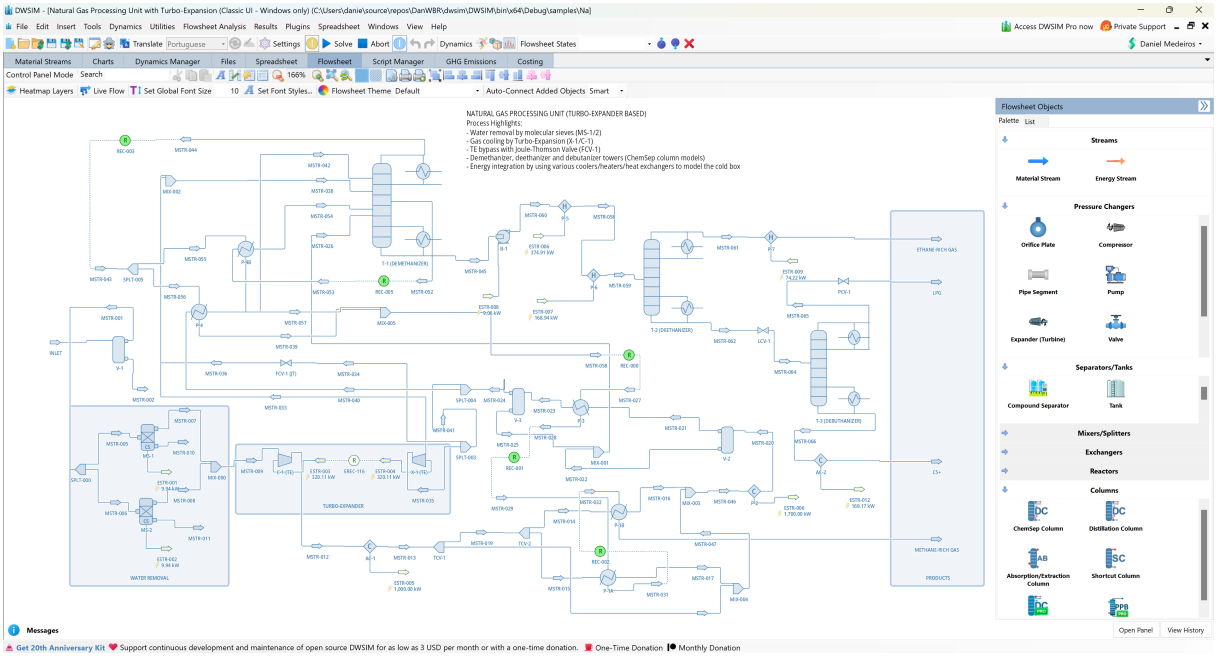


Figure 17: DWSIM simulation window.

- ***Menu Bar***: file handling, window arrangement, help, simulation settings, solver controls, undo/redo buttons.
- ***Flowsheet Objects Palette*** window: shows objects which can be added by dragging them into the PFD;
- ***Flowsheet Objects List*** window: contains a searchable list of the flowsheet objects, including shortcuts to edit, rename and delete items;
- ***Material Streams*** window: lists the material streams in the flowsheet and their calculated properties;
- ***Flowsheet*** window: process flowsheet building and editing area;
- ***Information*** window: general information about the active simulation;
- ***Spreadsheet*** window: shows the spreadsheet, a utility to do math operations with data provided by the objects in the current simulation;
- ***Charts*** window: used to create and view charts from flowsheet objects or from spreadsheet data;
- ***Script Manager*** window: displays the script manager, which can be used to write Python scripts to automate certain simulation tasks.

When running DWSIM on a Windows platform, the simulation windows can be freely repositioned, with the arrangement information being saved together with the rest of simulation data. To reposition a window, the user should click with the left mouse button in the window's top bar and drag it to the desired place. A preview of how the window will be is shown in blue (Figure 40).

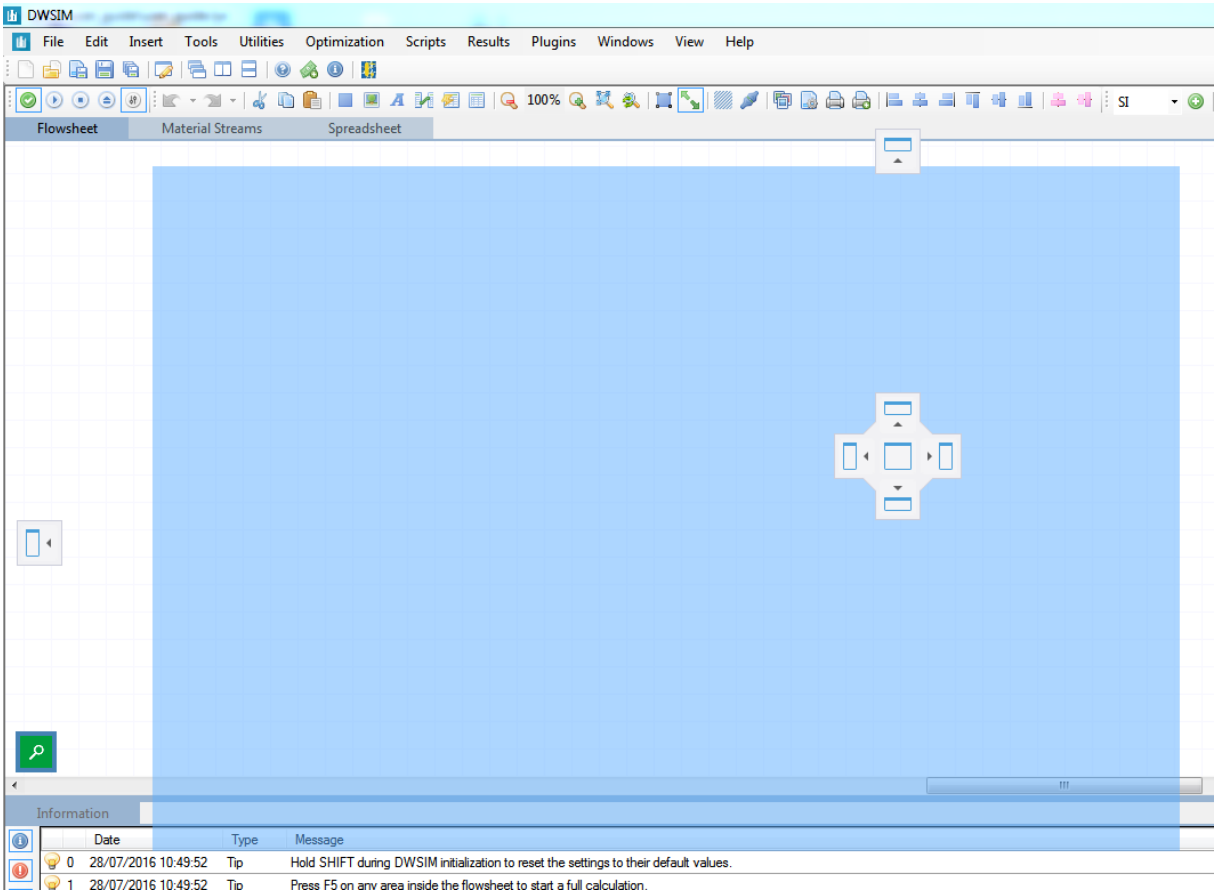



Figure 18: Window repositioning.

When running DWSIM on Mono (Linux), use the context menus (right-click with the mouse on the window caption bar) on each window to reposition/dock its contents.

### 2.3.1. Inserting Flowsheet Objects

To add an object to the flowsheet, you can:

➙ Use the **Insert > Flowsheet Object** menu item (keyboard shortcut: Ctrl+A):

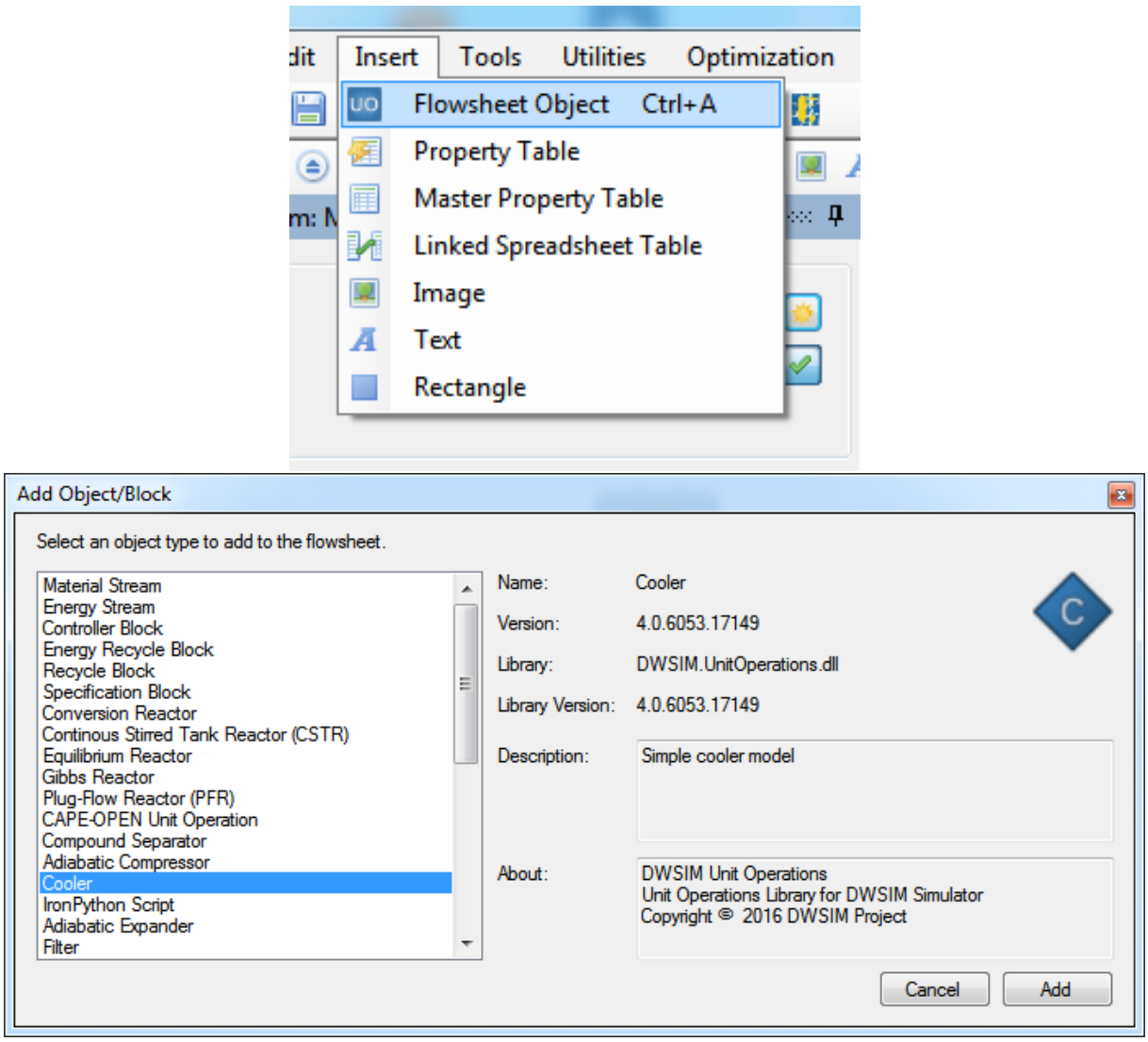



Figure 19: Inserting an object to the flowsheet.

➞ Drag an item from the **Object Pallette** window located on the bottom of the flowsheet panel:

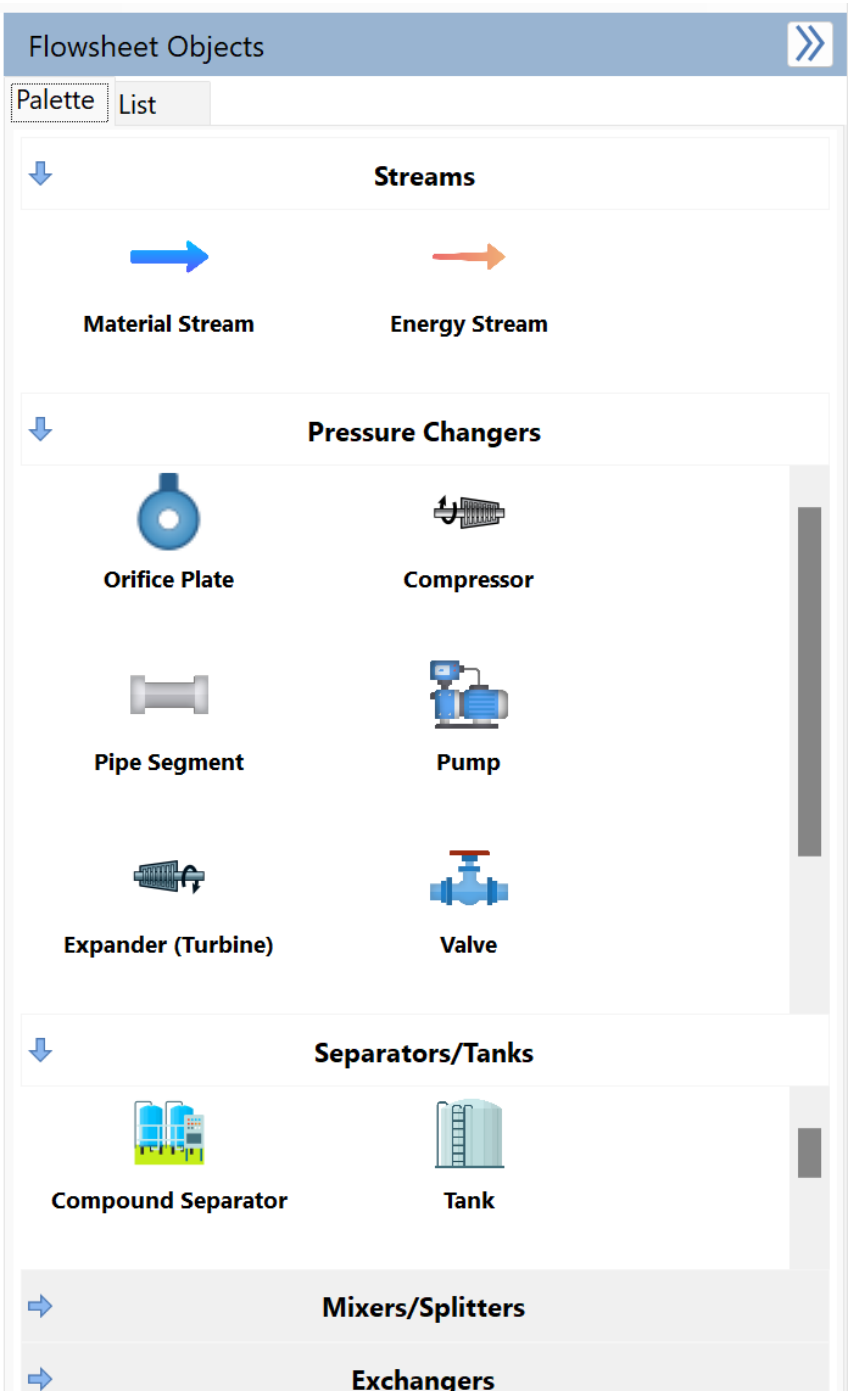



Figure 20: Inserting an object to the flowsheet by dragging from the Object Pallette window.

The elements of a simulation (objects) which can be added to the flowsheet are:

- ***Material Stream***: used to represent matter which enters and leaves the limits of the simulation and passes through the unit operations. The user should define their conditions and composition in order for DWSIM to calculate their properties accordingly;
- ***Energy Stream***: used to represent energy which enters and leaves the limits of the simulation and passes through the unit operations;
- ***Mixer***: used to mix up to three material streams into one, while executing all the mass and energy

balances;

- ***Energy Mixer:*** mix up to three energy streams into one;
- ***Splitter***: mass balance unit operation - divides a material stream into two or three other streams;
- ***Valve***: works like a fixed pressure drop for the process, where the outlet material stream properties are calculated beginning from the principle that the expansion is an isenthalpic process;
- ***Pipe***: simulates a fluid flow process (mono or two-phase). The pipe implementation in DWSIM provides the user with various configuration options, including heat transfer to environment or even to the soil in buried pipes. Two correlations for pressure drop calculations are available: Beggs & Brill and Lockhart & martinelli. Both reduces to Darcy equation in the case of single-phase flow;
- ***Pump***: used to provide energy to a liquid stream in the form of pressure. The process is isenthalpic, and the non-idealities are considered according to the pump efficiency, which is defined by the user;
- ***Tank***: in the current version of DWSIM, the tank works like a fixed pressure drop for the process;
- ***Separator Vessel***: used to separate the vapor and liquid phases of a stream into two other distinct streams;
- ***Compressor***: used to provide energy to a vapor stream in the form of pressure. The ideal process is isentropic (constant entropy) and the non-idealities are considered according to the compressor efficiency, which is defined by the user;
- ***Expander***: the expander is used to extract energy from a high-pressure vapor stream. The ideal process is isentropic (constant entropy) and the non-idealities are considered according to the expander efficiency, which is defined by the user;
- ***Heater***: simulates a stream heating process;
- ***Cooler***: simulates a stream cooling process;
- ***Conversion Reactor***: simulates a reactor where conversion reactions occur;
- ***Equilibrium Reactor***: simulates a reactor where equilibrium reactions occur;
- ***PFR***: simulates a Plug Flow Reactor (PFR);
- ***CSTR***: simulates a Continuous-Stirred Tank Reactor (CSTR);
- ***Shortcut Column***: simulates a simple distillation column with approximate results using shorcut calculations;
- ***Distillation Column***: simulates a distillation column using rigorous thermodynamic models;
- ***Absorption Column***: simulates an absorption column using rigorous thermodynamic models;
- ***Refluxed Absorber***: simulates a refluxed absorber column using rigorous thermodynamic models;
- ***Orifice Plate:*** model to simulate an orifice plate, used for flow metering;

- ***Component Separator:*** model to simulate a generic process for component separation;
- ***Custom Unit Operation:*** an user-defined model based on Python scripts;
- ***CAPE-OPEN Unit Operation:*** External CAPE-OPEN Unit Operation socket for adding CO Unit Operations in DWSIM;
- ***Spreadsheet Unit Operation:*** Unit Operation where the model is defined and calculated in Spreadsheet (XLS/XLSX/ODS) files;
- ***Solids Separator:*** model to simulate a generic process for solid compound separation;
- ***Continuous Cake Filter:*** continuous cake filter model for solids separation;
- ***Air Cooler 2:*** unit operation that is used to cool a material stream using air;
- ***Water Electrolyzer:*** electrolysis model for H2 generation from water;
- ***PEM Fuel Cell:*** Proton-exchange Membrane Fuel Cell model for energy generation from H2 and O2;
- ***Hydroelectric Turbine:*** generates energy from a water stream;
- ***Wind Turbine:*** generates energy from wind;
- ***Solar Panel:*** generates energy from solar energy;
- ***Gibbs Reactor (Reaktoro):*** general-purpose chemical reactor based on Reaktoro.

Additionally, the following logical operations are available in DWSIM:

- ***Controller:*** used to make a variable to be equal to a user-defined value by changing the value of other (independent) variable;
- ***Specification***: used to make a variable to be equal to a value that is a function of other variable, from other stream;
- ***Recycle***: used to mix downstream material with upstream material in a flowsheet,
- ***Energy Recycle***: used to mix downstream energy with upstream energy in a flowsheet.
- ***Input Box:*** use to quickly change a property of an object;
- ***Switch:*** used to switch the value of a property of an object between two values;

Figure 40 shows a material stream added to the flowsheet by one of the method described above. It can be observed that the stream is selected and that its property editor is shown as a panel on the left part of the main window.

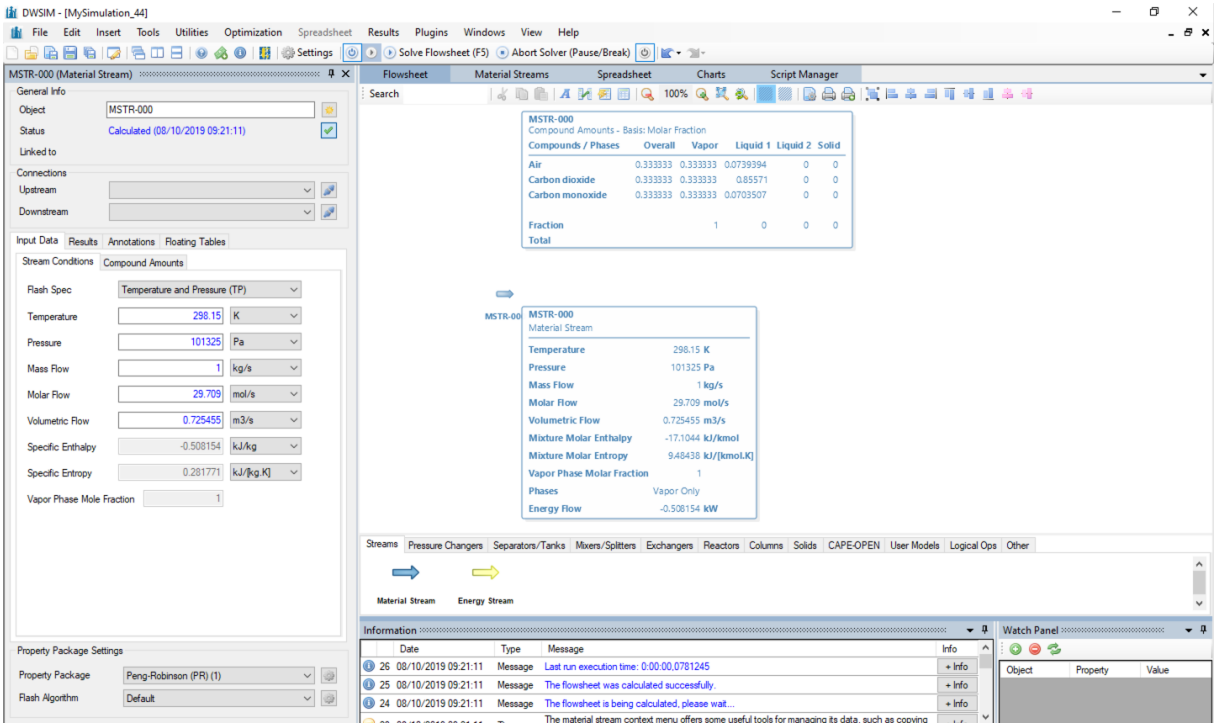



Figure 21: A material stream in the flowsheet.

***Connecting objects*** The material streams represent mass flowing between unit operations. There are two different ways in which a material stream can be connected to a unit operation (or *vice-versa*):

- Through the context menu activated with a right mouse button click over the object (Figure 40);

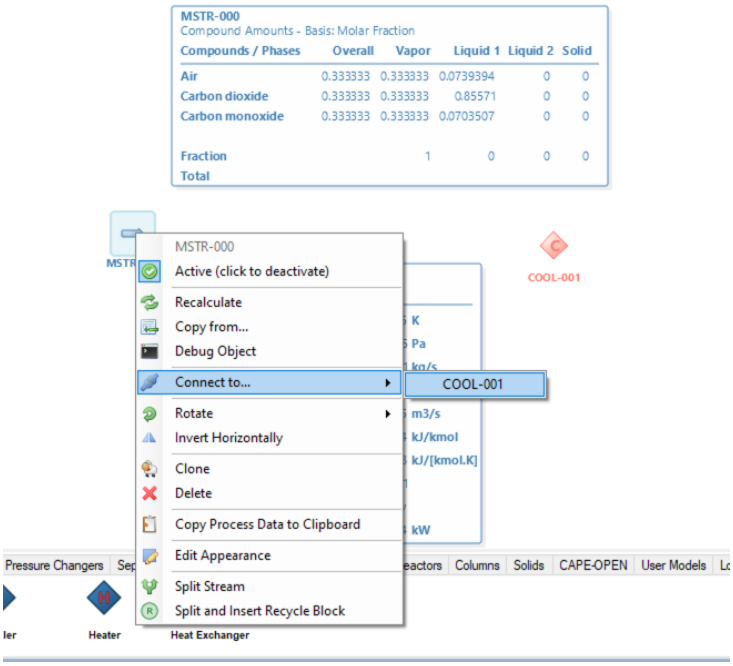



Figure 22: Selected object context menu.

- Through the property editor window - Connections section.

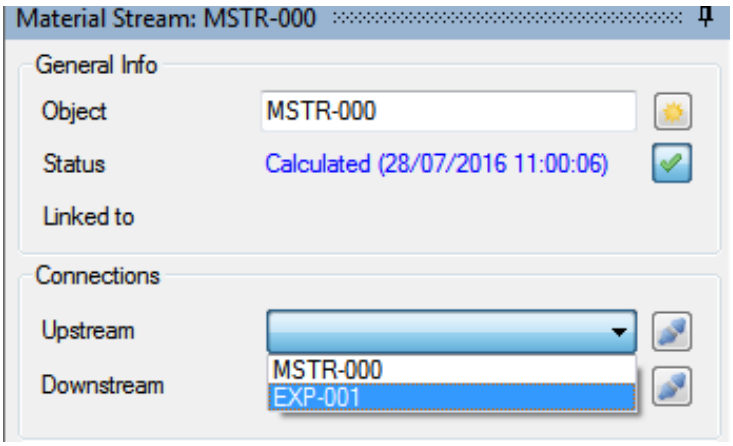



Figure 23: Connection selection menu.

➙ Through the "Create and Connect" buttons on the object editors. When you click on these buttons, DWSIM will automatically create and connect streams to the associated ports on the selected object.


Figure 24: Create and Connect tool.

An expander system with its connections is shown on Figure 40.

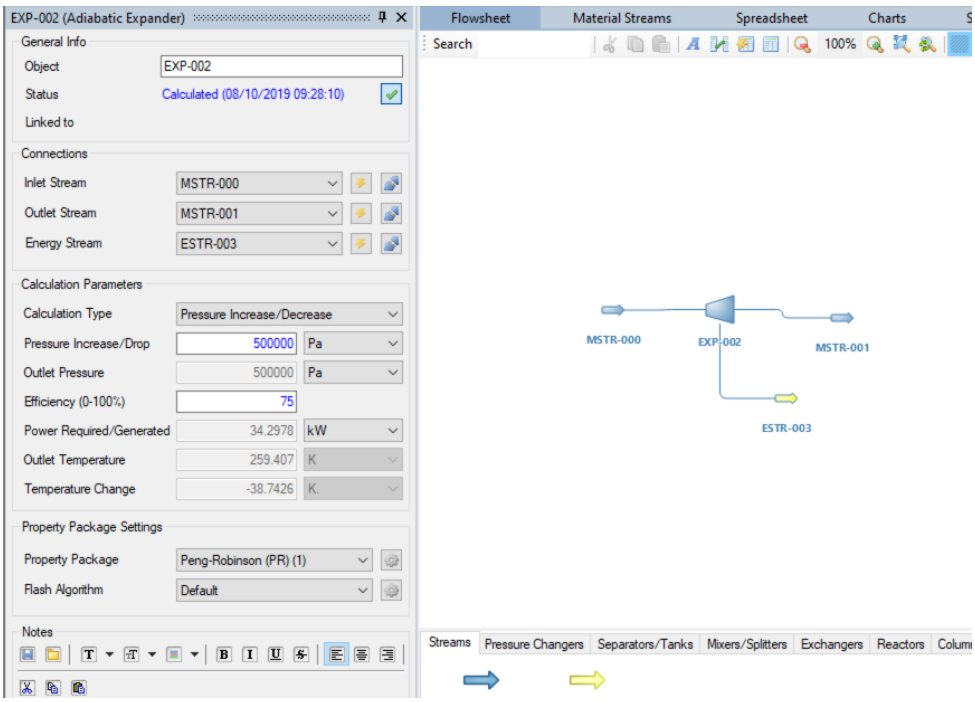



Figure 25: Expander with all connections correctly configured.

***Disconnecting objects*** Tools to disconnect objects from each other can be found on the same locations as the connecting ones.

***Removing objects from the flowsheet*** The selected object can be removed from the flowsheet by pressing the DEL keyboard button or by using the context menu - "Delete" item (Figure 40).

**2.3.1.1. Auto-Connect Added Objects** This is a new feature in DWSIM v7.4.0. It enables automatic connections between added objects and nearby ones.

There are three different options for this setting:

- **No**: No automatic connections are made when you add an object to the flowsheet.
- **Yes**: When you add a new unit operation, streams are automatically added to the flowsheet and connected to its ports.
- **Smart**: When you add a new unit operation, nearby streams are connected to it, and new streams are created to connect to the remaining ports.

### 2.3.2. Process data management

**Entering process data** The objects' process data (temperature, pressure, flow, composition and/or other parameters) can be entered in the property editor window (Figure 40). Properties that cannot be edited (read-only) are grayed-out.

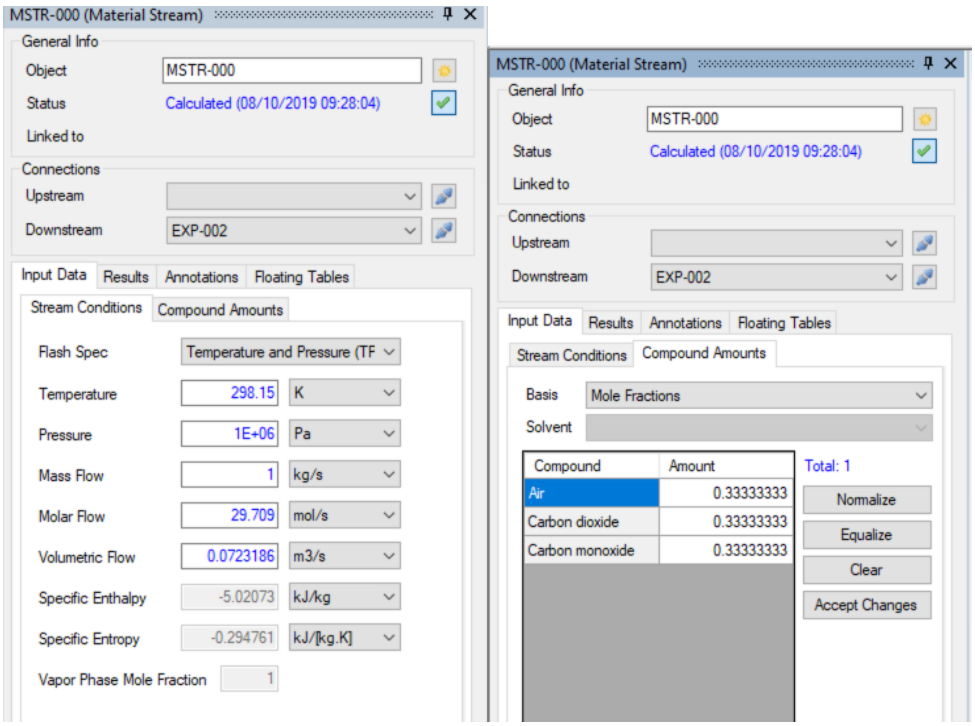



Figure 26: Viewing object properties in the editor window.

Most properties can be edited directly by typing a value in the textbox and pressing ENTER. DWSIM will then commit the new property value and trigger the flowsheet solver.

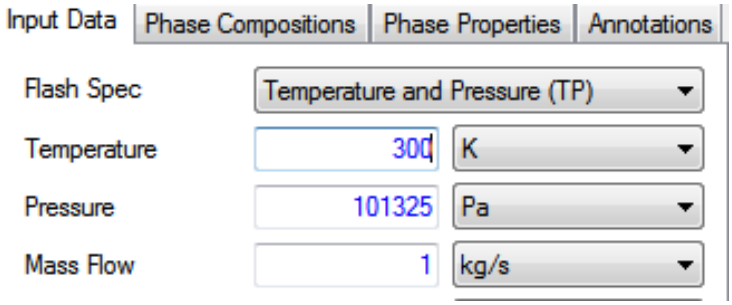



Figure 27: Direct editing of a property.

You can also use the inline units converter to convert the value of a property from the desired units to the current selected units. Type the value of the property on the textbox and select the unit to convert from at the combobox on the right. DWSIM will then convert the value from the selected units on the combobox to the actual units of the simulation system of units.

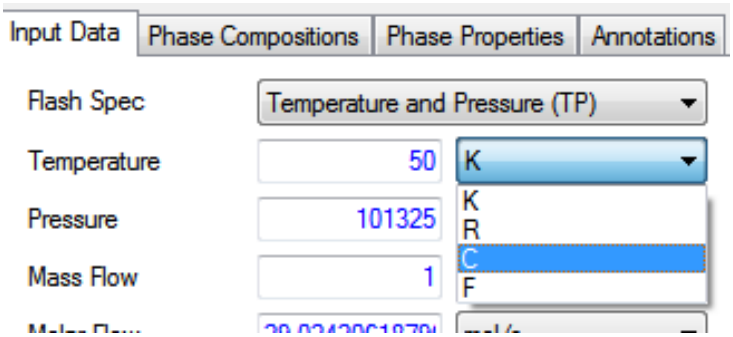



Figure 28: Converting 50 C to the current temperature units (K).

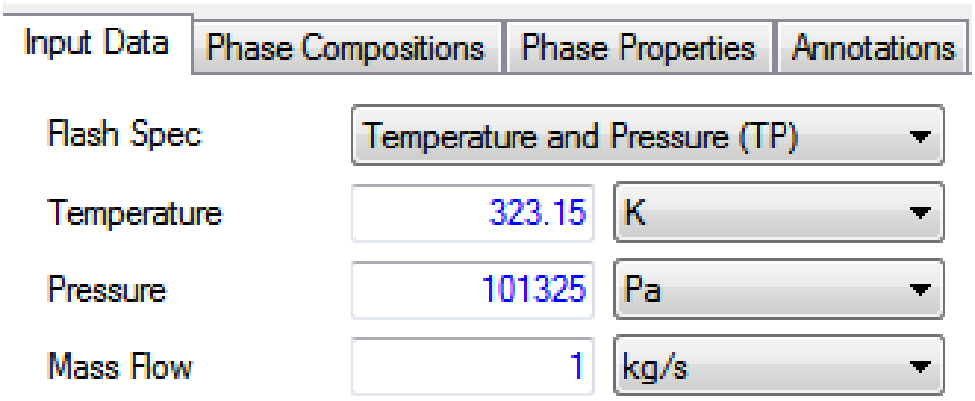



Figure 29: Converted temperature value (323.15 K).

If all object properties were correctly defined, it will be calculated by DWSIM and its flowsheet representation will have a blue border instead of a red one, indicating that the object was calculated successfully (Figure 40).

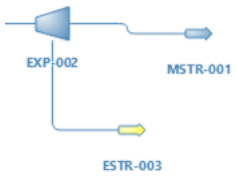



Figure 30: Calculated objects.

### 2.3.3. Cut/Copy/Paste objects

DWSIM also supports cutting, copying and pasting flowsheet objects inside a flowsheet or between different flowsheets. When copying objects between flowsheets, DWSIM may also copy compounds and property packages from one flowsheet to another. Cut/Copy/Paste behavior is an application setting and can be set in the General Settings menu (Section 5.6).

### 2.3.4. Simulation

DWSIM is a sequential modular process simulator, that is, all calculations are made in a per-module basis, according to the connections between the objects. The calculator checks if an object has all of its properties defined and, if yes, passes the data for the downstream object and calculates it, repeating the process in a loop until it reaches an object that doesn't have any of its dowstream connections attached to any object. This way, the entire flowsheet can be calculated as many times as necessary without having to "tell" DWSIM which object must be calculated. In fact, this is done indirectly if the user define all the properties and make all connections between objects correctly.

**DWSIM's calculation starts when the user edits a property which defines an object.** For example, editing a stream mass flow when its temperature, pressure and composition are already well-defined activates DWSIM's calculator.

It is possible to control DWSIM's calculator by using its button bar (Figure 40). Clicking on the button activates or deactivates the calculator. The Solve Flowsheet (F5) button performs a full flowsheet recalculation. DWSIM's calculator is enabled by default - if it is disabled, modifying of a property is accepted, but **does not** recalculate the object nor the ones that are downstream in the flowsheet. The Abort Solver (Pause/Break) button stops the any ongoing calculation.


Figure 31: DWSIM's calculator control bar.

As DWSIM's calculator does its job, messages are added to the "Information" window. These messages tell the user if the object was calculated successfully or if there was an error while calculating it, among others (Figure 40).

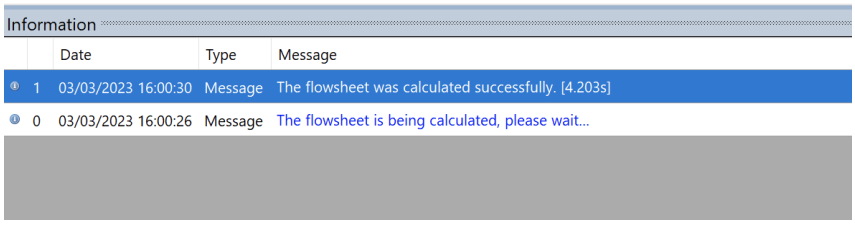



Figure 32: A DWSIM's calculator message.

### 2.3.5. Results

Results can be viewed in reports, generated (Figures 40 and 40) for printing. Report data can also be saved to ODT, ODS, XLS, TXT or XML files.

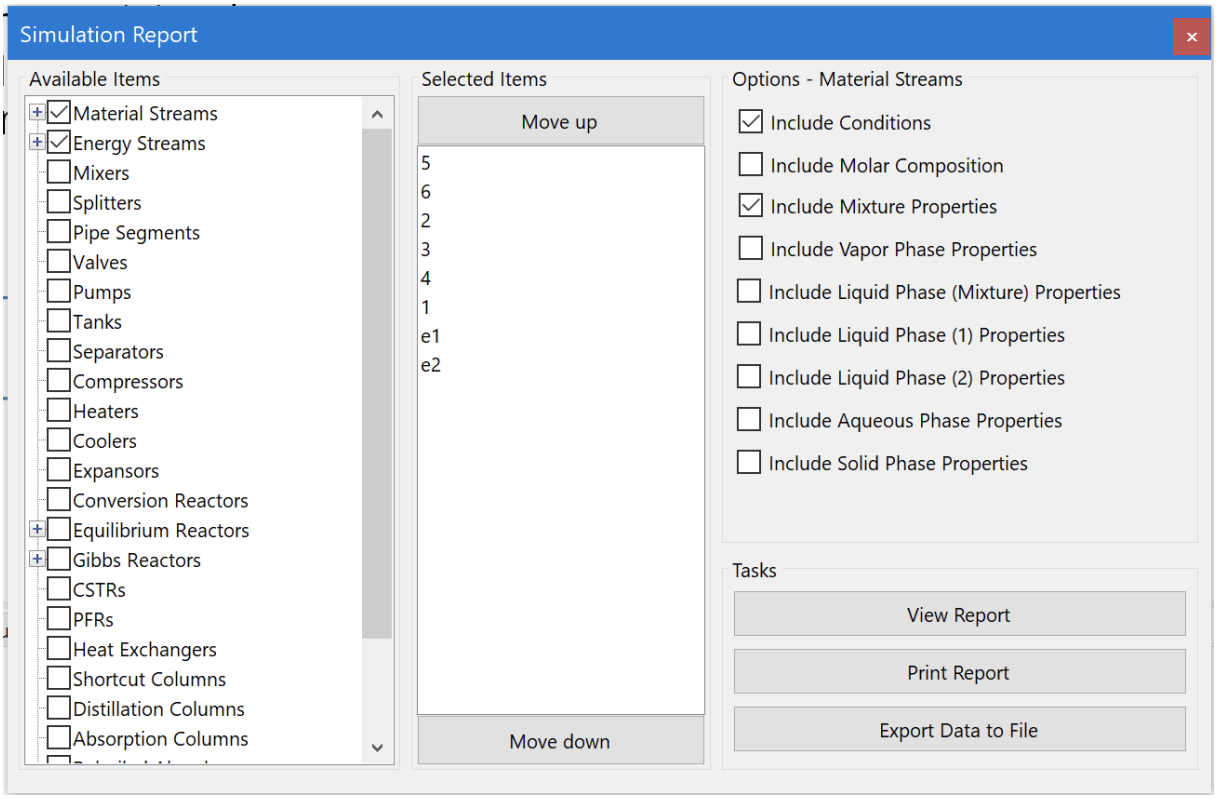
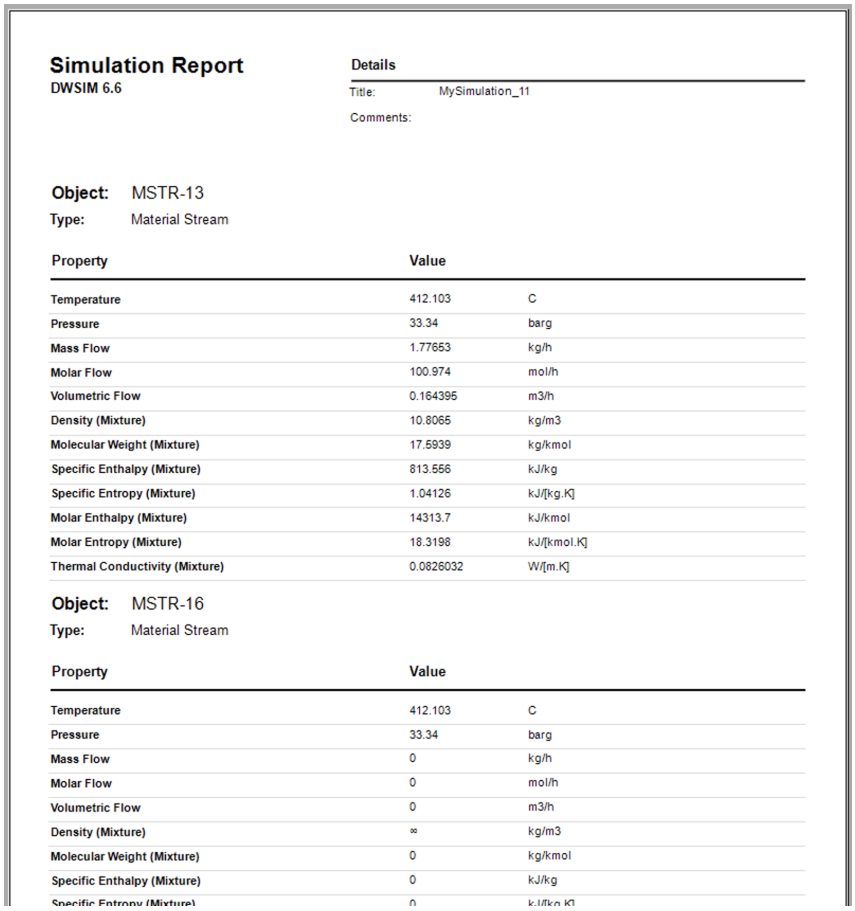



Figure 33: Results report configuration.



**Simulation Report**
DWSIM 6.6

**Details**
Title: MySimulation_11
Comments:

**Object:** MSTR-13
**Type:** Material Stream

| Property | Value | |
|---|---|---|
| Temperature | 412.103 | C |
| Pressure | 33.34 | barg |
| Mass Flow | 1.77653 | kg/h |
| Molar Flow | 100.974 | mol/h |
| Volumetric Flow | 0.164395 | m3/h |
| Density (Mixture) | 10.8065 | kg/m3 |
| Molecular Weight (Mixture) | 17.5939 | kg/kmol |
| Specific Enthalpy (Mixture) | 813.556 | kJ/kg |
| Specific Entropy (Mixture) | 1.04126 | kJ/[kg.K] |
| Molar Enthalpy (Mixture) | 14313.7 | kJ/kmol |
| Molar Entropy (Mixture) | 18.3198 | kJ/[kmol.K] |
| Thermal Conductivity (Mixture) | 0.0826032 | W/[m.K] |

**Object:** MSTR-16
**Type:** Material Stream

| Property | Value | |
|---|---|---|
| Temperature | 412.103 | C |
| Pressure | 33.34 | barg |
| Mass Flow | 0 | kg/h |
| Molar Flow | 0 | mol/h |
| Volumetric Flow | 0 | m3/h |
| Density (Mixture) | ∞ | kg/m3 |
| Molecular Weight (Mixture) | 0 | kg/kmol |
| Specific Enthalpy (Mixture) | 0 | kJ/kg |
| Specific Entropy (Mixture) | 0 | kJ/[kg.K] |

Figure 34: Results report.

## 2.4. Sensitivity Study

You can use the Sensitivity Study Utility in order to study the influence of up to 2 variables into other dependent flowsheet variables. The changes in variables are defined by a value range and a number of equally spaced points within this range. For example, you can analyze the influence of temperature and pressure in the enthalpy of a mixture, from 200 to 400 K and from 100 to 1000 kPa, nine points for temperature and 5 points for pressure, totaling 45 points on which the enthalpy will be calculated at different temperatures and pressures. This also means that the flowsheet will be recalculated 45 times (!), so be careful with the number of points you choose as the calculation time can be prohibitive.

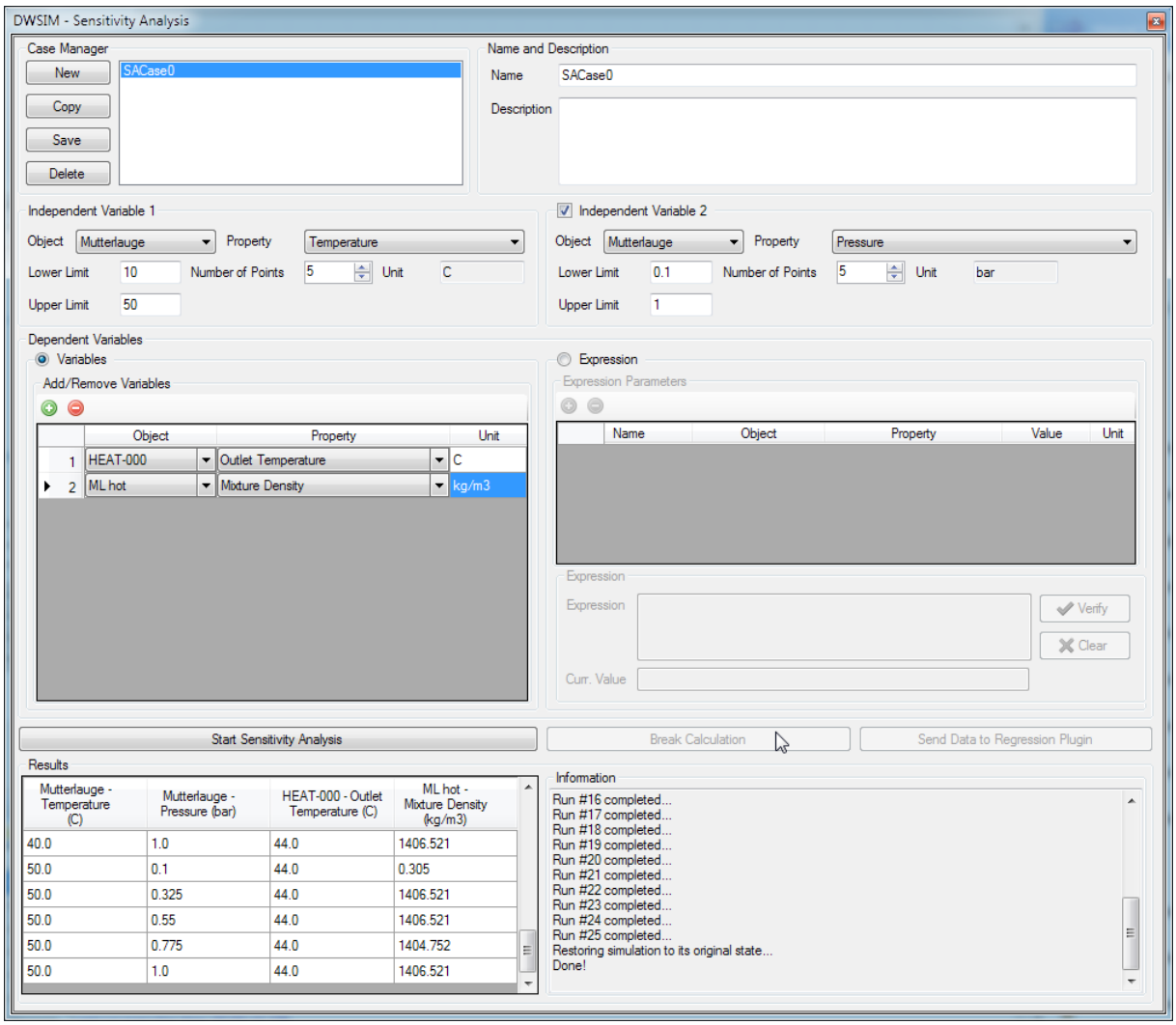



Figure 35: Sensitivity Analysis Utility (1).

The sensitivity analysis utility is based on case studies. In a single simulation one can define a number of cases, each one with its own variables, ranges and results. These cases will be saved together with the simulation, and cannot be exported to other ones. The results are shown in a table, so the data can be copied and pasted into another specialized data analysis software or sent directly to the data regression plugin.

## 2.5. Flowsheet Optimization

The Multivariate Optimizer in DWSIM handles single and multivariate optimization problems with or without bound constraints. The objective function can be either a variable in the flowsheet or an expression as a function of as many variables as you need.

The interface is very similar to Sensitivity Analysis's one. One can define a number of cases, each one with its own variables, ranges and results. These cases will be saved together with the current simulation, and cannot be exported to other simulations.

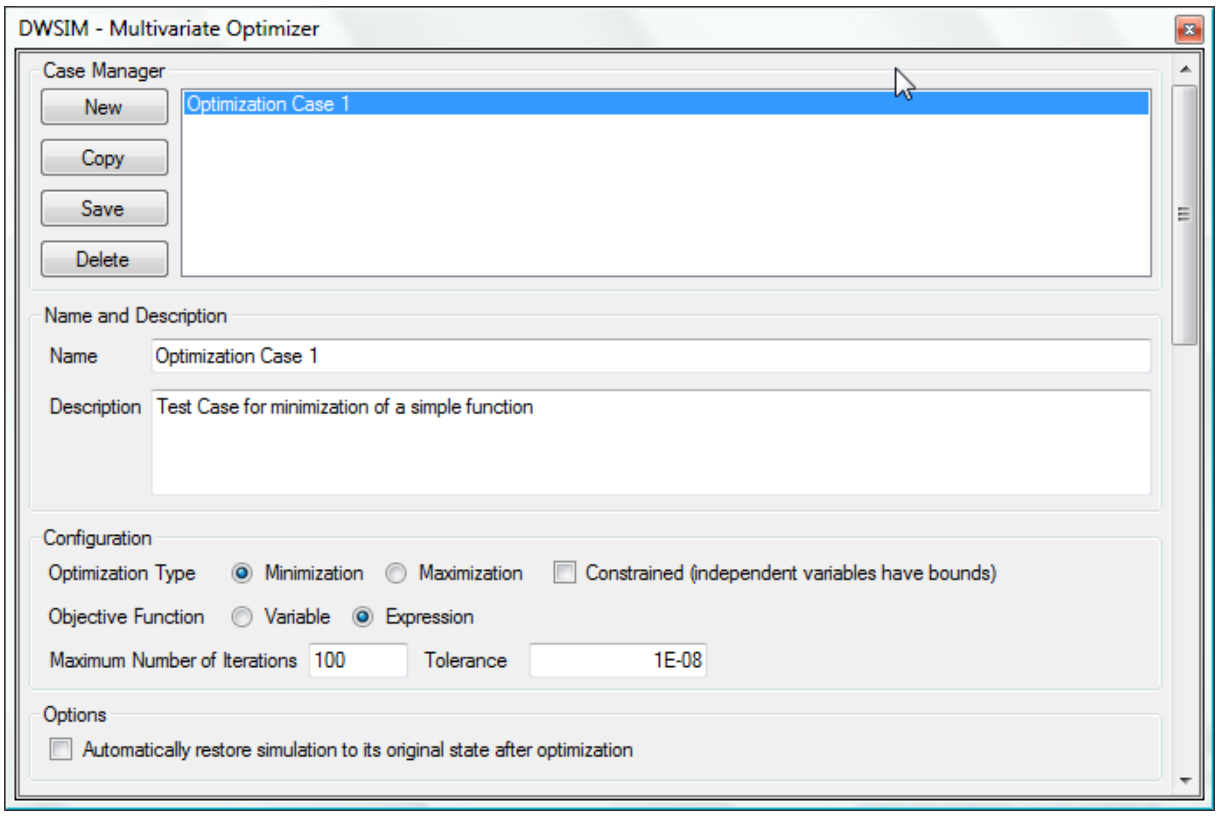



Figure 36: Multivariate Optimization Utility (1).

There are some options to choose from in DWSIM's Multivariate Optimizer. It is possible to select the type of the optimization (minimization or maximization of the objective function), choose if the indendent variables will have lower and/or upper bounds and if the objective function will be a flowsheet variable or an expression based on flowsheet variables. One can also define a maximum number for the iterations and a tolerance for the variation of the calculated value for the objective function - if the variation is less than the defined value, the flowsheet is considered optimized and the process stops. There is also an option to choose if the flowsheet will be returned to its original state after optimization, so the results will be shown only in the current window, and the flowsheet initial configuration will remain intact.

In order to define variables to be used in the optimization process, a variable can be added by clicking on the "+" button. With the variable row added to the list, one chooses an object, then the desired property and the type of variable (IND for independent, AUX for auxiliary or DEP for dependent variables). If necessary, one can define a lower and/or upper limit for the IND variables, according to the current unit system. The variable name is the one which will be used in the expression.

DWSIM only considers bounds for independent variables. Also, if the objective function is a DEP variable, and you defined multiple DEP variables, only the first one will be used. AUX variables are used by an expression when the objective function is set to evaluate the expression. To remove a variable, a row must be selected by clicking at the row header before pressing the "-" button.

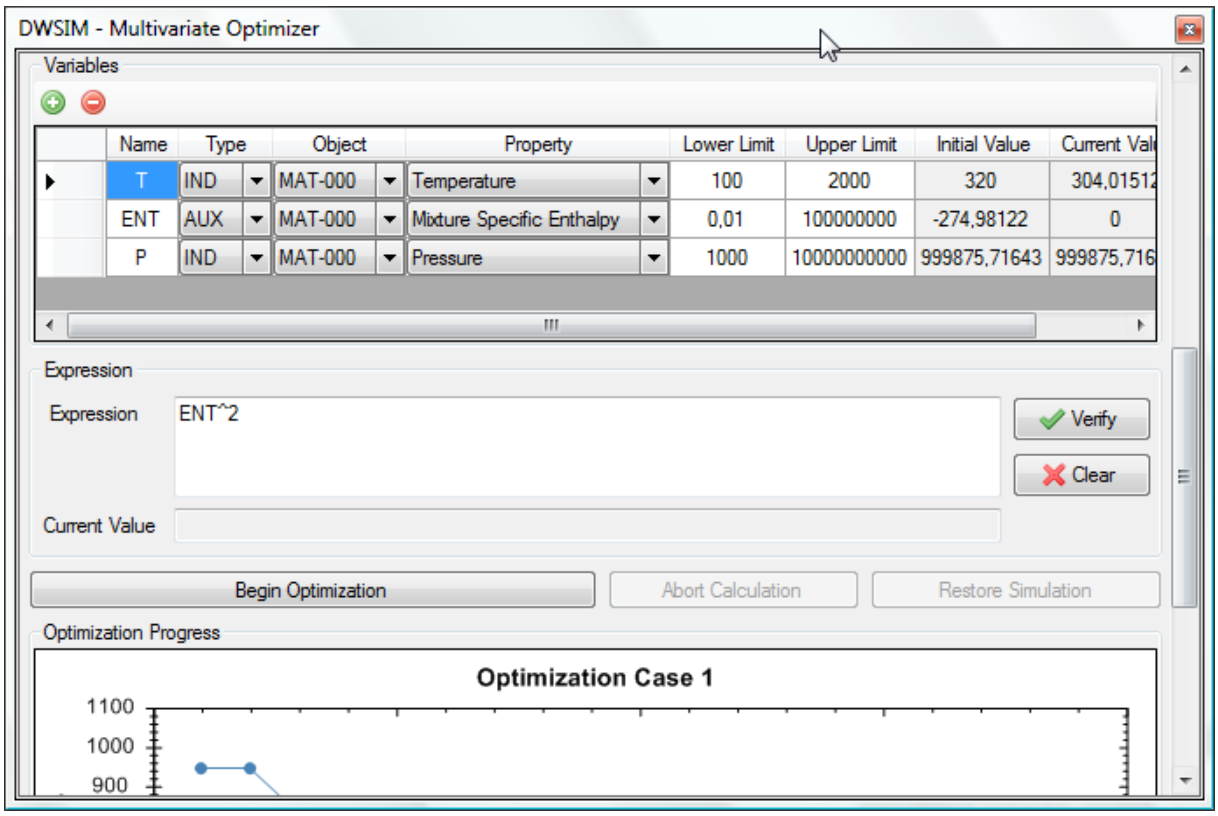



Figure 37: Multivariate Optimization Utility (2).

With all the variables defined and the case configured, the optimization can be carried out by clicking on the appropriate button - the button will become disabled. After some time, if the optimization converges, the button will become active again, indicating that the the optimization process is over.

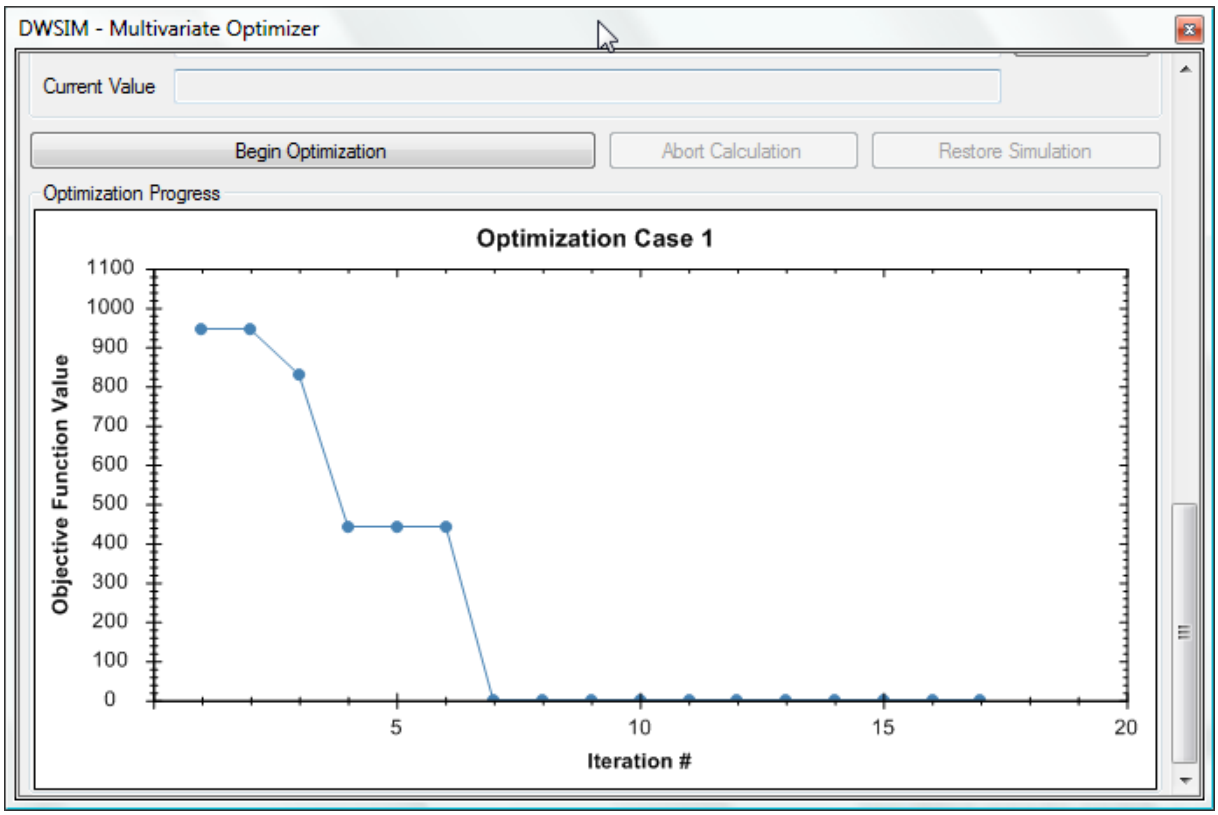



Figure 38: Multivariate Optimization Utility (3).

## 2.6. Mass and Energy Balance Summary

You can find the **Mass and Energy Balance Summary tool** in the Flowsheet Analysis menu:

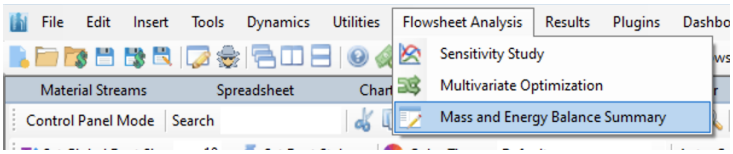



Figure 39: Mass and Energy Balance Summary tool location.

This tool gives you an overall view of the equipments and their energy consumption/generation, as well as the defined or calculated efficiencies, if applicable. There is also a list of all material streams and their associated energy flows (in SI, energy flow = enthalpy x mass flow = kJ/kg x kg/s = kJ/s = kW).

At the bottom of the tool window, you'll find the overall flowsheet mass balance residue and total flowsheet energy consumption/generation.

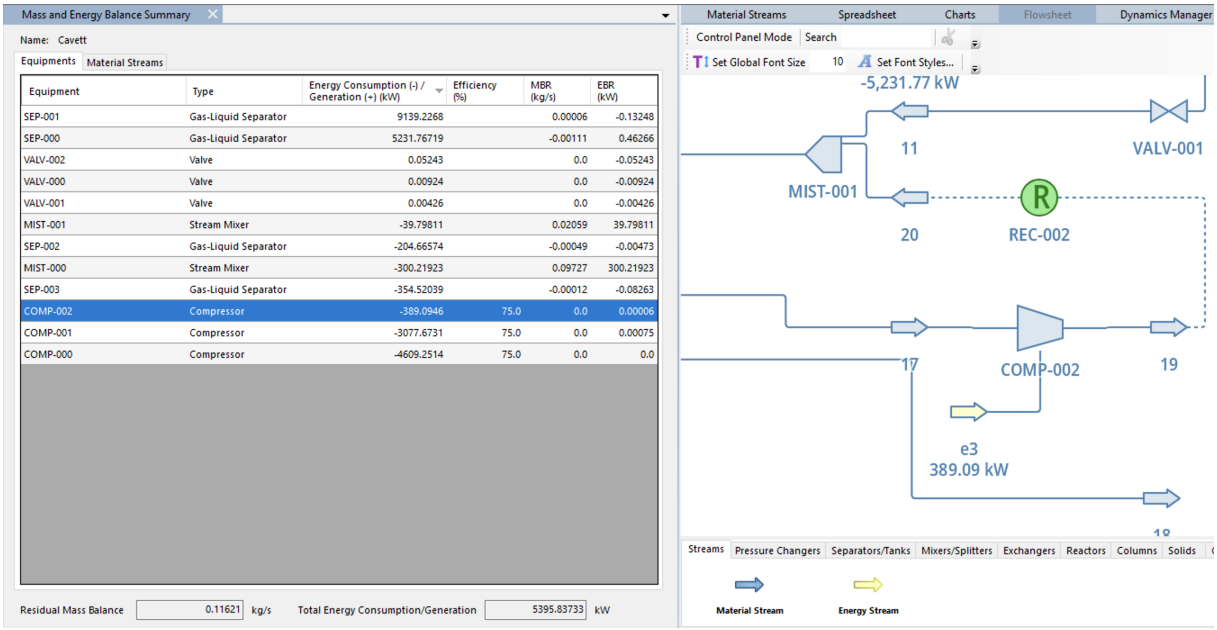



Figure 40: Mass and Energy Balance Summary tool.

## 2.7. Utilities

DWSIM includes some utilities which provides the user with more information about the process being simulated.

Utilities can be added and attached to Flowsheet objects (**Utilities > Add Utility** menu item). After being attached, they will be saved together with simulation data and restored upon reopening. Some data from the attached utilities will be available to be displayed on property tables and used on sensitivity analysis and optimization studies.

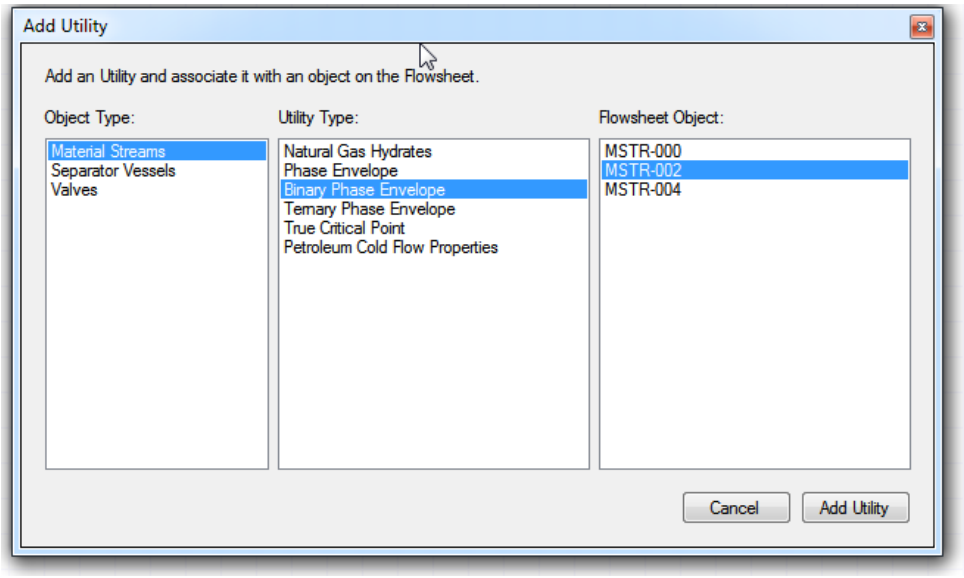



Figure 41: Attaching Utilities through the "Add Utility" window.

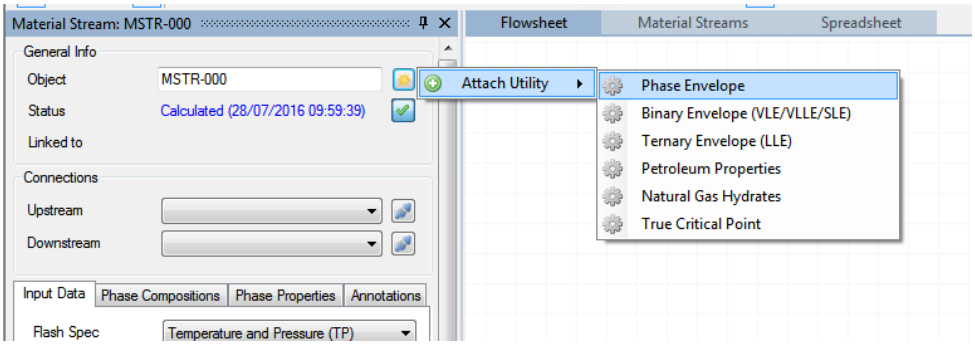



Figure 42: Attaching Utilities through the object editors.

Added/Attached Utilities will be visible on the context menu located on the object editors, on the right of the Object's Name textbox.

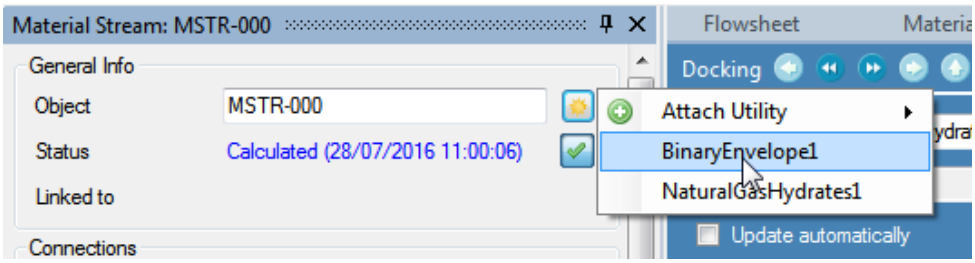



Figure 43: Accessing attached Utilities.

➙ ***True Critical Point*** - utility to calculate the true critical point of a mixture (Figure 40).

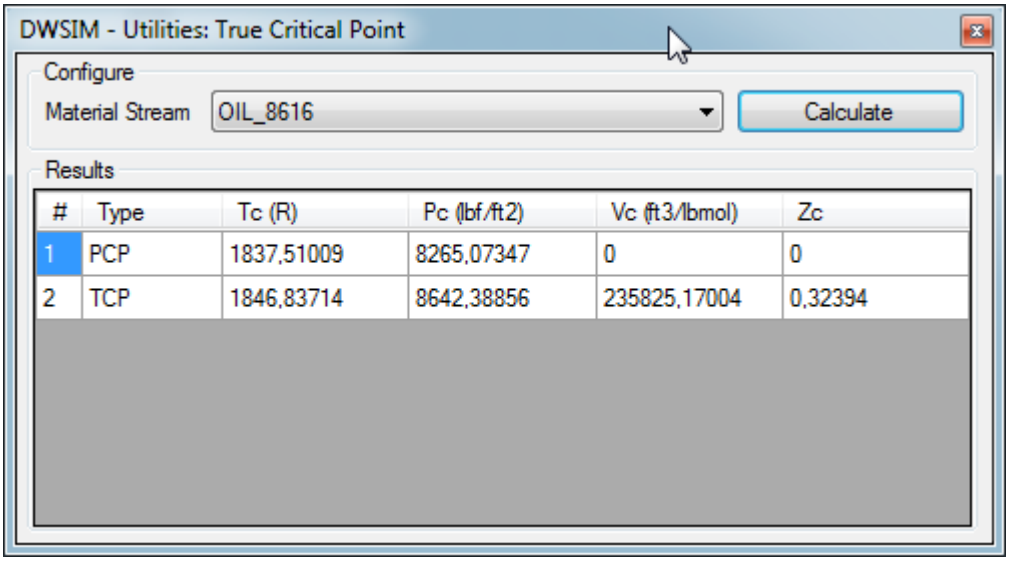



Figure 44: Utilities - True Critical Point.

➙ ***Phase Envelope*** - Material stream phase equilibria envelope calculation (Figure 40);

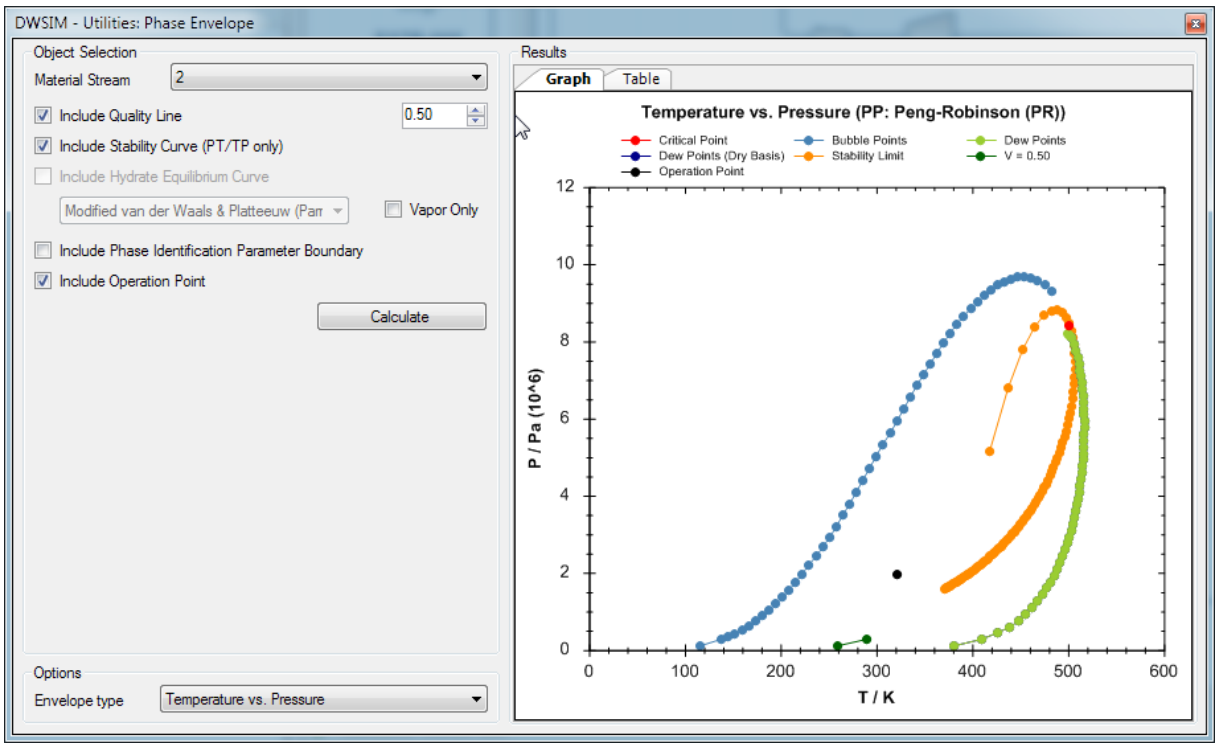



Figure 45: Utilities - Phase Envelope.

➙ ***Binary Envelope*** - special envelopes for binary mixtures (Figure 40).

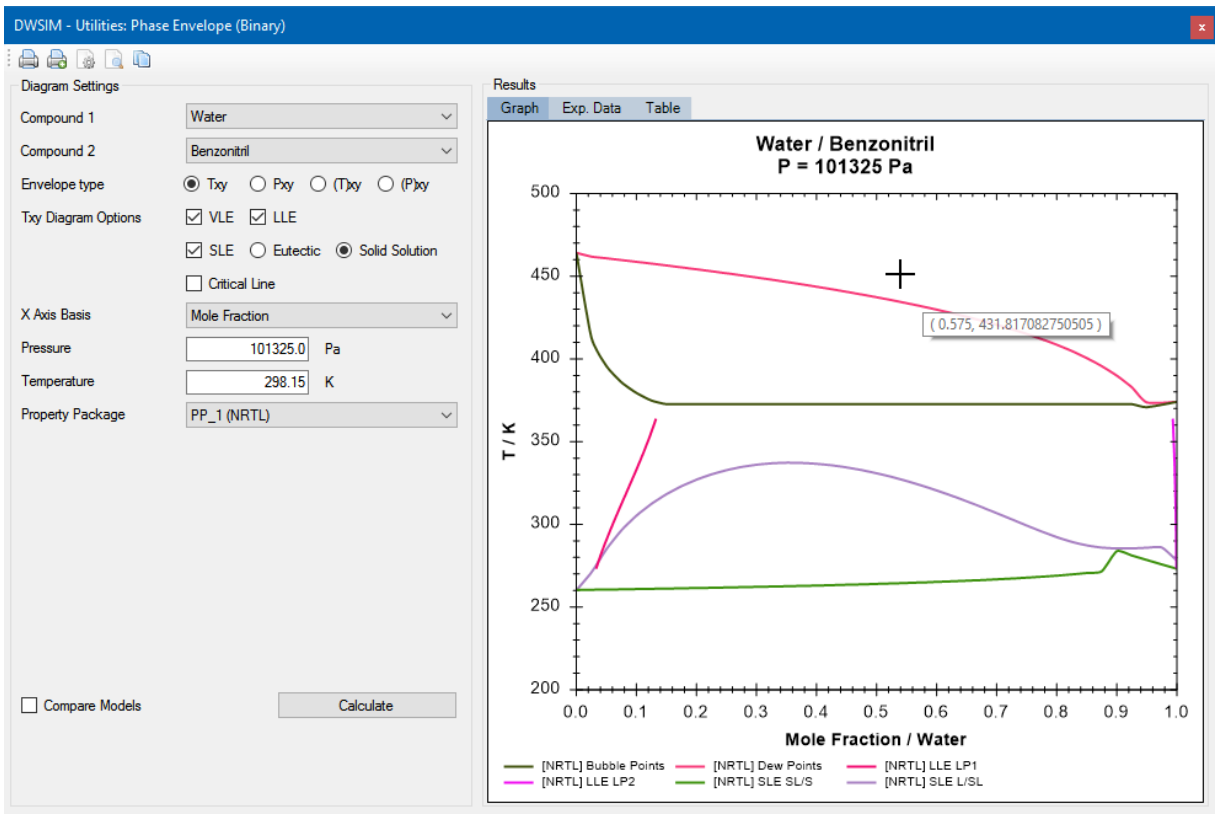



Figure 46: Utilities - Binary Envelope.

➙ ***Petroleum Cold Flow Properties*** - special properties of petroleum fractions, like cetane index, flash point, refraction index, etc. (Figure 40).

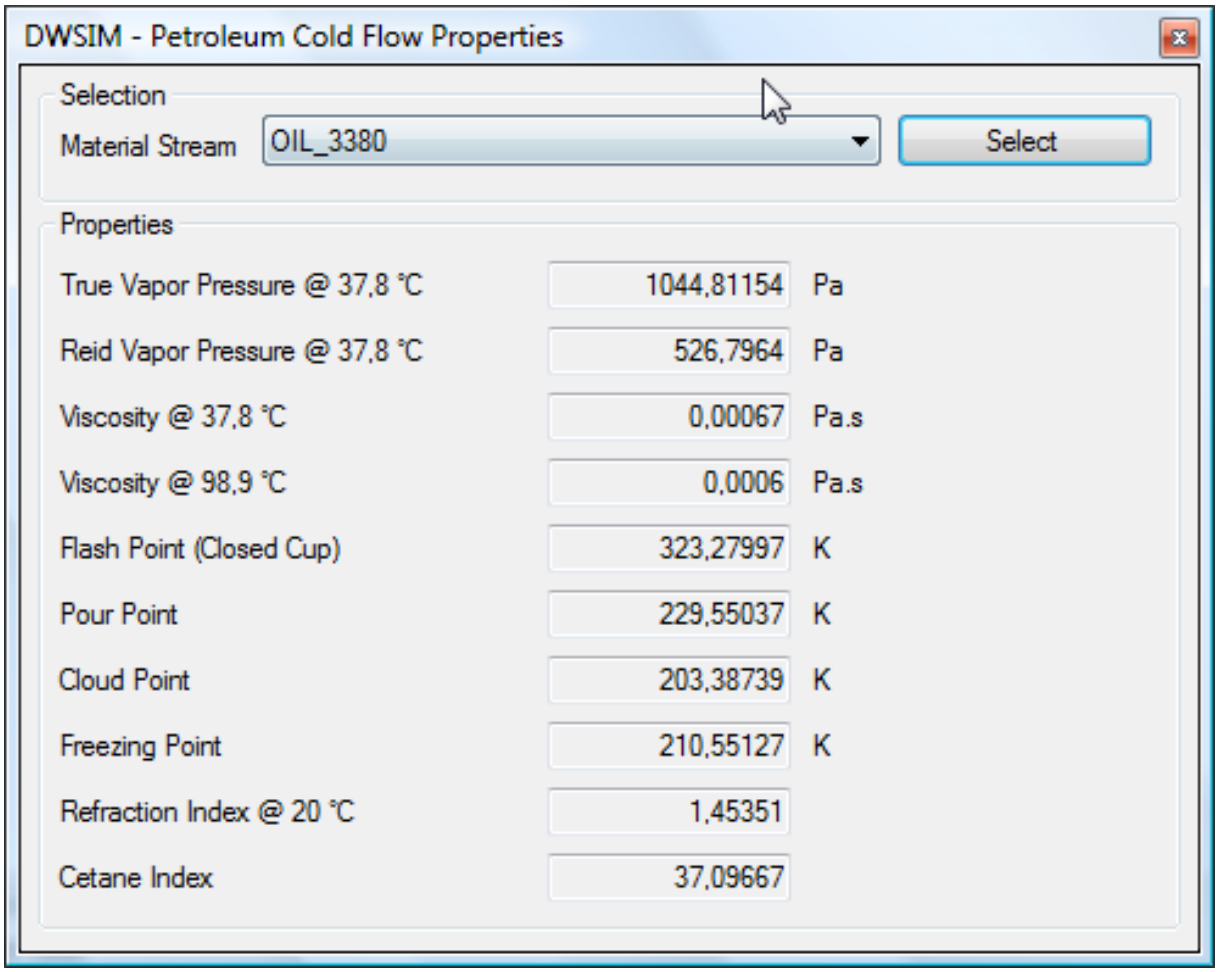



Figure 47: Utilities - Petroleum Cold Flow Properties.

Utilities calculate their properties for one object only, which is selected inside their own windows. In the majority of cases, this object must be calculated in order to be available for selection in the utility window.

***Please view DWSIM's Technical Manual for more details about the models and methods used by the Utilities.***

## 2.8. Chemical Reactions

DWSIM classifies chemical reactions in three different types: Conversion, where the conversion of a reagent can be specified as a function of temperature; Equilibrium, where the reaction is characterized by an equilibrium constant K, and Kinetic/Heterogeneous Catalytic, where the reaction is led by a velocity expression which is a function of concentration of reagents and/or products and/or a catalyst.

***Please view DWSIM's Technical Manual and Equipment and Utilities Guide for more details about chemical reactions and reactors, respectively.***

Chemical reactions in DWSIM are managed through the **Chemical Reactions Manager** (**Simulation Settings > Reactions** panel) (Figure 40):

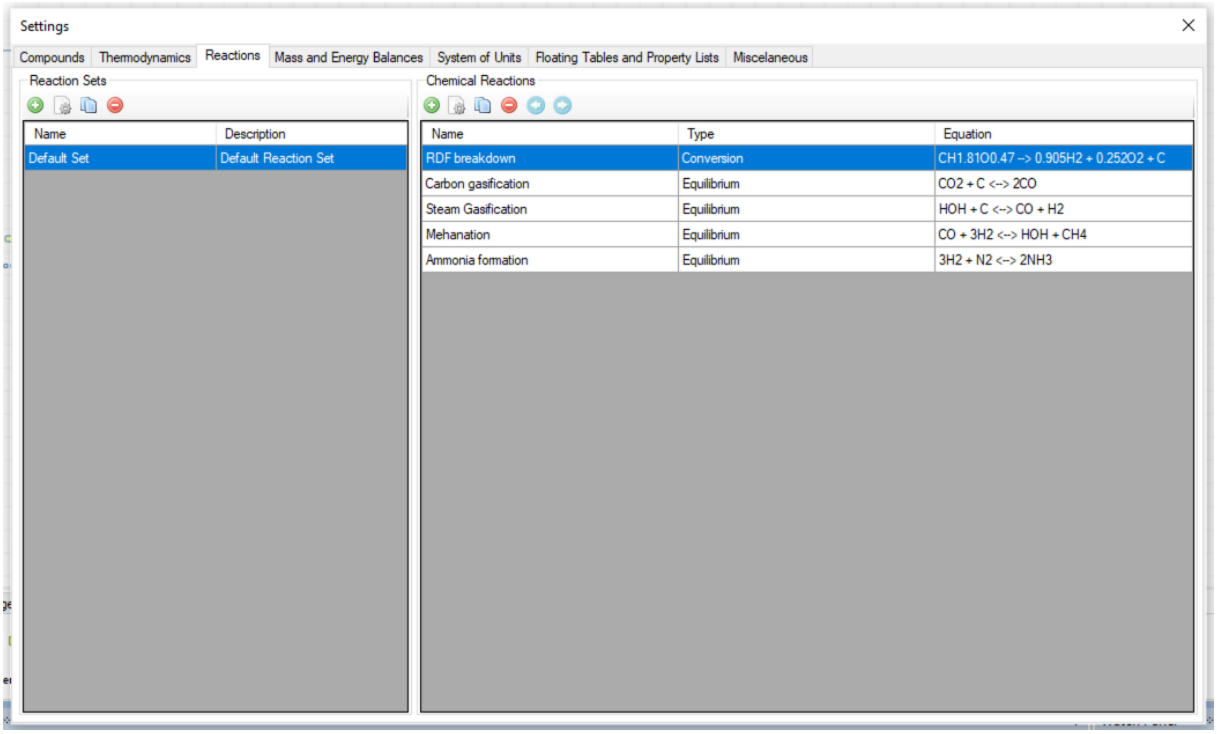



Figure 48: Chemical Reactions Manager.

The user can define various reactions which are grouped in *Reaction Sets*. These reaction sets list all chemical reactions, and the user must activate only those he/she wants to become available for one or more reactors, since the reactor's parameter is the **reaction set** and not the chemical reactions themselves. In the reaction set configuration window it is also possible to define the reaction ordering. Equal indexes define parallel reactions (Figure 40):

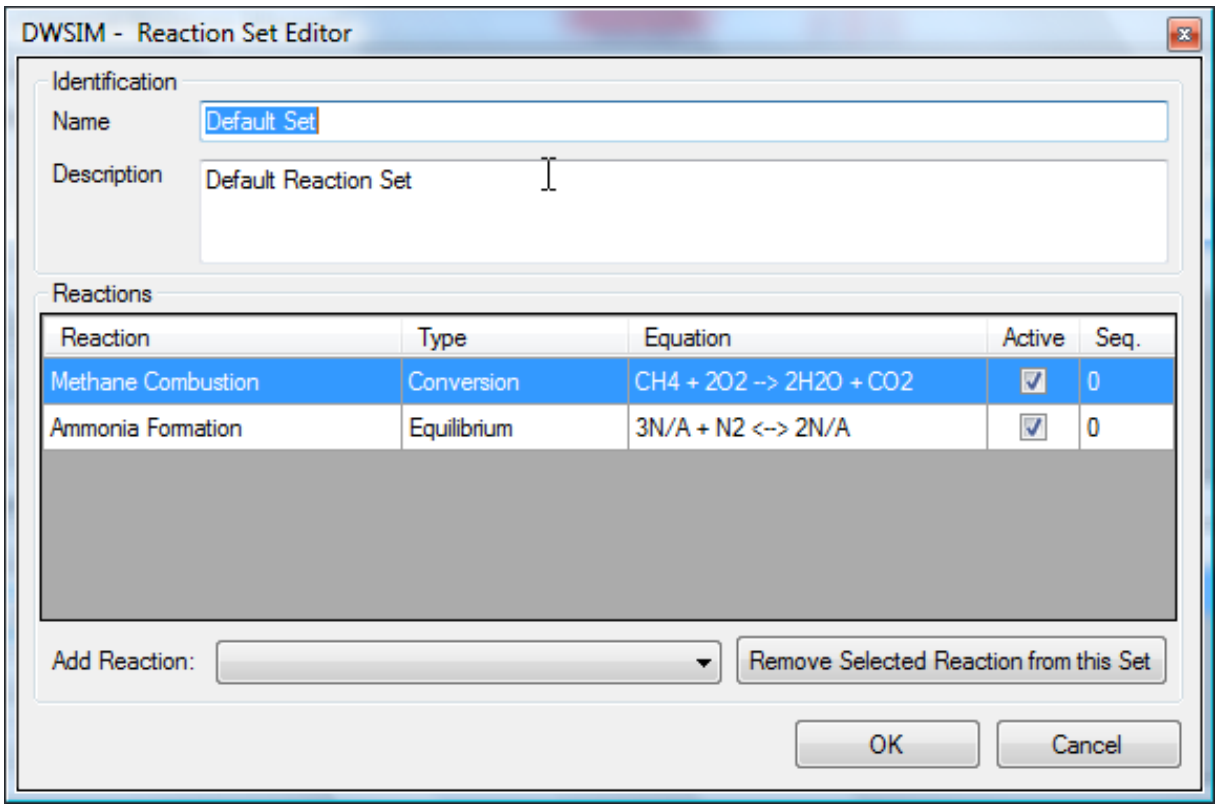



Figure 49: Reaction Set editor.

When the reactions and their respective reaction sets are correctly defined, the sets will be available for selection in the property window of a reactor in the simulation. When requested for a calculation, the reactor will then look for active reactions inside the selected set.

## 2.9. Characterization of Petroleum Fractions

DWSIM provides three tools for characterization of petroleum fractions. One of them characterizes C7+ fractions from bulk properties (Figure 40). The other characterizes the oil from an ASTM or TBP distillation curve (Figure 40). There is also a tool to create pseudocompounds from tabular data.

***- Characterization from bulk properties*** The method itself requires a minimum of information to generate the pseudocomponents, though the more data the user provides, the better will be the results (Figure 40). It is recommended that the user provides the specific gravity of the C7+ fraction at least. Viscosity data is also very important.

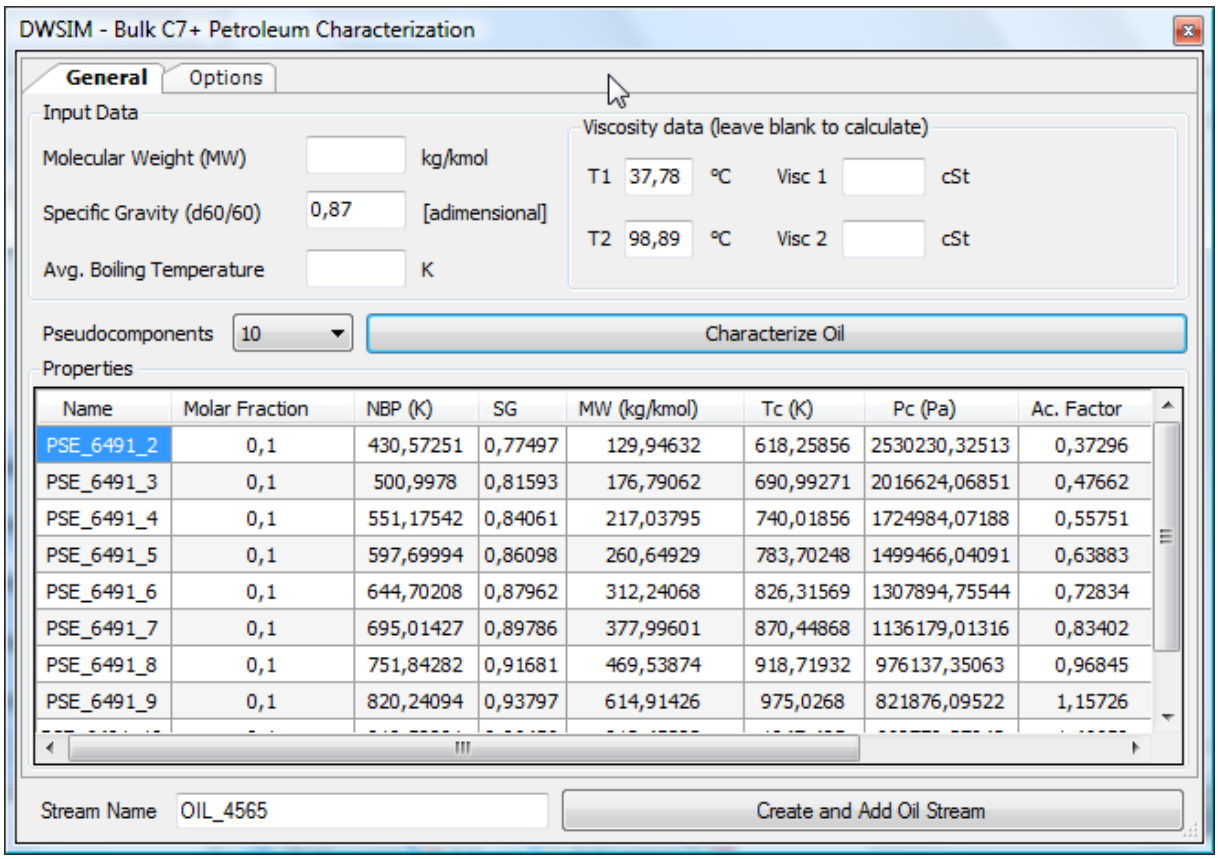



Figure 50: C7+ petroleum fraction characterization utility.

***- Characterization from distillation curves*** This tool gets data from an ASTM or TBP distillation curve to generate pseudocomponents. It is also possible to include viscosity, molecular weight and specific gravity curves to enhance the characterization.

The interface has a wizard-like style, with various customization options (Figure 40):

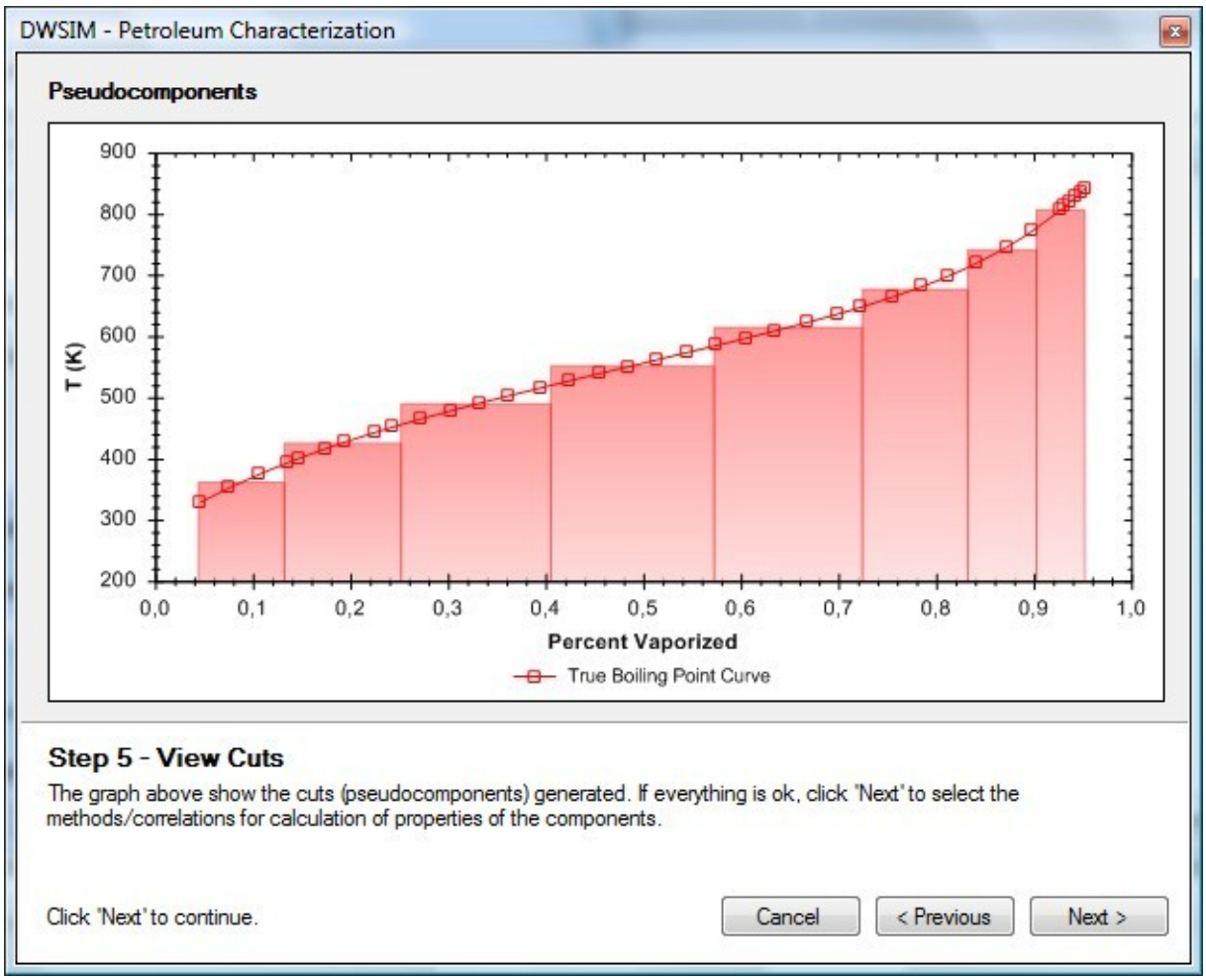



Figure 51: Characterizing petroleum from distillation curves.

After the pseudocomponents are created, a material stream with a defined composition is also created, which represents the characterized petroleum fraction.

***The hypo and pseudocomponents are available for use only in the simulation in which they were generated, even if there is more than one opened simulation in DWSIM. Nevertheless, the user can export these components to a file and import them into another simulation.***

***- Bulk/Batch creation of pseudocomponents/pseudocompounds*** The Bulk Create Pseudocompounds tool can be used to create pseudocompounds in a batch when you have the required data in a tabular format, or only part of the data. If some data is missing, DWSIM can estimate it before exporting the compounds to XML, JSON or add them to the Flowsheet (Figure 40).

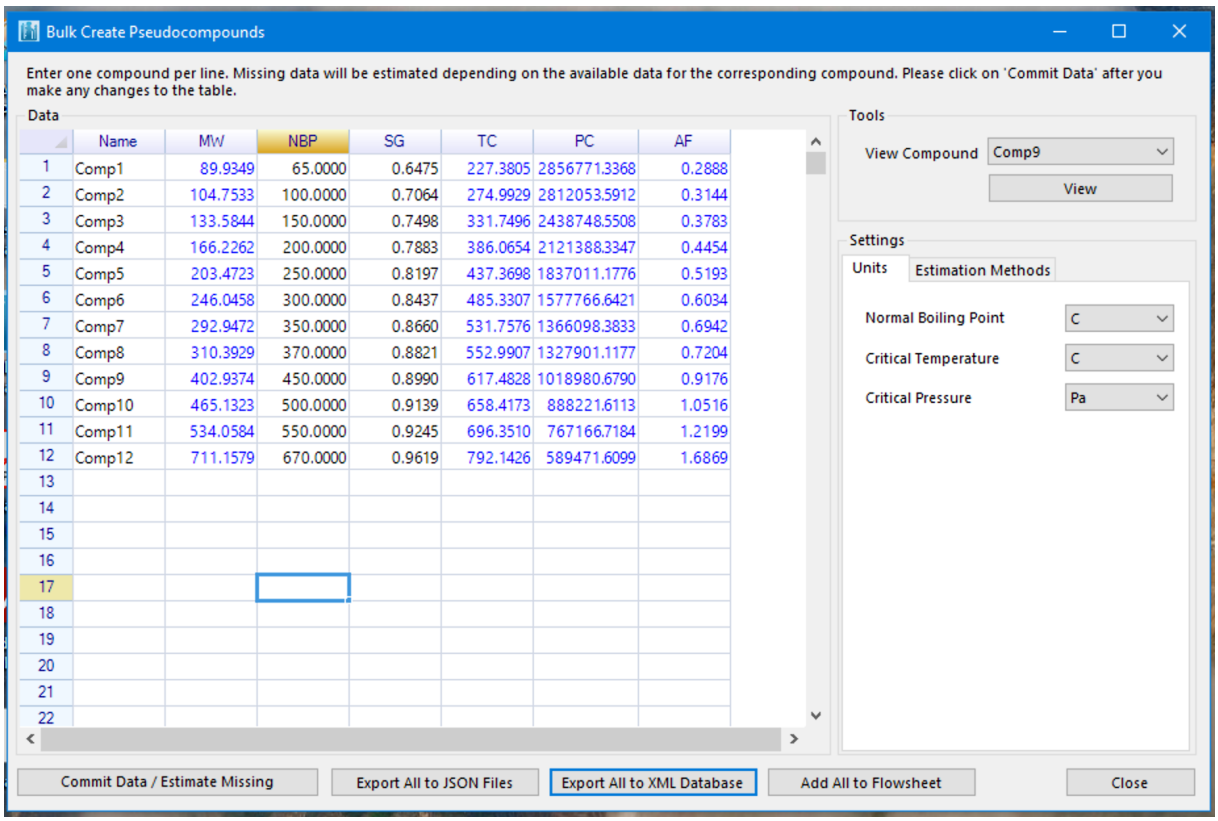
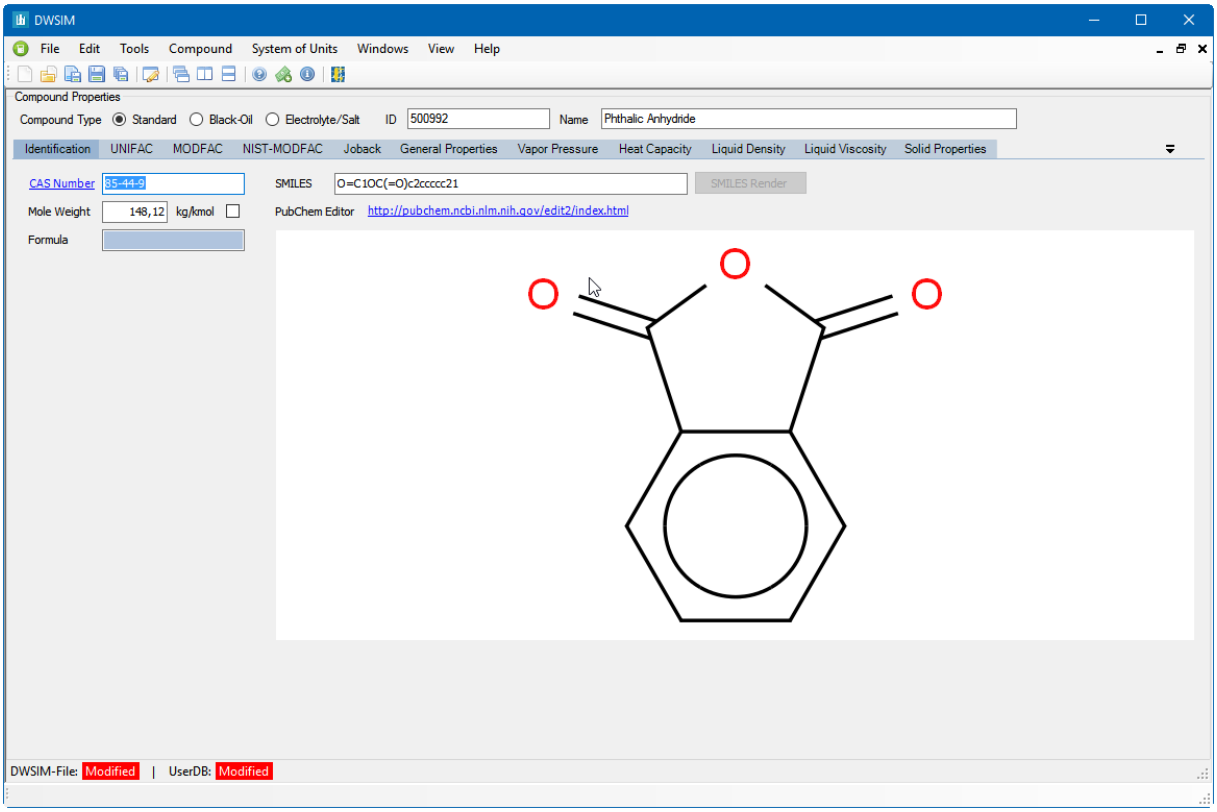



Figure 52: Bulk creation of pseudocomponents/pseudocompounds.

# 3. Compound Creator

## 3.1. Introduction

The new Compound Creator Utility is an all-in-one replacement for the User Compound and Hypothetical Creator utilities in DWSIM. It enables usage of experimental data as well as UNIFAC structure information to calculate and/or estimate all constant and temperature-dependent properties for a compound that isn't available on any of the default databases (DWSIM and ChemSep).

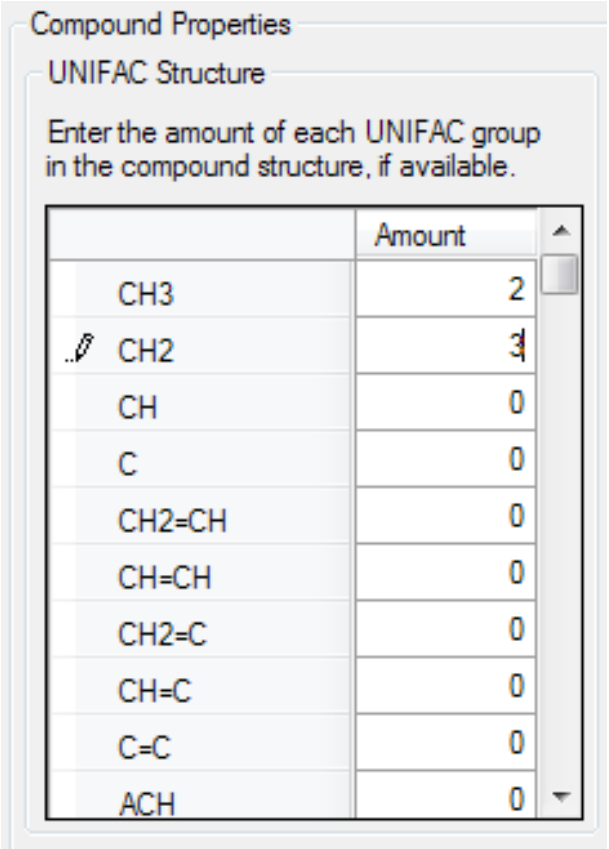


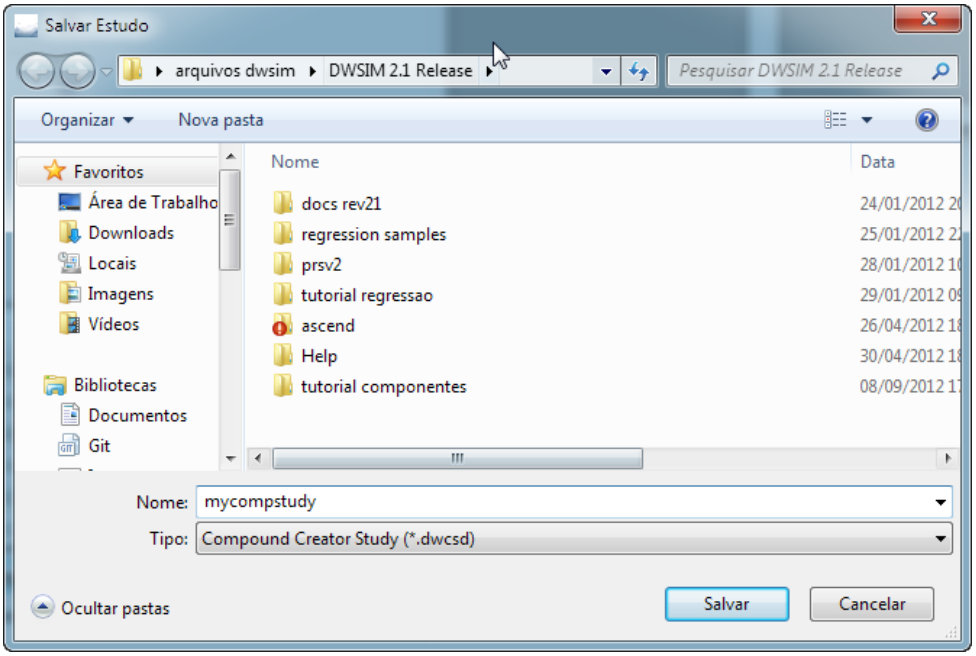

To open the utility, you can use the corresponding button on the Welcome screen or go to File > New > Compound Creator Study.

## 3.2. Data Input Constant Properties

Enter an unique ID for the compound. It can be any integer number (a random 5-digit integer is ok). Enter a name for the compound.

DWSIM makes it easier to calculate most properties if you enter some UNIFAC structure information. With UNIFAC structure info, DWSIM will calculate all properties that have its adjacent checkbox checked. Nothing stops you from entering your own value on these textboxes, but if the checkbox is checked and you change the UNIFAC structure info, DWSIM will update the value with its own calculation.



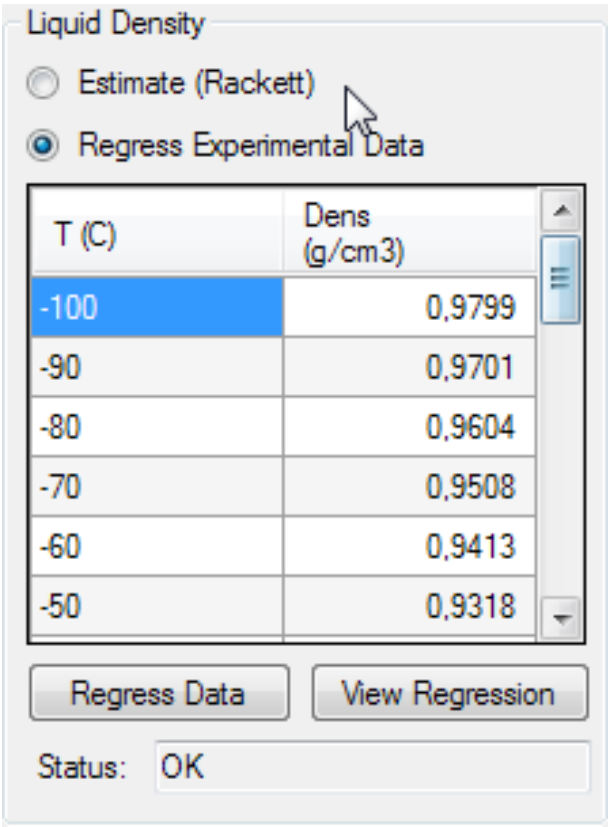



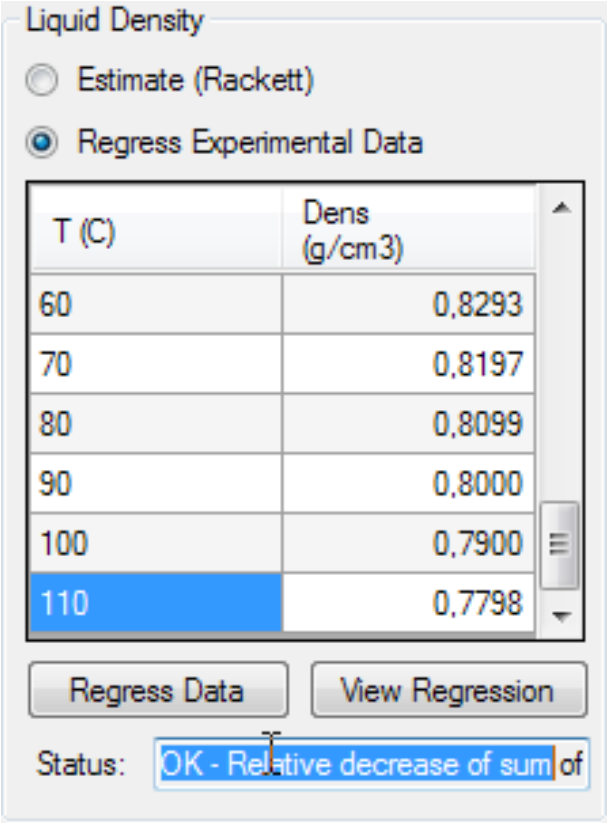

Property textboxes that have a blue background are not essential, but are required if you're planning to use your compound in a simulation with PC-SAFT, Chao-Seader and/or Grayson-Streed models, for example.

## 3.3. Temperature-dependent Properties

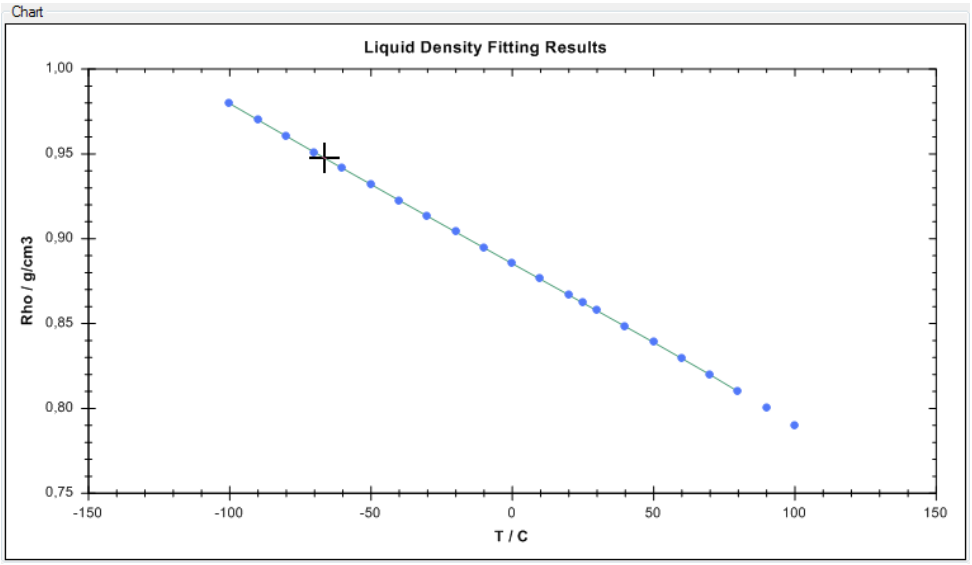
By default, temperature-dependent properties will be calculated by internal DWSIM routines, but if you have some tabulated data available, you can use it to make DWSIM generate coefficients and use them instead.

For instance, let's say that you have some liquid density data available. You can input it on the Liquid Density table (just make sure that the current units are the same as yours) and click on "Regress". DWSIM will let you know if anything went wrong during the regression on the textbox below the buttons.





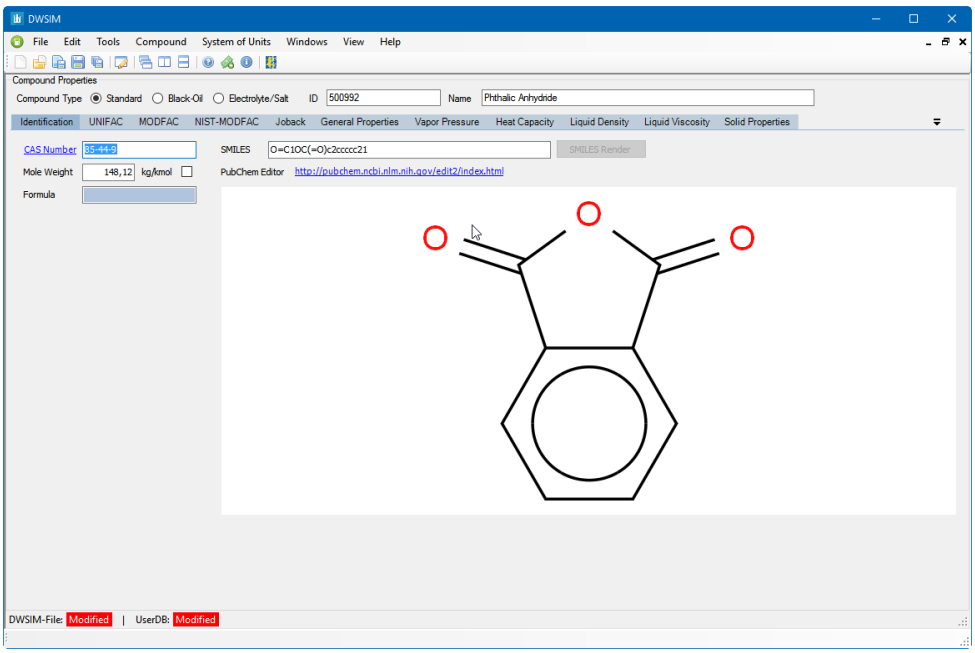

To view the regressed data, click on "View Regression". You should see your points and a line representing the fitted equation that will be used by DWSIM on your simulations.



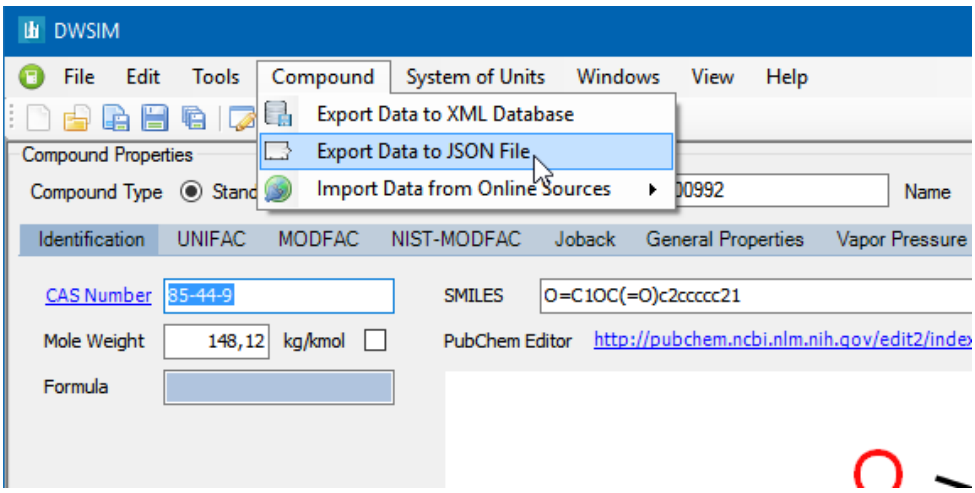

## 3.4. Importing Data from Online Sources

You can import compound data from some online sources like the Korean KDB Thermo Database, the Cheméo Database and UNIFAC/MODFAC Structure Data from the Dortmund Data Bank Online Interface. Go to Compound > Import Data from Online Sources and explore the available options.

After you finish importing data from the online sources, any data previously input on the textboxes will be overriden for the properties you've selected.

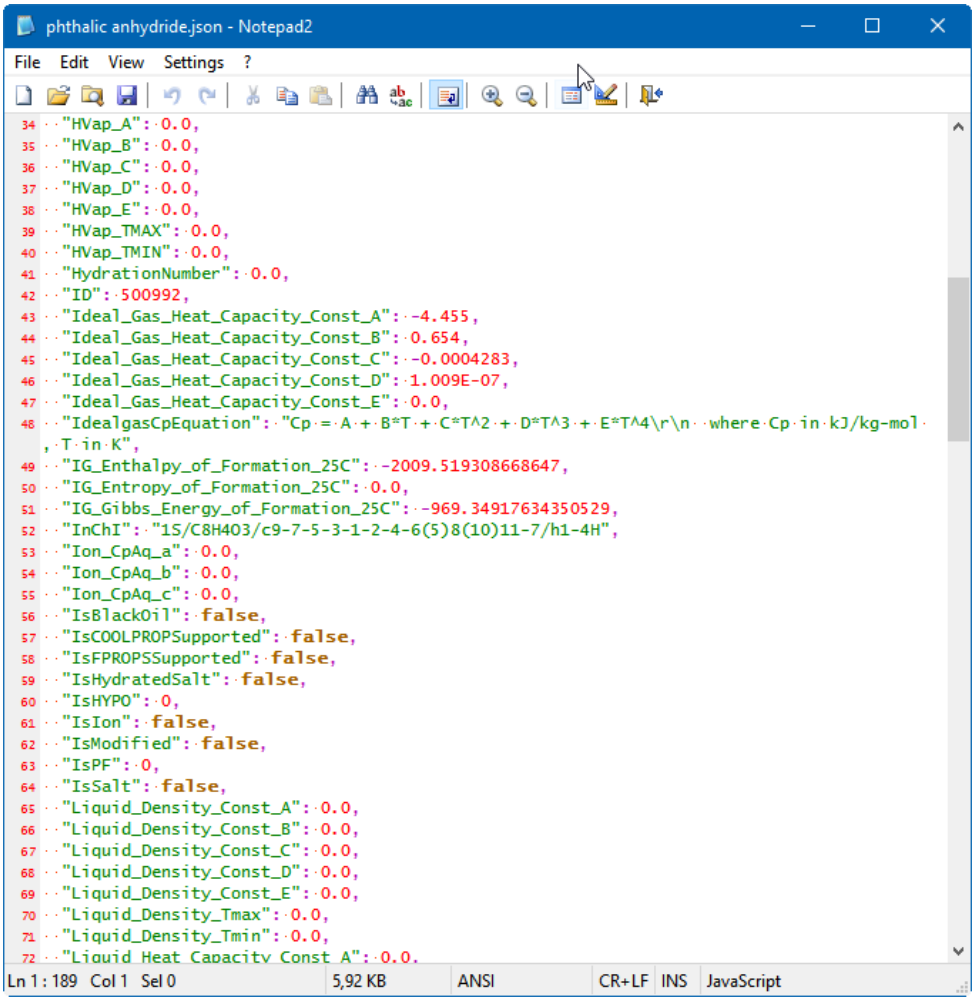



## 3.5. Creating the Compound

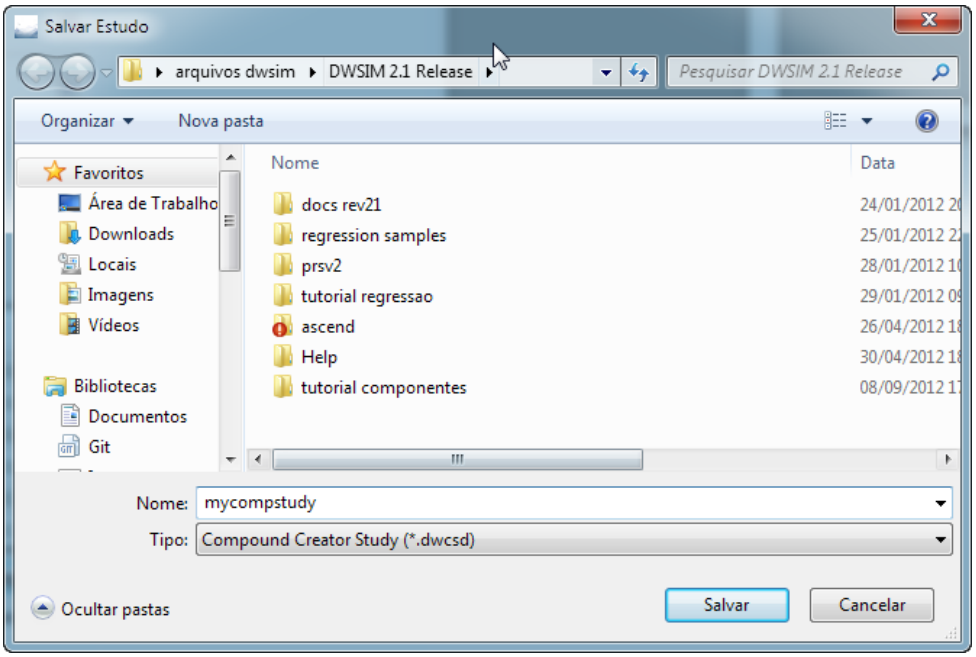



If everything is ok, you can save your compound data to a XML database file. Go to Compound > Export Data to XML Database. The XML database has the advantage of handling multiple compounds in a single file.

You can also export your compound to a single JSON file. The JSON file format is very easy to edit if you need to:



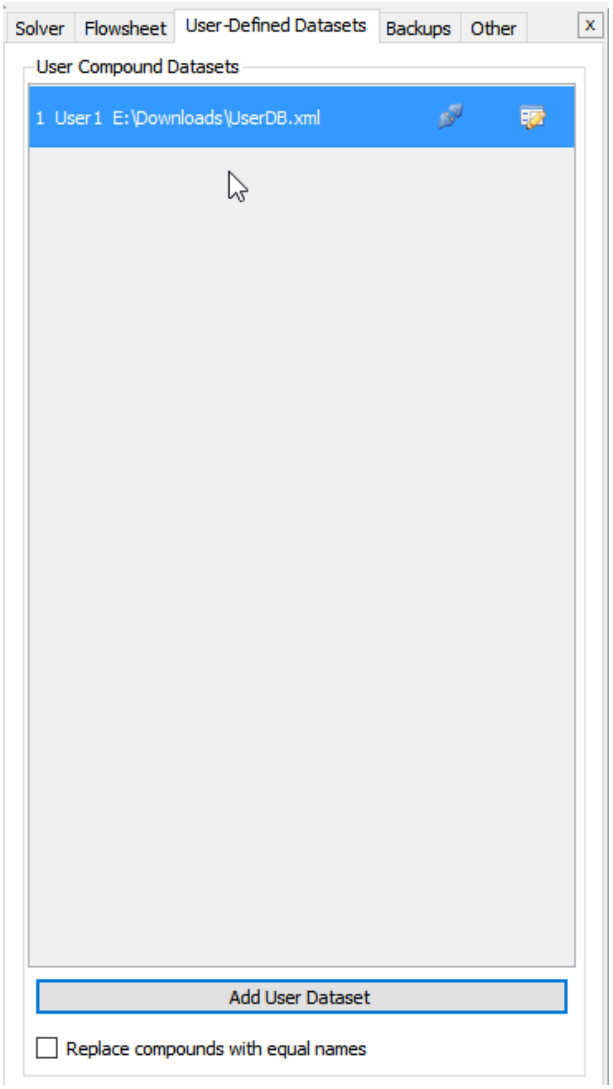

You can also save your compound creator data to a file if you think you'll need to change it later, or use it as a starting point for another compound (File > Save As):



## 3.6. Adding the Compound to a Simulation

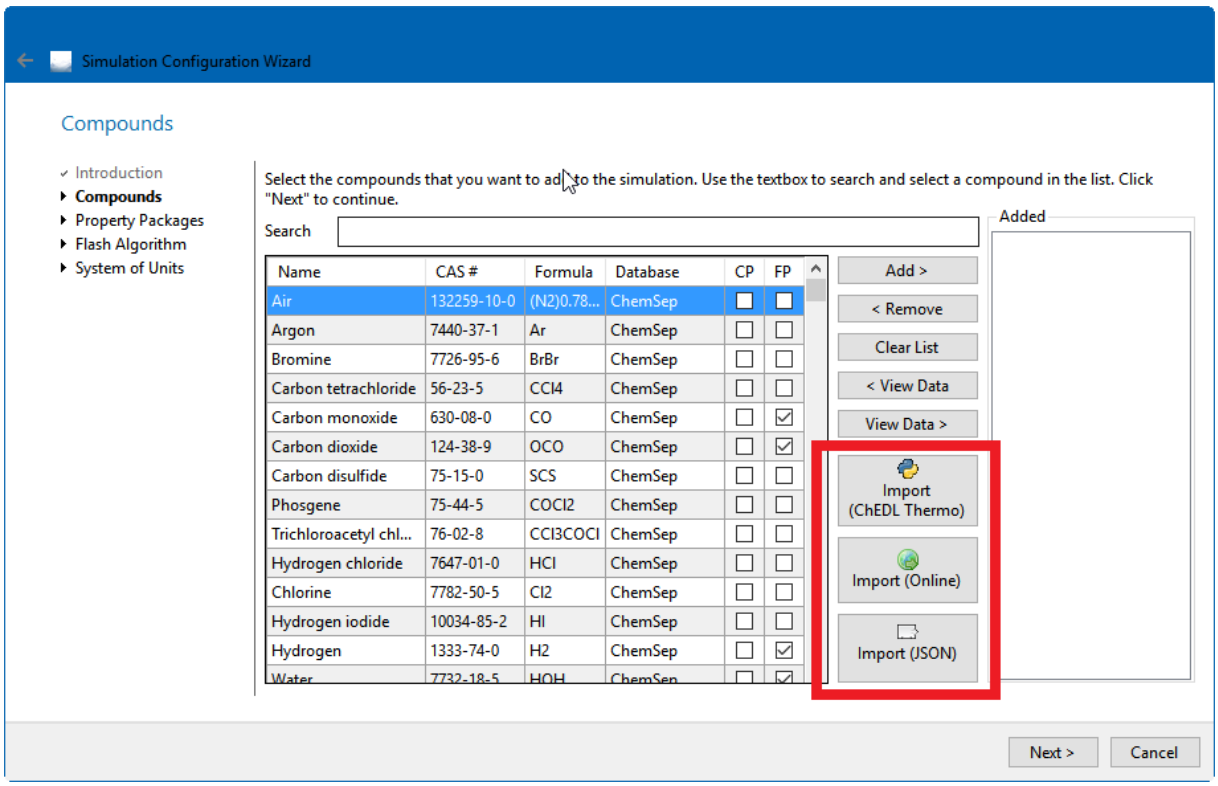
### 3.6.1. Loading Compounds from XML Databases

To load your compound into a simulation, go to Settings > General Settings > User-Defined Datasets and click on Add User Dataset. Select your XML database file and click Open. Create a new simulation and check if your compound is on the list (it should be the last one):




### 3.6.2. Loading Compounds from JSON files

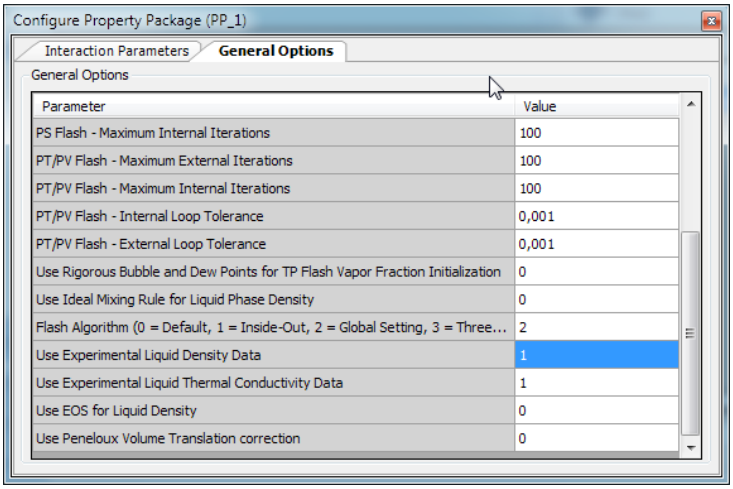
You can load a compound from a JSON file directly through the Compounds section in the Simulation Configuration Wizard and in the Simulation Settings Panel.



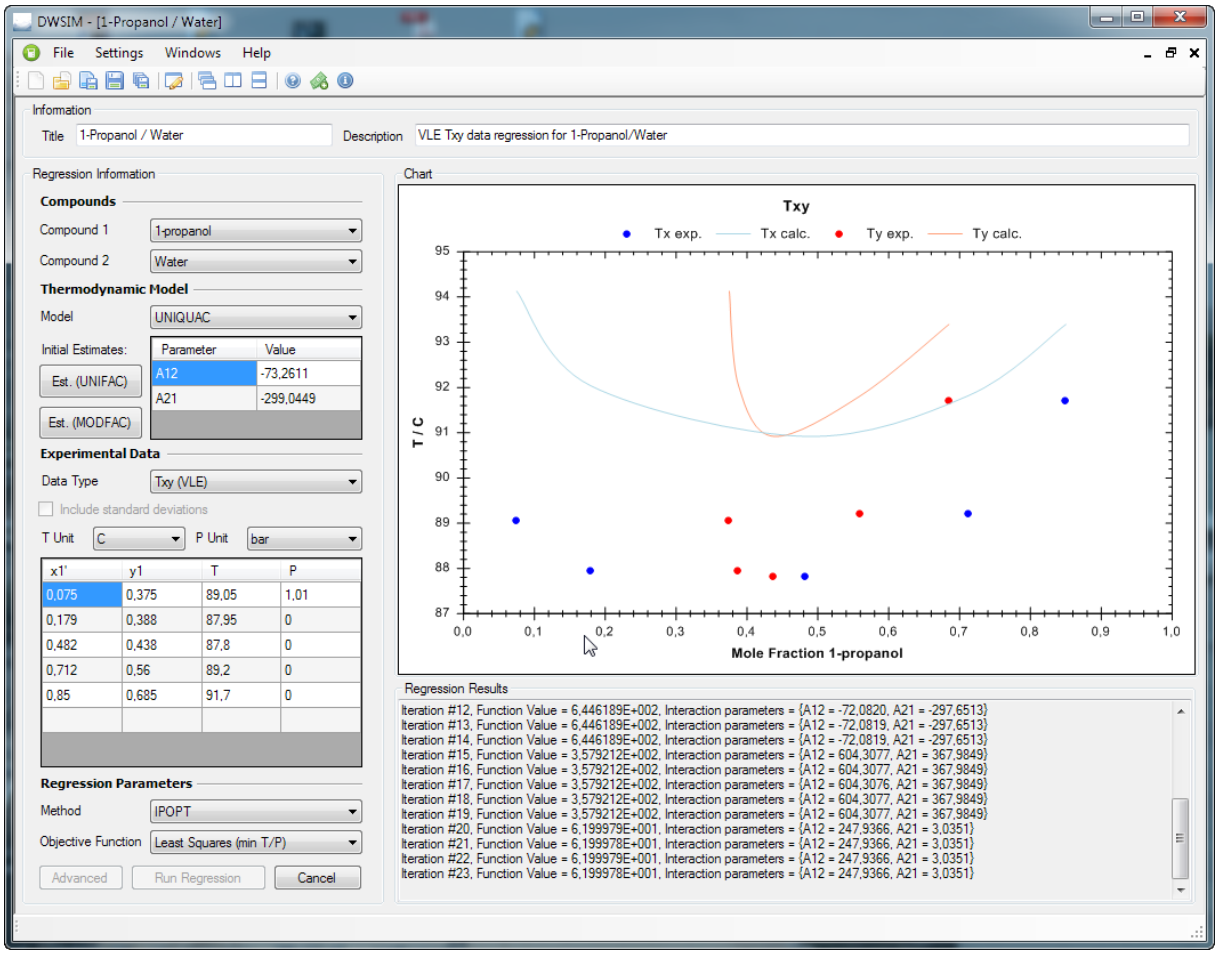

### 3.6.3. Remarks

When you add your compound to the simulation, you can use the Pure Compound Property Utility to edit the data, but those changes will be made only for the current simulation. If you need to make perpertual changes, you'll have to use the Compound Creator Utility to save your compound, or edit the XML or the JSON file directly and reload it.

If you input tabulated liquid density data to create an experimental curve, remember to activate the "Use Experimental Liquid Density Data" option on the Property Package configuration window, otherwise DWSIM will use the Rackett correlation for liquid density estimations.



# 4. Data Regression




The Data Regression Utility supports regression of experimental binary data for determination of interaction parameters for the following models:

- PC-SAFT
- Peng-Robinson
- Peng-Robinson-Stryjek-Vera 2
- Soave-Redlich-Kwong
- UNIQUAC
- NRTL
- Lee-Kesler-Plöcker

The following data sets are supported:

- VLE Temperature and mole fractions (Txy)
- VLE Pressure and mole fractions (Pxy)

- VLE Temperature, Pressure and mole fractions (TPxy)
- LLE Temperature and mole fractions (Txx)
- LLE Pressure and mole fractions (Pxx)
- LLE Temperature, Pressure and mole fractions (TPxx)

The Data Regression Utility also has some handy additional features like:

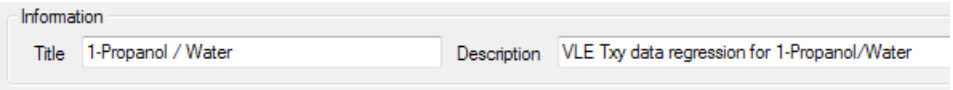
- Calculation of initial values for the binaries using UNIFAC/MODFAC structure information
- Calculation of missing experimental data using known models/binaries for determination of parameters for other models
- Optimization method selection
- Objective Function selection (Least Squares of temperature/pressure plus vapor fractions)

The Data Regression Utility also supports loading and saving of a regression study/case for later use. Currently, there is no way to export the generated binaries for a simulation or a database - you'll have to do this manually.

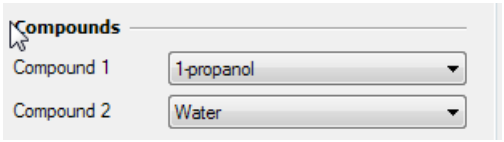
## 4.1. Data Regression - How To

In this example we will regress Txy experimental data for determination of UNIQUAC interaction parameters for the 1-Propanol/Water binary.

**Title and Description** Enter a title and a description for you regression case.

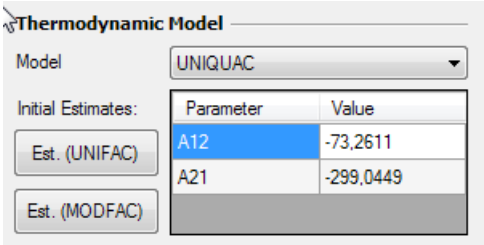



**Compound Selection** Select the compounds 1-Propanol and Water on the corresponding combo boxes.



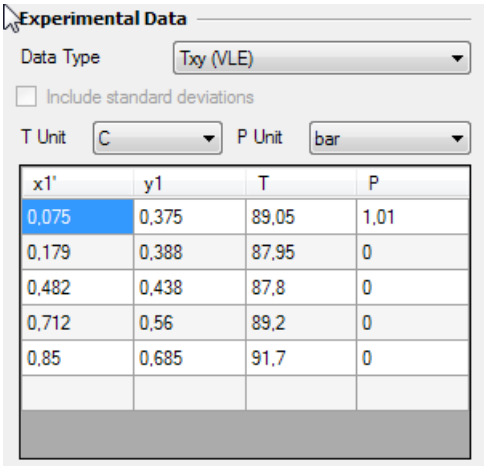

**Model Selection** Select UNIQUAC in the combobox for the Thermo Model. In the table below you can input your own initial estimates for the binaries, in order to help the optimizer on finding the values that minimize the difference between calculated and experimental values. You can also use the UNIFAC structure information to estimate initial values through UNIFAC and/or Modified UNIFAC models.

If the ***Use Ideal Vapor Phase Model*** is checked, DWSIM will calculate the vapor phase fugacities using the ideal model ($\phi = 1$), otherwise it will do it using the Peng-Robinson EOS.



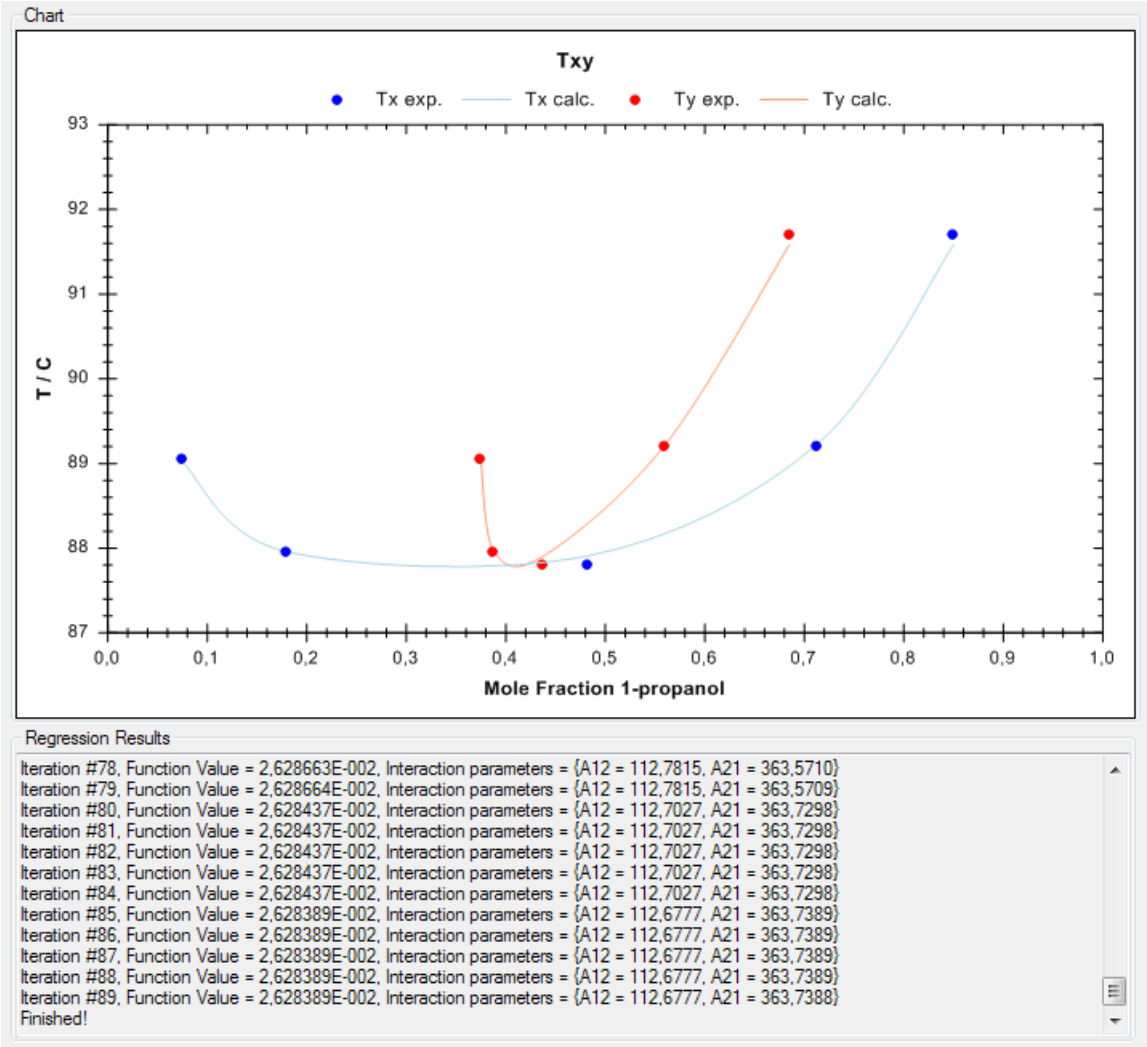

**Experimental Data Input** In the 'Experimental Data' section, select your data type (Txy), and the units for Temperature and Pressure. x1, x2 and y1 will always be in mole fractions.

Input your experimental data in the table below. If you selected the Txy data type, the pressure only needs to be informed on the first cell of the corresponding column. The same applies for Pxy data, you'll input all pressures and the temperature only has to be informed on the first cell.

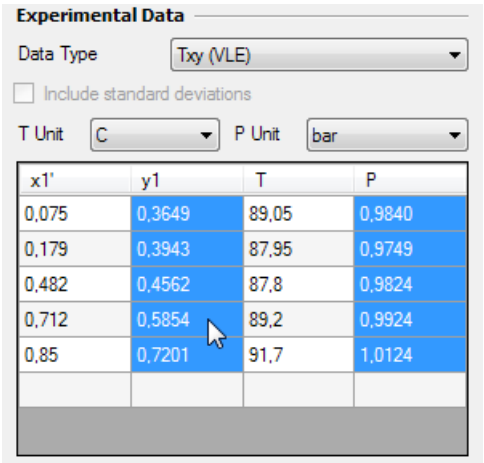



**Clearing cell and rows** DWSIM will only accept the last line as a blank one. If you want to remove data from the table, press the "Del" key after selecting the cells that you want to clear. To remove an entire row, select all cells on the row and press "Shift+Del".

**Calculating missing data** DWSIM can "complete" your experimental data set by using models with known interaction parameters. Say, for example, that you want to add a few data pairs but you only have the liquid phase mole fraction and pressure information. In this case you can use DWSIM to estimate the missing vapor phase mole fraction and temperature.



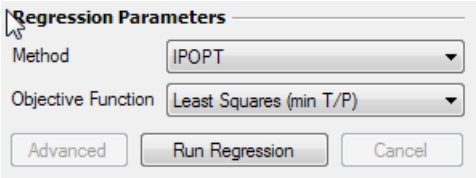

To estimate the missing data, just select the corresponding cells and click with the right mouse button, selecting the model that you want to use to calculate the values. DWSIM will then fill the selected cells with the calculated data.




**Regress data** In the "Regression Parameters" section, select the optimization method from the "Method" combobox and the Objective Function from the corresponding combobox. Currently, only the

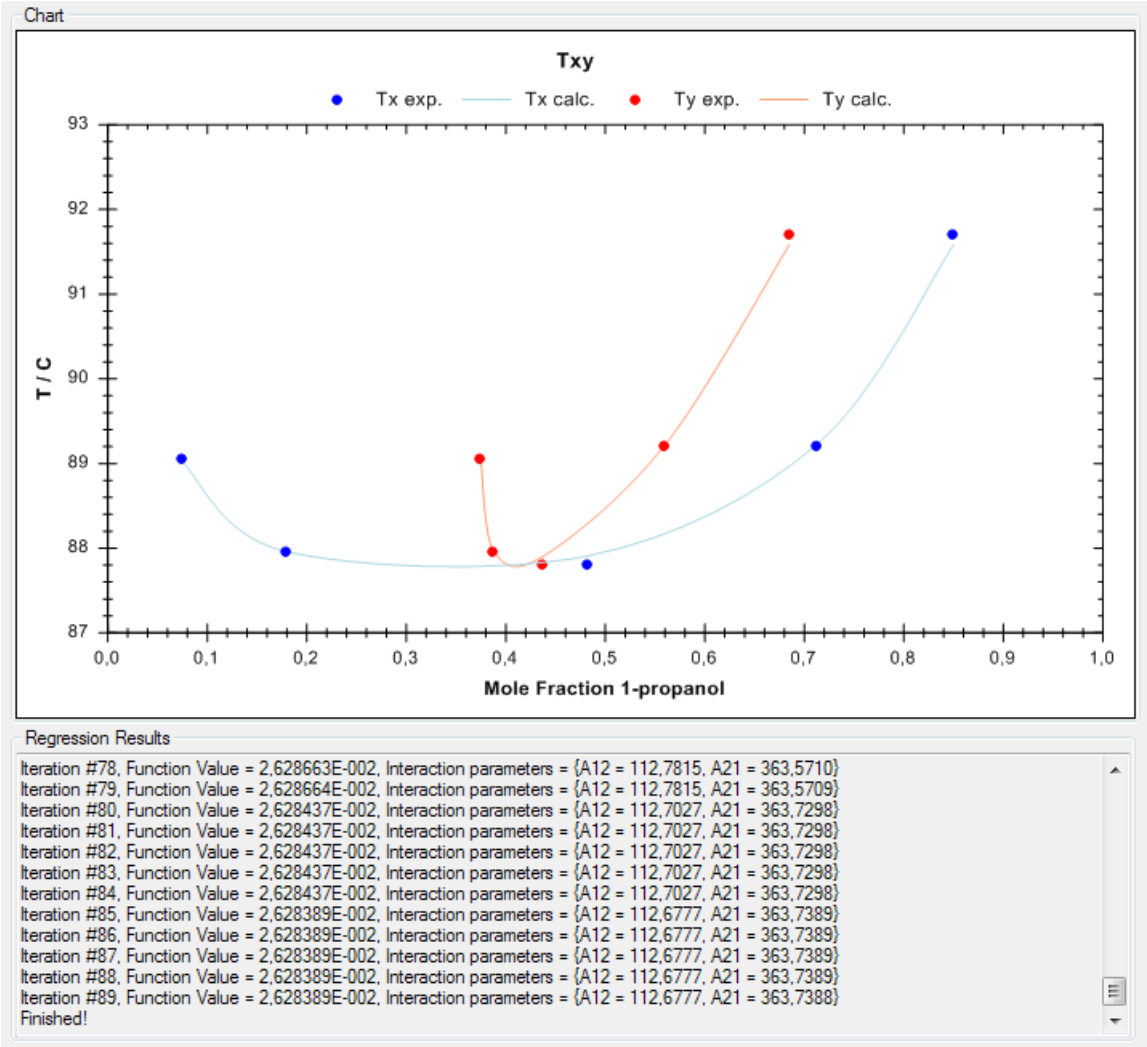
Least Squares obj. function is available. Here, You'll have the follwing options:

- Least Squares (min T/P): minimizes the difference between calculated and experimental pressures/temperatures ONLY
- Least Squares (min y): minimizes the difference between calculated and experimental vapor phase mole fractions ONLY
- Least Squares (min T/P+y): minimizes the difference between calculated and experimental pressures/temperatures plus vapor phase mole fractions

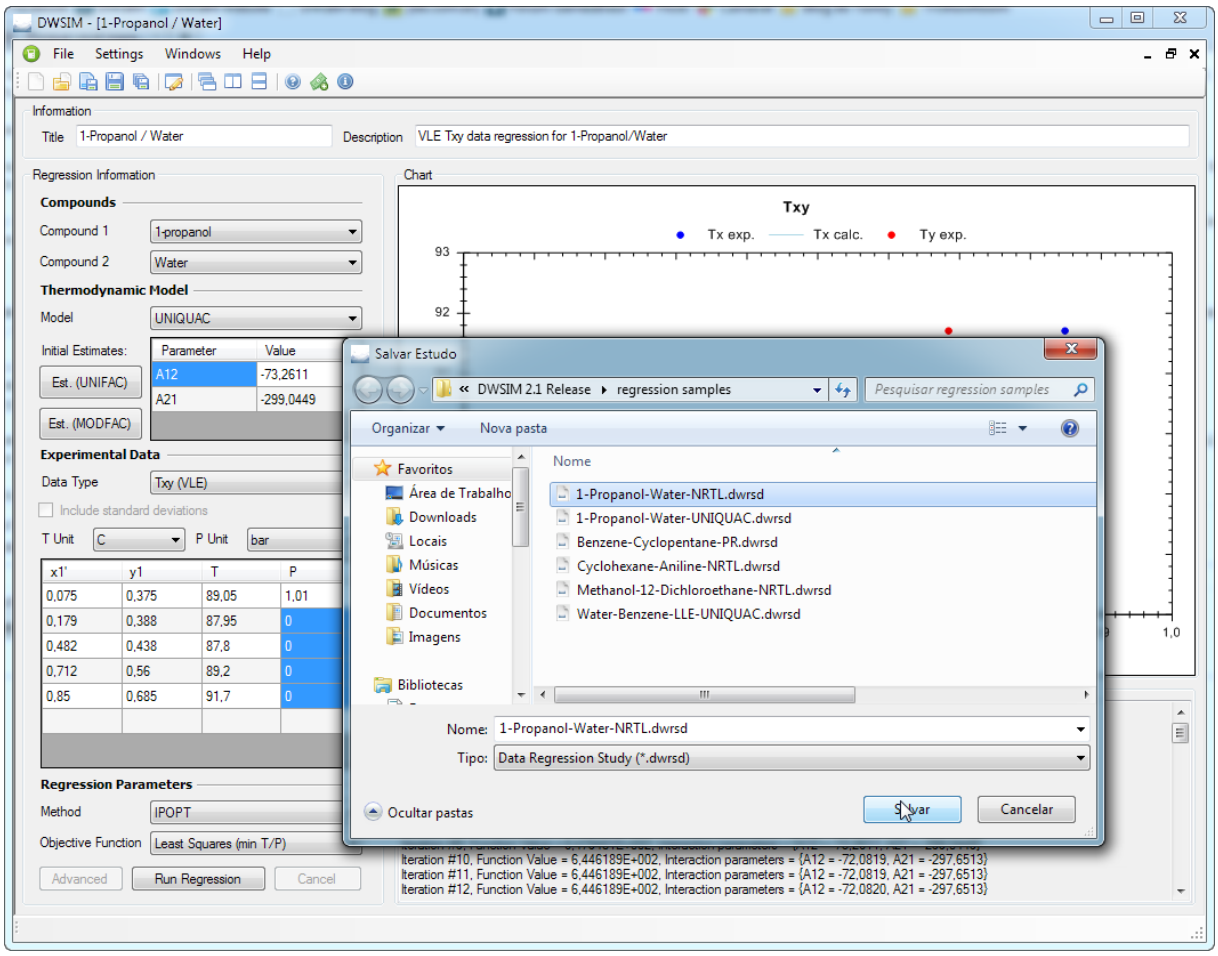



Now click on "Run Regression". DWSIM will then run the optimizer in order to find the optimum values for the binaries using the data you provided. DWSIM will tell you when it is done, and after that you can save your regression case for later use.

IMPORTANT: the calculated interaction parameters are not copied to the BIP databases - their values are only "shown" in the results textbox. You'll have to input them manually on your simulation.




DWSIM Data Regression files have the ".dwrsd" file extension.




# 5. General Settings

The application settings can be accessed through the **Edit > General Settings** menu item (Figure 40):

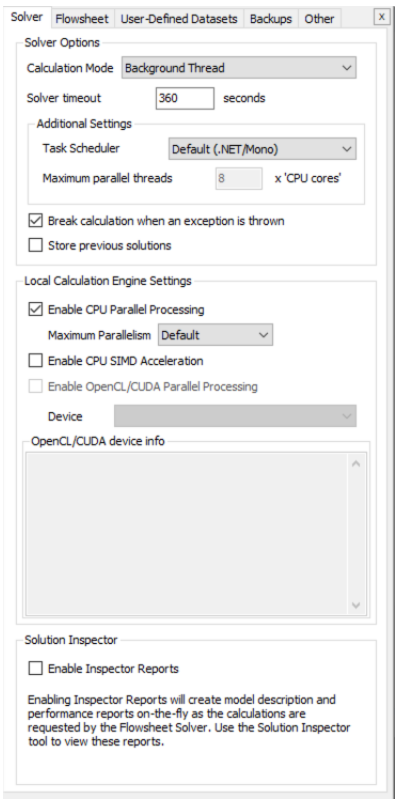



Figure 53: General Settings section.

## 5.1. Solver

The Solver configuration tab display a group of settings to control the behavior of DWSIM's solver. Check the Wiki articleSolver Configurationfor more details.

## 5.2. Flowsheet

### 5.2.1. Cut/Copy/Paste Flowsheet Objects

- **Compounds**: controls how compounds are handled during cut/copy/paste operations.
- **Property Packages**: controls how Property Packages are handled during cut/copy/paste operations.

### 5.2.2. Undo/Redo

- **Recalculate flowsheet**: defines if the flowsheet is to be recalculated after undo/redo operations.

### 5.2.3. Object Editors

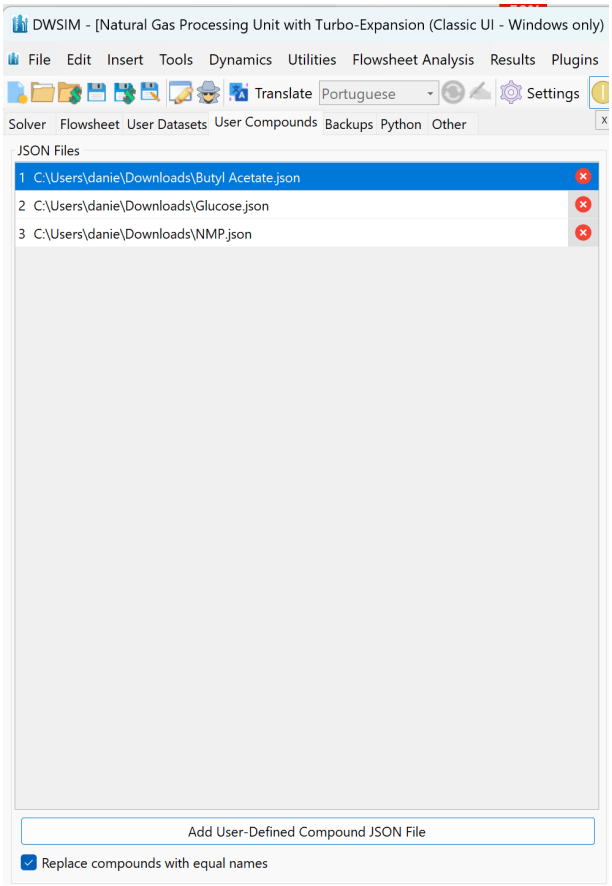
- **Enable multiple editors**: allows displaying of multiple object editors at once.
- **Close editors on deselecting**: closes the editors once the object being edited is deselected.
- **Default initial placement**: default location for displaying the object editors.

## 5.3. User Datasets

In the database tab, you have options to remove, add and edit user-defined compound and interaction parameter datasets.

## 5.4. User Compounds

Add references to JSON files containing pure compound data, so they are available on startup for all existing and new flowsheets:




## 5.5. Backup

The Backup tab has options to control the frequency of the backup file saving. You can also configure the option to save an existing file with another name instead of overwriting it.

## 5.6. Other

### 5.6.1. Messages

- **Show tips**: displays context-sensitive tips on the flowsheet information (log) window.
- **Show "What's New"**: displays a window with information about what's new on the running version.

### 5.6.2. Debug mode

- **Debug level**: controls the amount of information written to the flowsheet information (log) window when solving the simulation.
- **Redirect console output**: redirects the output of the console to the console window inside DWSIM.

### 5.6.3. UI Language

- **Language**: sets the UI language. Requires a restart.

### 5.6.4. CAPE-OPEN

- **Remove solid phases...**: This is for ChemSep compatibility. If enabled, DWSIM will hide the solid phase in Material Streams from CAPE-OPEN Unit Operations.

### 5.6.5. Compound Constant Properties

- **Ignore compound constant properties...**: If enabled, this will prevent DWSIM from using compound constant data from the loaded simulation files and use the data from the compound databases themselves.

### 5.6.6. DWSIM/Python Bridge Settings

- **Python Binaries Path**: Set the path where the GNU Octave binaries are located. This is only required if you're running DWSIM on Windows.
- **Python Process Timeout**: Set the timeout for the Octave processes, in minutes.

# Part III.

# Cross-Platform User Interface (CPUI)

## 6. Main Interface

When DWSIM is opened, you'll see a launcher menu, where you can create a new simulation or open a saved one, among other functions.

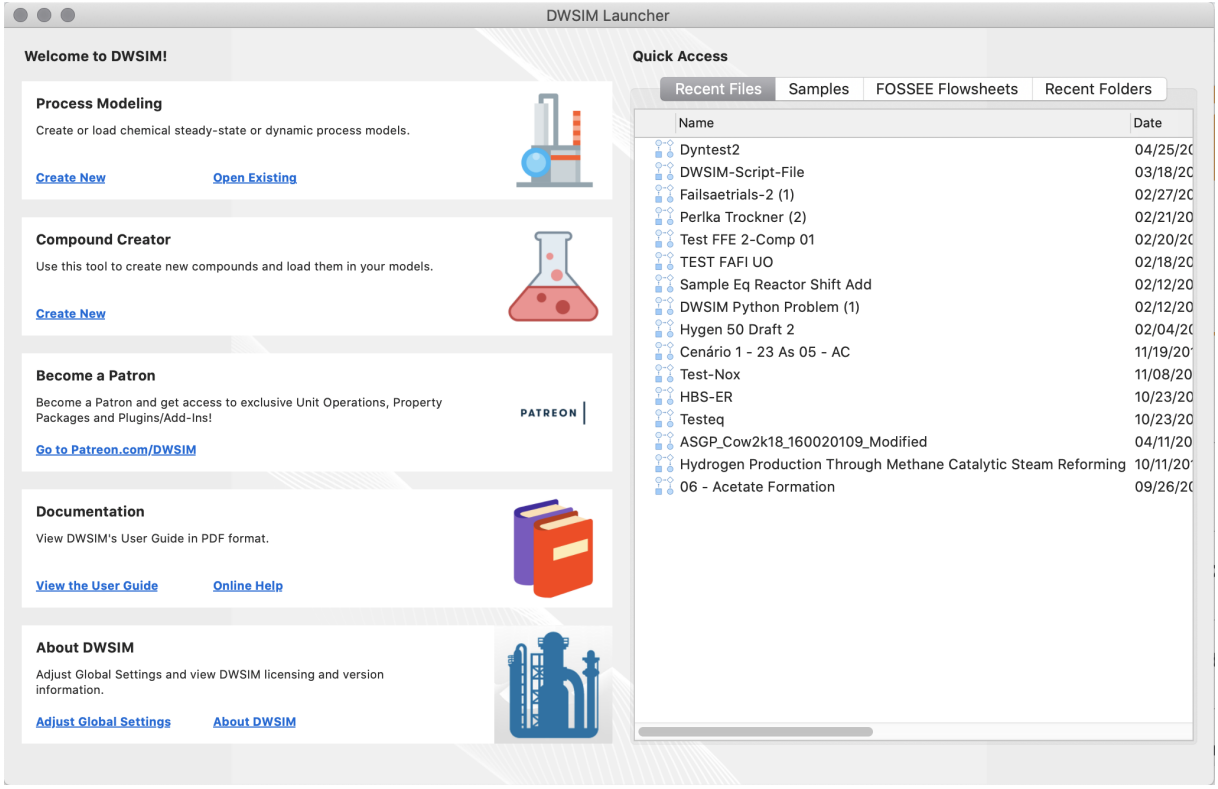



Figure 54: DWSIM Launcher.

If you create a new simulation, you'll get a mostly blank screen with a menu bar at the top and a Log Panel on the bottom part. The **Simulation Setup Wizard** will then appear.

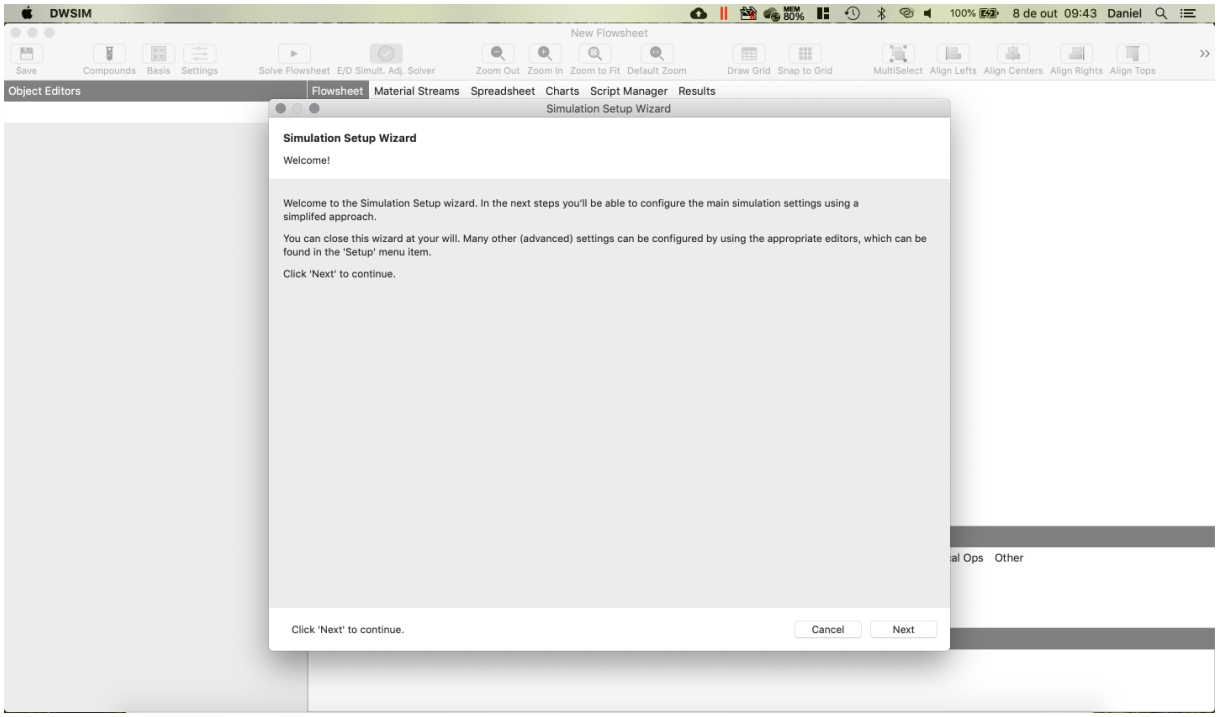



Figure 55: DWSIM Main Flowsheet Window with Simulation Setup Wizard.

# 7. Configuring a Simulation

In order to run a simulation/flowsheet, you need to add some Compounds, setup a Property Package, add Objects to the Flowsheet and connect them to each other following the process flow.

## 7.1. Components/Compounds

There are two essential information required by DWSIM in order to correctly start a simulation. The first refers to the available components (or compounds).

To add a compound to the simulation, select it on the list. To remove an added compound, just deselect it.

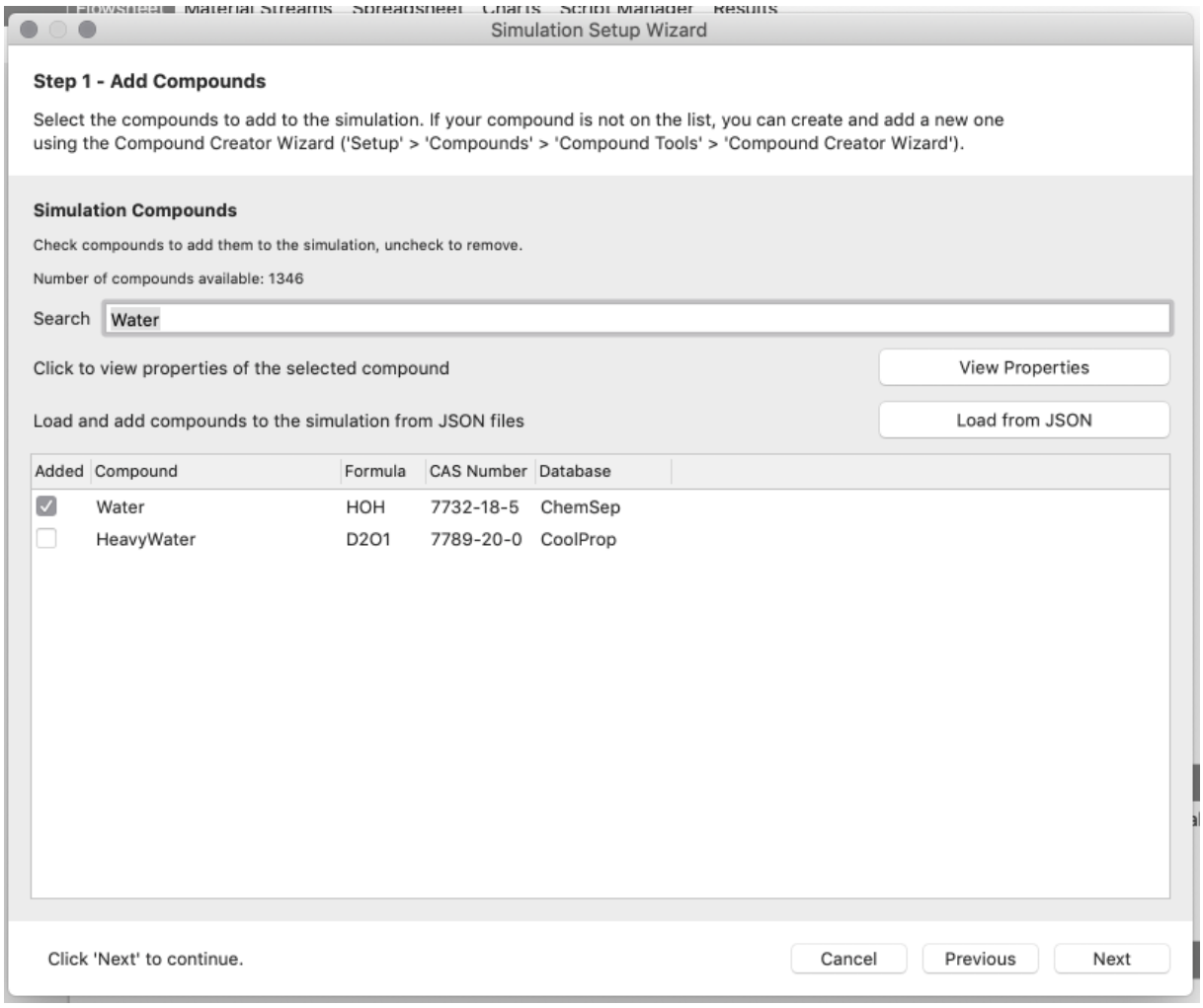



Figure 56: Selecting a Compound with the Simulation Setup Wizard.

## 7.2. Property Packages

The Property Package consists in a set of methods and models for the calculation of physical and chemical properties of material streams in the simulation. It is composed of a thermodynamic model - an equation of state or a hybrid model - and methods for property calculation, like the surface tension of the liquid phase.

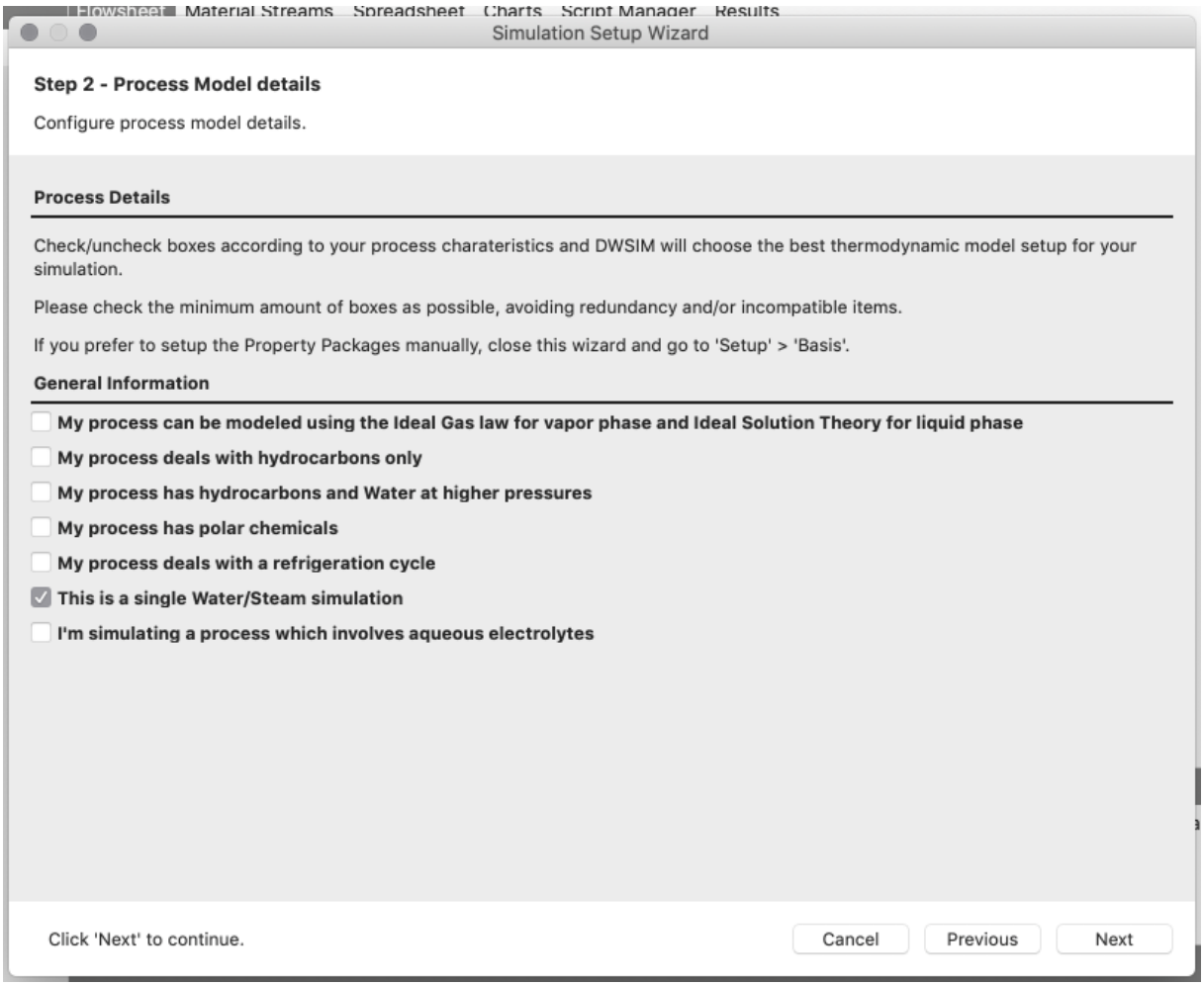



Figure 57: Selecting a Property Package with the Simulation Setup Wizard.

If the selected property package has any editable property, the "Configure" button becomes clickable and the user can click on it to show the property package configuration window.

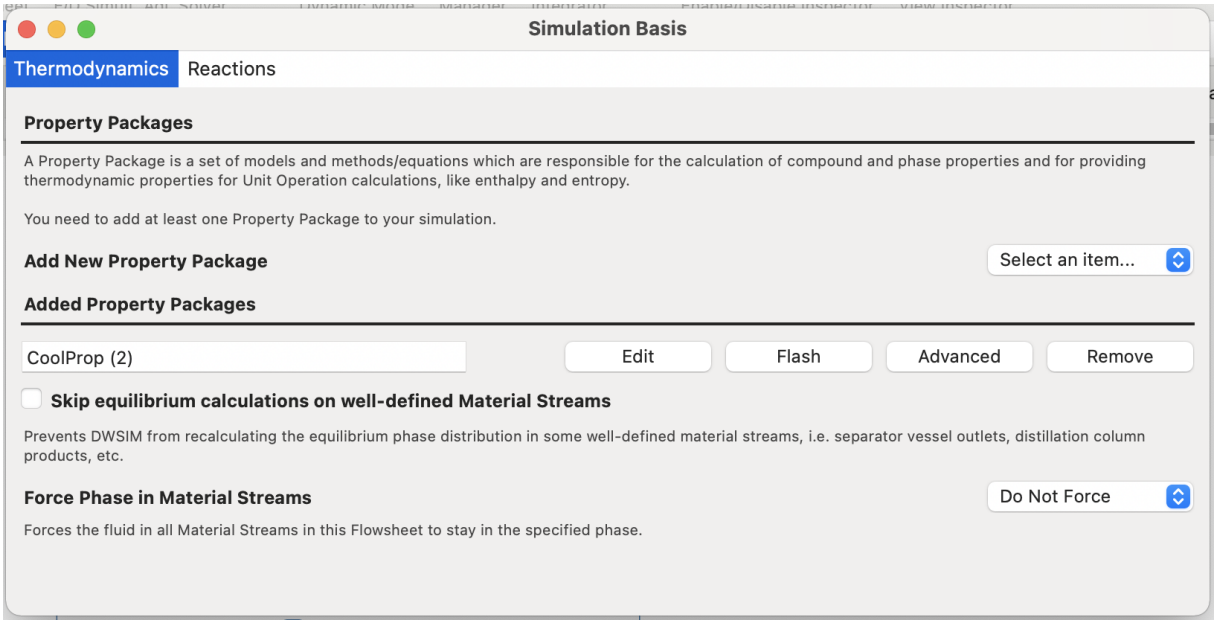



Figure 58: Viewing Property Packages on the Simulation Basis panel.

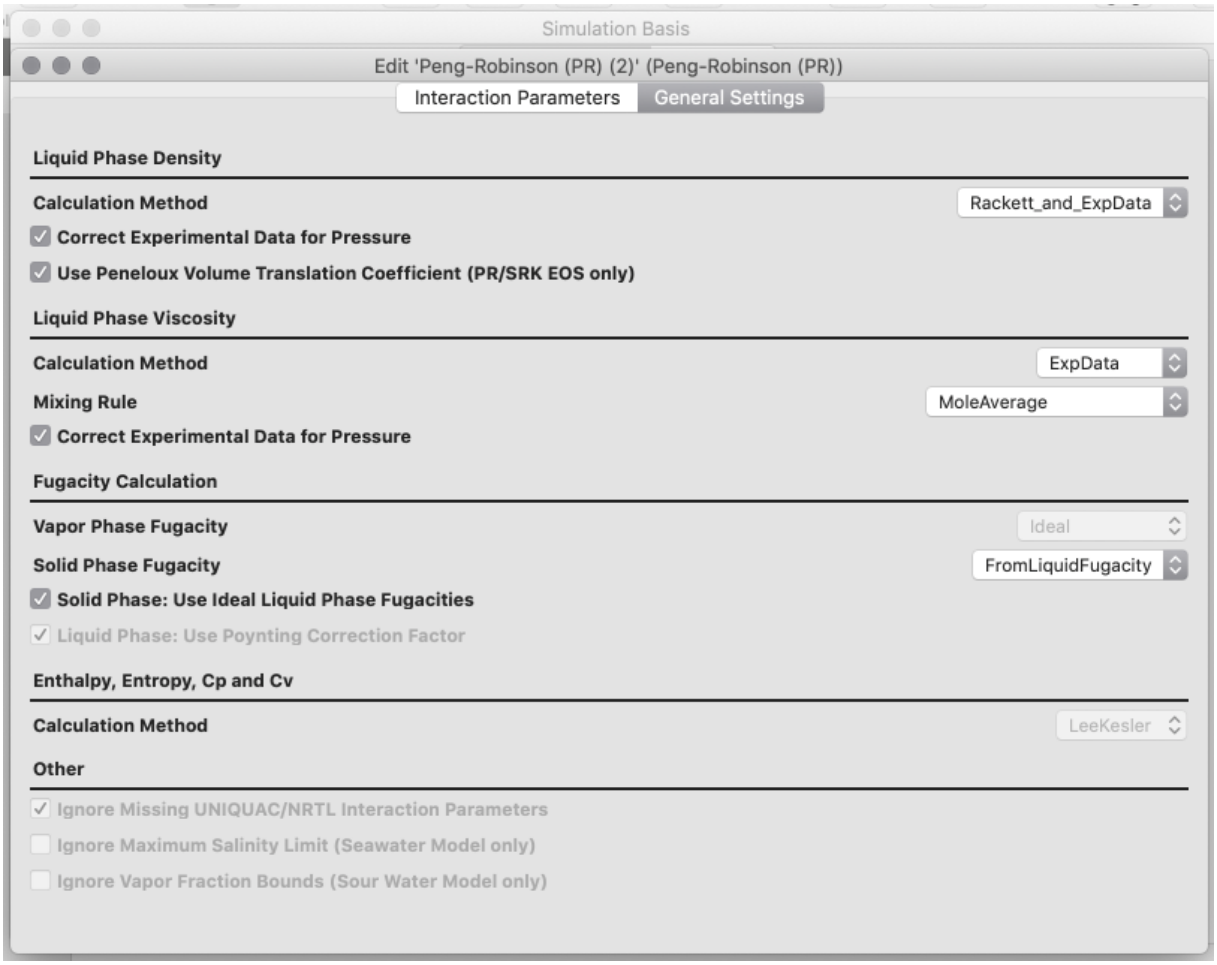



Figure 59: Editing Property Package Settings.

**Multiple Property Packages**

DWSIM allows multiple Property Packages to be added to a single simulation (Figure 40), which can be associated with each unit operation and material stream on an individual basis.

**Skip Equilibrium Calculation in Well-Defined Streams** Tells the Flowsheet Solver to avoid rework and skip equilibrium calculations in Material Streams connected to specific ports like Separator Vessel and Distillation Column outlets.

**Force Material Stream Phase** You can override the equilibrium phase for **all** Material Streams in the flowsheet by setting this property property to the desired value (*Vapor*, *Liquid* or *Solid*. The default value is *Do Not Force*.

When this property is set to one of the phase names (*Vapor*, *Liquid* or *Solid*), the equilibrium calculation for all Material Streams is bypassed and all compounds are put into the selected phase with the same composition as the mixture.

## 7.3. Systems of Units

Three basic units systems are present in DWSIM: **SI System** (selected by default), **CGS System** and **English (Imperial) System**. The simulation's units system can be viewed/modified in the "Units System" section of the "Simulation Settings" panel.

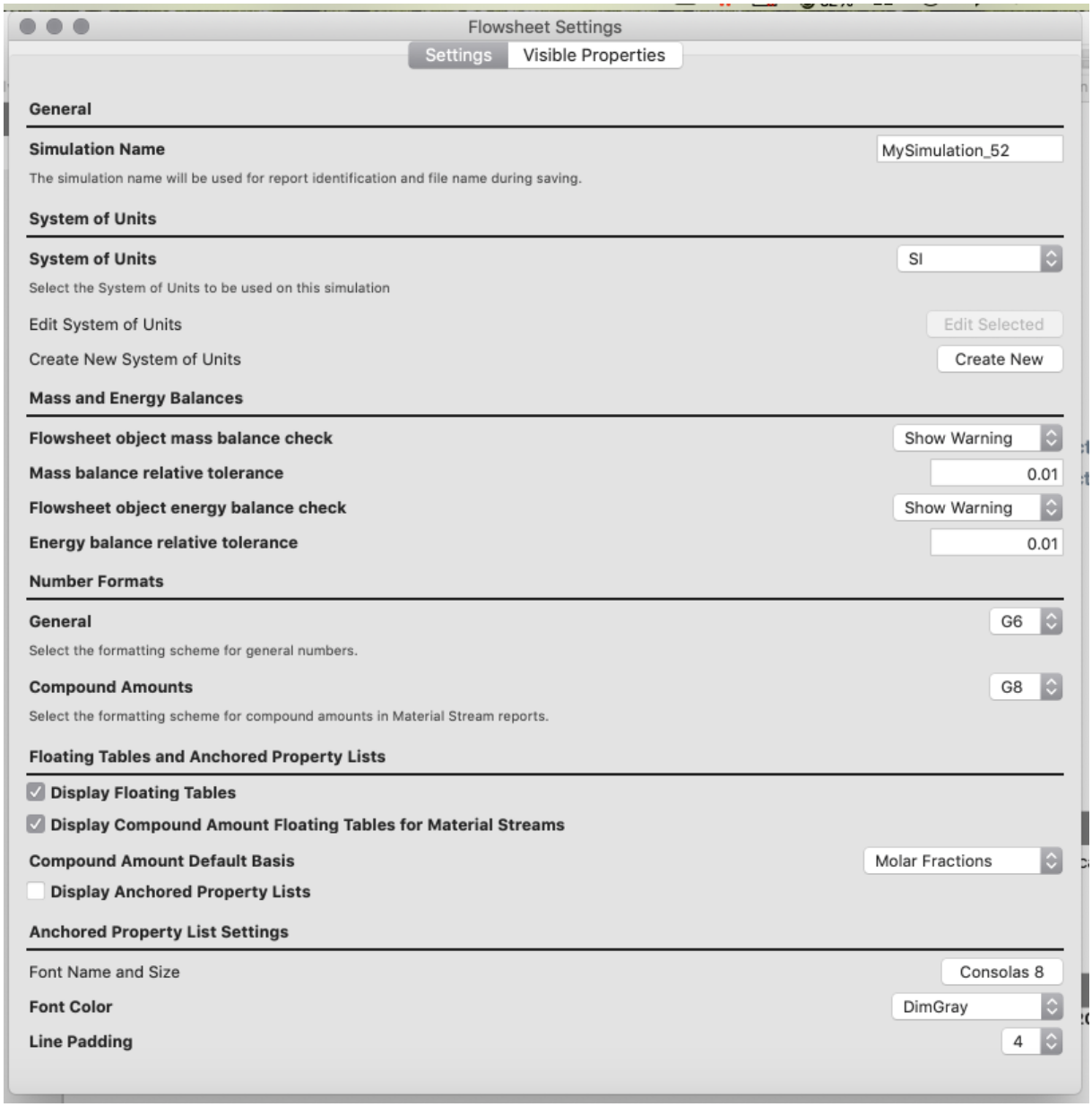



Figure 60: Viewing the Systems of Units on the Flowsheet Settings Panel.

You can also create a custom system of units. It is worth remembering that the units systems can also be modified at any time during the simulation - the changes are applied immediately.

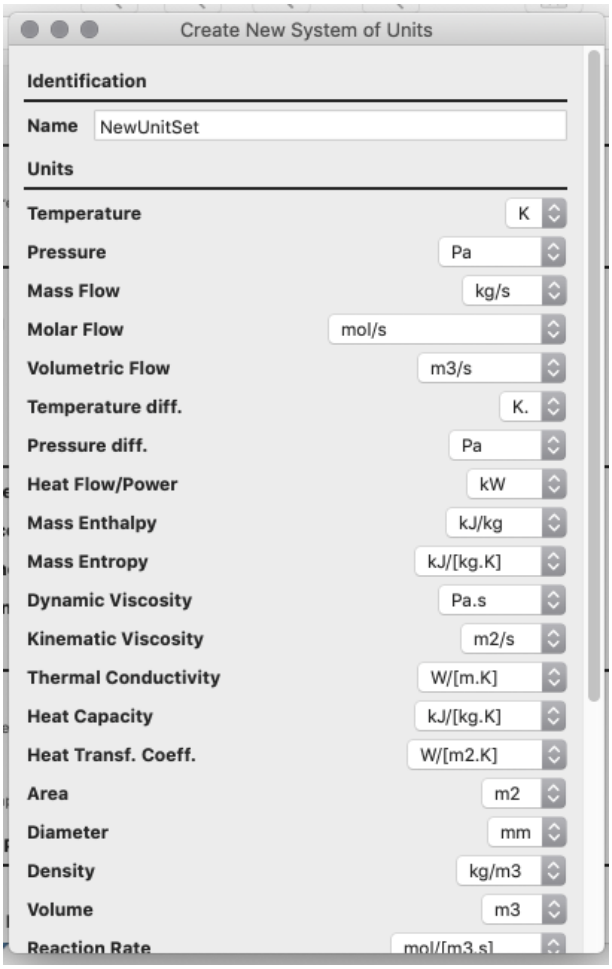



Figure 61: Creating a new System of Units.

# 8. Process Modeling (Flowsheeting)

## 8.1. Inserting Flowsheet Objects

To add an object to the flowsheet, go to 'Object' > 'Add New Simulation Object', or drag the icons from the **Object Palette** to the PFD.

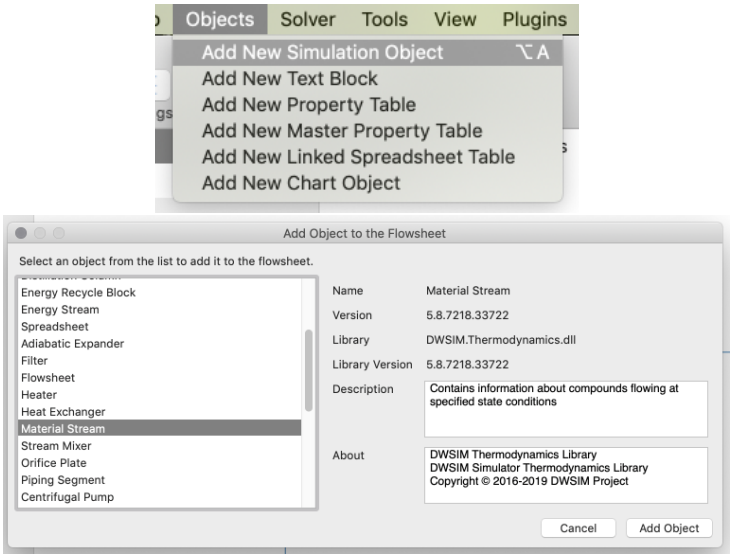



Figure 62: Add New Simulation Object.

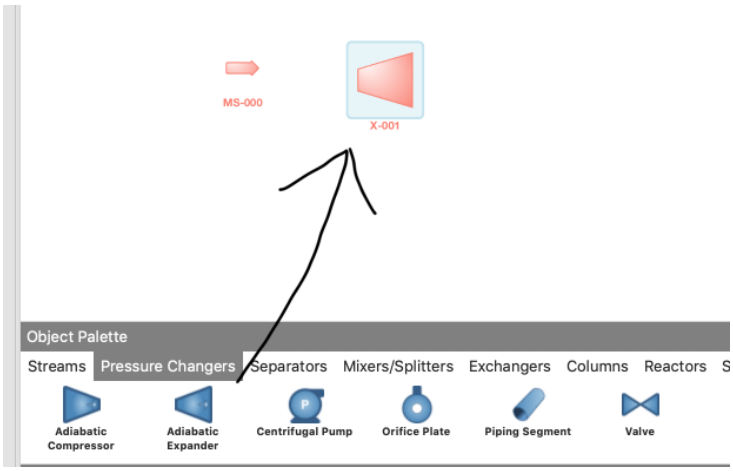



Figure 63: Dragging Objects from the Object Palette to the Flowsheet PFD.

The elements of a simulation (objects) which can be added to the flowsheet are:

- **Material Stream:** used to represent matter which enters and leaves the limits of the simulation and passes through the unit operations. The user should define their conditions and composition in order for DWSIM to calculate their properties accordingly;
- **Energy Stream:** used to represent energy which enters and leaves the limits of the simulation and passes through the unit operations;
- **Mixer:** used to mix up to three material streams into one, while executing all the mass and energy balances;
- **Splitter:** mass balance unit operation - divides a material stream into two or three other streams;
- **Valve:** works like a fixed pressure drop for the process, where the outlet material stream properties are calculated beginning from the principle that the expansion is an isenthalpic process;

- **Pipe:** simulates a fluid flow process (mono or two-phase). The pipe implementation in DWSIM provides the user with various configuration options, including heat transfer to environment or even to the soil in buried pipes. Two correlations for pressure drop calculations are available: Beggs and Brill and Lockhart and martinelli. Both reduces to Darcy equation in the case of single-phase flow;
- **Pump:** used to provide energy to a liquid stream in the form of pressure. The process is isenthalpic, and the non-idealities are considered according to the pump efficiency, which is defined by the user;
- **Separator Vessel:** used to separate the vapor and liquid phases of a stream into two other distinct streams;
- **Compressor:** used to provide energy to a vapor stream in the form of pressure. The ideal process is isentropic (constant entropy) and the non-idealities are considered according to the compressor efficiency, which is defined by the user;
- **Expander:** the expander is used to extract energy from a high-pressure vapor stream. The ideal process is isentropic (constant entropy) and the non-idealities are considered according to the expander efficiency, which is defined by the user;
- **Heater:** simulates a stream heating process;
- **Cooler:** simulates a stream cooling process;
- **Conversion Reactor:** simulates a reactor where conversion reactions occur;
- **Equilibrium Reactor:** simulates a reactor where equilibrium reactions occur;
- **PFR:** simulates a Plug Flow Reactor (PFR);
- **CSTR:** simulates a Continuous-Stirred Tank Reactor (CSTR);
- **Shortcut Column:** simulates a simple distillation column with approximate results using shorcut calculations;
- **Distillation Column:** simulates a distillation column using rigorous thermodynamic models;
- **Absorption Column:** simulates an absorption column using rigorous thermodynamic models;
- **Heat Exchanger:** simulates a countercurrent heat exchanger using rigorous thermodynamic models.
- **Component Separator:** model to simulate a generic process for component separation.
- **Solids Separator:** model to simulate a generic process for solid compound separation.

Additionally, the following logical operations are available in DWSIM:

- **Adjust:** used to make a variable to be equal to a user-defined value by changing the value of other (independent) variable;
- **Recycle:** used to mix downstream material with upstream material in a flowsheet.

## 8.2. Connecting/Disconnecting objects

The material and energy streams represent mass and energy flowing between unit operations. You can connect/disconnect streams to/from Unit Operations or Logical Blocks by selecting the object and working with the Combo Boxes on the ”Connections” panel within the Object Editor.

### 8.2.1. Auto-Connect Added Objects

This is a new feature in DWSIM v7.4.0. It enables automatic connections between added objects and nearby ones.

There are three different options for this setting:

- **No**: No automatic connections are made when you add an object to the flowsheet.
- **Yes**: When you add a new unit operation, streams are automatically added to the flowsheet and connected to its ports.
- **Smart**: When you add a new unit operation, nearby streams are connected to it, and new streams are created to connect to the remaining ports.

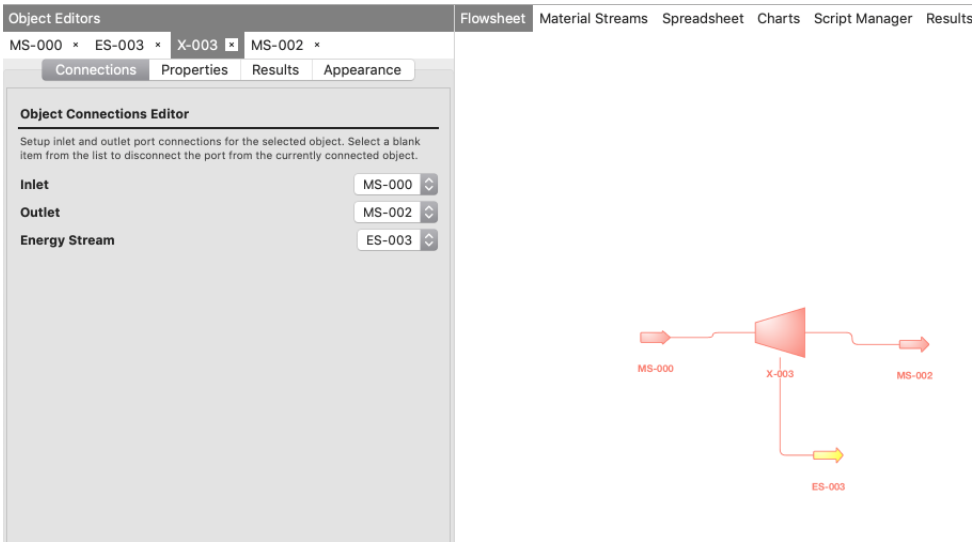



Figure 64: Editing the Connections of an Object.

## 8.3. Process data management

### 8.3.1. Entering process data

The objects’ process data (temperature, pressure, flow, composition and/or other parameters) can be entered in the property editor window, accessible through the ’Edit Properties’ context menu item.

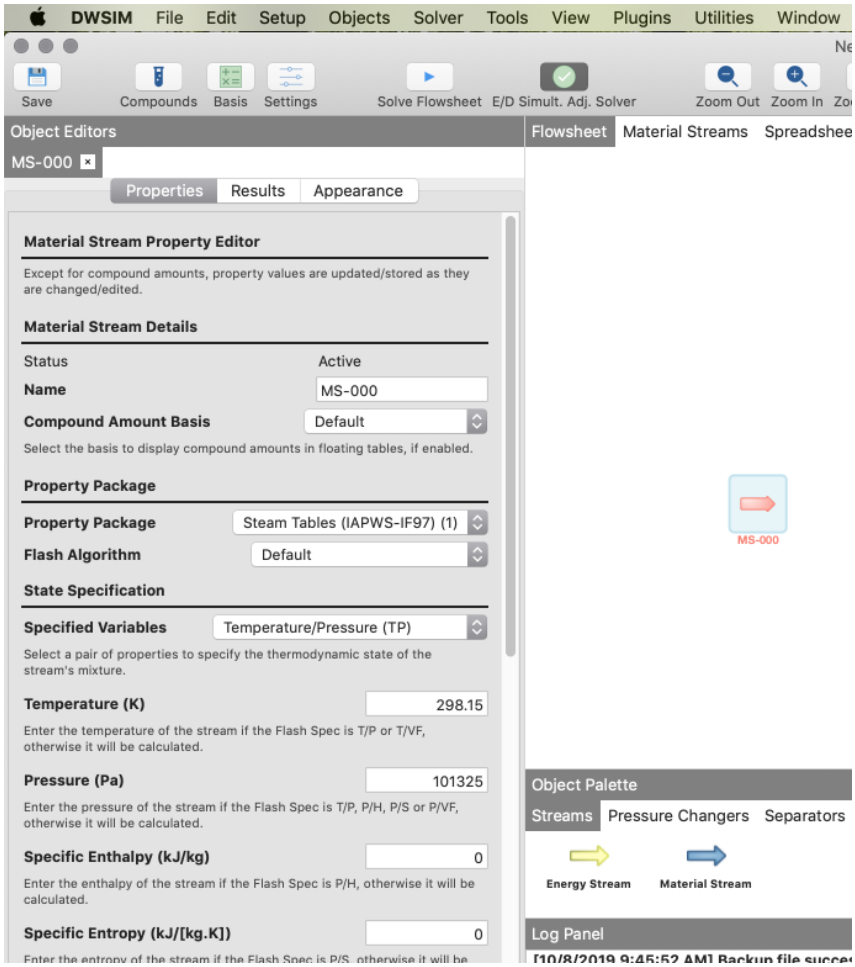



Figure 65: Entering process data.

## 8.4. Running a Simulation

DWSIM is a sequential modular process simulator, that is, all calculations are made in a per-module basis, according to the connections between the objects. The calculator checks if an object has all of its properties defined and, if yes, passes the data for the downstream object and calculates it, repeating the process in a loop until it reaches an object that doesn't have any of its dowstream connections attached to any object. This way, the entire flowsheet can be calculated as many times as necessary without having to "tell" DWSIM which object must be calculated. In fact, this is done indirectly if the user define all the properties and make all connections between objects correctly.

To solve the flowsheet, press F5 or click on 'Solver' > 'Solve Flowsheet'. As DWSIM's solver does its job, messages are shown in the log panel. These messages tell the user if the object was calculated successfully or if there was an error while calculating it.

If the calculation finishes without errors, you can proceed to viewing the results.

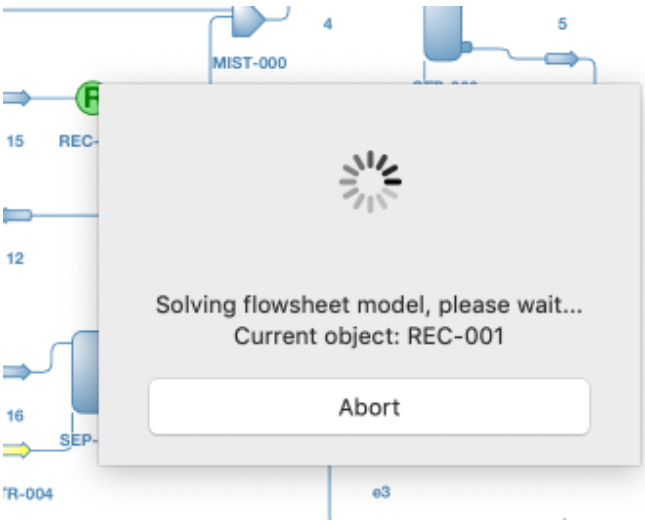



Figure 66: The Flowsheet Solver doing its work.

## 8.5. Viewing Results

Simulation Results can be viewed either in full reports in PDF/ODS/ODT format, or individually through the Results tab, by selecting the object on the list, or by going to the Results tab on the Object Editor, if it has already been calculated.

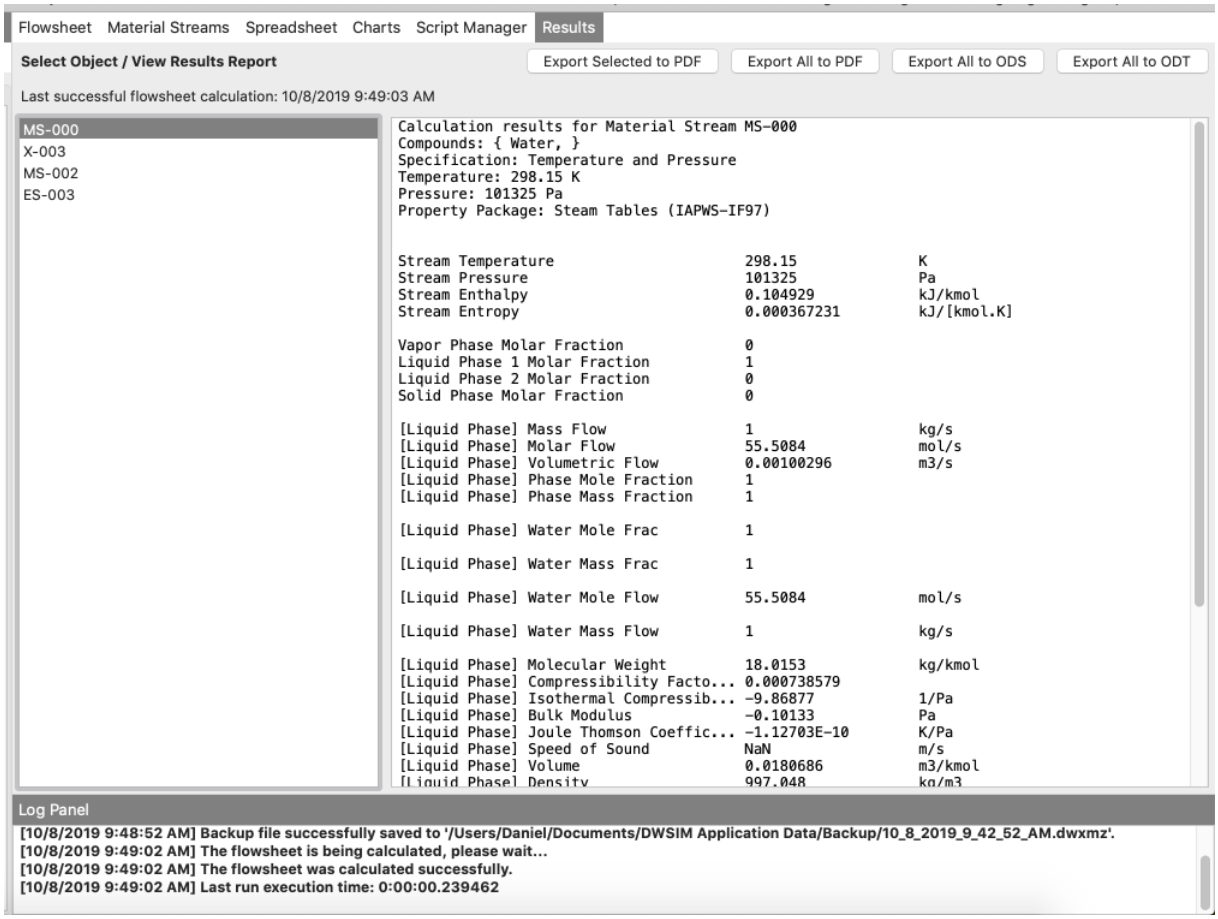



Figure 67: The Results Panel.

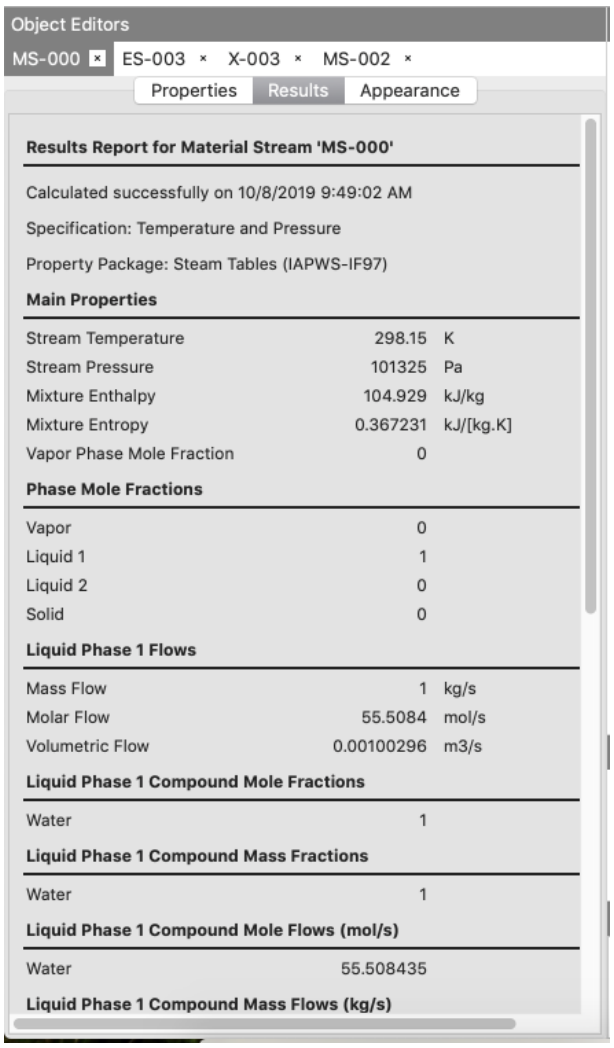



Figure 68: Viewing results from a single object.

## 8.6. Flowsheet Utilities

DWSIM includes some utilities which provides the user with more information about the process being simulated.

- ➙ True Critical Point - utility to calculate the true Critical Point of a mixture (PR/SRK EOS only)
- ➙ Phase Envelope - Material Stream phase equilibria envelope calculation
- ➙ Binary Envelope - special envelopes for binary mixtures
- ➙ Pure Compound Properties Viewer

Utilities calculate their properties for a single stream only. In the majority of cases, this object must be calculated in order to be available for selection in the utility window.

### 8.6.1. Pure Component Properties

The Pure Component Properties utility is used to view and edit pure component constants, view molecular properties and general temperature dependent properties like ideal gas Cp, vapor pressure, liquid

viscosity and vaporization enthalpy.

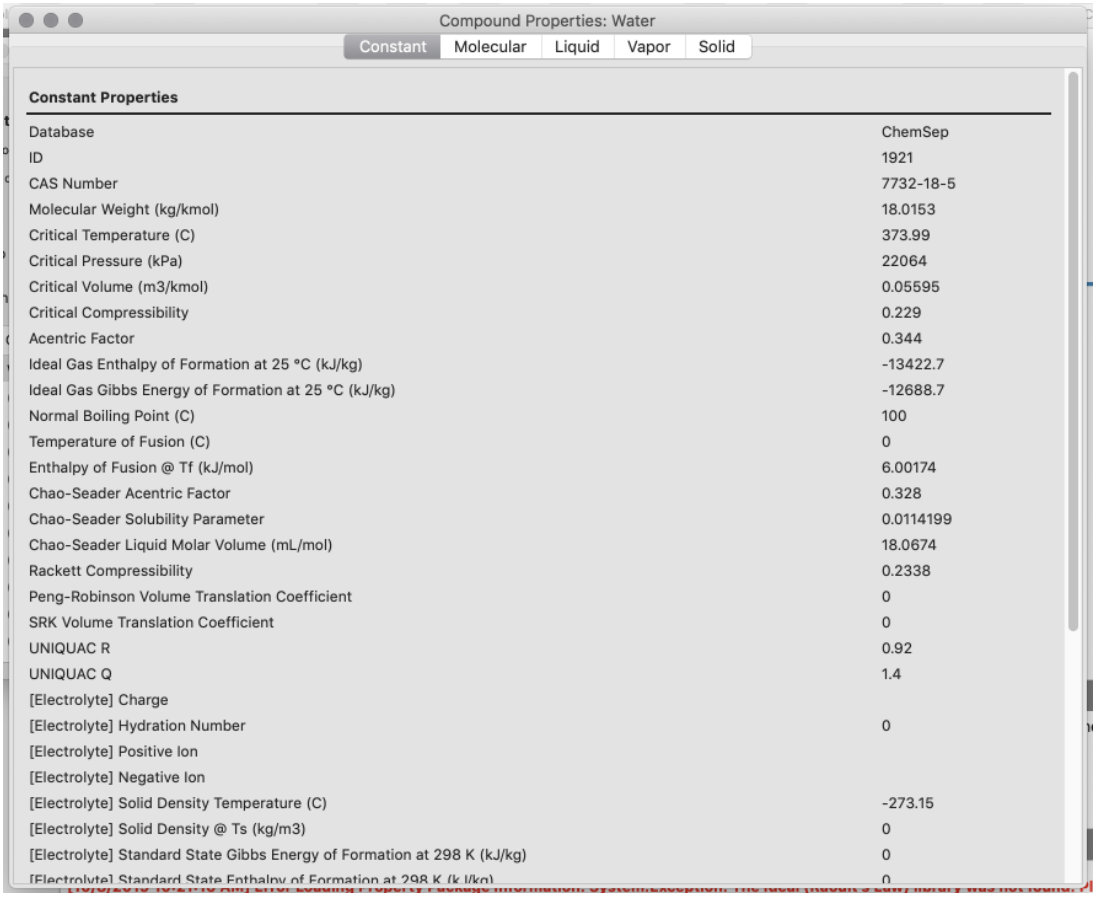



Figure 69: Pure Compound Property Viewer.

### 8.6.2. Phase Envelope

The Phase Envelope utility allows the visualization of the existing relations between thermodynamic properties of a mixture of components in a material stream. The following phase envelopes can be generated in DWSIM: Pressure-Temperature, Pressure-Enthalpy, Pressure-Entropy, Pressure-Volume, Temperature-Pressure, Temperature-Enthalpy, Temperature-Entropy, Temperature-Volume, Volume-Pressure, Volume-Temperature, Volume-Enthalpy and Volume-Entropy.

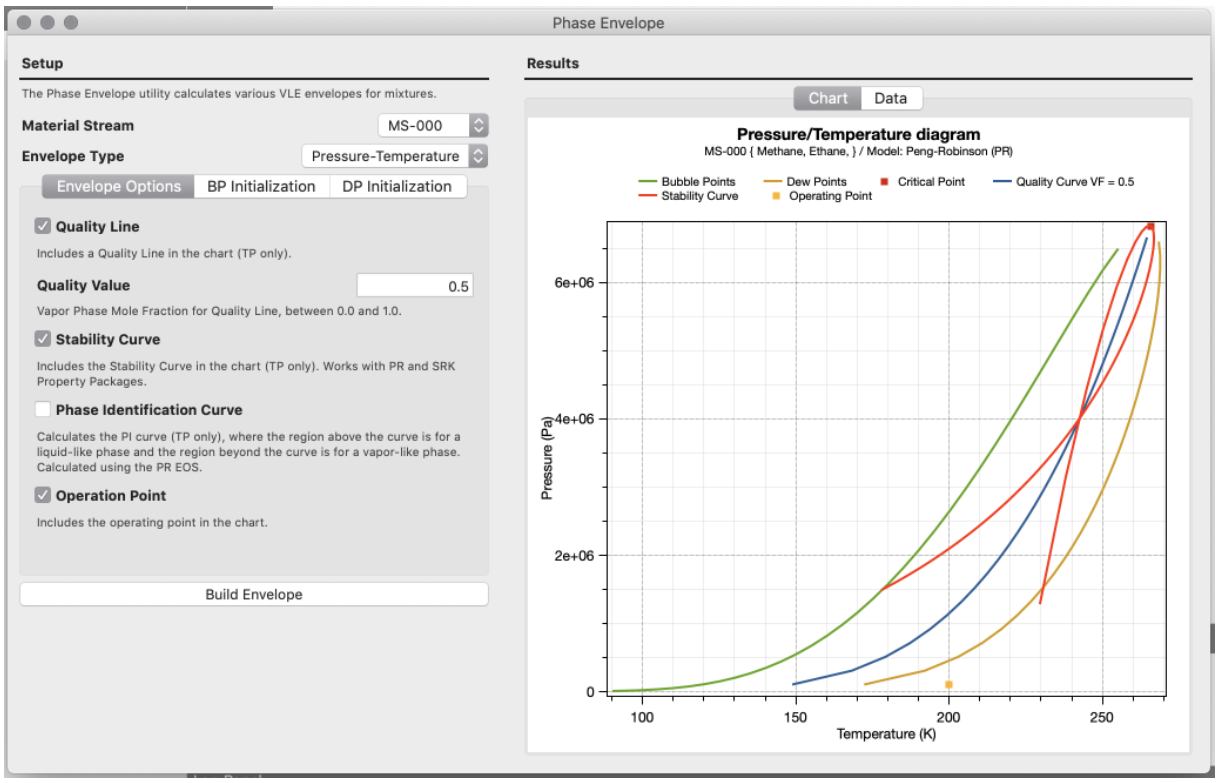



Figure 70: Phase Envelope Utility.

### 8.6.3. Binary Envelope

The Binary Envelope utility is a specialized phase envelope builder for viewing specific two-component diagrams (T-x-y, P-x-y, etc.). For the T-x-y diagram type, different equilibrium lines can be calculated, depending on Property Package and Flash Algorithm selections.

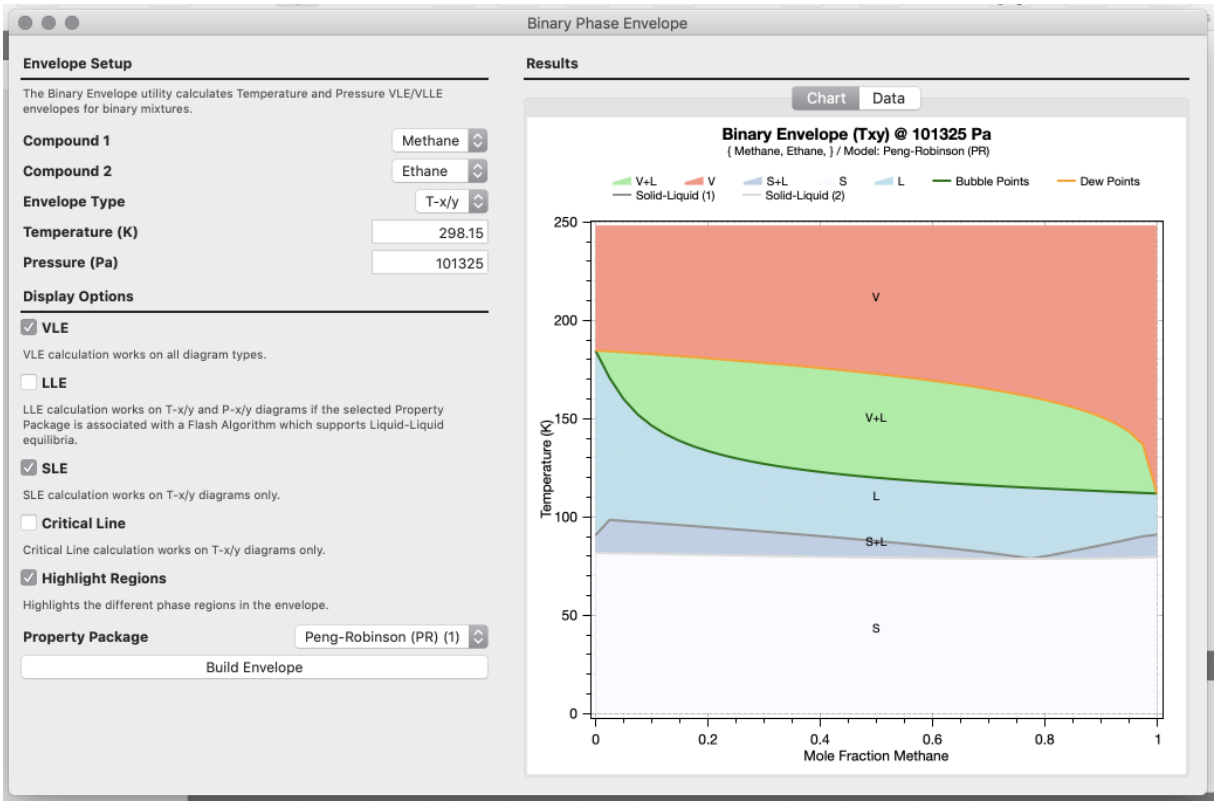



Figure 71: Binary Envelope Utility.

# Part IV.

# Unit Operation Models

## 9. Streams

### 9.1. Material Stream

The Material Stream is used to represent matter which enters and leaves the limits of the simulation, passing through the unit operations. Input Parameters Specification: One of the following flash types may be selected:

- Temperature and Pressure
- Pressure and Enthalpy
- Pressure and Entropy
- Pressure and Vapor Fraction
- Temperature and Vapor Fraction

The above selection dictates which variables should be defined in order to enable DWSIM calculate the others:

- Temperature: stream temperature
- Pressure: stream pressure Specific
- Enthalpy: stream's specific enthalpy Specific
- Entropy: stream's specific entropy
- Molar Fraction (Vapor Phase): the vapor phase mole fraction of the stream

**9.1.0.1. Composition** The stream composition can be entered on basis of: Mole/Mass fraction/flow Standard liquid volumetric fraction Molality or molarity The material stream composition can only be edited if the stream is not connected to any unit operation upstream, which means that it doesn't work as an output of any operation. If that is the case, the stream configures itself as "read-only" and the user will not be able to edit any of its properties directly. These are written into the stream as a result of calculation from its connected unit operation.

For the mole/mass fraction basis options, the user must enter the composition in such a way that the amounts sum up to 1. Regardless of the input basis option, new values are saved only when the "Apply / Commit Changes" button is pressed.

Molarity and Molality input composition options only make sense in the context of an electrolyte simulation. In this case, the solute amounts should be entered in moles and the solvent amount in L (molarity) or kg (molality).

**9.1.0.2. Flow** One of the three types of flow (mass, molar or volumetric) must be given. The other two are calculated automatically in the case when temperature and pressure are already defined. If composition is defined by either mole or mass flow of each component, the total flow of the stream is calculated automatically. If a definition of composition is utilised, you will have to define flowrate of the stream on stream level outside of composition dialog.

### 9.1.1. Calculation Method

When the four properties described above are defined, the material stream is calculated and its properties are shown in the same window. The calculation sequence for the material stream is the following:

1. A flash calculation is done to know the component distribution between phases
2. Properties of each phase are calculated individually
3. Finally, the mixture properties are calculated

In the first step, a TP Flash is done by default, but it can be replaced by a PH flash if the user defines this option in the simulation configuration window. When in "read-only" mode, the stream properties are calculated according to parameters given by the upstream unit operation (in the majority of cases, a TP flash is done as well).

**Overriding the Equilibrium Phase** You can override the equilibrium phase by setting the **Force Phase** property to the desired value (*Global Definition*, *Vapor*, *Liquid* or *Solid*. *Global Definition* is selected by default and reads the value from the Flowsheet Settings. The default setting for the Flowsheet-wide property is *Do Not Force*.

When this property is set to one of the phase names (*Vapor*, *Liquid* or *Solid*), the equilibrium calculation is bypassed and all compounds are put into the selected phase with the same composition as the mixture.

### 9.1.2. Output parameters

The following items are calculated:

- ➞ Component distribution between phases. The compositions can be displayed either as molar or mass fraction by selecting the correspondent option at Composition basis drop down field.
- ➞ Phase properties: specific enthalpy, specific entropy, molecular weight, density, volumetric flow rate @ T and P, phase molar and mass fraction, compressibility factor, constant-pressure heat capacity (Cp), Cp/Cv, thermal conductivity, surface tension (liquid phase only), kinematic viscosity, dynamic viscosity.
- ➞ Mixture properties: specific enthalpy, specific entropy, molecular weight, density, thermal conductivity.

If the flash algorithm setting ”Calculate bubble and dew points at stream conditions” is activated, these are also shown in the ”Mixture” properties section.

### 9.2. Energy Stream

The energy stream is used to represent energy entering and leaving the limits of the simulation, used by the unit operations, either to represent loss, demand or power generation. As we are dealing with steady-state simulations, one defines the energy stream in terms of power (energy by unit of time) and not energy itself.

The energy either is written by user to be used for energy consuming unit operations or or from its connected unit operation as a result of its calculations.

**9.2.0.1. Input Parameters** Energy: Energy by unit of time (power) which is represented by the stream;

**9.2.0.2. Output Parameters** There are no output parameters for this object.

## 10. Unit Operations

### 10.1. Mixer

The Mixer is used to mix up to six material streams into one, while executing all the mass and energy balances.

***Input Parameters***

- Downstream pressure: defines how the downstream pressure must be calculated from the pressure of the streams connected to the mixer inlets.

***Calculation Method*** The mixer does the mass balance in the equipment and determines the mass flow and the composition of the outlet stream. Pressure is calculated according to the parameter defined by the user. Temperature is calculated by doing a PH Flash in the outlet stream, with the enthalpy calculated from the inlet streams (energy balance).

***Output Parameters*** There are no output parameters for this object.

#### 10.1.1. Splitter

The splitter is a mass balance unit operation - divides a material stream into two or three other streams.

***Input Parameters***

- Stream fractions: this property defines the mass flow fraction to be passed to each outlet stream in the splitter. Each fraction must have a value between 0 and 1 and the total sum must not be bigger than 1.

***Output Parameters*** There are no output parameters for this object.

## 10.2. Separator Vessel

The separator vessel (also known as *flash drum*) is used to separate liquid phases from vapor in a mixed material stream.

***Input Parameters***

- Override separation temperature and pressure: these properties define if the flash calculation will be done with the temperature and pressure of the inlet stream or the ones defined by the user.

***Calculation Method*** The separator vessel just divides the inlet stream phases into two or three distinct streams. If the user defines values for the separation temperature and/or pressure, a TP Flash is done in the new conditions before the distribution of phases through the outlet streams.

***Output Parameters*** There are no output parameters for this object.

## 10.3. Tank

In the current version of DWSIM, the Tank works like a fixed pressure drop for the process.

***Input Parameters***

- Pressure drop: pressure difference between the outlet and inlet streams.

***Calculation Method*** The outlet stream pressure is calculated from the inlet pressure and the pressure drop given. The temperature remains the same. A TP Flash is done to calculate the properties of the outlet stream.

***Output Parameters*** There are no output parameters for this object.

## 10.4. Pipe Segment

The Pipe Segment unit operation can be used to simulate fluid flow process in a pipe. Two of the most used correlations for the calculation of pressure drop are available in DWSIM. Temperature can be rigorously calculated considering the influence of the environment. With the help of the Recycle Logical Operation, the user can build large water distribution systems, as an example.

The pipe segment is divided in sections, which can be straight tubes, valves, curves, etc. Each section is subdivided in small sections for calculation purposes, as defined by the user.

***Input Parameters***

- Hydraulic profile: clicking on the ellipsis button opens the pipe hydraulic profile editor. In the hydraulic profile editor, the user adds sections, define their type and in how many increments it will be divided during the calculations, the pipe material, length, elevation and internal and external diameters. Each change can be saved by clicking in the "Apply" button.

- Pressure drop correlation: select the model to be used for the pressure drop calculation in the pipe segment.
- Thermal profile: In the thermal profile editor it is possible to define how the temperature profile in the pipe should be calculated. The configurations in this window are valid for the **entire** pipe segment. Changes are saved automatically.

***Calculation Method*** The pipe segment is calculated based on incremental mass and energy balances. The complete algorithm consists in three nested loops. The external loop iterates on the sections (increments), the middle loop iterates on the temperature and the internal loop calculates the pressure. The pressure and temperature are calculated as follows:

1. The inlet temperature and pressure are used to estimate the increment outlet pressure and temperature.
2. Fluid properties are calculated based in a arithmetic mean of inlet and outlet conditions.
3. The calculated properties and the inlet pressure are used to calculate the pressure drop. With it, the outlet pressure is calculated.
4. The calculated and estimated pressure are compared, and if their difference exceeds the tolerance, a new outlet pressure is estimated, and the steps 2 and 3 are repeated.
5. Once the internal loop has converged, the outlet temperature is calculated. If the global heat transfer coefficient (U) was given, the outlet temperature is calculated from the following equation:

$$Q = UA\Delta T_{ml} \tag{10.1}$$

   where: $Q$ = heat transferred, $A$ = heat transfer area (external surface) and $\Delta T_{ml}$ = logarithmic mean temperature difference.
6. The calculated temperature is compared to the estimated one, and if their difference exceeds the specified tolerance, a new temperature is estimated and new properties are calculated (return to step 2).
7. When both pressure and temperature converges, the results are passed to the next increment, where calculation restarts.

***Output Parameters***

- Delta-T: temperature variation in the pipe segment.
- Delta-P: pressure variation in the pipe segment.
- Heat exchanged: amount of heat exchanged with the environment, or lost by friction in the pipe walls.
- Results (table): results are show section by section in a table.

➞ Results (graph): a graph shows the temperature, pressure, liquid holdup, velocity and heat exchanged profiles.

## 10.5. Valve

The Valve works like a fixed pressure drop for the process, where the outlet material stream properties are calculated beginning from the principle that the expansion is an isenthalpic process.

***Input Parameters***

➞ Pressure drop: pressure difference between the outlet and inlet streams.

***Calculation Method*** The outlet stream pressure is calculated from the inlet pressure and the pressure drop. The outlet stream temperature is found by doing a PH Flash. This way, in the majority of cases, the outlet temperature will be less than or equal to the inlet one.

***Output Parameters***

➞ Delta-T: temperature drop observed in the valve expansion process.

### 10.5.1. Opening (OP)/Flow Coefficient (Kv[Cv]) Relationship Types

➞ **Linear**

$$K_V = K_{Vmax} \times OP \tag{10.2}$$

➞ **Quick Opening**

$$K_V = K_{Vmax} \times \sqrt{OP} \tag{10.3}$$

➞ **Equal Percentage**

$$K_V = K_{Vmax} \times R^{OP-1} \tag{10.4}$$

where $R$ is the characteristic parameter (range: 20 - 50)

➞ **User-Defined Expression**

$$K_V = K_{Vmax} \times f(OP) \tag{10.5}$$

➞ **Data Table**

Actual flow coefficient value is determined by interpolating data from a user-defined table.

## 10.6. Pump

The pump is used to provide energy to a liquid stream in the form of pressure. The process is isenthalpic, and the non-idealities are considered according to the pump efficiency, which is defined by the user.

***Input Parameters***

- Delta-P: pressure rise in the pump.
- Efficiency: pump adiabatic efficiency;
- Ignore vapor in the inlet stream: defines if the calculator should ignore any vapor in the inlet stream;
- Use the provided Delta-P: defines if the pressure of the outlet stream will be calculated by the user-defined Delta-P or the energy stream connected to the pump.

***Calculation Method*** The calculation method for the pump is different for the two cases (when the provided delta-p or the potency of the energy stream is used). In the first method, we have the following sequence:

- Outlet stream enthalpy:

$$H_2 = H_1 + \frac{\Delta P}{\rho}, \tag{10.6}$$

- Pump discharge pressure:

$$P_2 = P_1 + \Delta P \tag{10.7}$$

- Pump required power:

$$Pot = \frac{W\left(H_2 - H_1\right)}{\eta}, \tag{10.8}$$

where:

$Pot$ pump power

$W$ mass flow

$H_2$ outlet stream specific enthalpy

$H_1$ inlet stream specific enthalpy

$\eta$ pump efficiency

- Outlet temperature: PH Flash (with P2 and H2).

In the second case (calculated outlet pressure), we have the following sequence:

➙ Outlet stream enthalpy:

$$H_2 = H_1 + \frac{Pot\,\eta}{W}, \tag{10.9}$$

➙ $\Delta P$:

$$\Delta P = \rho(H_2 - H_1), \tag{10.10}$$

➙ Discharge pressure:

$$P_2 = P_1 + \Delta P \tag{10.11}$$

➙ Outlet temperature: PH Flash.

***Outlet Parameters***

➙ Delta-T: temperature variation in the pumping process.

➙ Power required: power required by the pump.

## 10.7. Compressor/Expander

The compressor/expander is used to provide or generate energy to/from a vapor phase stream. The ideal process is isentropic (constant entropy) and the non-idealities are considered according to the compressor efficiency and thermodynamic path (adiabatic or polytropic), which are defined by the user.

***Input Parameters***

➙ Delta-P: pressure change in the equipment.

➙ Efficiency: adiabatic/polytropic efficiency;

➙ Ignore liquid in the inlet stream: defines if the calculator should ignore any liquid in the inlet stream;

➙ Thermodynamic path: select the thermodynamic path according to the experimental/field data available.

***Calculation Method*** Isentropic (Adiabatic) or Polytropic power is calculated from:

$$P = \frac{H_{2s} - H_1}{\eta} W \tag{10.12}$$

for compressor, and

$$P = (H_{2s} - H_1) \times W \times \eta \tag{10.13}$$

for expander, where:

$H_{2s}$ Outlet Enthalpy for Isentropic Process

$H_1$ Inlet Enthalpy

$W$ Mass Flow

$\eta$ Adiabatic or Polytropic Efficiency

Isentropic (Adiabatic) and Polytropic Coefficients are calculated from:

$$n_i = \frac{\ln(P_2/P_1)}{\ln(\rho_{2i}/\rho_1)} \tag{10.14}$$

$$n_p = \frac{\ln(P_2/P_1)}{\ln(\rho_2/\rho_1)} \tag{10.15}$$

where:

$\rho_{2i}$ Outlet Gas Density calculated with Inlet Gas Entropy

Adiabatic and Polytropic Heads are calculated from:

$$H = \frac{P}{W \times g} \tag{10.16}$$

where:

$H$ Adiabatic or Polytropic Head

$P$ Adiabatic or Polytropic Power

$W$ Mass Flow

$\eta$ Adiabatic or Polytropic Efficiency

$g$ Gravitational Constant (9.8 m/s2)

## 10.8. Heater/Cooler

The Heater and Cooler share a similar calculation model, where heat is added or removed from an inlet stream to simulate a heat exchange when the source is not another fluid/stream.

***Calculation Modes***

- **Energy Stream:** the energy flow from a connected stream is used to heat or cool the inlet stream.
- **Define Outlet Temperature:** the outlet temperature is defined and the amount of heat added or removed is calculated and written to the energy stream.
- **Define Outlet Vapor Fraction:** the quality of the outlet fluid is defined and the required amount of heat to add or remove is written to the energy stream.
- **Temperature Change:** the temperature difference is defined and the required amount of heat to add or remove is written to the energy stream.
- **Heat Added/Removed:** the heat added or removed is defined directly from the unit operation.

***Input/Output Parameters***

- **Pressure Drop (input only):** defines the pressure drop in the heater/cooler.
- **Heating/Cooling:** used as input in *Heat Added/Removed* calculation mode, otherwise it is a calculated value.
- **Efficiency (input only):** heating/cooling efficiency.
- **Outlet Vapor Fraction:** used as input in *Define Outlet Vapor Fraction* mode, otherwise it is a calculated value.
- **Outlet Temperature:** used as input in *Define Outlet Temperature* mode, otherwise it is a calculated value.
- **Temperature Change:** used as input in *Temperature Change* mode, otherwise it is a calculated value.

## 10.9. Shortcut Column

The shortcut column is used to calculate the minimum reflux and distribution of products in a distillation column by the method of Fenske-Underwood-Gilliland [1]. The column should have a single feed stage, two products (top and bottom), condenser (partial or total) and reboiler. The results are the minimum reflux, thermal loads and temperature of the condenser and reboiler for a fixed reflux ratio, in addition to determining the optimum feed stage and the minimum number of stages.

***Input Parameters***

- **Connections:** feed, product, top, bottom and heat loads (condenser / reboiler).
- **Type of condenser:** partial or total.
- **Reflux Ratio:** ratio between the flow of liquid that returns from the condenser to the column and the one that leaves the condenser as the top product.
- **Light Key:** component used as a reference so that the lighter ones are present only in the top product.
- **Heavy Key:** component used as a reference so that the heavier ones are present only in the product of fund.
- **Condenser pressure:** pressure of the condenser.
- **Reboiler pressure:** pressure of the reboiler.

***Output Parameters***

- **Minimum reflux:** reflux ratio of minimum to ensure the separation specified.
- **Minimum number of stages:** the minimum number of training which ensures the separation specified.
- **Optimal feed stage:** the feed stage that minimizes the thermal load of the reboiler.
- **Liquid** / **Vapor flows:** internal flows in sections of rectification and stripping of the column.
- **Thermal loads:** thermal loads of condenser and reboiler.

## 10.10. Rigorous Column (Distillation/Absorption)

The rigorous column is an unit operation that represents the fractionating towers, where components in a mixture are separated in various equilibrium stages. It is called *rigorous* because of the thermodynamic models used in the solution of the mass and energy balances throughout the column.

In DWSIM, two types of rigorous columns are available: **Distillation Column** and **Absorption Column**. They share the same basic interface, with the following characteristics:

- Supports multiple feed streams
- Supports multiple side draws
- Supports energy streams representing heat exchangers on each stage
- Definition of pressure and efficiency by stage

***Solving methods*** In the current version of DWSIM, three methods are available to solve the mass and energy balances in a column. They are:

- Bubble-Point method of Wang and Henke (distillation column) [2]
- *Sum-Rates* method of Burningham and Otto (absorption column)[2]
- Simultaneous Correction method by Napthali and Sandholm (distillation/absorption columns) [2]

***Results*** After the solution of mass and energy balances for the entire column, the output streams are calculated and the following results are shown in the properties window of the column:

- Heat exchanged in the condenser (where applicable)
- Heat exchanged in the reboiler (where applicable)
- Temperature profiles and flow of liquid / vapor entering at each stage
- Distribution profiles of components on each stage

## 10.11. Heat Exchanger

DWSIM has a model for the countercurrent, two-stream heat exchanger which supports phase change and multiple phases in a stream.

***Input Parameters*** The heat exchanger in DWSIM has seven calculation modes:

1. Calculate hot fluid outlet temperature: you must provide the cold fluid outlet temperature and the exchange area to calculate the hot fluid temperature.
2. Calculate cold fluid outlet temperature: in this mode, DWSIM needs the hot fluid outlet temperature and the exchange area to calculate the cold fluid temperature.
3. Calculate both temperatures: in this mode, DWSIM needs the exchange area and the heat exchanged to calculate both temperatures.
4. Calculate area: in this mode you must provide the HTC and both temperatures to calculate the exchange area.
5. Rate a Shell and Tube exchanger: in this mode you must provide the exchanger geometry and DWSIM will calculate output temperatures, pressure drop on the shell and tubes, overall HTC, LMTD, and exchange area. This calculation mode uses a simplified version of Tinker's method [3] for Shell and Tube exchanger calculations.
6. Pinch-Point (minimum temperature differente between outlet streams)
7. Specify Outlet Vapor Fraction (Stream 1 or Stream 2)

***Calculation Mode*** The heat exchanger in DWSIM is calculated using the simple convection heat equation:

$$Q = UA\Delta T_{ml}, \tag{10.17}$$

where: $Q$ = heat transferred, $A$ = heat transfer area (external surface) and $\Delta T_{ml}$ = Logarithmic Mean Temperature Difference (LMTD). We also remember that:

$$Q = m\Delta H, \tag{10.18}$$

where: $Q$ = heat transferred from/to the fluid and $\Delta H$ = outlet-inlet enthalpy difference.

The calculation procedure depends on the mode selected:

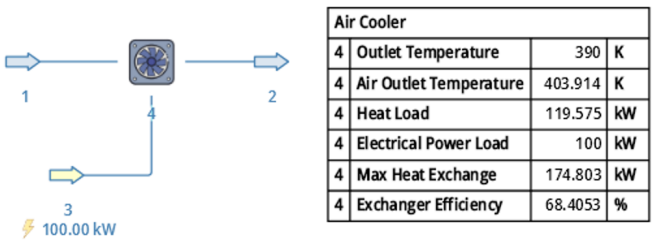
1. Calculate hot fluid outlet temperature: HTC (Heat Transfer Coefficient), hot fluid outlet temperature, heat load and LMTD.
2. Calculate cold fluid outlet temperature: HTC, cold fluid outlet temperature, heat load and LMTD.
3. Calculate both temperatures: HTC, cold and hot fluid outlet temperatures and LMTD.

4. Calculate area: exchange area and LMTD.
5. Rate Shell and Tube exchanger: exchanger geometry information.

***Results*** The results given by the heat exchanger after calculation depends on the mode selected:

1. Calculate hot fluid outlet temperature: overall HTC, hot fluid outlet temperature, heat load and LMTD.
2. Calculate cold fluid outlet temperature: overall HTC, cold fluid outlet temperature, heat load and LMTD.
3. Calculate both temperatures: overall HTC, cold and hot fluid outlet temperatures and LMTD.
4. Calculate area: exchange area and LMTD.
5. Rate Shell and Tube exchanger: area, LMTD, LMTD correction factor (F), overall HTC, hot fluid outlet temperature, cold fluid outlet temperature, hot fluid pressure drop (shell/tubes only), cold fluid pressure drop (shell/tubes only).

## 10.12. Air Cooler




| Air Cooler | | | |
|---|---|---|---|
| 4 | Outlet Temperature | 390 | K |
| 4 | Air Outlet Temperature | 403.914 | K |
| 4 | Heat Load | 119.575 | kW |
| 4 | Electrical Power Load | 100 | kW |
| 4 | Max Heat Exchange | 174.803 | kW |
| 4 | Exchanger Efficiency | 68.4053 | % |

Figure 72: Air Cooler model

The Air Cooler model is a subset of the Shell and Tube HX model, with air being the 'shell side' fluid.

***Input Parameters*** The Air Cooler model in DWSIM has three calculation modes:

- Specify Outlet Temperature: you must provide the fluid outlet temperature and DWSIM will calculate Overall UA and Heat Exchanged.
- Specify Tube Geometry: in this mode you must provide the tube geometry and DWSIM will calculate output temperatures, pressure drop at the tubes, overall HTC (U), LMTD, and exchange area. This calculation mode uses a simplified version of Tinker's method for Shell and Tube exchanger calculations, with a modification for the outside heat transfer coefficient (convection).
- Specify Overall UA: in this mode you must provide the Overall UA and DWSIM will calculate the Heat Exchanged and Outlet Temperature.

You can provide the pressure drop for the hot fluid in the exchanger for modes 1 and 3 only.

***Calculation Mode*** The Air Cooler is calculated using the simple convection heat equation:

$$Q = UA\Delta T_{ml}, \tag{10.19}$$

where: $Q$ = heat transferred, $A$ = heat transfer area (external surface) and $\Delta T_{ml}$ = Logarithmic Mean Temperature Difference (LMTD). We also remember that:

$$Q = m\Delta H, \tag{10.20}$$

where: $Q$ = heat transferred from/to the fluid and $\Delta H$ = outlet-inlet enthalpy difference.

## 10.13. Component Separator

The Component Separator is a mass balance unit operation. The components are separated between two streams, specified as fractions or absolute flow rates. The energy balance is then calculated after the separation.

### *Input Parameters*

- Specified stream: sets the stream to which the separation specifications will be applied. ”0” corresponds to the Outlet stream 1 (overhead) and ”1” corresponds to the Outlet stream 2 (bottoms).

### *Results*

- Energy imbalance: Difference between enthalpy of outlet and inlet streams. in some cases it can be interpreted as the energy necessary to do the separation.

## 10.14. Orifice Plate

This model corresponds to the ISO 5167 specification for Orifice Plates, and can be used to measure flow rates if used in conjunction with the Adjust logical operation (change flow rates until specified pressure drop matches the calculated one).

### *Input Parameters*

- Pressure tappings: select the option which corresponds to the arrangement of the tappings for pressure reading.
- Orifice diameter: inner diameter of the plate.
- Beta (d/D): ratio between plate’s inner and outer diameters.
- Correction factor: multiplier for the mass flow rate used in the calculation of the pressure drop across the orifice. Default is 1.

***Results***

- Orifice pressure drop: Pressure drop across the orifice. This is the value that is read through the tappings.
- Overall pressure drop: Pressure drop after recovery. This values should always be less or equal to the orifice pressure drop.
- Delta T: temperature drop across the orifice, considering that the process is an adiabatic expansion.

## 10.15. Custom Unit Operation

This Unit Operation lets you run scripts as the main calculation routine (inside the Calculate() procedure which is called by the calculator for all flowsheet simulation objects). There are up to six streams available for the UO, three as input and three as output.

Supported Languages are IronPython, IronRuby, VBScript and JScript. You can use some predefined reference variables inside your script, defined as shortcuts to the most common objects:

**ims1** Input Material Stream in slot 1 (MaterialStream class instance)

**ims2** Input Material Stream in slot 2 (MaterialStream class instance)

**ims3** Input Material Stream in slot 3 (MaterialStream class instance)

**oms1** Output Material Stream in slot 1 (MaterialStream class instance)

**oms2** Output Material Stream in slot 2 (MaterialStream class instance)

**oms3** Output Material Stream in slot 3 (MaterialStream class instance)

**Me** Reference variable to the Custom UO object instance (CustomUO UnitOperation class)

**Flowsheet** Reference variable to the active flowsheet object (FormChild class)

**Solver** Flowsheet solver class instance, used to send commands to the calculator (COMSolver class)

**Spreadsheet** Reference variable to the Spreadsheet object.

## 10.16. Solids Separator

The solids separator is used to separate solids from a liquid phase in a mixed material stream.

***Input Parameters***

- Solids Separation Efficiency: defines the amount of solids in the liquid stream. 100% efficiency means no solids in the liquid stream.
- Liquids Separation Efficiency: defines the amount of liquid in the solids stream. 100% efficiency means no liquid in the solids stream.

***Calculation Method*** The solids separator does a mass balance and splits the inlet stream phases into two distinct streams.

### 10.16.1. Continuous Cake Filter

In a continuous filter, the feed, filtrate and cake move at steady constant rates. It is evident that the process consists of several steps in series - cake formation, washing, drying and discharging - and that each step involves progressive and continual change in conditions. The pressure drop across the filter during cake formation is, however, held constant.

***Calculation Method*** For a continuous cake filter, the equation that relates the filter characteristics with the rate of solids production is [4]

$$\frac{\dot{m_c}}{A_T} = \frac{\left[2c\alpha\triangle Pfn/\mu + (nR_m)^2\right]^{0,5} - nR_m}{\alpha}, \tag{10.21}$$

where:

$\dot{m_c}$ rate of solids production, $kg/s$

$A_T$ total filter area, $m^2$

$\Delta P$ total pressure drop, $Pa$

$f$ fraction of filter area available for cake formation

$c$ solids concentration in the solids stream

$\alpha$ specific cake resistance, $m/kg$

$R_m$ filter medium resistance, $m^{-1}$

$n$ drum speed, $s^{-1}$

***Input and Output Parameters***

- Filter Medium Resistance: filter medium flow resistance;
- Specific Cake Resistance: specific cake flow resistance;
- Cycle Time: filter cycle time.
- Cake Relative Humidity: filter cake moisture in % wet basis;
- Filter Calculation Mode: Design or Simulation. If ***Design*** is selected, DWSIM will calculate the filter area given the total pressure drop. If ***Simulation*** is selected, it will do the opposite;
- Total Filter Area: filter area measured perpendicularly to the direction of flow;
- Pressure Drop: total pressure drop across the filter (cake + medium).

## 10.17. Excel Unit Operation

The Excel (Spreadsheet) Unit Operation is used to enable user defined models of unit operations which are calculated by Excel spreadsheet calculator.

Each instance of that unit operation model is associated with a separate calculation file. This file is preconfigured in a special way to recieve data from DWSIM for executing the calculations and hold results to be transfered back. The results are fetched from DWSIM to be fed back into the exit streams of that unit operation. You also may transfere user defined parameters from DWSIM to Excel and read other user defined parameters as results back to DWSIM. Up to four streams may be connected both to inlet and outlet of that unit. The energy stream recieves the calculatd energy from enthalpy balance.

**10.17.0.1. Calculation Parameters** Calculation parameters are defined inside the Excel definition file as input parameters.

**Editor:** This field displays the Excel file being associated with that unit. Click on file name to open Unit editor. Results Heat added After calculating the Excel user model DWSIM calculates the enthalpy balance of input and output streams. Other results are parameters calculated by the user model as output parameters. Annotation The selected parameter may contain an anotation which is defined inside the Excel file for.

**Search Button:** to search Excel file to be associated with Unit Operation

**Create New Button:** to create new definition file from template. Edit Opens definition file with Excel for editing. After editing you just save the Excel file from within Excel.

**Excel definition file:** the excel definition file has a preconfigured structure which should not be altered. Otherwise DWSIM may not be able to transfere information forth and back in the right way!

**10.17.0.2. Input Tab** DWSIM is writing informations of all connected input streams into the blue area. From line 12 downward all components and their molar flows are listed. The red area contains the parameters which are required for calculation. This parameters are displayed in DWSIM as "Calculation Parameters" inside the property tab of ExcelUO. You may list as many properties as you want here. DWSIM starts to read the list below the heading downwards until it finds an empty cell. Each parameter features a name, a value, a unit and an annotation.

**10.17.0.3. Output Tab** The output tab has the same structure as the input tab. Molare flows of each component of every output stream are to be written into their fields by the user defined calculation procedures. You also have to calculate temperature and pressure of all streams leaving the unit. The enthalpy of each stream is calculated by DWSIM automatically after finishing Excel calculations.

**10.17.0.4. Results** DWSIM is calculating the enthalpy balance of the unit from enthalpy of outlet streams minus enthalpy of inlet streams. The result of this calculation is written to the energy stream. After finishing the calculations in Excel, DWSIM checks the mass balance of that unit. If mass balance is not ok DWSIM will issue an error message.

## 10.18. Flowsheet Unit Operation

The Flowsheet Unit Operation allows you to run a XML simulation file as a block inside another flowsheet. This can be useful if you have a large simulation and want to split it in several, smaller blocks which can be run as independent simulations, making it easier to mantain, make modifications and fix errors in the smaller blocks.

Mass transfer between flowsheets is done in a per-compound basis, that is, any compound that doesn't match in both simulations will have its flow data erased in the inner flowsheet and will be ignored in the outer one. The mass is transferred to and from the inner flowsheet by matching inlet and outlet

Material Streams connected to the Flowsheet UO with streams in the inner flowsheet. Besides mass/-mole flow information, temperature, pressure and enthalpy are also written and read to and from the inner flowsheet.

DWSIM uses the Property Package models defined in the inner flowsheet to do the mass and energy balances. Only settings like Parallel CPU and Parallel GPU calculations affect the way that DWSIM does its calculations inside the block, since it will use the parameters defined when the inner flowsheet was last saved.

You can select Parameters/Properties from objects in the inner flowsheet to expose them to the outer flowsheet, allowing usage of these parameters in Optimization and Sensitivity Analysis cases, Script blocks and for displaying in the outer flowsheet as well.

**10.18.0.1. Connections** Ten inlet and ten outlet Material Stream ports are available for connecting with Material Streams from the inner flowsheet.

After connecting streams to the ports, you must open the Control Panel to map the connected streams to streams in the inner flowsheet.

**10.18.0.2. Calculation Parameters** **Simulation file:** selects the XML simulation file to use as the inner flowsheet.

**Control Panel:** opens the Control Panel to initialize the flowsheet, define stream mapping, expose input and output parameters from the inner flowsheet and define the mass transfer mode.

**Initialize on load:** if true, initializes the inner flowsheet during the opening of the main flowsheet.

**Update process data on saving:** when saving the main simulation file, DWSIM will update the process data from the inner flowsheet in the selected XML file. Only object process data calculated by the solver will be updated. Other settings will remain unchanged.

**Redirect calculator messages:** show calculation details from the inner flowsheet in the main flowsheet's log window.

**View flowsheet:** shows the inner flowsheet PFD.

**10.18.0.3. Linked Input Parameters** This section will display the parameters that you've selected in the control panel to use as input parameters in the outer flowsheet. They can be changed anytime and will trigger the calculation of the flowsheet block.

**10.18.0.4. Linked Output Parameters** This section will display the parameters that you've selected in the control panel to use as output parameters in the outer flowsheet. They are read-only and will be

updated only after the flowsheet block is calculated successfully.

**10.18.0.5. Results** **Mass balance error:** shows the mass balance error in %. It can be useful to detect orphan streams in the inner flowsheet, that is, streams that work as inlet or outlet streams in the inner flowsheet but aren't connected to any stream in the main flowsheet, as this may lead to large mass balance errors.

**Control Panel:** Use the Control Panel to initialize the flowsheet. Only after initialization you'll be able to make the connection and expose parameters from the inner flowsheet. You can also use the "Initialize/Reload" button to reload the simulation file after you've done changes in the simulation by opening it in another DWSIM window.

**10.18.0.6. Viewing the inner flowsheet** By clicking on the "View Flowsheet" button in the property grid you'll be able to view the inner flowsheet, check object properties and the overall layout. You can't change anything here, so any attempt to do so will result in an error.

## 10.19. PEM Fuel Cell

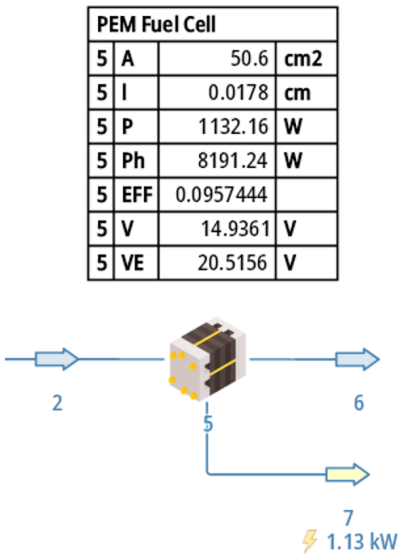



Figure 73: PEM Fuel Cell model

**Proton-exchange membrane fuel cells (PEMFC)**, also known as polymer electrolyte membrane (PEM) fuel cells, are a type of fuel cell being developed mainly for transport applications, as well as for stationary fuel-cell applications and portable fuel-cell applications. Their distinguishing features include lower temperature/pressure ranges (50 to 100 řC) and a special proton-conducting polymer electrolyte membrane. PEMFCs generate electricity and operate on the opposite principle to PEM electrolysis, which consumes electricity.

The PEM Fuel Cell model in DWSIM is an interface for the **Amphlett Static Model** from the **OPEM Python Library** (https://www.ecsim.ir/opem/).

The Amphlett static model has been used to predict the performance of proton exchange membrane fuel cell. Key concepts in Amphlett static model are Nernst voltage, activation polarization loss, ohmic polarization loss and concentration polarization loss. Amphlett static model has a mechanistic and empirical approach to describe the performance of proton exchange membrane fuel cell. The ideal standard potential of an H2/O2 fuel cell is 1.229 V with liquid water product. The actual cell potential is decreased from its reference potential because of irreversible losses.

For more information about the model inputs and outputs, please visit https://www.ecsim.ir/opem/doc/Static/Amphlett.

## 10.20. Water Electrolyzer

| Electrolyzer | | | |
|---|---|---|---|
| 4 | Voltage | 12 | V |
| 4 | Number of Cells | 5 | |
| 4 | Cell Voltage | 2.4 | V |
| 4 | Waste Heat | 0.0592673 | kW |
| 4 | Current | 10.1312 | A |
| 4 | Electron Transfer | 0.000525011 | mol/s |

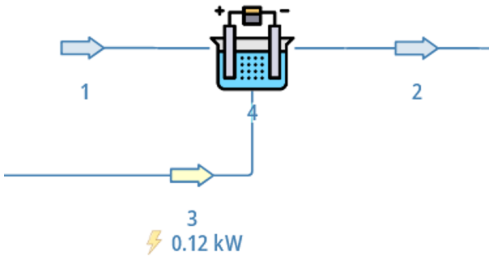



Figure 74: Water Electrolyzer model

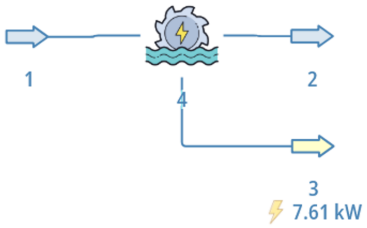
Electrolysis of water, also known as electrochemical water splitting, is the process of using electricity to decompose water into oxygen and hydrogen gas by a process called electrolysis. Hydrogen gas released in this way can be used as hydrogen fuel, or remixed with the oxygen to create oxyhydrogen gas, which is used in welding and other applications.

Electrolysis of water requires a minimum potential difference of 1.23 volts, though at that voltage external heat is required from the environment.

### Setup and Calculation Guide

- The Water Electrolyzer model requires Water, Hydrogen and Oxygen added to the simulation with Liquid Water present in the inlet stream.
- Input Parameters: Total Voltage and Number of Cells.
- Output Parameters: Cell Voltage, Current, Electron Transfer and Waste Heat.
- After the calculation, the generated power is directed to the energy stream, while the waste heat is added to the outlet material stream, increasing its temperature.

## 10.21. Hydroelectric Turbine




| Hydroelectric Turbine | | | |
|---|---|---|---|
| 4 | Efficiency | 75 | |
| 4 | Velocity Head | 0.0382653 | m |
| 4 | Static Head | 1 | m |
| 4 | Total Head | 1.03827 | m |
| 4 | Inlet Velocity | 1 | m/s |
| 4 | Outlet Velocity | 0.5 | m/s |
| 4 | Generated Power | 7.60872 | kW |

Figure 75: Hydroelectric Turbine model

A Hydroelectric Turbine is a rotary machine that converts kinetic energy and potential energy of water into mechanical work.

Water turbines were developed in the 19th century and were widely used for industrial power prior to electrical grids. Now, they are mostly used for electric power generation. Water turbines are mostly found in dams to generate electric power from water potential energy.

### Setup and Calculation Guide

- The Hydroelectric Turbine model converts head and velocity from the inlet stream into usable energy for the process.
- Input parameters: Static Head, Inlet Velocity, Outlet Velocity and Efficiency.
- Output parameters: Velocity Head, Total Head, Generated Power.
- The generated power is calculated from

$$P = \eta \, \rho \, g \, h \, q \tag{10.22}$$

where:

$P$ generated power (W)

$\eta$ efficiency (0.00-1.00)

$\rho$ fluid density (kg/m3)

$g$ acceleration of gravity (9.81 m/s2)

$h$ total head (m)

- Total head is calculated from

$$h = h_s + h_v \tag{10.23}$$

where

$h_s$ static head (m)

$h_v$ velocity head (m), calculated from

$$h_v = \frac{v_{in}^2 - v_{out}^2}{2g} \tag{10.24}$$

## 10.22. Wind Turbine


Figure 76: Wind Turbine model

A wind turbine is a device that converts the kinetic energy of wind into electrical energy. Hundreds of thousands of large turbines, in installations known as wind farms. They are an increasingly important source of intermittent renewable energy, and are used in many countries to lower energy costs and reduce reliance on fossil fuels.

### Setup and Calculation Guide

- The Wind Turbine model converts energy from air (wind) into usable energy for the process.
- Input parameters: Wind Speed, Atmospheric Temperature and Pressure, Relative Humidity, Rotor Diameter, Efficiency and Number of Units.
- Output parameters: Generated Power, Maximum Theoretical Power and Calculated Air Density.
- Atmospheric conditions are used to calculate the density of air.
- Conservation of mass requires that the amount of air entering and exiting a turbine must be equal. Accordingly, Betz's law gives the maximal achievable extraction of wind power by a wind turbine

as 16⁄27 (59.3%) of the rate at which the kinetic energy of the air arrives at the turbine (ref. "The Physics of Wind Turbines Kira Grogg Carleton College, 2005, p. 8").

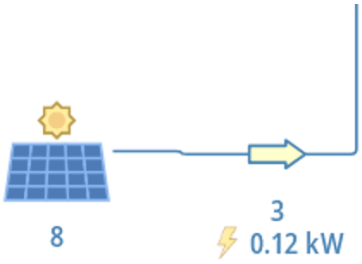
➞ The maximum theoretical power output of a wind machine is thus 16/27 times the rate at which kinetic energy of the air arrives at the effective disk area of the machine. If the effective area of the disk is $A$, and the wind velocity $v$, the maximum theoretical power output $P_{max}$ is:

$$P_{max} = \frac{16}{27}\frac{1}{2}\rho v^3 A = \frac{8}{27}\rho v^3 A, \tag{10.25}$$

where $\rho$ is the air density. The actual generated power is given by

$$P = n\,\eta\,P_{max} \tag{10.26}$$

## 10.23. Solar Panel




| Solar Panel | | | |
|---|---|---|---|
| 8 | Efficiency | 15 | % |
| 8 | User-Defined Solar Irradiation | 1 | kW/m2 |
| 8 | Panel Area | 1 | m2 |
| 8 | Number of Panels | 1 | |
| 8 | Generated Power | 0.121574 | kW |

Figure 77: Solar Panel model

A solar cell panel, solar electric panel, photo-voltaic (PV) module or solar panel is an assembly of photo-voltaic cells mounted in a framework for installation. Solar panels use sunlight as a source of energy to generate direct current electricity. A collection of PV modules is called a PV panel, and a system of PV panels is called an array. Arrays of a photovoltaic system supply solar electricity to electrical equipment.

### Setup and Calculation Guide

➞ The Solar Panel converts energy from solar irradiation into usable energy for the process.

➞ Input parameters: Solar Irradiation (kW/m2), Panel Area, Panel Efficiency and Number of Panels.

➞ Output parameters: Generated Power.

➞ Generated Power is calculated from

$$P = \eta\, n\, S\, A \tag{10.27}$$

where

$P$ generated power (kW)

$\eta$ panel efficiency

$n$ number of panels

$S$ solar irradiation (kW/m2)

$A$ panel area (m2)

# 11. Reactors

The reactors in DWSIM are specialized modules that solve a particular system of reactions of the same type in sequence or in parallel. There are five different types of reactors available:

➞ Continuous Stirred Tank Reactor (CSTR)

➞ Plug Flow Reactor (PFR)

➞ Gibbs Minimization Reactor

➞ Equilibrium Reactor

➞ Conversion Reactor

To run a simulation of a reactor, the user needs to define the chemical reactions which will take place in the reactor. This is done through the **Reactions Manager**, accessible through the **Simulation Settings** panel (Classic UI) or on the **Basis Panel** (Cross-Platform UI).

Reactions can be of Equilibrium, Conversion, Kinetic or Heterogeneous Catalytic types. One or more reactions can be combined to define a Reaction Set. The reactors then "see" the reactions through the reaction sets.

Equilibrium reactions are defined by an equilibrium constant ($K$). The source of information for the equilibrium constant can be a direct gibbs energy calculation, an expression defined by the user or a constant value. Equilibrium reactions can be used in Equilibrium and Gibbs reactors.

Conversion reactions are defined by the amount of a base compound which is consumed in the reaction. This amount can be a fixed value or a function of the system temperature. Conversion reactions are supported by the Conversion reactor.

Kinetic reactions are reactions defined by a kinetic expression. These reactions are supported by the PFR and CSTR reactors.

Heterogeneous Catalytic reactions in DWSIM must obey the Langmuir–Hinshelwood mechanism, where compounds react over a solid catalyst surface. In this model, reaction rates are a function of catalyst amount (i.e. mol/kg cat.s). These reactions are supported by the PFR and CSTR reactors.

## 11.1. Input Parameters

All reactors share the same basic interface. Properties that need to be set include Input and Output Streams Energy Streams that represent the heat exchanged with the environment Reaction Set to be used Operation Mode (isothermal or adiabatic) Pressure Drop through the reactor. For the PFR and CSTR, it is also necessary to define: Volume of the reaction medium Catalyst loading, particle diameter and void fraction, for Heterogeneus Catalytic reaction simulations only For the Gibbs Reactor, you must select the calculation mode: Direct Minimization: The equilibrium composition is found in such a way that the final gibbs energy is at is minimum. Reaction Extents: Like in the Equilibrium Reactor, the equilibrium composition is found according to the reaction information in the reaction set.

## 11.2. Output Parameters

The output of the reactor calculations includes conversions of the components involved in the reactions, the temperature change and the heat exchanged in the reactor, among other information.

For the PFR, the concentration profiles of reactants and products is shown along the longitudinal axis of the reactor (assuming that the concentration does not vary radially).

## 11.3. Calculation Methods

### 11.3.1. Conversion Reactor

The Conversion Reactor is solved by simple energy and mass balances, calculating the reaction heat by considering the variation in the amount of the base component.

The defined conversion for a compound in a particular reaction is used to calculate the final mole amounts with

$$n_j = n_{j0} - \sum X_{ji} \frac{\nu_{ji}}{\nu_{bi}} n_{b0} \tag{11.1}$$

where $n_j$ and $n_{j0}$ are the final and initial mole amounts of compound j, respectively, $X_{ji}$ is the conversion (0.0 - 1.0) of compound $j$ in reaction $i$, $\nu_{ji}$ is the stoichiometric coefficient of compound $j$ in reaction $i$, $\nu_{bi}$ is the stoichiometric coefficient of the defined base compound for reaction $i$, and $n_{b0}$ is the initial mole amount of defined base compound for reaction $i$.

Since a compound can be part of multiple reactions is parallel, DWSIM solves the above problem by minimizing the difference between defined and calculated (final) conversions, subject to all mole flows being equal to or higher than zero. This solution scheme for parallel reactions guarantees that the mass balance is preserved, even if the final conversion values aren't reached due to limited reactant amounts.

### 11.3.2. Equilibrium Reactor [5]

#### Chemical Reaction Equilibrium

In phase equilibrium calculations for a given feed at specified temperature and pressure a material balance must be satisfied for each component in the mixture, the total amount in the combined product phases being identical to that in the feed. When chemical reactions occur, additional degrees of freedom

are available, resulting in a set of material balance constraints, which is smaller than the number of components in the mixture.

The mixture composition at chemical equilibrium at constant $T$ and $P$ satisfies the condition of minimum Gibbs energy,

$$\min G = \min \sum_{i=1}^{C} n\mu_i \tag{11.2}$$

subject to a set of $M < C$ material balance constraints. In addition we must require that

$$n_i \geq 0, i = 1, 2, ..., C \tag{11.3}$$

The material balance constraints can be formulated in different ways, and the most important formulations are outlined below. To illustrate the concepts, we shall consider a specific example, the combustion of a mixture of 1 mole propane ($C_3H_8$) and 5 moles oxygen ($O_2$), at 2200 K, 4 MPa. Under these conditions the reaction mixture is assumed to contain the following species (components) at equilibrium: $CO_2$ (1), $CO$ (2), $H_2O$ (3), $O_2$ (4), $H_2$ (5), $O$ (6), $H$ (7) and $OH$ (8).

**Independent Chemical Reactions and Reaction Extents**

One approach eliminates the material balance constraints by formulating a complete set of independent chemical reactions between the mixture components. In our example, the following reactions could be chosen:

$2CO2 \leftrightarrow 2CO + O2$

$2H2O2 \leftrightarrow H2 + O2$

$H2 \leftrightarrow 2H$

$O2 \leftrightarrow 2O$

$H2 + O2 \leftrightarrow 2OH$

Each reaction is characterized by a stoichiometric vector, $\nu$, where the i'th element of the vector represents the stoichiometric coefficient of component i in the reaction, with a negative sign for components on the left hand side and positive on the right hand side. Thus,

$\nu_1 = (-2, 2, 0, 1, 0, 0, 0, 0)^T$

The chosen set of chemical reactions must have linearly independent stoichiometric vectors, i.e., none of the reactions may be linear combinations of other reactions. If e.g. the reaction

$CO2 + H2 \leftrightarrow CO + H2O$

was chosen a candidate for a potential 6th reaction, its stoichiometric vector

$$\nu_6 = (-1, 1, 1, 0, -1, 0, 0, 0)^T$$

shows that the choice is illegal, since $\nu_6 = 1/2(\nu_1 - \nu_2)$.

The $R$ (here, $R = 5$) stoichiometric vectors are combined into an $C\,x\,R$ matrix $E$, given by

$$\mathbf{E} = (\nu_1, \nu_2, ..., \nu_R)$$

A test for linear independence of the stoichiometric vectors is that the matrix $E$ must be of rank $R$. Given an initial composition vector $\nu_0$ consistent with the overall feed composition, the vector of moles n

can be written in the general form

$$\mathbf{n} = \mathbf{n_0} + \sum_{k=1}^{R} \nu_k \zeta_k \tag{11.4}$$

or

$$\mathbf{n} = \mathbf{n_0} + \mathbf{E}\zeta \tag{11.5}$$

where $\zeta_k$ is called the extent of the $k$'th reaction.

The composition vector no can be chosen as the feed composition, if all components present in the feed are also found in the equilibrium mixture. For the present example, the equilibrium concentration of propane is likely to be extremely small (below 10-30) and propane is therefore not included in the vector of possible product components. Some consistent choices of no are

$$\mathbf{n_0} = (3, 0, 4, 0, 0, 0, 0, 0)^T$$

(3 moles $CO_2$, 4 moles $H_2O$) or

$$\mathbf{n_0} = (0, 3, 0, 3.5, 4, 0, 0, 0)^T$$

(3 moles $CO$, 3.5 moles $O_2$, 4 moles $H_2$). An alternative possibility is of course to introduce $C_3H_8$ as an additional component in the reaction mixture, together with a reaction involving propane, e.g.

$C_3H_8 + 5O_2 \leftrightarrow 3CO_2 + 4H_2O$

in which case the feed composition is chosen as the no-vector.

The substitution of the mole vector enables us to formulate the equilibrium calculation in terms of the $R$ independent variable $\zeta_k$, i.e.,

$$\min G = \min G(\mathbf{n}(\zeta_k)) \tag{11.6}$$

At equilibrium the derivatives of $G$ with respect to the reaction extents must equal zero

$$\frac{\partial G}{\partial \zeta_k} = \sum_{i=1}^{C} \frac{\partial G}{\partial n_i} \frac{\partial n_i}{\partial \zeta_k} = \sum_{i=1}^{C} \mu_i E_{i,k} = 0,\ k = 1, 2, ..., R \tag{11.7}$$

Any suitable procedure can be used for $G$, and the approach based on reaction extents can be used without essential differences for ideal as well as for non-ideal mixtures.

The reaction extent approach is best suited for problems where the number of independent reactions is small, or where the mixture is highly nonideal.

**Solving Method**

DWSIM uses an inner loop to converge the reaction extents using information about the equilibrium constant for each reaction,

$$\ln K = \prod basis^{\nu} \tag{11.8}$$

or, in a non-linear system of equations notation,

$$\ln K_i - \prod basis_j^{\nu_{ji}} = 0 \tag{11.9}$$

where $K_i$ is the equilibrium constant or the reaction $i$, $basis_j$ is the reaction basis (activity, fugacity, partial pressure, etc.) for compound $j$, and $\nu_{ji}$ is the stoichiometric coefficient of compound $j$ in reaction $i$.

The equilibrium constant is evaluated at $T$ plus a **Temperature Approach** value which can be defined in the equilibrium reaction editor.

The system of nonlinear equations defined by 11.9 can be solved using a Newton-like solver or by a local minimization solver type like IPOPT, where the problem is rearranged in a single equation to be minimized using the reaction extents as variables.

### 11.3.3. Gibbs Reactor [5]

The calculation of chemical equilibrium at specified temperature and pressure is in many ways similar to the calculation of phase equilibrium. In both cases the equilibrium state corresponds to a global minimum of the Gibbs energy subject to a set of material balance constraints.

The alternative formulation of the constraints Is based on the requirement of conservation of chemical elements. A key concept in this approach is the formula matrix for the reaction components. In this matrix, $A_{ji}$ Is the formula content of element $j$ in component $i$.

The element conservation constraints can be written

$$An = b \tag{11.10}$$

where $b_k$ is the total amount of element $k$ in the reaction mixture.

#### Solving Method

DWSIM solves the multiphase Gibbs minimization problem

$$\min G = \min \sum_{i=1}^{C} n_i \mu_i \tag{11.11}$$

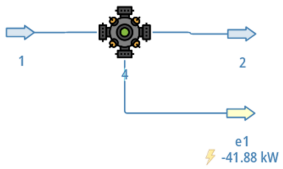
using the reactive compound overall molar fractions as the minimization variables, while obeying the element and mass conservation constraints.

IPOPT (or an external solver) is used to minimize $G$. For each solver iteration, a multiphase flash calculation is performed using the current tentative molar fractions, and the $G$ for each phase is calculated using the compound fugacities as provided by the currently selected Property Package. Total $G$ is calculated as the sum of the phase $G$s multiplied by their amounts.")

Chemical potentials are calculated from fugacities using the following relationship:

$$\mu_{ik} = \mu_0 + RT \ln f_{ik} + RT \ln x_{ik} \tag{11.12}$$

### 11.3.4. Gibbs Reactor (Reaktoro)




| Compositions | | | |
|---|---|---|---|
| Object | 1 | 2 | |
| Temperature | 1000.0 | 1000.0 | K |
| Pressure | 101325.0 | 101325.0 | Pa |
| Mass Flow | 0.08613 | 0.08613 | kg/s |
| Molar Flow | 5.0 | 5.5034 | mol/s |
| Molar Fraction (Mixture) / Carbon dioxide | 0.0 | 0.04381 | |
| Molar Fraction (Mixture) / Hydrogen | 0.0 | 0.18101 | |
| Molar Fraction (Mixture) / Carbon monoxide | 0.0 | 0.00193 | |
| Molar Fraction (Mixture) / Methane | 0.4 | 0.31768 | |
| Molar Fraction (Mixture) / Water | 0.6 | 0.45557 | |
| Molar Fraction (Mixture) / Ethane | 0.0 | 0.0 | |
| Molar Fraction (Mixture) / Oxygen | 0.0 | 0.0 | |

Figure 78: Gibbs Reactor (Reaktoro) model

The Reaktoro version of the Gibbs Reactor is a general-purpose isothermal reactor where the product species are determined on-the-fly. The user defines the reactive compounds, active elements and other parameters. The products formed will be a function of the global system Gibbs Free Energy according to the selected reactants and elements.

#### Setup and Calculation Guide

1. Setup the reactor connections.
2. Phases: select the active phases (Aqueous, Gaseous, Liquid and Mineral) in the reactor output. You need to select at least one phase.
3. Pressure Drop: set the pressure drop across the reactor.
4. Component Database: select the component database to be used.
5. Property Package: select the Property Package to be used.
6. Compounds: select the compounds that will be fed to the reactor.
7. Elements: select the active elements in the reactor.
8. Compound Names: you can change the compound name mappings to match specific compounds in the currently selected database.
9. Run the calculation once to get a list of product species.
10. Go back to the editor and map the product species back to DWSIM compounds. If more than one species is mapped to the same compound, the outlet flow of the compound will be the sum of the flows of the mapped species.
11. Run the calculation again to obtain the correct product flows.

### 11.3.5. PFR/CSTR

The PFR and CSTR are solved by a numerical method for systems of ordinary differential equations (ODEs):

$$N_i = N_{i0} + \int_0^V r_i dV \tag{11.13}$$

or

$$\frac{dN_i}{dV} = r_i \tag{11.14}$$

which forms a system of equations that must be solved using an ODE solver for each $dV$, since $r_i$ depends on $N_i$.

If $r_i$ is defined through a standard rate equation (Arrhenius),

$$k_{iF} = A_{iF} \exp\left(-E_{iF}/RT\right) \tag{11.15}$$

$$k_{iR} = A_{iR} \exp\left(-E_{iR}/RT\right) \tag{11.16}$$

$$r_i = k_{iF} \prod basis_{Fj}^{\nu_{ji}} - k_{iR} \prod basis_{Rj}^{\nu_{ji}} \tag{11.17}$$

where A is the pre-exponential factor and E is the Activation Energy, for Forward ($F$) and Reverse ($R$) reactions, $basis$ is the reaction basis (concentration, activity, fugacity, partial pressure, etc.) for compound $j$, and $\nu_{ji}$ is the stoichiometric coefficient of compound $j$ in reaction $i$.

$r_i$ can also be defined as a temperature-dependent equation or as a more advanced formulation defined through a Python script.

If the reaction is heterogeneous,

$$r_i = \frac{numerator = f(T,\, basis_{ji})}{denominator = f(T,\, basis_{ji})} \tag{11.18}$$

#### ➙ Non-Adiabatic Non-Isothermal Calculation Mode

The PFR supports and additional calculation mode called *Non-Adiabatic Non-Isothermal*. In this mode, the heat exchanged is defined by the Energy Stream attached to the PFR. A fraction of the total energy flow entering or leaving the reactor is added or removed at each volume step calculation.

## 12. Logical Operations

### 12.1. Recycle

The Recycle operation is composed by a block in the flowsheet which does convergence verification in systems were downstream material connects somewhere upstream in the diagram. With this tool it is possible to build complex flowsheets, with many recycles, and solve them in an efficient way by using the acceleration methods presents in this logical operation.

The user can define convergence parameters for temperature, pressure and mass flow in the recycle, that is, the minimum acceptable values for the difference in these values between the inlet and outlet streams, which, rigorously, must be identical. The smaller these values are, the more time is used by the calculator in order to converge the recycle.

A new feature added to the Recycle block in DWSIM v8.8 is the Smoothing Factor ($\alpha$), a parameter that varies from 0.1 to 1.0 and controls how updated parameters (Temperature, Pressure and Flows) are written to the outlet stream. A value equal to 0.7 means that the new parameters will be calculated taking into consideration 70% of the new value plus 30% of the existing one:

$$T_{new} = \alpha T_{new} + (1 - \alpha) T_{old} \tag{12.1}$$

$$P_{new} = \alpha P_{new} + (1 - \alpha) P_{old} \tag{12.2}$$

$$W_{new} = \alpha W_{new} + (1 - \alpha) W_{old} \tag{12.3}$$

## 12.2. Energy Recycle

The Energy Recycle logical operation works the same way as the normal Recycle, except that it is aimed at Energy Streams. Here, instead of defining convergence parameters for temperature, pressure and flow rate, you'll define the convergence error for the energy flow only.

## 12.3. Adjust

The Adjust is a logical operation which changes the value of a variable in a stream in order to attain a specification which can be a user-defined value or the value of other variable, in other stream. The adjust operation is very useful when there is a specification which cannot be accomplished directly, imposing the necessity of doing a trial and error calculation. If this is the case, the Adjust does everything automatically.

The user selects the controlled (specified) variable and the manipulated one. Then he defines the parameters:

- Adjust value (or offset): desired value for the variable or the value to be added or subtracted from the referenced variable.
- Maximum iterations: maximum number of iterations to be executed by the adjust;

In the Adjust Control Panel, the user controls the operation convergence. Two convergence methods are available.

## 12.4. Specification

The Specification Logical Operation is used to make a variable assume a value which is function of other variable, from other object. In opposition to the Adjust Op, the Spec is calculated automatically if the source variable is modified during the flowsheet calculation. In order for the calculator do not make any repeated calculation in the target object, one can define the "Calculate target object?" as *False*.

In the Specification Control panel, the user can define the relation between the variables by writing a function as complex as required. The user can click on the "Evaluate Expression" to check if the expression is valid.

# Part V.

# Dynamic Modeling and Simulation

## 13. Introduction

The key difference between the steady-state models and dynamic process models is the ability to take into account variation over time. However, as can be seen below the difference is not simply the addition of a time dimension; dynamic modeling often brings a whole different approach that results in dynamic models being a much truer-to-life representation of the process in many other respects.

While steady-state analysis is mainly used for process flowsheet design, usually to determine mass and energy balances and approximate equipment sizes, or perhaps stream properties, the ability of dynamic models to model transient behavior opens up a whole new world of application. Typical applications of dynamic models are as follows:

- ➙ analysis of transient behavior, including performance during start-up, shutdown, and load change;
- ➙ regulatory (i.e., PID) control scheme analysis and design;
- ➙ design of optimal operating procedures – for example, to optimize transition between product grades;
- ➙ design of batch processes;
- ➙ design of inherently dynamic continuous processes;
- ➙ fitting data from non steady-state operations – for example, dynamic experiments, which contain much more information than steady-state experiments, or estimation of process parameters from transient plant data;
- ➙ safety analysis;
- ➙ inventory accounting and reconciliation of plant data;
- ➙ online or offline parameter re-estimation to determine key operating parameters;
- ➙ online soft-sensing;
- ➙ operator training.

A dynamic simulation requires increased calculation time and is mathematically more complex than a steady-state one. It can be seen as a multiply repeated steady-state simulation (based on a fixed time step) with constantly changing parameters.

The dynamic model shares the same physical property packages as the steady-state model, simulating the behavior of the chemical system in a similar manner. On the other hand, the dynamic model uses a different set of conservation equations which account for changes occurring over time.

The equations for material, energy, and composition balances include an additional accumulation (volume) term which is differentiated with respect to time. Numerical integration is used to determine the

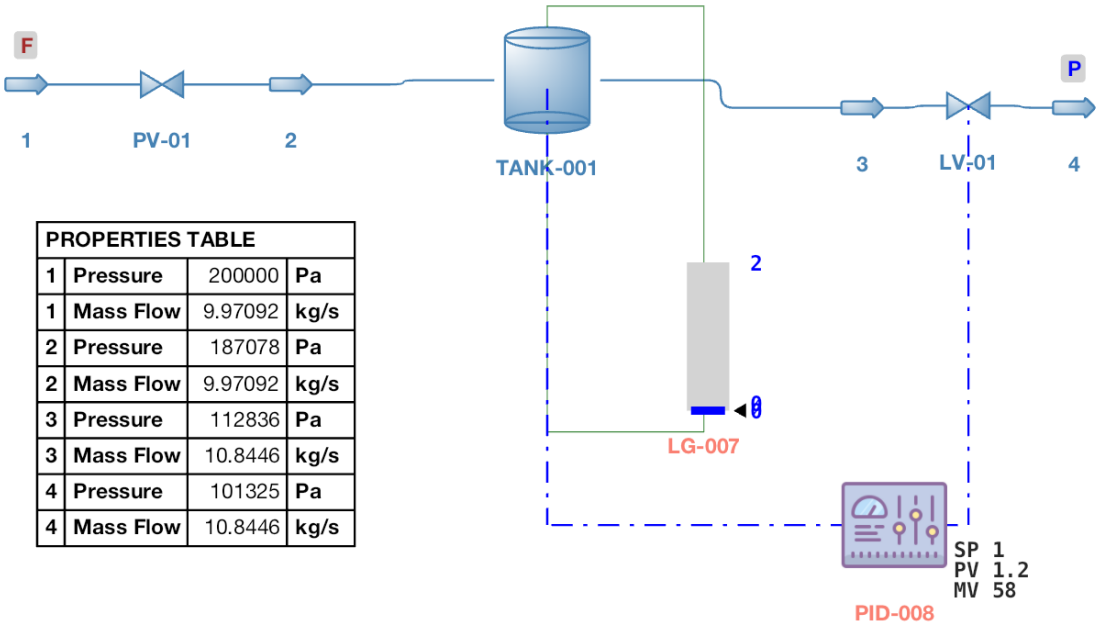
process behavior at sequential time steps. The smaller the step, the more closely the calculated solution matches the analytic solution. However, this gain in rigor is offset by the additional calculation time required to simulate the same amount of elapsed real time. A reasonable compromise is achieved by using the largest possible step size, while maintaining an acceptable degree of accuracy without becoming unstable.

In DWSIM, dynamic modeling is handled by the **Dynamics Manager**. Dynamic modeling components include **Events** and **Event Sets**, **Cause-and-Effect Matrices**, **Integrators**, **Monitored Variables** and **Schedules**.

# 14. Dynamic Simulation Structure and Configuration

## 14.1. Dynamic Model Setup

A dynamic model in DWSIM can be configured by starting from a solved flowsheet in steady-state mode. Additionally, each Boundary Material Stream must have a Pressure or Flow specification. In dynamic mode, a Valve Unit Operation connected to these streams will determine the Pressure-Flow relationships for the rest of the flowsheet as the remaining Unit Operation blocks get solved at each time step (Figure 40).




| PROPERTIES TABLE | | | |
|---|---|---|---|
| 1 | Pressure | 200000 | Pa |
| 1 | Mass Flow | 9.97092 | kg/s |
| 2 | Pressure | 187078 | Pa |
| 2 | Mass Flow | 9.97092 | kg/s |
| 3 | Pressure | 112836 | Pa |
| 3 | Mass Flow | 10.8446 | kg/s |
| 4 | Pressure | 101325 | Pa |
| 4 | Mass Flow | 10.8446 | kg/s |

Figure 79: Example of a valid flowsheet for dynamic mode. Both boundary streams are connected to valves. The inlet stream has a Flow specification while the outlet stream is Pressure-specified.

## 14.2. Event Sets

Dynamic Events are property changes that occur in a defined time step. A group of Dynamic Events is called an Event Set.

Property change events can transition from a previous state 'suddenly' (step change), linearly, log-linearly, inverse log-linearly or randomly between the two states (reference state and the one defined by the event itself).

### 14.3. Cause-and-Effect Matrices

Cause-and-Effect Matrices consist in a set of property changes that occur after an alarm is activated during the dynamic simulation. Each alarm activation triggers a property change defined by the user.

### 14.4. Integrators

Integrators are the components responsible to run the flowsheet sequentially in a dynamic simulation. They can be configured to run during a specified duration in fixed intervals, triggering the mass and balance solvers according to the user preference.

#### 14.4.1. Monitored Variables

Each Integrator can have a set of monitored variables, that is, a set of object properties which will have their values stored after each integration step. Monitored variables can have their values exported for later visualization in the Spreadsheet.

### 14.5. Schedules

The Schedule is a set of definitions which, together, can be considered as a *case study* in the context of a dynamic simulation. A Schedule must have the following items defined in order to run:

- Associated Integrator (Required)
- Event Set (Optional)
- Cause-and-Effect Matrix (Optional)
- Initial Flowsheet State (Optional)

A Schedule can be run from the **Integrator Controls** window. After running a Schedule, the results (values of monitored variables at each step) can be copied to a new spreadsheet.

## 15. Dynamic Simulation Flowsheet Blocks

### 15.1. Analog Gauge

Displays an analog gauge on the flowsheet. The gauge is associated to a property of an object on the flowsheet.

### 15.2. Digital Gauge

Displays a digital gauge on the flowsheet. The gauge is associated to a property of an object on the flowsheet.

## 15.3. Level Gauge

Displays a level gauge on the flowsheet. The gauge is associated to a property of an object on the flowsheet.

## 15.4. PID Controller

Introductory text taken from Wikipedia: `https://en.wikipedia.org/wiki/PID_controller`

### 15.4.1. Introduction

A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value $e(t)$ as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.

In practical terms, it automatically applies accurate and responsive correction to a control function. An everyday example is the cruise control on a car, where ascending a hill would lower speed if only constant engine power were applied. The controller's PID algorithm restores the measured speed to the desired speed with minimal delay and overshoot by increasing the power output of the engine.

### 15.4.2. Fundamental Operation

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram on (40) shows the principles of how these terms are generated and applied. It shows a PID controller, which continuously calculates an error value $e(t)$ as the difference between a desired setpoint $\mathrm{SP} = r(t)$ and a measured process variable $\mathrm{PV} = y(t)$, and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error over time by adjustment of a control variable $u(t)$, such as the opening of a control valve, to a new value determined by a weighted sum of the control terms.

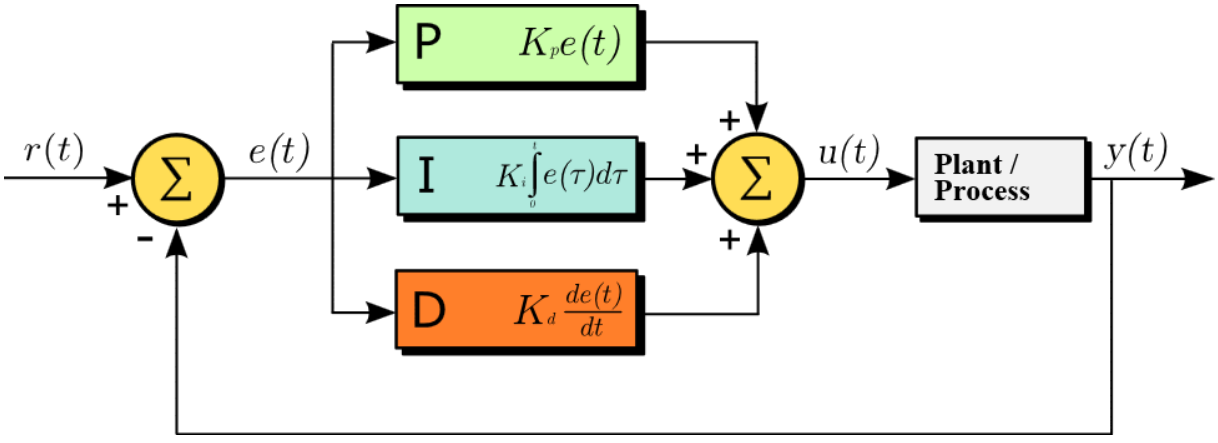



Figure 80: A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV). (source: wikipedia)

In this model:

- Term **P** is proportional to the current value of the PV – SP error $e(t)$. For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor "K". Using proportional control alone will result in an error between the setpoint and the actual process value, because it requires an error to generate the proportional response. If there is no error, there is no corrective response.
- Term **I** accounts for past values of the PV – SP error and integrates them over time to produce the I term. For example, if there is a residual PV – SP error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.
- Term **D** is a best estimate of the future trend of the PV – SP error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP – PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or dampening effect.

**Tuning** The balance of these effects is achieved by loop tuning to produce the optimal control function. The tuning constants are shown as "K" and must be derived for each control application, as they depend on the response characteristics of the complete loop external to the controller. These are dependent on the behaviour of the measuring sensor, the final control element (such as a control valve), any control signal delays and the process itself. Approximate values of constants can usually be initially entered knowing the type of application, but they are normally refined, or tuned, by "bumping" the process in practice by introducing a setpoint change and observing the system response.

**Control action** The mathematical model and practical loop above both use a "direct" control action for all the terms, which means an increasing positive error results in an increasing positive control output for the summed terms to apply correction. However, the output is called "reverse" acting if it is necessary to apply negative corrective action. For instance, if the valve in the flow loop was 100–0% valve opening for 0–100% control output – meaning that the controller action has to be reversed. Some process control schemes and final control elements require this reverse action. An example would be a valve for cooling water, where the fail-safe mode, in the case of loss of signal, would be 100% opening of the valve; therefore 0% controller output needs to cause 100% valve opening.

**Integral wind-up** Integral wind-up, also known as integrator wind-up or reset wind-up, refers to the situation in a PID feedback controller where a large change in setpoint occurs (say a positive change) and the integral term accumulates a significant error during the rise (wind-up), thus overshooting and continuing to increase as this accumulated error is unwound (offset by errors in the other direction). The specific problem is the excess overshooting. This problem can be addressed by preventing the integral term from accumulating above or below pre-determined bounds (wind-up guard).

### 15.4.3. Mathematical form

The overall control function

$$u(t) = K_{\mathsf{p}} e(t) + K_{\mathsf{i}} \int_0^t e(t')\,dt' + K_{\mathsf{d}} \frac{de(t)}{dt},$$

where $K_{\mathsf{p}}$, $K_{\mathsf{i}}$ and $K_{\mathsf{d}}$, all non-negative, denote the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted P, I, and D).

### 15.4.4. Selective use of control terms

Although a PID controller has three control terms, some applications need only one or two terms to provide appropriate control. This is achieved by setting the unused parameters to zero and is called a PI, PD, P or I controller in the absence of the other control actions. PI controllers are fairly common in applications where derivative action would be sensitive to measurement noise, but the integral term is often needed for the system to reach its target value.

### 15.4.5. Overview of tuning methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively time-consuming, particularly for systems with long loop times.

The choice of method will depend largely on whether or not the loop can be taken offline for tuning, and on the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

**15.4.5.1. Manual tuning** If the system must remain online, one tuning method is to first set $K_i$ and $K_d$ values to zero. Increase the $K_p$ until the output of the loop oscillates, then the $K_p$ should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase $K_i$ until any offset is corrected in sufficient time for the process. However, too much $K_i$ will cause instability. Finally, increase $K_d$, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much $K_d$ will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an overdamped closed-loop system is required, which will require a $K_p$ setting significantly less than half that of the $K_p$ setting that was causing oscillation.

**Effects of *increasing* a parameter independently**[22][23]

| Parameter | Rise time | Overshoot | Settling time | Steady-state error | Stability |
|---|---|---|---|---|---|
| $K_p$ | Decrease | Increase | Small change | Decrease | Degrade |
| $K_i$ | Decrease | Increase | Increase | Eliminate | Degrade |
| $K_d$ | Minor change | Decrease | Decrease | No effect in theory | Improve if $K_d$ small |

Figure 81: PID manual tuning summary.

**15.4.5.2. Ziegler–Nichols method** Another heuristic tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols in the 1940s. As in the method above, the $K_i$ and $K_d$ gains are first set to zero. The proportional gain is increased until it reaches the ultimate gain, $K_u$, at which the output of the loop starts to oscillate constantly. $K_u$ and the oscillation period $T_u$ are used to set the gains as follows:

**Ziegler–Nichols method**

| Control Type | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| ***P*** | $0.50K_u$ | $-$ | $-$ |
| ***PI*** | $0.45K_u$ | $0.54K_u/T_u$ | $-$ |
| ***PID*** | $0.60K_u$ | $1.2K_u/T_u$ | $3K_uT_u/40$ |

Figure 82: Ziegler-Nichols tuning.

These gains apply to the ideal, parallel form of the PID controller. When applied to the standard PID form, only the integral and derivative time parameters $T_i$ and $T_d$ are dependent on the oscillation period $T_u$.

### 15.4.6. DWSIM Implementation

The PID Controller model in DWSIM allows you to configure $K_p$, $K_i$ and $K_d$, set the action type (direct or reverse), set the Integral Wind-Up Guard value and enable/disable the controller. In Control Panel (Real-Time) mode, clicking over the PID Controller icon will display a small panel where you can enable or disable it, set it to AUTO or MANUAL operating mode and change the SP and/or MV values.

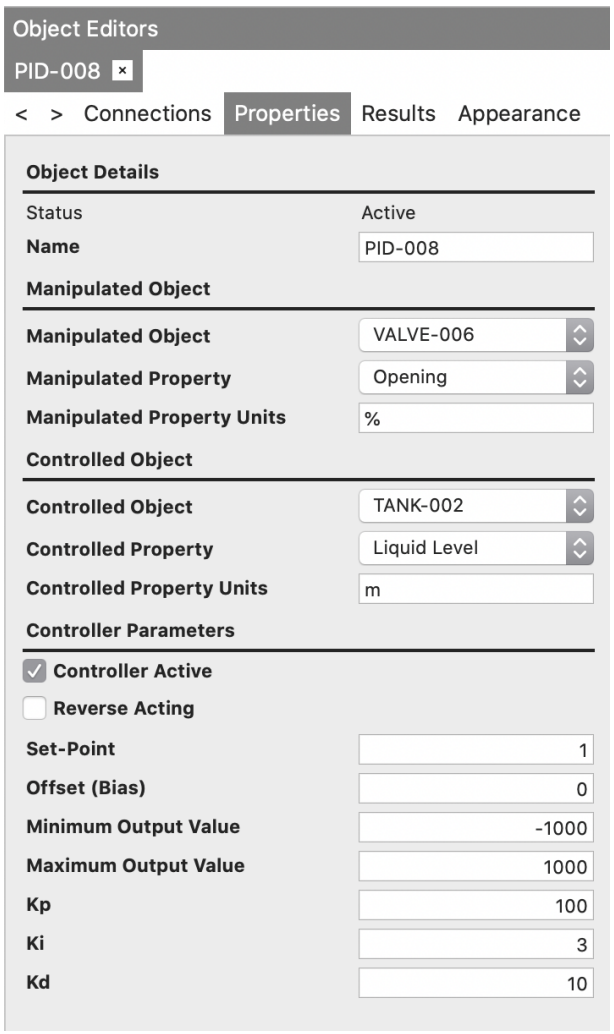



Figure 83: PID Controller Parameters.

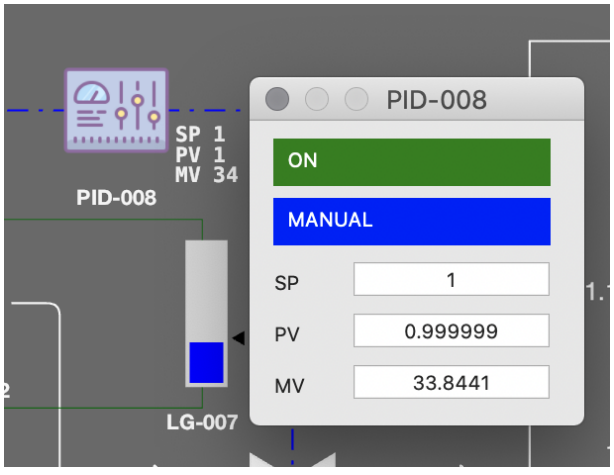



Figure 84: PID Editor in Control Panel (Real-Time) mode.

## 15.5. Python Controller

The **Python Controller** is a custom dynamic mode controller where the relationship between the Process Variable, Set-Point and Manipulated Variable is defined by an IronPython script.

For more information, check the *Dynamic Water Tank Level Control with Custom Controller* sample.

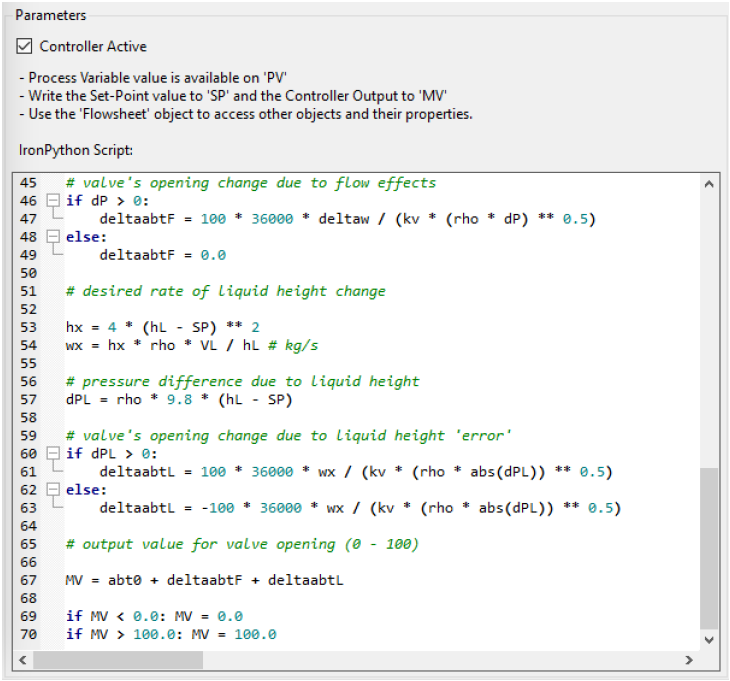



Figure 85: Sample Python Controller code.

## 15.6. Input Box

The Input Box can be associated to a property of an object on the flowsheet. Clicking on the box will display an input field from where the user can update the property's value directly.

## 15.7. Switch

The Switch can be associated to a property of an object on the flowsheet. Changing the value of the switch will update the associated property between two predefined values.

# 16. Dynamic Simulation Tools

## 16.1. Control Panel Mode

Control Panel mode makes the Flowsheet read-only and must be used when a Schedule is running in real-time mode. In such scenario, variables can have their values changed through input boxes. You can also change some PID Controller parameters by clicking on their icons on the flowsheet.

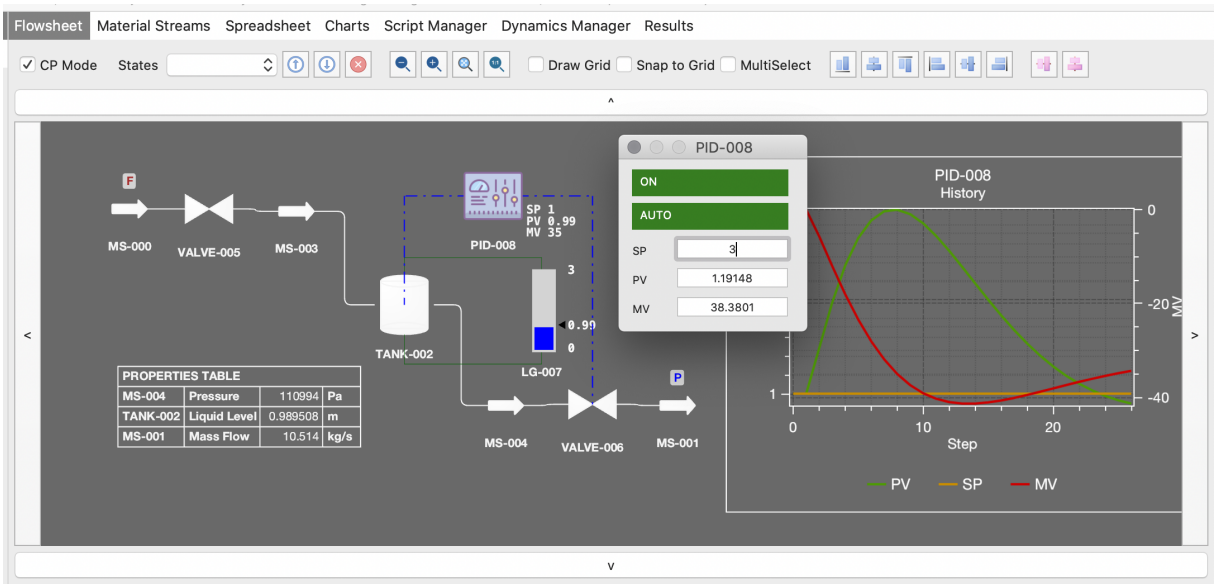



Figure 86: A Flowsheet running in Control Panel (real-time) dynamic mode.

## 16.2. PID Controller Tuning

The PID Controller Tuning tool can be used to tune the parameters (Kp, Ki, and Kd) of one or more PID Controllers. It uses a numerical optimizer (Simplex) to obtain the PID parameters which minimize the squared sum of errors (PV - SP) for an entire schedule run.

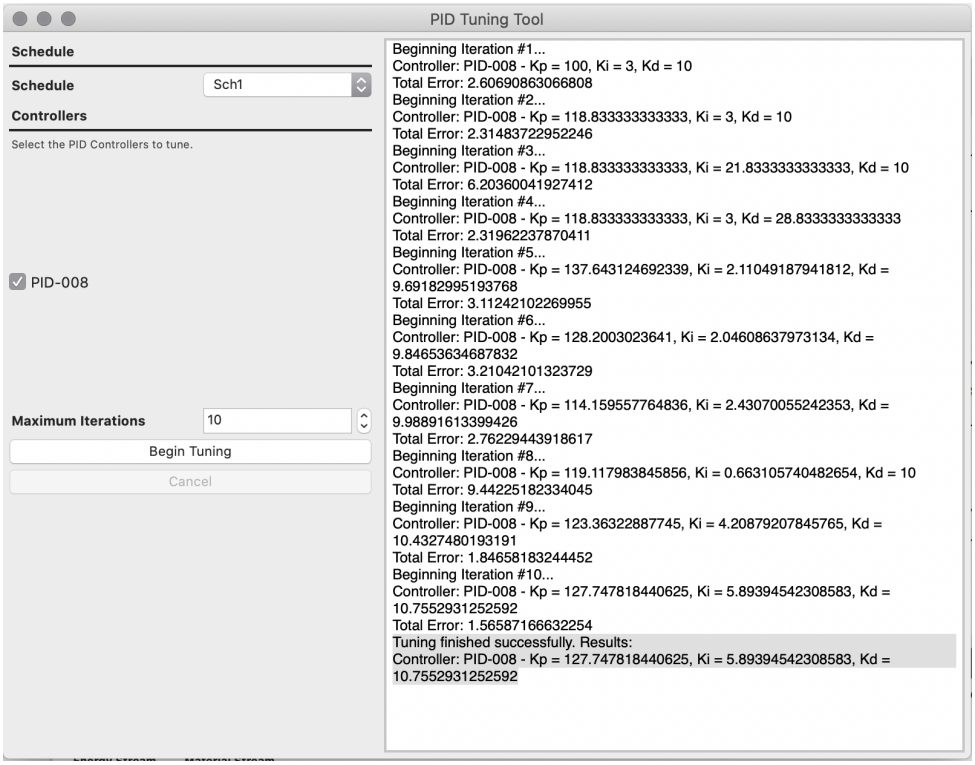



Figure 87: PID Controller Tuning Tool

# 17. Supported Unit Operations

As of v8.3, the following Unit Operations are supported in Dynamic Mode:

- Valve
- Gas-Liquid Separator
- Tank
- Compressor
- Expander
- Pump
- Heater
- Cooler
- Mixer
- Splitter
- Python Script Block
- CAPE-OPEN Block
- Heat Exchanger
- General Reactors (Conversion, Equilibrium and Gibbs)
- CSTR
- PFR
- Specification Logical Block

Unsupported Unit Operations will display an exclamation icon over their flowsheet objects when Dynamic Mode is enabled.


Figure 88: Unsupported Unit Operation in Dynamic Mode.

# Part VI.

# Technical Basis: Methods and Procedures

## 18. Introduction

The thermodynamic calculations are the basis of the simulations in DWSIM. It is important for a process simulator to cover a variety of systems, which can go from simple water handling processes to complex, more elaborated cases, such as simulations of processes in the petroleum/chemical industry.

DWSIM is able to model phase equilibria between solids, vapor and up to two liquid phases where possible. External CAPE-OPEN Property Packages may have different equilibrium capabilities.

The following sections describe the calculation methods used in DWSIM for the physical and chemical description of the elements of a simulation.

# 19. Thermodynamic Properties

## 19.1. Phase Equilibria Calculation

In a mixture which finds itself in a vapor-liquid equilibria state (VLE), the component fugacities are the same in all phases, that is [6]:

$$f_i^L = f_i^V \tag{19.1}$$

The fugacity of a component in a mixture depends on temperature, pressure and composition. in order to relate $f_i^V$ with temperature, pressure and molar fraction, we define the fugacity coefficient,

$$\phi_i = \frac{f_i^V}{y_i P}, \tag{19.2}$$

which can be calculated from PVT data, commonly obtained from an equation of state. For a mixture of ideal gases, $\phi_i = 1$.

The fugacity of the $i$ component in the liquid phase is related to the composition of that phase by the activity coefficient $\gamma_i$, which by itself is related to $x_i$ and standard-state fugacity $f_i^0$ by

$$\gamma_i = \frac{f_i^L}{x_i f_i^0}. \tag{19.3}$$

The standard state fugacity $f_i^0$ is the fugacity of the $i$-th component in the system temperature, i.e. mixture, and in an arbitrary pressure and composition. in DWSIM, the standard-state fugacity of each component is considered to be equal to pure liquid $i$ at the system temperature and pressure.

If an Equation of State is used to calculate equilibria, fugacity of the $i$-th component in the liquid phase is calculated by

$$\phi_i = \frac{f_i^L}{x_i P}, \tag{19.4}$$

with the fugacity coefficient $\phi_i$ calculated by the EOS, just like it is for the same component in the vapor phase.

The fugacity coefficient of the $i$-th component either in the liquid or in the vapor phase is obtained from the same Equation of State through the following expressions

$$RT \ln \phi_i^L = \int_{V^L}^{\infty} \left[ \left( \frac{\partial P}{\partial n_i} \right)_{T,V,n_j} - \frac{RT}{V} \right] dV - RT \ln Z^L, \tag{19.5}$$

$$RT \ln \phi_i^V = \int_{V^V}^{\infty} \left[ \left( \frac{\partial P}{\partial n_i} \right)_{T,V,n_j} - \frac{RT}{V} \right] dV - RT \ln Z^V, \tag{19.6}$$

where the compressibility factor $Z$ is given by

$$Z^L = \frac{PV^L}{RT} \tag{19.7}$$

$$Z^V = \frac{PV^V}{RT} \tag{19.8}$$

### 19.1.1. Fugacity Coefficient calculation models

***Peng-Robinson Equation of State*** The Peng-Robinson equation [7] is an cubic Equation of State (characteristic related to the exponent of the molar volume) which relates temperature, pressure and molar volume of a pure component or a mixture of components at equilibrium. The cubic equations are, in fact, the simplest equations capable of representing the behavior of liquid and vapor phases simultaneously. The Peng-Robinson EOS is written in the following form

$$P = \frac{RT}{(V-b)} - \frac{a(T)}{V(V+b)+b(V-b)} \tag{19.9}$$

where

$P$ pressure

$R$ ideal gas universal constant

$v$ molar volume

$b$ parameter related to hard-sphere volume

$a$ parameter related to intermolecular forces

For pure substances, the $a$ and $b$ parameters are given by:

$$a(T) = [1 + (0.37464 + 1.54226\omega - 0.26992\omega^2)(1 - T_r^{(1/2)})]^2 0.45724(R^2 T_c^2)/P_c \tag{19.10}$$

$$b = 0.07780(RT_c)/P_c \tag{19.11}$$

where

$\omega$ acentric factor

$T_c$ critical temperature

$P_c$ critical pressure

$T_r$ reduced temperature, $T/Tc$

For mixtures, equation 19.9 can be used, replacing $a$ and $b$ by mixture-representative values. $a$ and $b$ mixture values are normally given by the basic mixing rule,

$$a_m = \sum_i \sum_j x_i x_j \sqrt{(a_i a_j)}(1 - k_{ij}) \tag{19.12}$$

$$b_m = \sum_i x_i b_i \tag{19.13}$$

where

$x_{i,j}$ molar fraction of the $i$ or $j$ component in the phase (liquid or vapor)

$a_{i,j}$ $i$ or $j$ component $a$ constant

$b_{i,j}$ $i$ or $j$ component $b$ constant

$k_{ij}$ binary interaction parameter which characterizes the i-j pair

***The binary interaction parameters used by DWSIM are loaded from a databank (file) and can be modified in the Property Package configuration window.***

The fugacity coefficient obtained with the Peng-Robinson EOS in given by

$$\ln \frac{f_i}{x_i P} = \frac{b_i}{b_m}(Z-1) - \ln(Z-B) - \frac{A}{2\sqrt{2}B}\left(\frac{\sum_k x_k a_{ki}}{a_m} - \frac{b_i}{b_m}\right)\ln\left(\frac{Z+2,414B}{Z-0,414B}\right), \quad (19.14)$$

where $Z$ in the phase compressibility factor (liquid or vapor) and can be obtained from the equation 19.9,

$$Z^3 - (1-B)Z^2 + (A - 3B^2 - 2B)Z - (AB - B^2 - 2B) = 0, \quad (19.15)$$

$$A = \frac{a_m P}{R^2 T^2} \quad (19.16)$$

$$B = \frac{b_m P}{RT} \quad (19.17)$$

$$Z = \frac{PV}{RT} \quad (19.18)$$

***Peng-Robinson (1978) Equation of State*** The 1978 version [8] of the PR EOS introduces a modification to the $a$ term:

If $\omega_i \leq 0.491$:

$$c_i = 0.37464 + 1.5422\omega_i - 0.26992\omega_i^2 \quad (19.19)$$

Else:

$$c_i = 0.379642 + 1.48503\omega_i - 0.164423\omega_i^2 + 0.016666\omega_i^3 \quad (19.20)$$

$$a_i = [1 + c_i(1 - T_r^{(1/2)})]^2 0.45724(R^2 T_c^2)/P_c \quad (19.21)$$

***Soave-Redlich-Kwong Equation of State*** The Soave-Redlich-Kwong Equation [9] is also a cubic equation of state in volume,

$$P = \frac{RT}{(V-b)} - \frac{a(T)}{V(V+b)}, \tag{19.22}$$

The $a$ and $b$ parameters are given by:

$$a(T) = [1 + (0.48 + 1.574\omega - 0.176\omega^2)(1 - T_r^{(1/2)})]^2 0.42747(R^2 T_c^2)/P_c \tag{19.23}$$

$$b = 0.08664(RT_c)/P_c \tag{19.24}$$

The equations 19.12 and 19.13 are used to calculate mixture parameters. Fugacity is calculated by

$$\ln \frac{f_i}{x_i P} = \frac{b_i}{b_m}(Z-1) - \ln(Z-B) - \frac{A}{B}\left(\frac{\sum_k x_k a_{ki}}{a_m} - \frac{b_i}{b_m}\right) \ln\left(\frac{Z+B}{Z}\right) \tag{19.25}$$

The phase compressibility factor $Z$ is obtained from the equation 19.22,

$$Z^3 - Z^2 + (A - B - B^2)Z - AB = 0, \tag{19.26}$$

$$A = \frac{a_m P}{R^2 T^2} \tag{19.27}$$

$$B = \frac{b_m P}{RT} \tag{19.28}$$

$$Z = \frac{PV}{RT} \tag{19.29}$$

The equations 19.15 and 19.26, in low temperature and pressure conditions, can provide three roots for $Z$. In this case, if liquid properties are being calculated, the smallest root is used. If the phase is vapor, the largest root is used. The remaining root has no physical meaning; at high temperatures and pressures (conditions above the pseudocritical point), the equations 19.15 and 19.26 provides only one real root.

***Peng-Robinson with Volume Translation*** Volume translation solves the main problem with two-constant EOS's, poor liquid volumetric predictions. A simple correction term is applied to the EOS-calculated molar volume,

$$v = v^{EOS} - c, \tag{19.30}$$

where $v$ =corrected molar volume, $v^{EOS}$ =EOS-calculated volume, and $c$ =component-specific constant. The shift in volume is actually equivalent to adding a third constant to the EOS but is special because equilibrium conditions are unaltered.

It is also shown that multicomponent VLE is unaltered by introducing the volume-shift term *c* as a mole-fraction average,

$$v_L = v_L^{EOS} - \sum x_i c_i \tag{19.31}$$

Volume translation can be applied to any two-constant cubic equation, thereby eliminating the volumetric defficiency suffered by all two-constant equations [10].

#### *Peng-Robinson-Stryjek-Vera*

**PRSV1**

A modification to the attraction term in the Peng-Robinson equation of state published by Stryjek and Vera in 1986 (PRSV) significantly improved the model's accuracy by introducing an adjustable pure component parameter and by modifying the polynomial fit of the acentric factor.

The modification is:

$$\kappa = \kappa_0 + \kappa_1 \left(1 + T_r^{0.5}\right) (0.7 - T_r) \tag{19.32}$$

$$\kappa_0 = 0.378893 + 1.4897153\,\omega - 0.17131848\,\omega^2 + 0.0196554\,\omega^3 \tag{19.33}$$

where $\kappa_1$ is an adjustable pure component parameter. Stryjek and Vera published pure component parameters for many compounds of industrial interest in their original journal article.

**PRSV2**

A subsequent modification published in 1986 (PRSV2) [11] further improved the model's accuracy by introducing two additional pure component parameters to the previous attraction term modification.

The modification is:

$$\kappa = \kappa_0 + \left[\kappa_1 + \kappa_2 \left(\kappa_3 - T_r\right)\left(1 - T_r^0.5\right)\right]\left(1 + T_r^{0.5}\right)(0.7 - T_r) \tag{19.34}$$

$$\kappa_0 = 0.378893 + 1.4897153\,\omega - 0.17131848\,\omega^2 + 0.0196554\,\omega^3 \tag{19.35}$$

where $\kappa_1$, $\kappa_2$, and $\kappa_3$ are adjustable pure component parameters.

PRSV2 is particularly advantageous for VLE calculations. While PRSV1 does offer an advantage over the Peng-Robinson model for describing thermodynamic behavior, it is still not accurate enough, in general, for phase equilibrium calculations. The highly non-linear behavior of phase-equilibrium calculation methods tends to amplify what would otherwise be acceptably small errors. It is therefore recommended that PRSV2 be used for equilibrium calculations when applying these models to a design. However, once the equilibrium state has been determined, the phase specific thermodynamic values at equilibrium may be determined by one of several simpler models with a reasonable degree of accuracy.

### 19.1.2. Chao-Seader and Grayson-Streed models

Chao-Seader ([12]) and Grayson-Streed ([13]) are older, semi-empirical models. The Grayson-Streed correlation is an extension of the Chao-Seader method with special applicability to hydrogen. In DWSIM, only the equilibrium values produced by these correlations are used in the calculations. The Lee-Kesler method is used to determine the enthalpy and entropy of liquid and vapor phases.

***Chao Seader*** Use this method for heavy hydrocarbons, where the pressure is less than 10 342 kPa (1 500 psia) and the temperature is between the range -17.78 žC and 260 žC.

***Grayson Streed*** Recommended for simulating heavy hydrocarbon systems with a high hydrogen content.

### 19.1.3. Calculation models for the liquid phase activity coefficient

The activity coefficient $\gamma$ is a factor used in thermodynamics to account for deviations from ideal behaviour in a mixture of chemical substances. In an ideal mixture, the interactions between each pair of chemical species are the same (or more formally, the enthalpy of mixing is zero) and, as a result, properties of the mixtures can be expressed directly in terms of simple concentrations or partial pressures of the substances present. Deviations from ideality are accommodated by modifying the concentration by an activity coefficient. . The activity coefficient is defined as

$$\gamma_i = [\frac{\partial(nG^E/RT)}{\partial n_i}]_{P,T,n_{j\neq i}} \tag{19.36}$$

where $G^E$ represents the excess Gibbs energy of the liquid solution, which is a measure of how far the solution is from ideal behavior. For an ideal solution, $\gamma_i$ = 1. Expressions for $G^E/RT$ provide values for the activity coefficients.

***UNIQUAC and UNIFAC models*** The UNIQUAC equation considers $g \equiv G^E/RT$ formed by two additive parts, one combinatorial term $g^C$ to take into account the size of the molecules, and one residual term $g^R$, which take into account the interactions between molecules:

$$g \equiv g^C + g^R \tag{19.37}$$

The $g^C$ function contains only pure species parameters, while the $g^R$ function incorporates two binary parameters for each pair of molecules. For a multicomponent system,

$$g^C = \sum_i x_i \ln \phi_i/x_i + 5 \sum_i q_i x_i \ln \theta_i/\phi_i \tag{19.38}$$

and

$$g^R = -\sum_i q_i x_i \ln(\sum_j \theta_j \tau_j i) \tag{19.39}$$

where

$$\phi_i \equiv (x_i r_i)/(\sum_j x_j r_j) \tag{19.40}$$

and

$$\theta_i \equiv (x_i q_i)/(\sum_j x_j q_j) \tag{19.41}$$

The $i$ subscript indicates the species, and $j$ is an index that represents all the species, $i$ included. All sums are over all the species. Note that $\tau_{ij} \neq \tau_{ji}$. When $i = j$, $\tau_{ii} = \tau_{jj} = 1$. In these equations, $r_i$ (a relative molecular volume) and $q_i$ (a relative molecular surface area) are pure species parameters. The influence of temperature in $g$ enters by means of the $\tau_{ij}$ parameters, which are temperature-dependent:

$$\tau_{ij} = \exp(u_{ij} - u_{jj})/RT \tag{19.42}$$

This way, the UNIQUAC parameters are values of $(u_{ij} - u_{jj})$.

An expression for $\gamma_i$ is found through the application of the following relation:

$$\ln \gamma_i = \left[\partial(nG^E/RT)/\partial n_i\right]_{(P,T,n_{j\neq i})} \tag{19.43}$$

The result is represented by the following equations:

$$\ln \gamma_i = \ln \gamma_i^C + \ln \gamma_i^R \tag{19.44}$$

$$\ln \gamma_i^C = 1 - J_i + \ln J_i - 5q_i(1 - J_i/L_i + \ln J_i/L_i) \tag{19.45}$$

$$\ln \gamma_i^R = q_i(1 - \ln s_i - \sum_j \theta_j \tau_{ij}/s_j) \tag{19.46}$$

where

$$J_i = r_i/(\sum_j r_j x_j) \tag{19.47}$$

$$L = q_i/(\sum_j q_j x_j) \tag{19.48}$$

$$s_i = \sum_l \theta_l \tau_{li} \tag{19.49}$$

Again the $i$ subscript identify the species, $j$ and $l$ are indexes which represent all the species, including $i$. all sums are over all the species, and $\tau_{ij} = 1$ for $i = j$. The parameters values $(u_{ij} - u_{jj})$ are found by regression of binary VLE/LLE data.

The UNIFAC method for the estimation of activity coefficients depends on the concept of that a liquid mixture can be considered a solution of its own molecules. These structural units are called subgroups. The greatest advantage of this method is that a relatively small number of subgroups can be combined to form a very large number of molecules.

The activity coefficients do not only depend on the subgroup properties, but also on the interactions between these groups. Similar subgroups are related to a main group, like “CH2”, “OH”, “ACH” etc.; the identification of the main groups are only descriptive. All the subgroups that belongs to the same main group are considered identical with respect to the interaction between groups. Consequently, the parameters which characterize the interactions between the groups are identified by pairs of the main groups.

The UNIFAC method is based on the UNIQUAC equation, where the activity coefficients are given by the equation 19.43. When applied to a solution of groups, the equations 19.45 and 19.46 are written in the form:

$$\ln \gamma_i^C = 1 - J_i + \ln J_i - 5q_i(1 - J_i/L_i + \ln J_i/L_i) \tag{19.50}$$

$$\ln \gamma_i^R = q_i(1 - \sum_k (\theta_k \beta_{ik}/s_k) - e_{ki} ln \beta_{ik}/s_k) \tag{19.51}$$

The parameters $J_i$ e $L_i$ are still given by eqs. 19.61 and **??**. Furthermore, the following definitions apply:

$$r_i = \sum_k \nu_k^{(i)} R_k \tag{19.52}$$

$$q_i = \sum_k \nu_k^{(i)} Q_k \tag{19.53}$$

$$e_{ki} = (\nu_k^{(i)} Q_k)/q_i \tag{19.54}$$

$$\beta_{ik} = \sum_m e_{mk} \tau_{mk} \tag{19.55}$$

$$\theta_k = (\sum_i x_i q_i e_{ki})/(\sum_i x_j q_j) \tag{19.56}$$

$$s_k = \sum_m \theta_m \tau_{mk} \tag{19.57}$$

$$s_i = \sum_l \theta_l \tau_{li} \tag{19.58}$$

$$\tau_{mk} = \exp(-a_{mk})/T \tag{19.59}$$

The $i$ subscript identify the species, and $j$ is an index that goes through all the species. The $k$ subscript identify the subgroups, and $m$ is an index that goes through all the subgroups. The parameter $\nu_k^{(i)}$ is the number of the $k$ subgroup in a molecule of the $i$ species. The subgroup parameter values $R_k$ and $Q_k$ and the interaction parameters $-a_{mk}$ are obtained in the literature.

***Modified UNIFAC (Dortmund) model*** The UNIFAC model, despite being widely used in various applications, has some limitations which are, in some way, inherent to the model. Some of these limitations are:

1. UNIFAC is unable to distinguish between some types of isomers.
2. The $\gamma - \phi$ approach limits the use of UNIFAC for applications under the pressure range of 10-15 atm.
3. The temperature is limited within the range of approximately 275-425 K.

4. Non-condensable gases and supercritical components are not included.
5. Proximity effects are not taken into account.
6. The parameters of liquid-liquid equilibrium are different from those of vapor-liquid equilibrium.
7. Polymers are not included.
8. Electrolytes are not included.

Some of these limitations can be overcome. The insensitivity of some types of isomers can be eliminated through a careful choice of the groups used to represent the molecules. The fact that the parameters for the liquid-liquid equilibrium are different from those for the vapor-liquid equilibrium seems not to have a theoretical solution at this time. One solution is to use both data from both equiibria to determine the parameters as a modified UNIFAC model. The limitations on the pressure and temperature can be overcome if the UNIFAC model is used with equations of state, which carry with them the dependencies of pressure and temperature.

These limitations of the original UNIFAC model have led several authors to propose changes in both combinatorial and the residual parts. To modify the combinatorial part, the basis is the suggestion given by Kikic et al. (1980) in the sense that the Staverman-Guggenheim correction on the original term of Flory-Huggins is very small and can, in most cases, be neglected. As a result, this correction was empirically removed from the UNIFAC model. Among these modifications, the proposed by Gmehling and coworkers [Weidlich and Gmehling, 1986; Weidlich and Gmehling, 1987; Gmehling et al., 1993], known as the model UNIFAC-Dortmund, is one of the most promising. In this model, the combinatorial part of the original UNIFAC is replaced by:

$$\ln \gamma_i^C = 1 - J_i + \ln J_i - 5q_i(1 - J_i/L_i + \ln J_i/L_i) \tag{19.60}$$

$$J_i = r_i^{3/4}/(\sum_j r_j^{3/4} x_j) \tag{19.61}$$

where the remaining quantities is defined the same way as in the original UNIFAC. Thus, the correction in-Staverman Guggenheim is empirically taken from the template. It is important to note that the in the UNIFAC-Dortmund model, the quantities $R_k$ and $Q_k$ are no longer calculated on the volume and surface area of Van der Waals forces, as proposed by Bondi (1968), but are additional adjustable parameters of the model.

The residual part is still given by the solution for groups, just as in the original UNIFAC, but now the parameters of group interaction are considered temperature dependent, according to:

$$\tau_{mk} = \exp(-a_{mk}^{(0)} + a_{mk}^{(1)} T + a_{mk}^{(2)} T^2)/T \tag{19.62}$$

These parameters must be estimated from experimental phase equilibrium data. Gmehling et al. (1993) presented an array of parameters for 45 major groups, adjusted using data from the vapor-liquid equilibrium, excess enthalpies, activity coefficients at infinite dilution and liquid-liquid equilibrium. enthalpy and entropy of liquid and vapor.

***Modified UNIFAC (NIST) model*** This model [14] is similar to the Modified UNIFAC (Dortmund), with new modified UNIFAC parameters reported for 89 main groups and 984 group–group interactions using critically evaluated phase equilibrium data including vapor–liquid equilibrium (VLE), liquid–liquid equilibrium (LLE), solid–liquid equilibrium (SLE), excess enthalpy (HE), infinite dilution activity coefficient (AINF) and excess heat capacity (CPE) data. A new algorithmic framework for quality assessment of phase equilibrium data was applied for qualifying the consistency of data and screening out possible erroneous data. Substantial improvement over previous versions of UNIFAC is observed due to inclusion of experimental data from recent publications and proper weighting based on a quality assessment procedure. The systems requiring further verification of phase equilibrium data were identified where insufficient number of experimental data points is available or where existing data are conflicting.

***NRTL model*** Wilson (1964) presented a model relating $g^E$ to the molar fraction, based mainly on molecular considerations, using the concept of local composition. Basically, the concept of local composition states that the composition of the system in the vicinity of a given molecule is not equal to the overall composition of the system, because of intermolecular forces.

Wilson's equation provides a good representation of the Gibbs' excess free energy for a variety of mixtures, and is particularly useful in solutions of polar compounds or with a tendency to association in apolar solvents, where Van Laar's equation or Margules' one are not sufficient. Wilson's equation has the advantage of being easily extended to multicomponent solutions but has two disadvantages: first, the less important, is that the equations are not applicable to systems where the logarithms of activity coefficients, when plotted as a function of *x*, show a maximum or a minimum. However, these systems are not common. The second, a little more serious, is that the model of Wilson is not able to predict limited miscibility, that is, it is not useful for LLE calculations.

Renon and Prausnitz [15] developed the *NRTL* equation *(Non-Random, Two-Liquid)* based on the concept of local composition but, unlike Wilson's model, the NRTL model is applicable to systems of partial miscibility. The model equation is:

$$\ln \gamma_i = \frac{\sum_{j=1}^{n} \tau_{ji} x_j G_{ji}}{\sum_{k=1}^{n} x_k G_{ki}} + \sum_{j=1}^{n} \frac{x_j G_{ij}}{\sum_{k=1}^{n} x_k G_{kj}} \left( \tau_{ij} - \frac{\sum_{m=1}^{n} \tau_{mj} x_m G_{mj}}{\sum_{k=1}^{n} x_k G_{kj}} \right), \tag{19.63}$$

$$G_{ij} = exp(-\tau_{ij} \alpha_{ij}), \tag{19.64}$$

$$\tau_{ij} = a_{ij}/RT, \tag{19.65}$$

where

$\gamma_i$ Activity coefficient of component *i*

$x_i$ Molar fraction of component *i*

$a_{ij}$ Interaction parameter between *i-j* $(a_{ij} \neq a_{ji})$ (cal/mol)

$T$ Temperature (K)

$\alpha_{ij}$ non-randomness parameter for the *i-j* pair $(\alpha_{ij} = \alpha_{ji})$

The significance of $G_{ij}$ is similar to $\Lambda_{ij}$ from Wilson's equation, that is, they are characteristic energy parameters of the *ij* interaction. The parameter is related to the non-randomness of the mixture, i.e. that the components in the mixture are not randomly distributed but follow a pattern dictated by the local composition. When it is zero, the mixture is completely random, and the equation is reduced to the two-suffix Margules equation.

For ideal or moderately ideal systems, the NRTL model does not offer much advantage over Van Laar and three-suffix Margules, but for strongly non-ideal systems, this equation can provide a good representation of experimental data, although good quality data is necessary to estimate the three required parameters.

## 19.2. Enthalpy, Entropy and Heat Capacities

***$H^{id}$ values are calculated from the ideal gas heat capacity. For mixtures, a molar average is used. The value calculated by the EOS is for the phase, independently of the number of components present in the mixture.***

***Peng-Robinson, Soave-Redlich-Kwong*** For the cubic equations of state, enthalpy, entropy and heat capacities are calculated by the *departure functions*, which relates the phase properties in the conditions of the mixture with the same mixture property in the ideal gas state.This way, the following departure functions are defined [16],

$$\frac{H - H^{id}}{RT} = X; \; \frac{S - S^{id}}{R} = Y \tag{19.66}$$

values for $X$ and $Y$ are calculated by the PR and SRK EOS, according to the table 40:

Table 1: Enthalpy/Entropy calculation with an EOS

| | $\frac{H-H^{id}}{RT}$ | $\frac{S-S^{id}}{R}$ |
|---|---|---|
| PR | $Z - 1 - \frac{1}{2^{1,5}bRT}\left[a - T\frac{da}{dT}\right] \times \times \ln\left[\frac{V+2,414b}{V+0,414b}\right]$ | $\ln(Z-B) - \ln\frac{P}{P^0} - \frac{A}{2^{1,5}bRT}\left[\frac{T}{a}\frac{da}{dT}\right] \times \times \ln\left[\frac{V+2,414b}{V+0,414b}\right]$ |
| SRK | $Z - 1 - \frac{1}{bRT}\left[a - T\frac{da}{dT}\right] \times \times \ln\left[1 + \frac{b}{V}\right]$ | $\ln(Z-B) - \ln\frac{P}{P^0} - \frac{A}{B}\left[\frac{T}{a}\frac{da}{dT}\right] \times \times \ln\left[1 + \frac{B}{Z}\right]$ |

***In DWSIM, $P_o$= 1 atm.***

Heat capacities are obtained directly from the EOS, by using the following thermodynamic relations:

$$C_p - C_p^{id} = T \int_{\infty}^{V} \left( \frac{\partial^2 P}{\partial T^2} \right) dV - \frac{T(\partial P/\partial T)_V^2}{(\partial P/\partial V)_T} - R \tag{19.67}$$

$$C_p - C_v = -T \frac{\left(\frac{\partial P}{\partial T}\right)_V^2}{\left(\frac{\partial P}{\partial V}\right)_T} \tag{19.68}$$

***Lee-Kesler*** Enthalpies, entropies and heat capacities are calculated by the Lee-Kesler model [17] through the following equations:

$$\frac{H - H^{id}}{RT_c} = T_r \left( Z - 1 - \frac{b_2 + 2b_3/T_r + 3b_4/T_r^2}{T_r V_r} - \frac{c_2 - 3c_3/T_r^2}{2T_r V_r^2} + \frac{d_2}{5T_r V_r^2} + 3E \right) \tag{19.69}$$

$$\frac{S - S^{id}}{R} + \ln\left(\frac{P}{P_0}\right) = \ln Z - \frac{b_2 + b_3/T_r^2 + 2b_4/T_r^3}{V_r} - \frac{c_1 - 2c_3/T_r^3}{2V_r^2} + \frac{d_1}{5V_r^5} + 2E \tag{19.70}$$

$$\frac{C_v - C_v^{id}}{R} = \frac{2\left(b_3 + 3b_4/T_r\right)}{T_r^2 V_r} - \frac{3c_3}{T_r^3 V_r^2} - 6E \tag{19.71}$$

$$\frac{C_p - C_p^{id}}{R} = \frac{C_v - C_v^{id}}{R} - 1 - T_r \frac{\left(\frac{\partial P_r}{\partial T_r}\right)_{V_r}^2}{\left(\frac{\partial P_r}{\partial V_r}\right)_{T_r}} \tag{19.72}$$

$$E = \frac{c_4}{2T_r^3 \gamma} \left[ \beta + 1 - \left( \beta + 1 + \frac{\gamma}{V_r^2} \right) \exp\left( -\frac{\gamma}{V_r^2} \right) \right] \tag{19.73}$$

***An iterative method is required to calculate $V_r$. The user should always watch the values generated by DWSIM in order to detect any issues in the compressibility factors generated by the Lee-Kesler model.***

$$Z = \frac{P_r V_r}{T_r} = 1 + \frac{B}{V_r} + \frac{C}{V_r^2} + \frac{D}{V_r^5} + \frac{c_4}{T_r^3 V_r^2} \left( \beta + \frac{\gamma}{V_r^2} \right) \exp\left( -\frac{\gamma}{V_r^2} \right) \tag{19.74}$$

$$B = b_1 - b_2/T_r - b_3/T_r^2 - b_4/T_r^3 \tag{19.75}$$

$$C = c_1 - c_2/T_r + c_3/T_r^3 \tag{19.76}$$

$$D = d_1 + d_2/T_r \tag{19.77}$$

Each property must be calculated based in two fluids apart from the main one, one simple and other for reference. For example, for the compressibility factor,

$$Z = Z^{(0)} + \frac{\omega}{\omega^{(r)}}\left(Z^{(r)} - Z^{(0)}\right), \tag{19.78}$$

where the $(0)$ superscript refers to the simple fluid while the $(r)$ superscript refers to the reference fluid. This way, property calculation by the Lee-Kesler model should follow the sequence below (enthalpy calculation example):

1. $V_r$ and $Z^{(0)}$ are calculated for the simple fluid at the fluid $T_r$ and $P_r$. using the equation 19.69, and with the constants for the simple fluid, as shown in the table 40, $(H - H^0)/RT_c$ is calculated. This term is $\left[(H - H^0)/RT_c\right]^{(0)}$. in this calculation, $Z$ in the equation 19.69 is $Z^{(0)}$.

2. The step 1 is repeated, using the same $T_r$ and $P_r$, but using the constants for the reference fluid as shown in table 40. With these values, the equation 19.69 allows the calculation of $\left[(H - H^0)/RT_c\right]^{(r)}$. In this step, $Z$ in the equation 19.69 is $Z^{(r)}$.

3. Finally, one determines the residual enthalpy for the fluid of interest by

$$\begin{aligned}\left[(H - H^0)/RT_c\right] &= \left[(H - H^0)/RT_c\right]^{(0)} + \\ &\frac{\omega}{\omega^{(r)}}\left(\left[(H - H^0)/RT_c\right]^{(r)} - \left[(H - H^0)/RT_c\right]^{(0)}\right),\end{aligned} \tag{19.79}$$

where $\omega^{(r)} = 0,3978$.

Table 3: Constants for the Lee-Kesler model

| Constant | Simple Fluid | Reference Fluid |
|---|---|---|
| $b_1$ | 0.1181193 | 0.2026579 |
| $b_2$ | 0.265728 | 0.331511 |
| $b_3$ | 0.154790 | 0.027655 |
| $b_4$ | 0.030323 | 0.203488 |
| $c_1$ | 0.0236744 | 0.0313385 |
| $c_2$ | 0.0186984 | 0.0503618 |
| $c_3$ | 0.0 | 0.016901 |
| $c_4$ | 0.042724 | 0.041577 |
| $d_1 \times 10^4$ | 0155488 | 0.48736 |
| $d_2 \times 10^4$ | 0.623689 | 0.0740336 |
| $\beta$ | 0.65392 | 1.226 |
| $\gamma$ | 0.060167 | 0.03754 |

## 19.3. Speed of Sound

The speed of sound in a given phase is calculated by the following equations:

$$c = \sqrt{\frac{K}{\rho}}, \tag{19.80}$$

where:

$c$ Speed of sound (m/s)

$K$ Bulk Modulus (Pa)

$\rho$ Phase Density (kg/m³)

### 19.4. Joule-Thomson Coefficient

In thermodynamics, the Joule–Thomson effect (also known as the Joule–Kelvin effect, Kelvin–Joule effect, or Joule–Thomson expansion) describes the temperature change of a real gas or liquid when it is forced through a valve or porous plug while kept insulated so that no heat is exchanged with the environment. This procedure is called a throttling process or Joule–Thomson process. At room temperature, all gases except hydrogen, helium and neon cool upon expansion by the Joule–Thomson process. The rate of change of temperature with respect to pressure in a Joule–Thomson process is the Joule–Thomson coefficient.

The Joule-Thomson coefficient for a given phase is calculated by the following definition:

$$\mu = \left(\frac{\partial T}{\partial P}\right)_H, \tag{19.81}$$

The JT coefficient is calculated rigorously by the PR and SRK equations of state, while the Goldzberg correlation is used for all other models,

$$\mu = \frac{0.0048823 T_{pc}\left(18/T_{pr}^2 - 1\right)}{P_{pc} C_p \gamma}, \tag{19.82}$$

for gases, and

$$\mu = -\frac{1}{\rho C_p}, \tag{19.83}$$

for liquids.

## 20. Transport Properties

### 20.1. Density

***Liquid Phase*** Liquid phase density is calculated with the Rackett equation for non-EOS models when experimental data is not available [16],

$$V_s = \frac{RT_C}{P_C} Z_{RA}^{\left[1+(1-T_r)^{2/7}\right]}, \tag{20.1}$$

where:

$V_s$ Saturated molar volume (m³/mol)

$T_c$ Critical temperature (K)

$P_c$ Critical pressure (Pa)

$T_r$ Reduced temperature

$Z_{RA}$ Rackett constant of the component (or the mixture)

$R$ Ideal Gas constant (8,314 J/[mol.K])

***If $T \geqslant T_{cm}$, the Rackett method does not provide a value for $V_s$ and, in this case, DWSIM uses the EOS-generated compressibility factor to calculate the density of the liquid phase.***

For mixtures, the equation 20.1 becomes

$$V_s = R\left(\sum \frac{x_i T_{c_i}}{P_{c_i}}\right) Z_{RA}^{\left[1+(1-T_r)^{2/7}\right]}, \tag{20.2}$$

with $T_r = T/T_{cm}$, and

$$T_{c_m} = \sum\sum \phi_i \phi_j T_{c_{ij}}, \tag{20.3}$$

$$\phi_i = \frac{x_i V_{c_i}}{\sum x_i V_{c_i}}, \tag{20.4}$$

$$T_{c_{ij}} = \left[\frac{8\left(V_{c_i} V_{c_j}\right)^{1/2}}{\left(V_{c_i}^{1/3} + V_{c_j}^{1/3}\right)^3}\right]\left(T_{c_i} T_{c_j}\right)^{1/2}, \tag{20.5}$$

where:

$x_i$ Molar fraction

$V_{c_i}$ Critical volume (m³/mol)

If $Z_{RA}$ isn't available, it is calculated from the component acentric factor,

$$Z_{RA} = 0.2956 - 0.08775\omega, \tag{20.6}$$

If the component (or mixture) isn't saturated, a correction is applied in order to account for the effect of pressure in the volume,

$$V = V_s\left[1 - (0.0861488 + 0.0344483\omega)\ln\frac{\beta + P}{\beta + P_{vp}}\right], \tag{20.7}$$

with

$$\begin{aligned}\frac{\beta}{P} &= -1 - 9.070217\left(1 - T_r\right)^{1/3} + 62.45326\left(1 - T_r\right)^{2/3} - 135.1102\left(1 - T_r\right) + \\ &\quad + \exp\left(4.79594 + 0.250047\omega + 1.14188\omega^2\right)\left(1 - T_r\right)^{4/3},\end{aligned} \tag{20.8}$$

where:

$V$ Compressed liquid volume (m³/mol)

$P$ Pressure (Pa)

$P_{vp}$ Vapor pressure / Bubble point pressure (Pa)

Finally, density is calculated from the molar volume by the following relation:

$$\rho = \frac{MM}{1000V}, \tag{20.9}$$

where:

$\rho$ Density (kg/m³)

$V$ Specific volume of the fluid (m³/mol)

$MM$ Liquid phase molecular volume (kg/kmol)

***For the* Ideal Gas *Property Package, the compressibility factor is considered to be equal to 1.***

***Vapor Phase*** Vapor phase density is calculated from the compressiblity factor generated by the EOS model, according with the following equation:

$$\rho = \frac{MM\,P}{1000ZRT}, \tag{20.10}$$

where:

$\rho$ Density (kg/m³)

$MM$ Molecular weight of the vapor phase (kg/kmol)

$P$ Pressure (Pa)

$Z$ Vapor phase compressibility factor

$R$ Ideal Gas constant (8,314 J/[mol.K])

$T$ Temperature (K)

For ideal gases, the same equation is used, with Z = 1.

***Mixture*** If there are two phases at system temperature and pressure, the density of the mixture is calculated by the following expression:

$$\rho_m = f_l \rho_l + f_v \rho_v, \tag{20.11}$$

where:

$\rho_{m,l,v}$ Density of the mixture / liquid phase / vapor phase (kg/m³)

$f_{l,v}$ Volumetric fraction of the liquid phase / vapor phase (kg/kmol)

## 20.2. Viscosity

***Liquid Phase*** When experimental data is not available, liquid phase viscosity is calculated from

$$\eta_L = \exp\left(\sum_i x_i \ln \eta_i\right), \tag{20.12}$$

where $\eta_i$ is the viscosity of each component in the phase, which depends on the temperature and is calculated from experimental data. Dependence of viscosity with the temperature is described in the equation

$$\eta = \exp\left(A + B/T + C\ln T + DT^E\right), \tag{20.13}$$

where *A, B, C, D* and *E* are experimental coefficients (or generated by DWSIM in the case of pseudocomponents or hypotheticals).

***Vapor Phase*** Vapor phase viscosity is calculated in two steps. First, when experimental data is not available, the temperature dependence is given by the *Lucas* equation [16],

$$\eta\xi = \left[0, .807T_r^{0,618} - 0.357\exp(-0.449T_r) + 0.34\exp(-4.058T_r) + 0.018\right] \tag{20.14}$$

$$\xi = 0,176\left(\frac{T_c}{MM^3P_c^4}\right)^{1/6}, \tag{20.15}$$

where

$\eta$ Viscosity ($\mu P$)

$T_c, P_c$ Component (or mixture) critical properties

$T_r$ Reduced temperature, $T/T_c$

$MM$ Molecular weight (kg/kmol)

In the second step, the experimental or calculated viscosity with the Lucas method is corrected to take into account the effect of pressure, by the *Jossi-Stiel-Thodos* method [16],

$$\left[(\eta-\eta_0)\left(\frac{T_c}{MM^3P_c^4}\right)^{1/6}+1\right]^{1/4} = 1.023+0.23364\rho_r+ \\ + \quad 0.58533\rho_r^2-0.40758\rho_r^3+0.093324\rho_r^4, \tag{20.16}$$

where

| | |
|---|---|
| $\eta, \eta_0$ | Corrected viscosity / Lucas method calculated viscosity ($\mu P$) |
| $T_c, P_c$ | Component critical properties |
| $\rho_r$ | Reduced density, $\rho/\rho_c = V/V_c$ |
| $MM$ | Molecular weight (kg/kmol) |

If the vapor phase contains more than a component, the viscosity is calculated by the same procedure, but with the required properties calculated by a molar average.

## 20.3. Surface Tension

When experimental data is not available, the liquid phase surface tension is calculated by doing a molar average of the individual component tensions, which are calculated with the *Brock-Bird* equation [16],

$$\frac{\sigma}{P_c^{2/3}T_c^{1/3}} = (0.132\alpha_c-0.279)\,(1-T_r)^{11/9} \tag{20.17}$$

$$\alpha_c = 0.9076\left[1+\frac{T_{br}\ln(P_c/1.01325)}{1-T_{br}}\right], \tag{20.18}$$

where

| | |
|---|---|
| $\sigma$ | Surface tension (N/m) |
| $T_c$ | Critical temperature (K) |
| $P_c$ | Critical pressure (Pa) |
| $T_{br}$ | Reduced normal boiling point, $T_b/T_c$ |

## 20.4. Isothermal Compressibility

Isothermal compressiblity of a given phase is calculated following the thermodynamic definition:

$$\beta = -\frac{1}{V}\frac{\partial V}{\partial P} \tag{20.19}$$

The above expression is calculated rigorously by the PR and SRK equations of state. For the other models, a numerical derivative approximation is used.

## 20.5. Bulk Modulus

The Bulk Modulus of a phase is defined as the inverse of the isothermal compressibility:

$$K = \frac{1}{\beta} \tag{20.20}$$

# 21. Thermal Properties

## 21.1. Thermal Conductivity

***Liquid Phase*** When experimental data is not available, the contribution of each component for the thermal conductivity of the liquid phase is calculated by the *Latini* method [16],

$$\lambda_i = \frac{A(1-T_r)^{0.38}}{T_r^{1/6}} \tag{21.1}$$

$$A = \frac{A^* T_b^{0.38}}{MM^{\beta} T_c^{\gamma}}, \tag{21.2}$$

where $A^*$, $\alpha$, $\beta$ and $\gamma$ depend on the nature of the liquid (Saturated Hydrocarbon, Aromatic, Water, etc). The liquid phase thermal conductivity is calculated from the individual values by the *Li* method [16],

$$\lambda_L = \sum\sum \phi_i \phi_j \lambda_{ij} \tag{21.3}$$

$$\lambda_{ij} = 2(\lambda_i^{-1} + \lambda_j^{-1})^{-1} \tag{21.4}$$

$$\phi_i = \frac{x_i V_{c_i}}{\sum x_i V_{c_i}}, \tag{21.5}$$

where

$\lambda_L$ liquid phase thermal conductivity (W/[m.K])

***Vapor Phase*** When experimental data is not available, vapor phase thermal conductivity is calculated by the *Ely and Hanley* method [16],

$$\lambda_V = \lambda^* + \frac{1000\eta^*}{MM} 1.32 \left( C_v - \frac{3R}{2} \right), \tag{21.6}$$

where

$\lambda_V$ vapor phase thermal conductivity (W/[m.K])

$C_v$ constant volume heat capacity (J/[mol.K])

$\lambda^*$and $\eta^*$ are defined by:

$$\lambda^* = \lambda_0 H \tag{21.7}$$

$$H = \left( \frac{16.04E-3}{MM/1000} \right)^{1/2} f^{1/2}/h^{2/3} \tag{21.8}$$

$$\lambda_0 = 1944\eta_0 \tag{21.9}$$

$$f = \frac{T_0\theta}{190.4} \tag{21.10}$$

$$h = \frac{V_c}{99.2}\phi \tag{21.11}$$

$$\theta = 1 + (\omega - 0.011)(0.56553 - 0.86276\ln T^+ - 0.69852/T^+ \tag{21.12}$$

$$\phi = \left[1 + (\omega - 0.011)(0.38650 - 1.1617\ln T^+)\right] 0.288/Z_c \tag{21.13}$$

If $T_r \leqslant 2$, $T^+ = T_r$. If $T_r > 2$, $T^+ = 2$.

$$h = \frac{V_c}{99.2}\phi \tag{21.14}$$

$$\eta^* = \eta_0 H \frac{MM/1000}{16.04E - 3} \tag{21.15}$$

$$\eta_0 = 10^{-7}\sum_{n=1}^{9} C_n T_0^{(n-4)/3} \tag{21.16}$$

$$T_0 = T/f \tag{21.17}$$

# 22. Aqueous Solution Properties

## 22.1. Mean salt activity coefficient

The mean salt activity coefficient is calculated from the activity coefficients of the ions,

$$\ln\left(\gamma_{\pm}^{*,m}\right) = \frac{\nu_+}{\nu}\ln\left(\gamma_{+}^{*,m}\right) + \frac{\nu_-}{\nu}\ln\left(\gamma_{-}^{*,m}\right) \tag{22.1}$$

In this equation $\nu_+$ and $v_-$ are the stoichiometric coefficients of the cations and anions of the salt, while $\nu$ stands for the sum of these stoichiometric coefficients. With the mean salt activity coefficient the real behavior of a salt can be calculated and it can, e.g. be used for the calculation of electromotoric forces EMF.

## 22.2. Osmotic coefficient

The osmotic coefficient represents the reality of the solvent in electrolyte systems. It is calculated by the logarithmic ratio of the activity and mole fraction of the solvent:

$$\Phi = -\frac{\ln a_i}{M_s \sum_{ion} m_{ion}} \tag{22.2}$$

## 22.3. Freezing point depression

The Schröder and van Laar equation is used:

$$\frac{\ln a_i}{(1-(T_{m,i}/T))} = \frac{\triangle_m h_i}{RT_{m,i}} \tag{22.3}$$

On the right hand side of the equation a constant factor is achieved, while on the left hand side the activity depends on temperature and composition. For a given composition the freezing point of the system can be calculated iteratively by varying the system temperature. The best way to do this is by starting at the freezing point of the pure solvent. This equation also allows calculating the freezing point of mixed solvent electrolyte systems.

# 23. Specialized Models / Property Packages

## 23.1. IAPWS-IF97 Steam Tables

Water is used as cooling medium or heat transfer fluid and it plays an important role for air-condition. For conservation or for reaching desired properties, water must be removed from substances (drying). In other cases water must be added (humidification). Also, many chemical reactions take place in hydrous solutions. That's why a good deal of work has been spent on the investigation and measurement of water properties over the years. Thermodynamic, transport and other properties of water are known better than of any other substance. Accurate data are especially needed for the design of equipment in steam power plants (boilers, turbines, condensers). In this field it's also important that all parties involved, e.g., companies bidding for equipment in a new steam power plant, base their calculations on the same property data values because small differences may produce appreciable differences.

A standard for the thermodynamic properties of water over a wide range of temperature and pressure was developed in the 1960's, the 1967 IFC Formulation for Industrial Use (IFC-67). Since 1967 IFC-67 has been used for "official" calculations such as performance guarantee calculations of power cycles.

In 1997, IFC-67 has been replaced by a new formulation, the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam or **IAPWS-IF97** for short. IAPWS-IF97 was developed in an international research project coordinated by the International Association for the Properties of Water and Steam (IAPWS). The formulation is described in a paper by W. Wagner et al., "The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam," ASME J. Eng. Gas Turbines and Power, Vol. 122 (2000), pp. 150-182 and several steam table books, among others ASME Steam Tables and Properties of Water and Steam by W. Wagner, Springer 1998.

The IAPWS-IF97 divides the thermodynamic surface into five regions:

- Region 1 for the liquid state from low to high pressures,
- Region 2 for the vapor and ideal gas state,
- Region 3 for the thermodynamic state around the critical point,
- Region 4 for the saturation curve (vapor-liquid equilibrium),
- Region 5 for high temperatures above 1073.15 K (800 řC) and pressures up to 10 MPa (100 bar).

For regions 1, 2, 3 and 5 the authors of IAPWS-IF97 have developed fundamental equations of very high accuracy. Regions 1, 2 and 5 are covered by fundamental equations for the Gibbs free energy g(T,p), region 3 by a fundamental equation for the Helmholtz free energy f(T,v). All thermodynamic properties can then be calculated from these fundamental equations by using the appropriate thermodynamic relations. For region 4 a saturation-pressure equation has been developed.

In chemical engineering applications mainly regions 1, 2, 4, and to some extent also region 3 are of interest. The range of validity of these regions, the equations for calculating the thermodynamic properties, and references are summarized in Attachment 1. The equations of the high-temperature region 5 should be looked up in the references. For regions 1 and 2 the thermodynamic properties are given as a function of temperature and pressure, for region 3 as a function of temperature and density. For other independent variables an iterative calculation is usually required. So-called backward equations are provided in IAPWS-IF97 which allow direct calculation of properties as a function of some other sets of variables (see references).

Accuracy of the equations and consistency along the region boundaries are more than sufficient for engineering applications.

More information about the IAPWS-IF97 Steam Tables formulation can be found at `http://www.thermo.ruhr-uni-bochum.de/en/prof-w-wagner/software/iapws-if97.html?id=172`.

## 23.2. IAPWS-08 Seawater

The IAPWS-08 Seawater Property Package is based on the **Seawater-Ice-Air (SIA)** library. The Seawater-Ice-Air (SIA) library contains the **TEOS-10** subroutines for evaluating a wide range of thermodynamic properties of pure water (using IAPWS-95), seawater (using IAPWS-08 for the saline part), ice Ih (using IAPWS-06) and for moist air (using Feistel et al. (2010a), IAPWS (2010)).

**TEOS-10** is based on a Gibbs function formulation from which all thermodynamic properties of seawater (density, enthalpy, entropy sound speed, etc.) can be derived in a thermodynamically consistent manner. TEOS-10 was adopted by the Intergovernmental Oceanographic Commission at its 25th Assembly in June 2009 to replace EOS-80 as the official description of seawater and ice properties in marine science.

A significant change compared with past practice is that TEOS-10 uses Absolute Salinity SA (mass fraction of salt in seawater) as opposed to Practical Salinity SP (which is essentially a measure of the conductivity of seawater) to describe the salt content of seawater. Ocean salinities now have units of g/kg.

Absolute Salinity (g/kg) is an SI unit of concentration. The thermodynamic properties of seawater, such as density and enthalpy, are now correctly expressed as functions of Absolute Salinity rather than being functions of the conductivity of seawater. Spatial variations of the composition of seawater mean that Absolute Salinity is not simply proportional to Practical Salinity; TEOS-10 contains procedures to correct for these effects.

More information about the SIA library can be found at `http://www.teos-10.org/software.htm`.

## 23.3. Black-Oil

When fluids flow from a petroleum reservoir to the surface, pressure and temperature decrease. This affects the gas/liquid equilibrium and the properties of the gas and liquid phases. The black-oil model enables estimation of these, from a minimum of input data.

The black-oil model employs 2 pseudo components:

1. Oil which is usually defined as the produced oil, at stock tank conditions.
2. Gas which then is defined as the produced gas at atmospheric standard conditions.

The basic modeling assumption is that the gas may dissolve in the liquid hydrocarbon phase, but no oil will dissolve in the gaseous phase. This implies that the composition of the gaseous phase is assumed the same at all pressure and temperatures.

The black-oil model assumption is reasonable for mixtures of heavy and light components, like many reservoir oils. The assumption gets worse for mixtures containing much of intermediate components (propane, butane), and is directly misleading for mixtures of light and intermediate components typically found in condensate reservoirs.

In DWSIM, a set of models calculates properties for a black oil fluid so it can be used in a process simulation. Black-oil fluids are defined in DWSIM through a minimum set of properties:

- Oil specific gravity (SGo) at standard conditions
- Gas specific gravity (SGg) at standard conditions
- Gas-to-oil ratio (GOR) at standard conditions
- Basic Sediments and Water (%)

Black oil fluids are defined and created through the **Compound Creator** tool. If multiple black-oil fluids are added to a simulation, a single fluid is calculated (based on averaged black-oil properties) and used to calculate stream equilibrium conditions and phase properties.

The Black-Oil Property Package is a simplified package for quick process calculations involving the black-oil fluids described above. All properties required by the unit operations are calculated based on the set of four basic properties (SGo, SGg, GOR and BSW), so the results of the calculations cannot be considered precise in any way. They can exhibit errors of several orders of magnitude when compared to real-world data.

For more accurate petroleum fluid simulations, use the petroleum characterization tools available in DWSIM together with an Equation of State model like Peng-Robinson or Soave-Redlich-Kwong.

### 23.4. CoolProp

CoolProp [18] is a C++ library that implements pure and pseudo-pure fluid equations of state and transport properties for 114 components.

The CoolProp library currently provides thermophysical data for 114 pure and pseudo-pure working fluids. The literature sources for the thermodynamic and transport properties of each fluid are summarized in a table in the Supporting Information available in the above reference.

For the CoolProp Property Package, DWSIM implements simple mixing rules based on mass fraction averages in order to calculate mixture enthalpy, entropy, heat capacities, density (and compressibility factor as a consequence). For equilibrium calculations, DWSIM requires values of fugacity coefficients at system's temperature and pressure. In the CoolProp Property Package, the vapor and liquid phases are considered to be ideal.

More information about CoolProp can be found at `http://www.coolprop.org`.

## 24. Reactions

DWSIM includes support for chemical reactions through the Chemical Reactions Manager. Three types of reactions are available to the user:

**Conversion,** where you must specify the conversion (%) of the limiting reagent as a function of temperature

**Equilibrium,** where you must specify the equilibrium constant (K) as a function of temperature, a constant value or calculated from the Gibbs free energy of reaction ($\Delta G/R$). The orders of reaction of the components are obtained from the stoichiometric coefficients.

**Kinetic,** where you should specify the frequency factor (A) and activation energy (E) for the direct reaction (optionally for the reverse reaction), including the orders of reaction (direct and inverse) of each component.

For each chemical reaction is necessary to specify the stoichiometric coefficients of the compounds and a base compound, which must be a reactant. This base component is used as reference for calculating the heat of reaction.

### 24.1. Conversion Reaction

In the conversion reaction it is assumed that the user has information regarding the conversion of one of the reactants as a function of temperature. By knowing the conversion and the stoichiometric coefficients,

the quantities of the other components in the reaction can be calculated.

Considering the following chemical reaction:

$$aA + bB \rightarrow cC, \tag{24.1}$$

where *a*, *b* and *c* are the stoichiometric coefficients of reactants and product, respectively. *A* is the limiting reactant and *B* is in excess. The amount of each component at the end of the reaction can then be calculated from the following stoichiometric relationships:

$$N_A = N_{A_0} - N_{A_0} X_A \tag{24.2}$$

$$N_B = N_{B_0} - \frac{b}{a} N_{A_0} X_A \tag{24.3}$$

$$N_C = N_{C_0} + \frac{c}{a} (N_{A_0} X_A) \tag{24.4}$$

where $N_{A,B,C}$ are the molar amounts of the components at the end of the reaction, $N_{A_0,B_0,C_0}$ are the molar amount of the components at the start of the reaction and $X_A$ is the conversion of the base-reactant *A*.

## 24.2. Equilibrium Reaction

In the equilibrium reaction, the quantity of each component at the equilibrium is related to equilibrium constant by the following relationship:

$$K = \prod_{j=1}^{n} (q_j)^{\nu_j}, \tag{24.5}$$

where *K* is the equilibrium constant, *q* is the basis of components (partial pressure in the vapor phase or activity in the liquid phase) $\nu$ is the stoichiometric coefficient of component *j* and *n* is the number of components in the reaction.

The equilibrium constant can be obtained by three different means. One is to consider it a constant, another is considering it as a function of temperature, and finally calculate it automatically from the Gibbs free energy at the temperature of the reaction. The first two methods require user input.

### 24.2.1. Solution method

For each reaction that is occurring in parallel in the system, we can define $\xi$ as the *reaction extent*, so that the molar amount of each component in the equilibrium is obtained by the following relationship:

$$N_j = N_{j_0} + \sum_i \nu_{ij} \xi_i, \tag{24.6}$$

where$\xi_i$ is the coordinate of the reaction *i* and $\nu_{ij}$ is the stoichiometric coefficient of the *j* component at

reaction *i*. Defining the molar fraction of the component *i* as $x_j = n_j/n_t$, where $n_t$ is the total number of mols, including inerts, whe have the following expression for each reaction *i*:

$$f_i(\xi) = \sum_i \ln(x_i) - \ln(K_i) = 0, \tag{24.7}$$

where the system of equations F can be easily solved by Newton-Raphson's method [5].

### 24.3. Kinetic Reaction

The kinetic reaction is defined by the parameters of the equation of Arrhenius (frequency factor and activation energy) for both the direct order and for the reverse order. Suppose we have the following kinetic reaction:

$$aA + bB \rightarrow cC + dD \tag{24.8}$$

The reaction rate for the *A* component can be defined as

$$r_A = k[A][B] - k'[C][D] \tag{24.9}$$

where

$$k = A\exp\left(-E/RT\right) \tag{24.10}$$

$$k' = A'\exp\left(-E'/RT\right) \tag{24.11}$$

The kinetic reactions are used in Plug-Flow Reactors (PFRs) and in Continuous-Stirred Tank Reactors (CSTRs). In them, the relationship between molar concentration and the rate of reaction is given by

$$F_A = F_{A_0} + \int^V r_A dV, \tag{24.12}$$

where $F_A$ is the molar flow of the *A* component and *V* is the reactor volume.

## 25. Property Estimation Methods

### 25.1. Petroleum Fractions

#### 25.1.1. Molecular weight

**Riazi and Al Sahhaf method [19]**

$$MM = \left[\frac{1}{0.01964}(6.97996 - \ln(1080 - T_b)\right]^{3/2}, \tag{25.1}$$

where

$MM$ Molecular weight (kg/kmol)

$T_b$ Boiling point at 1 atm (K)

If the specific gravity ($SG$) is available, the molecular weight is calculated by

$$\begin{aligned} MM &= 42.965[\exp(2.097 \times 10^{-4} T_b - 7.78712 SG + \\ &\quad +2.08476 \times 10^{-3} T_b SG)] T_b^{1.26007} SG^{4.98308} \end{aligned} \tag{25.2}$$

**Winn [20]**

$$MM = 0.00005805 PEMe^{2.3776} / d15^{0.9371}, \tag{25.3}$$

where

$PEMe$ Mean Boiling Point (K)

$d15$ Specific Gravity @ 60 řF

**Riazi[20]**

$$\begin{aligned} MM &= 42.965 \exp(0.0002097 PEMe - 7.78 d15 + 0.00208476 \times PEMe \times d15) \times \\ &\quad \times PEMe^{1.26007} d15^{4.98308} \end{aligned} \tag{25.4}$$

**Lee-Kesler[20]**

$$t_1 = -12272.6 + 9486.4 d15 + (8.3741 - 5.9917 d15) PEMe \tag{25.5}$$

$$\begin{aligned} t_2 &= (1 - 0.77084 d15 - 0.02058 d15^2) \times \\ &\quad \times (0.7465 - 222.466/PEMe) \times 10^7 / PEMe \end{aligned} \tag{25.6}$$

$$\begin{aligned} t_3 &= (1 - 0.80882 d15 - 0.02226 d15^2) \times \\ &\quad \times (0.3228 - 17.335/PEMe) \times 10^{12} / PEMe^3 \end{aligned} \tag{25.7}$$

$$MM = t_1 + t_2 + t_3 \tag{25.8}$$

**Farah**

$$MM = \exp(6.8117 + 1.3372 A - 3.6283 B) \tag{25.9}$$

$$MM = \exp(4.0397 + 0.1362 A - 0.3406 B - 0.9988 d15 + 0.0039 PEMe), \tag{25.10}$$

where

$A, B$ *Walther-ASTM* equation parameters for viscosity calculation

### 25.1.2. Specific Gravity

**Riazi e Al Sahhaf [19]**

$$SG = 1.07 - \exp(3.56073 - 2.93886MM^{0.1}), \tag{25.11}$$

where

$SG$ Specific Gravity

$MM$ Molecular weight (kg/kmol)

### 25.1.3. Critical Properties

**Lee-Kesler [19]**

$$T_c = 189.8 + 450.6SG + (0.4244 + 0.1174SG)T_b + (0.1441 - 1.0069SG)10^5/T_b \tag{25.12}$$

$$\begin{aligned} \ln P_c &= 5.689 - 0.0566/SG - (0.43639 + 4.1216/SG + 0.21343/SG^2) \times \\ &\times 10^{-3}T_b + (0.47579 + 1.182/SG + 0.15302/SG^2) \times 10^{-6} \times T_b^2 - \\ &-(2.4505 + 9.9099/SG^2) \times 10^{-10} \times T_b^3, \end{aligned} \tag{25.13}$$

where

$T_b$ NBP (K)

$T_c$ Critical temperature (K)

$P_c$ Critical pressure (bar)

**Farah**

$$T_c = 731.968 + 291.952A - 704.998B \tag{25.14}$$

$$T_c = 104.0061 + 38.75A - 41.6097B + 0.7831PEMe \tag{25.15}$$

$$T_c = 196.793 + 90.205A - 221.051B + 309.534d15 + 0.524PEMe \tag{25.16}$$

$$P_c = \exp(20.0056 - 9.8758\ln(A) + 12.2326\ln(B)) \tag{25.17}$$

$$P_c = \exp(11.2037 - 0.5484A + 1.9242B + 510.1272/PEMe) \tag{25.18}$$

$$P_c = \exp(28.7605 + 0.7158\ln(A) - 0.2796\ln(B) + 2.3129\ln(d15) - 2.4027\ln(PEMe)) \tag{25.19}$$

**Riazi-Daubert[20]**

$$\begin{aligned} T_c &= 9.5233\exp(-0.0009314PEMe - 0.544442d15 + 0.00064791 \times PEMe \times d15) \times \\ &\times PEMe^{0.81067}d15^{0.53691} \end{aligned} \tag{25.20}$$

$$P_c = 31958000000\exp(-0.008505PEMe - 4.8014d15 + 0.005749 \times PEMe \times d15) \times \times PEMe^{-0.4844}d15^{4.0846} \quad (25.21)$$

**Riazi[20]**

$$T_c = 35.9413\exp(-0.00069PEMe - 1.4442d15 + 0.000491 \times PEMe \times d15) \times \times PEMe^{0.7293}d15^{1.2771} \quad (25.22)$$

### 25.1.4. Acentric Factor

**Lee-Kesler method [19]**

$$\omega = \frac{-\ln\frac{P_c}{1.10325} - 5.92714 + 6.09648/T_{br} + 1.28862\ln T_{br} - 0.169347T_{br}^6}{15.2518 - 15.6875/T_{br} - 13.472\ln T_{br} + 0.43577T_{br}^6} \quad (25.23)$$

**Korsten[20]**

$$\omega = 0.5899 \times ((PEMV/T_c)^{1.3})/(1 - (PEMV/T_c)^{1.3}) \times \log(P_c/101325) - 1 \quad (25.24)$$

### 25.1.5. Vapor Pressure

**Lee-Kesler method[19]**

$$\ln P_r^{pv} = 5.92714 - 6.09648/T_{br} - 1.28862\ln T_{br} + 0.169347T_{br}^6 + + \omega(15.2518 - 15.6875/T_{br} - 13.4721\ln T_{br} + 0.43577T_{br}^6), \quad (25.25)$$

where

$P_r^{pv}$ Reduced vapor pressure, $P^{pv}/P_c$

$T_{br}$ Reduced NBP, $T_b/T_c$

$\omega$ Acentric factor

### 25.1.6. Viscosity

**Letsou-Stiel [16]**

$$\eta = \frac{\xi_0 + \xi_1}{\xi} \quad (25.26)$$

$$\xi_0 = 2.648 - 3.725T_r + 1.309T_r^2 \quad (25.27)$$

$$\xi_1 = 7.425 - 13.39T_r + 5.933T_r^2 \quad (25.28)$$

$$\xi = 176\left(\frac{T_c}{MM^3P_c^4}\right)^{1/6} \quad (25.29)$$

where

$\eta$ Viscosity (Pa.s)

$P_c$ Critical pressure (bar)

$T_r$ Reduced temperature, $T/T_c$

$MM$ Molecular weight (kg/kmol)

**Abbott[20]**

$$\begin{aligned} t_1 &= 4.39371 - 1.94733Kw + 0.12769Kw^2 + 0.00032629API^2 - 0.0118246KwAPI + \\ &\quad +(0.171617Kw^2 + 10.9943API + 0.0950663API^2 - 0.869218KwAPI \qquad (25.30) \\ \log v_{100} &= \frac{t_1}{API + 50.3642 - 4.78231Kw}, \qquad (25.31) \end{aligned}$$

$$\begin{aligned} t_2 &= -0.463634 - 0.166532API + 0.000513447API^2 - 0.00848995APIKw + \\ &\quad +(0.080325Kw + 1.24899API + 0.19768API^2 \qquad (25.32) \\ \log v_{210} &= \frac{t_2}{API + 26.786 - 2.6296Kw}, \qquad (25.33) \end{aligned}$$

where

$v_{100}$ Viscosity at 100 řF (cSt)

$v_{210}$ Viscosity at 210 řF (cSt)

$K_w$ Watson characterization factor

$API$ Oil API degree

## 25.2. Hypothetical Components

The majority of properties of the hypothetical components is calculated, when necessary, using the group contribution methods, with the UNIFAC structure of the hypo as the basis of calculation. The table 40 lists the properties and their calculation methods.

Table 4: Hypo calculation methods.

| Property | Symbol | Method |
|---|---|---|
| Critical temperature | $T_c$ | Joback [16] |
| Critical pressure | $P_c$ | Joback [16] |
| Critical volume | $V_c$ | Joback [16] |
| Normal boiling point | $T_b$ | Joback [16] |
| Vapor pressure | $P^{pv}$ | Lee-Kesler (Eq. 25.25) |
| Acentric factor | $\omega$ | Lee-Kesler (Eq. 25.23) |
| Vaporization enthalpy | $\Delta H_{vap}$ | Vetere [16] |
| Ideal gas heat capacity | $C_p^{gi}$ | Harrison-Seaton [21] |
| Ideal gas enthalpy of formation | $\Delta H_f^{298}$ | Marrero-Gani [22] |

# 26. Other Properties

## 26.1. True Critical Point

The Gibbs criteria for the true critical point of a mixture of $n$ components may be expressed of various forms, but the most convenient when using a pressure explicit cubic equation of state is

$$L = \begin{vmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & & \\ \vdots & & & \\ A_{n1} & \dots & \dots & A_{nn} \end{vmatrix} = 0 \tag{26.1}$$

$$M = \begin{vmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & & \\ \vdots & & & \\ A_{n-1,1} & \dots & \dots & A_{n-1,n} \\ \frac{\partial L}{\partial n_1} & \dots & \dots & \frac{\partial L}{\partial n_n} \end{vmatrix} = 0, \tag{26.2}$$

where

$$A_{12} = \left( \frac{\partial^2 A}{\partial n_1 \partial n_2} \right)_{T,V} \tag{26.3}$$

All the $A$ terms in the equations 26.1 and 26.2 are the second derivatives of the total Helmholtz energy $\underline{A}$ with respect to mols and constant $T$ and $V$. The determinants expressed by 26.1 and 26.2 are simultaneously solved for the critical volume and temperature. The critical pressure is then found by using the original EOS.

DWSIM utilizes the method described by Heidemann and Khalil [23] for the true critical point calculation using the *Peng-Robinson* and *Soave-Redlich-Kwong* equations of state.

## 26.2. Petroleum Cold Flow Properties

### 26.2.1. Refraction Index

**API Procedure 2B5.1**

$$I = 0.02266 \exp(0.0003905 \times (1.8MeABP) + 2.468SG - 0.0005704(1.8MeABP) \times SG) \times \times (1.8MeABP)^{0.0572} SG^{-0.72} \quad (26.4)$$

$$r = \left(\frac{1+2I}{1-I}\right)^{1/2} \quad (26.5)$$

where

| | |
|---|---|
| $r$ | Refraction Index |
| $SG$ | Specific Gravity |
| $MeABP$ | Mean Averaged Boiling Point (K) |

### 26.2.2. Flash Point

**API Procedure 2B7.1**

$$PF = \{[0.69 \times ((t_{10ASTM} - 273.15) \times 9/5 + 32) - 118.2] - 32\} \times 5/9 + 273.15 \quad (26.6)$$

where

| | |
|---|---|
| $PF$ | Flash Point (K) |
| $t_{10ASTM}$ | ASTM D93 10% vaporized temperature (K) |

### 26.2.3. Pour Point

**API Procedure 2B8.1**

$$PFL = [753 + 136(1 - \exp(-0.15v_{100})) - 572SG + 0.0512v_{100} + 0.139(1.8MeABP)] / 1.8 \quad (26.7)$$

where

| | |
|---|---|
| $PFL$ | Pour Point (K) |
| $v_{100}$ | Viscosity @ 100 řF (cSt) |

#### 26.2.4. Freezing Point

**API Procedure 2B11.1**

$$PC = -2390.42 + 1826SG + 122.49K - 0.135 \times 1.8 \times MeABP \tag{26.8}$$

where

$PC$ Freezing Point (K)

$K$ API characterization factor (API)

#### 26.2.5. Cloud Point

**API Procedure 2B12.1**

$$PN = \left[10^{(-7.41+5.49\log(1.8MeABP)-0.712\times(1.8MeABP)^{0.315}-0.133SG)}\right]/1.8 \tag{26.9}$$

where

$PN$ Cloud Point (K)

#### 26.2.6. Cetane Index

**API Procedure 2B13.1**

$$\begin{aligned} IC &= 415.26 - 7.673API + 0.186 \times (1.8MeABP - 458.67) + 3.503API \times \\ &\quad \times \log(1.8MeABP - 458.67) - 193.816 \times \log(1.8MeABP - 458.67) \end{aligned} \tag{26.10}$$

where

$IC$ Cetane Index

$API$ API degree of the oil

### 26.3. Chao-Seader Parameters

The Chao-Seader parameters needed by the CS/GS models are the Modified Acentric Factor, Solubility Parameter and Liquid Molar Volume. When absent, the Modified Acentric Factor is taken as the normal acentric factor, either read from the databases or calculated by using the methods described before in this document. The Solubility Parameter is given by

$$\delta = \left(\frac{\Delta H_v - RT}{V_L}\right)^{1/2} \tag{26.11}$$

where

$\Delta H_v$ Molar Heat of Vaporization

$V_L$ Liquid Molar Volume at 20 řC

# Part VII.

# How-To Tutorials

## 27. Methane Steam Reforming

### 27.1. Introduction

This tutorial is based on the document entitled ”Simulation of a Methane Steam Reforming Reactor”, which can be found here.

It deals with the task of developing a numerical model to predict the conversion and hydrogen yield within a steam reforming reactor.

### 27.2. Background

Natural gas has been proposed as a source of hydrogen for fuel cell vehicle applications because of the existing infrastructure. In a process known as steam reforming, natural gas and steam are reacted into mostly carbon monoxide and hydrogen with some carbon dioxide also produced. There can also be excess water in the reformate stream.

The steam reforming reaction is given as:

CH4 + H2O $\leftrightarrow$ 3 H2 + CO (1)

In the steam reformer, the water gas shift reaction also takes place as:

CO + H2O $\leftrightarrow$ H2 + CO2 (2)

Adding together the steam reforming and water gas shift reactions gives the overall reaction:

CH4 + 2 H2O $\leftrightarrow$ 4 H2 + CO2 (3)

The equilibrium constants can be expressed in terms of partial pressures (in atm) and temperature in degrees Kelvin. The subscript on the following equilibrium constants refers to the equation number given above:

$$K_1 = \frac{P_{H2}^3 P_{CO}}{P_{CH4} P_{H2O}} = \exp(30.42 - 27106/T)$$

$$K_2 = \frac{P_{H2} P_{CO2}}{P_{CO} P_{H2O}} = \exp(-3.798 + 4160/T)$$

$$K_3 = \frac{P_{H2}^4 P_{CO2}}{P_{CH4} P_{H2O}^2} = \exp(34.218 - 31266/T)$$

In the reactor, methane (CH4) and water (H2O) are fed as reactants and carbon dioxide (CO2), carbon monoxide (CO), and hydrogen (H2) are produced over a nickel catalyst on an alumina support.

In laboratory experiments, a nonreacting inert gas such as helium (He) may also be present. In the most general form, the governing conservation equations for each of these species is given below, where denotes the molar flow rate of species i in mol/h, ***W*** denotes the catalyst weight in g, and ***Ri*** denotes the reaction rate of equation i in units of mol/(g-h):

$$\frac{dF_{CH4}}{dW} = -(R_1 + R_3) \text{ with } F_{CH4}(W=0) = F^0_{CH4}$$

$$\frac{dF_{H2O}}{dW} = -(R_1 + R_2 + 2R_3) \text{ with } F_{H2O}(W=0) = F^0_{H2O}$$

$$\frac{dF_{H2}}{dW} = (3R_1 + R_2 + 4R_3) \text{ with } F_{H2}(W=0) = F^0_{H2}$$

$$\frac{dF_{CO}}{dW} = (R_1 - R_2) \text{ with } F_{CO}(W=0) = F^0_{CO}$$

$$\frac{dF_{CO2}}{dW} = (R_2 + R_3) \text{ with } F_{CO2}(W=0) = F^0_{CO2}$$

$$\frac{dF_{He}}{dW} = 0 \text{ with } F_{He}(W=0) = F^0_{He}$$

The reaction rates are given by:

$$R_1 = \frac{\frac{k_1}{P^{2.5}_{H2}}\left[P_{CH4}P_{H2O} - \frac{P^3_{H2}P_{CO}}{K_1}\right]}{DEN^2}$$

$$R_2 = \frac{\frac{k_2}{P_{H2}}\left[P_{CO}P_{H2O} - \frac{P_{H2}P_{CO2}}{K_2}\right]}{DEN^2}$$

$$R_3 = \frac{\frac{k_3}{P^{3.5}_{H2}}\left[P_{CH4}P^2_{H2O} - \frac{P^4_{H2}P_{CO2}}{K_3}\right]}{DEN^2}$$

$$DEN = 1 + K_{CH4}P_{CH4} + K_{CO}P_{CO} + K_{H2}P_{H2} + \frac{K_{H2O}P_{H2O}}{P_{H2}}$$

Furthermore, the coefficients in the equations above are given by the Arrhenius relationships as:

$$k_1 = 4.22 \times 10^{15} \exp(-240100 / RT)$$

$$k_2 = 1.96 \times 10^{6} \exp(-67130 / RT)$$

$$k_3 = 1.02 \times 10^{15} \exp(-243900 / RT)$$

$$K_{CH4} = 6.65 \times 10^{-4} \exp(38280 / RT)$$

$$K_{H2O} = 1.77 \times 10^{5} \exp(-88680 / RT)$$

$$K_{H2} = 6.12 \times 10^{-9} \exp(82900 / RT)$$

$$K_{CO} = 8.23 \times 10^{-5} \exp(70650 / RT)$$

Note that in the above expressions, R = 8.314 J/(mol-K) is the gas constant.

## 27.3. Problem Statement

Consider a feed of 10000 mol/h CH4, 10000 mol/h H2O, and 100 mol/h H2 to a steam reforming reactor that operates at 1000 K and a 1 atm feed pressure. Determine the overall methane conversion as a function of catalyst weight up to 382 g.

The overall methane conversion as found on the original reference is equal to **76%**. We'll try to obtain the same result in DWSIM.

## 27.4. DWSIM Model (Classic UI)

1. Create a New Steady State Simulation. Close the Simulation Wizard.

***Remember to Save your simulation at the end of each step.***

2. Go to **Edit** > **Simulation Settings** > **Compounds**, and select Methane, Hydrogen, Water, Carbon Monoxide and Carbon Dioxide to add these compounds to the simulation.

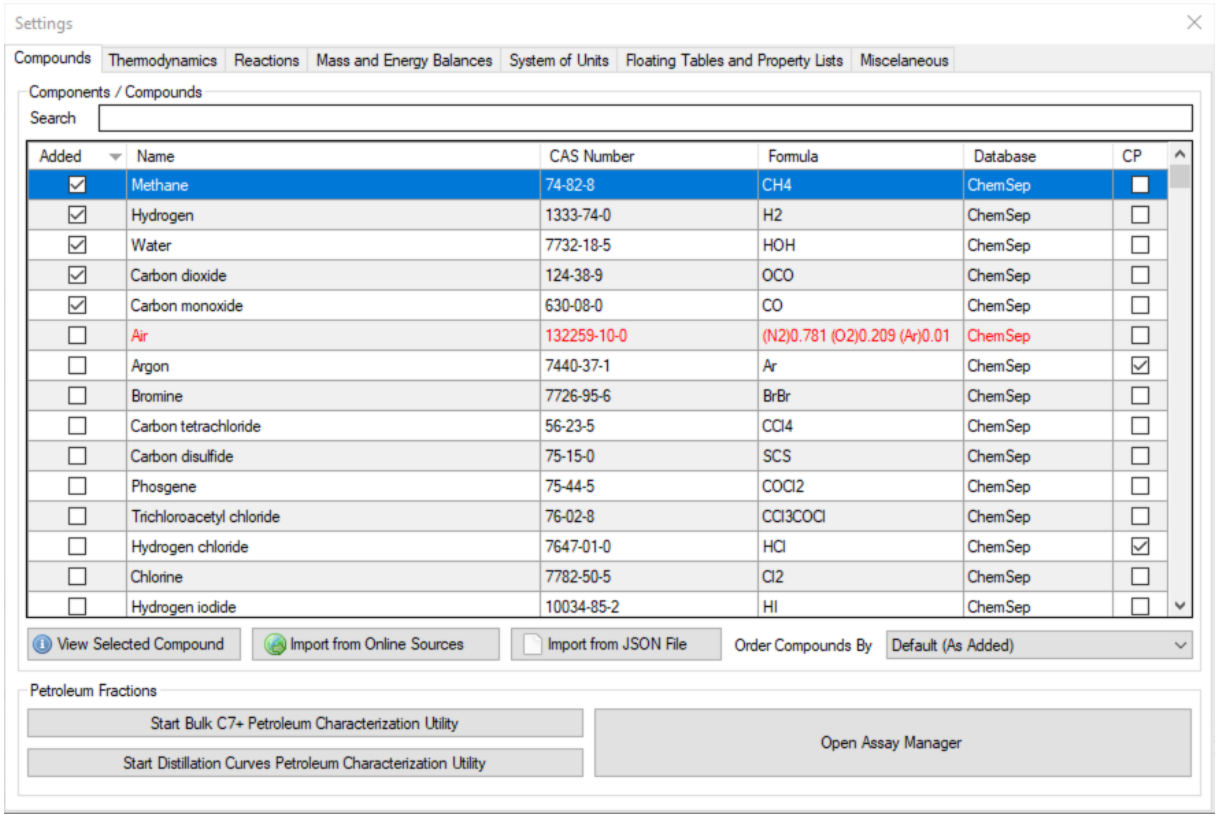



Figure 89: Compound Selection

3. Go to **Thermodynamics** tab, select **Peng-Robinson (PR)** on the Available Property Packages section and click **Add**.

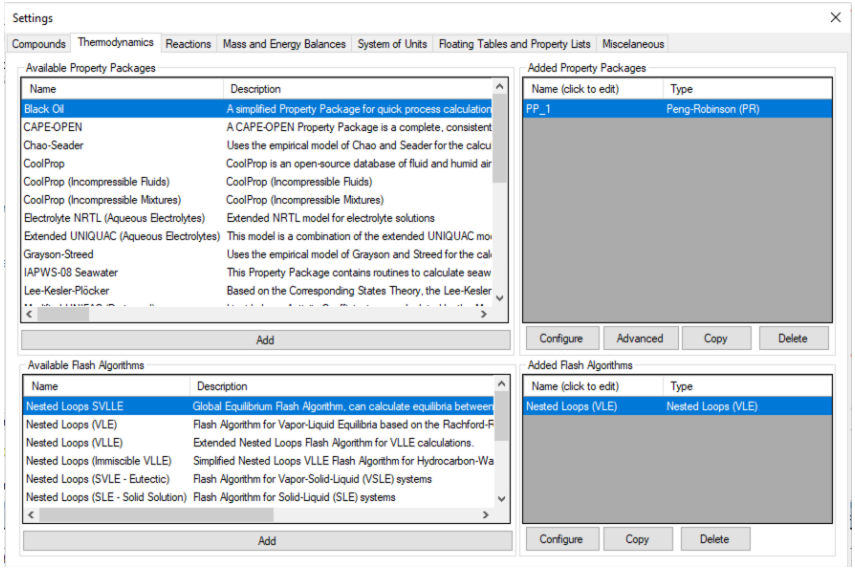



Figure 90: Property Package Selection

4. Go to the **System of Units** tab and create a new System of Units, with the following setup:

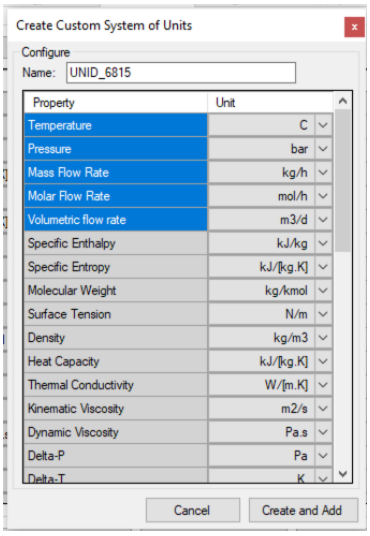



Figure 91: New System of Units

5. After creating this Units Set, select it on the System of Units combobox.

6. Go to **Reactions** and create three Heterogeneous Catalytic reactions, with the following configuration. The denominator expression is the same for all three reactions: **(1+1.77E+5*exp(-88680/8.314/T)*R1/P1+ 6.12E-9*exp(82900/8.314/T)*P1+8.23E-5*exp(70650/8.314/T)*R2)^2**

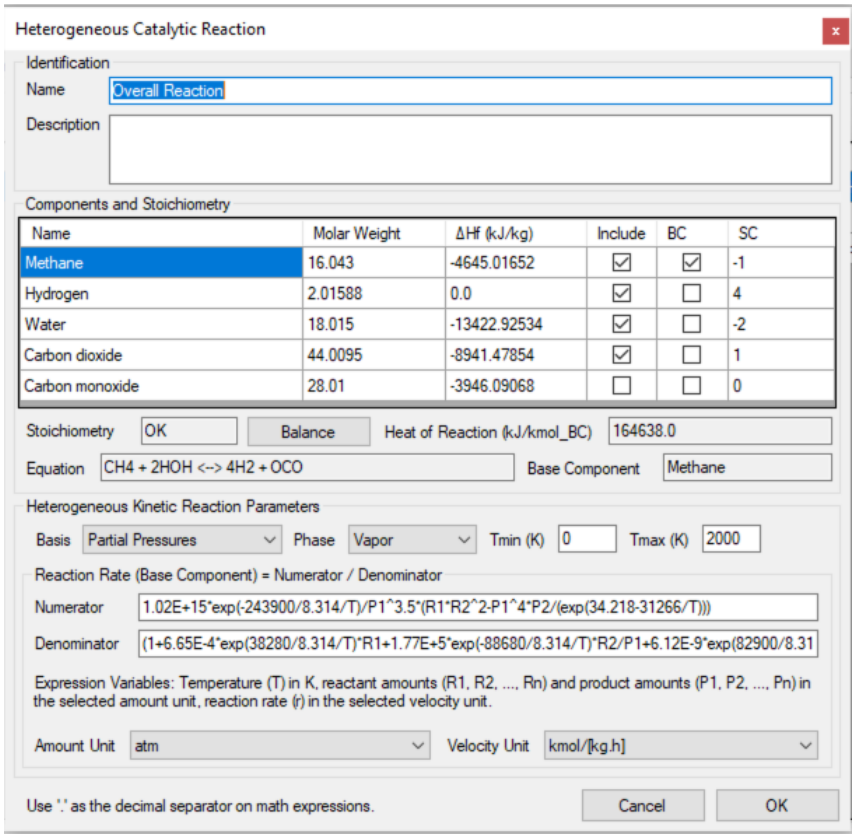



Figure 92: Overall Reaction setup

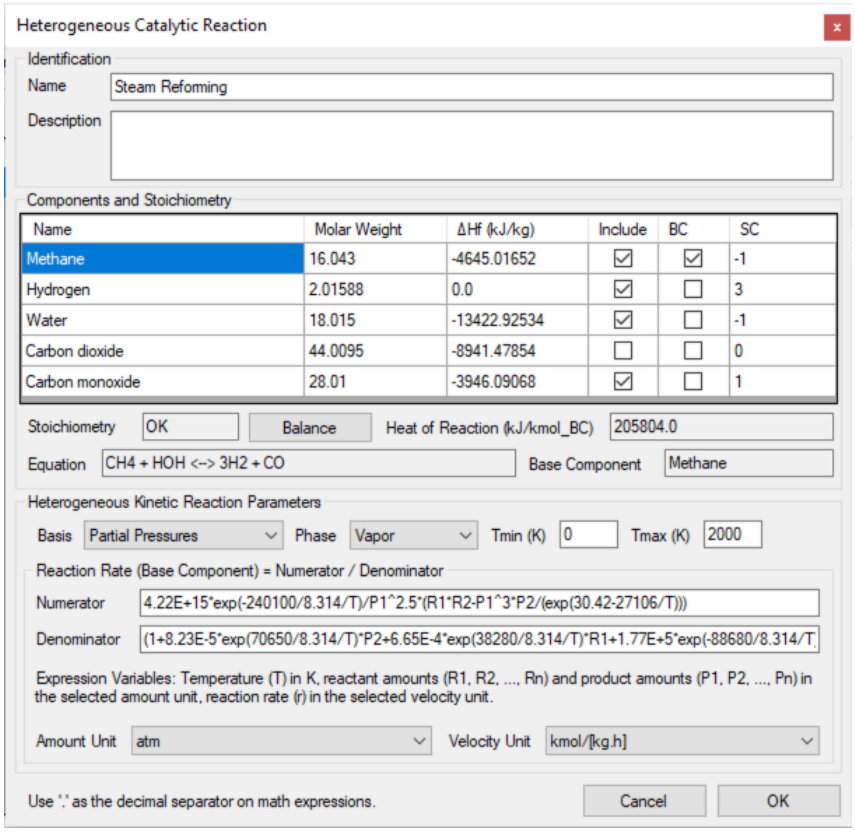



Figure 93: Steam Reforming Reaction setup

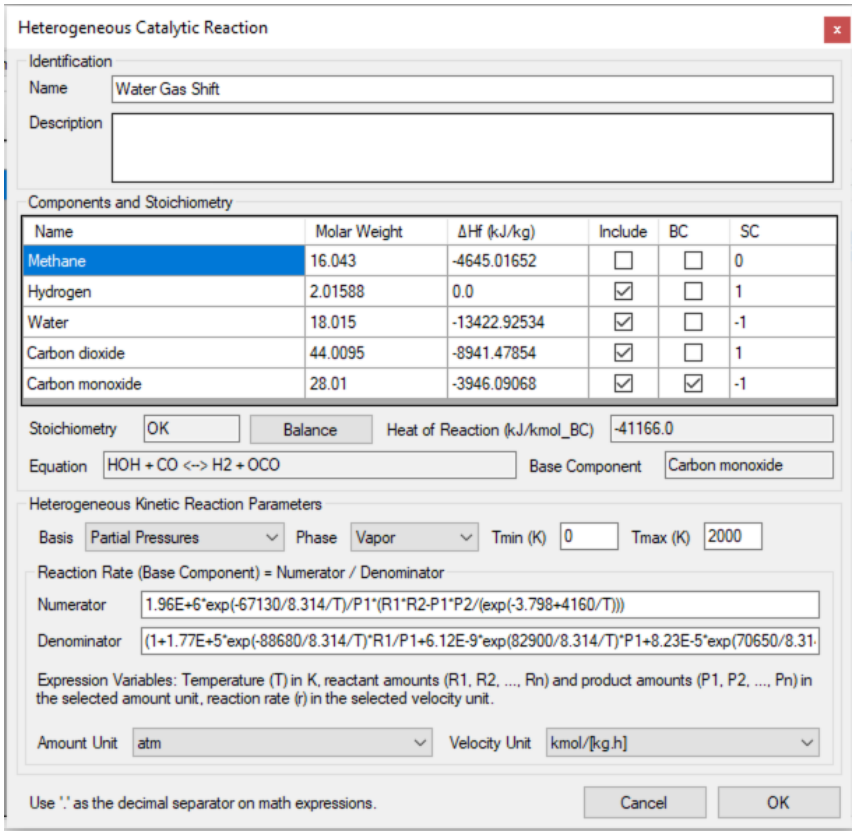



Figure 94: Water Gas Shift Reaction setup

7. Close the Settings panel, and drag two material streams, one energy stream and one PFR to the Flowsheet PFD. Connect the streams to the PFR as shown on the following figure:

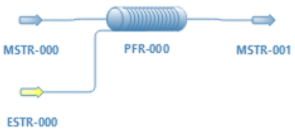



Figure 95: PFD setup

8. Configure the inlet stream (MSTR-000) as follows:

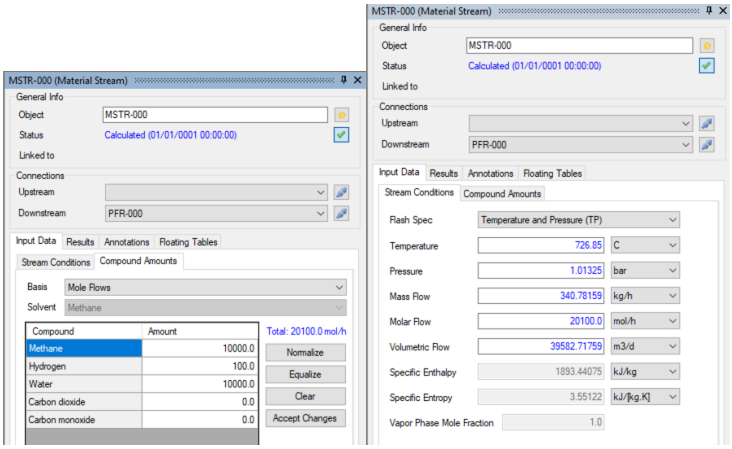



Figure 96: Inlet Stream setup

9. Configure the PFR as follows:

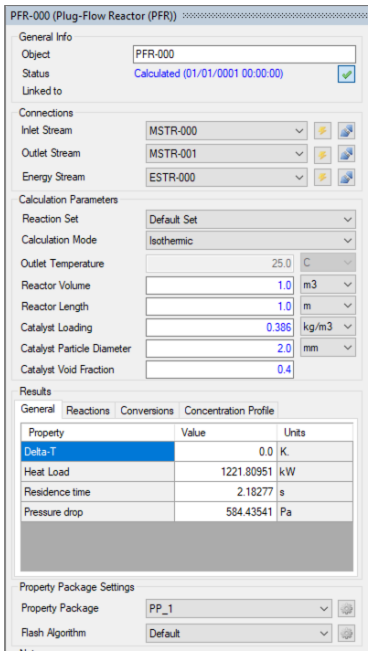



Figure 97: PFR setup

10. **Note:** when you create new reactions, they are automatically added to the **Default Reaction Set**. When you add new reactors to the flowsheet, they are automatically configured to use all supported and active reactions on the Default Reaction Set. You can create, edit and remove Reaction Sets at any time, and associate the individual reactors with different Reactions Sets too.

11. Run the simulation (press F5 or click on the Solve Flowsheet button on the toolbar). Wait for the calculation to finish.

12. Once finished, you should get the following results (methane conversion ~ **76.4%**):

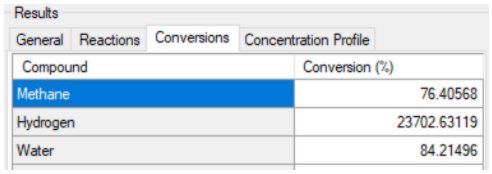



Figure 98: Final Methane conversion

13. Create a new Sensitivity Analysis case to study the influence of the temperature on Methane conversion from 700 to 1000 C. Go to **Tools** > **Sensitivity Analysis** and click on **Create New**.

14. Setup the Independent and Dependent Variables as follows:

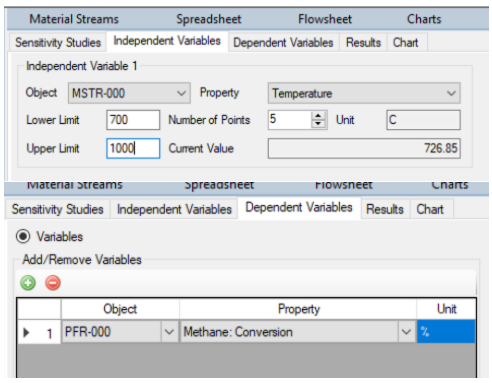



Figure 99: Sensitivity Analysis case setup

15. Go to Results and run the Case. Wait for the calculations to finish.

16. Once finished, click on **Send Data to New Worksheet**.

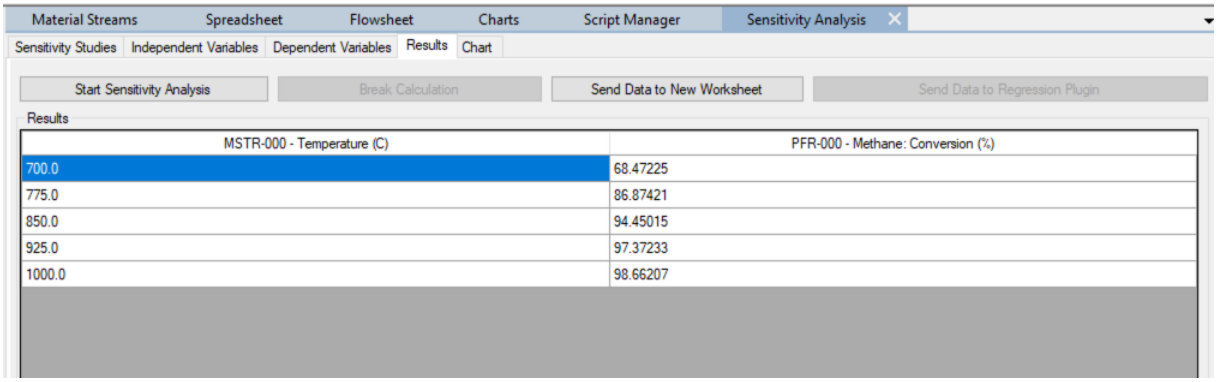



Figure 100: Analysis results

17. With the data range selected on the flowsheet, right-click on it and select **Create 2D XY Chart from Selection**.

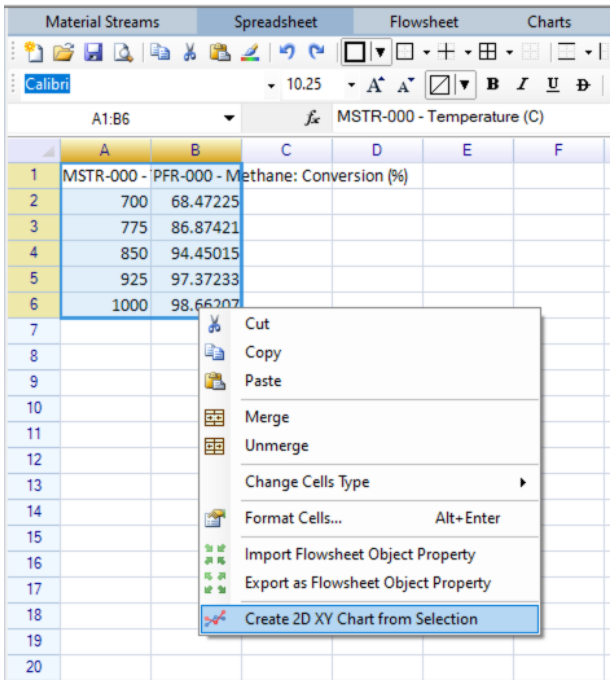



Figure 101: Create Chart from Spreadsheet Range

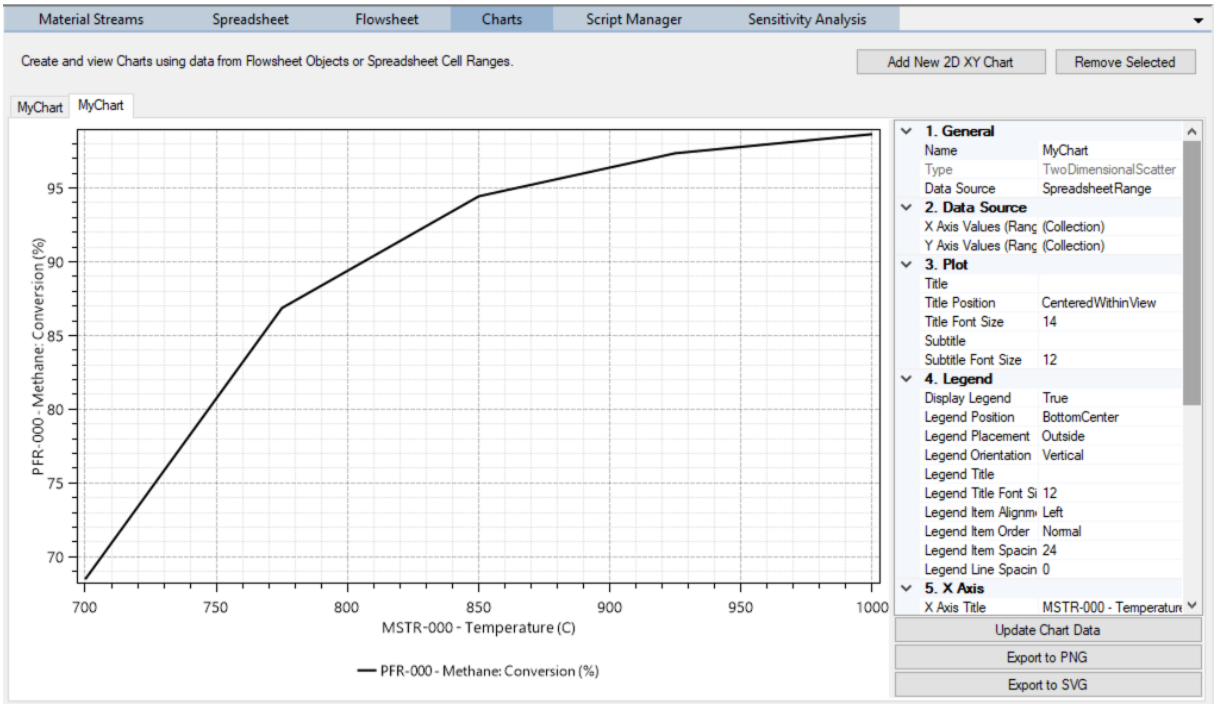



Figure 102: Created Chart

18. You can also view the concentration profile of the PFR using the Charts utility. Click on **Add New 2D XY Chart**, select the PFR as the data source of the chart, select *Concentration Profile* as the **Chart Type** and click on **Update Chart Data:**

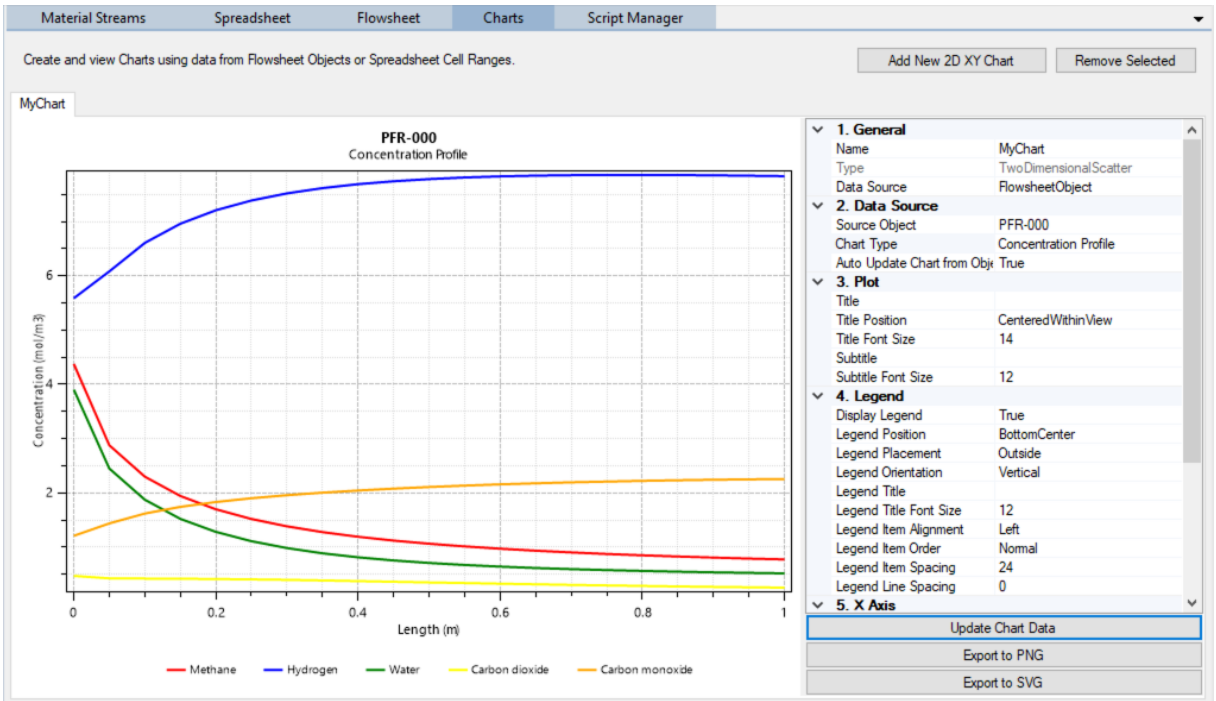



Figure 103: PFR concentration profile

## 27.5. DWSIM Model (Cross-Platform UI)

1. Create a New Simulation. Close the Simulation Wizard.

> ***Remember to Save your simulation at the end of each step.***

2. Go to **Setup** > **Compounds**, and select Methane, Hydrogen, Water, Carbon Monoxide and Carbon Dioxide to add these compounds to the simulation.

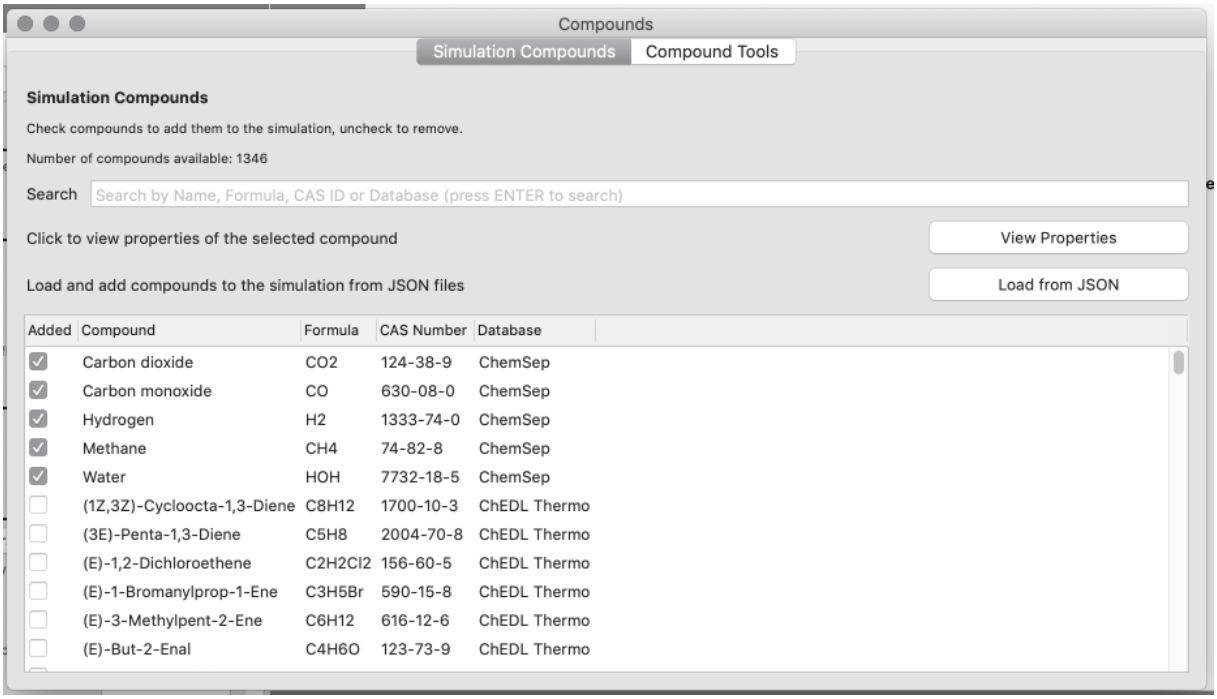



Figure 104: Compound Selection

3. Go to **Setup > Basis**, select **Peng-Robinson (PR)** on the **Add New Property Package** combobox to add a copy of this Property Package to the simulation.

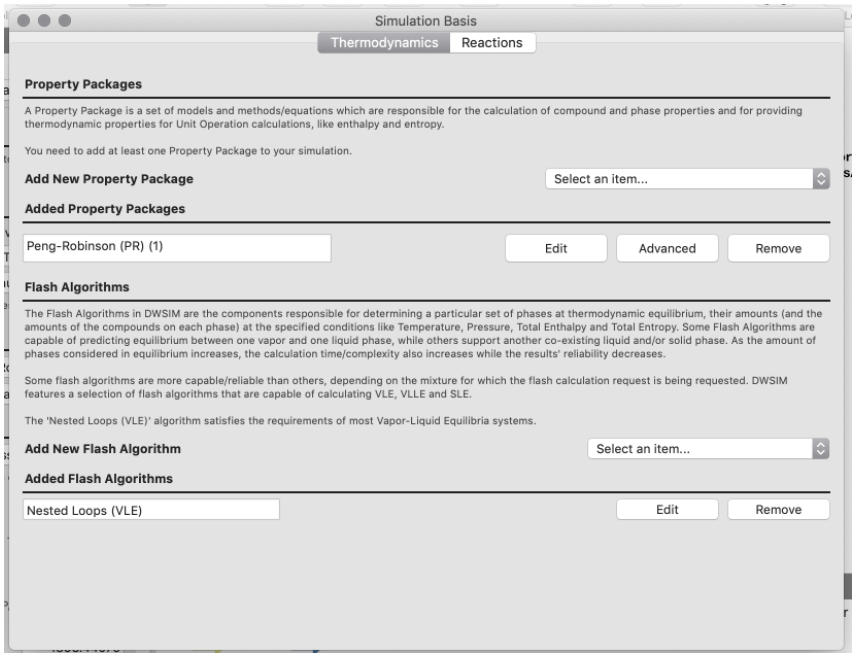



Figure 105: Property Package Selection

4. Go to the **Setup > Flowsheet Settings > System of Units** section and create a new System of Units, with the following setup:

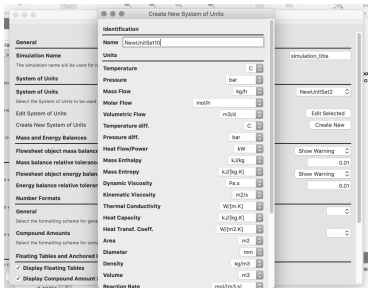


Figure 106: New System of Units

5. After creating this Units Set, select it on the System of Units combobox.

6. Go to **Setup > Basis > Reactions** and create three new Heterogeneous Catalytic reactions, with the following configuration. The denominator expression is the same for all three reactions: **(1+1.77E+5*exp(-88680/8.314/T)* R1/P1+6.12E-9*exp(82900/8.314/T)*P1+8.23E-5*exp(70650/8.314/T)*R2)^2**

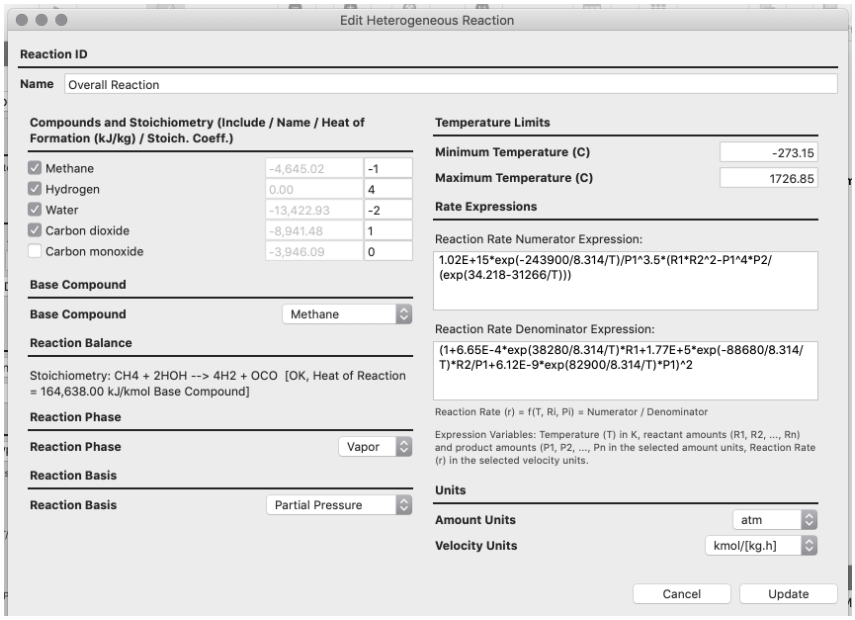



Figure 107: Overall Reaction setup

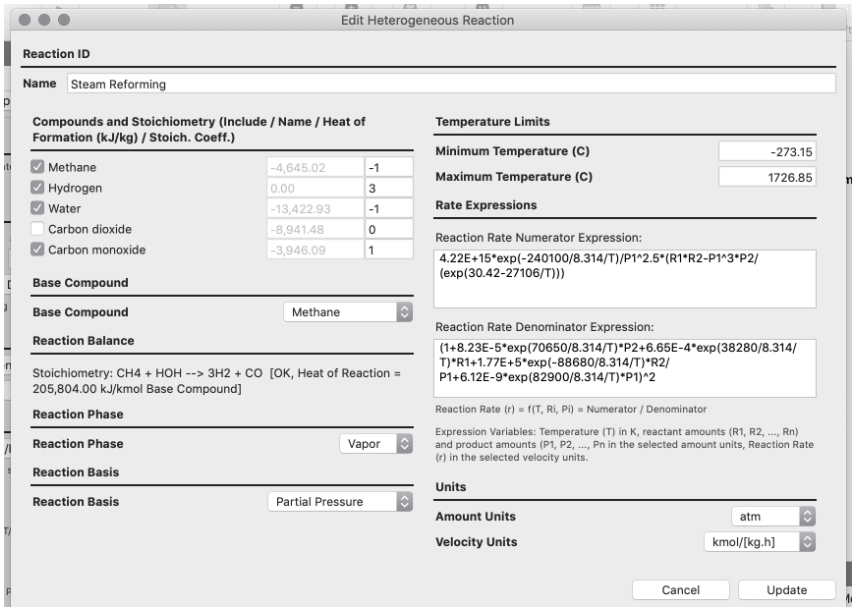



Figure 108: Steam Reforming Reaction setup

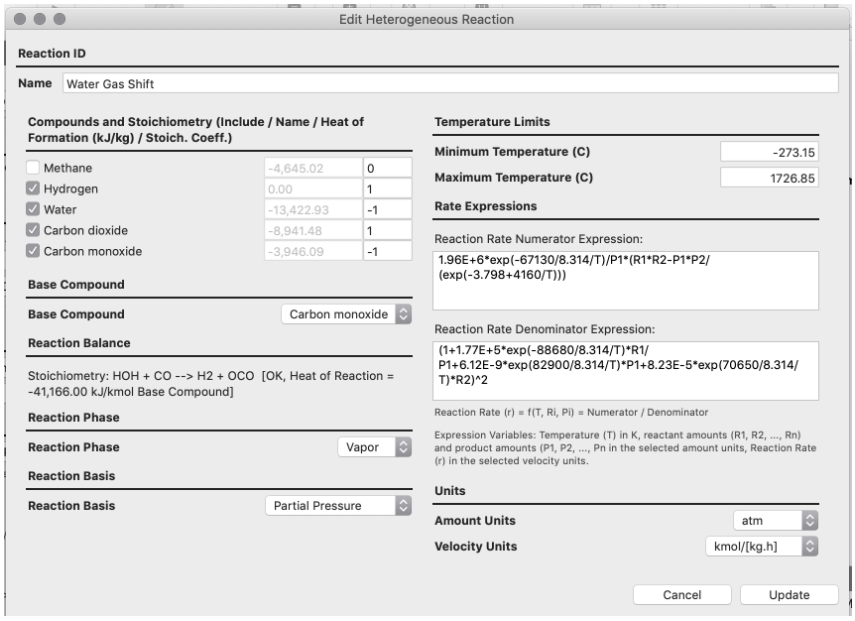



Figure 109: Water Gas Shift Reaction setup

7. Close the Basis panel, and drag two material streams, one energy stream and one PFR from the **Object Palette** to the **Flowsheet PFD**. Connect the streams to the PFR as shown on the following figure:

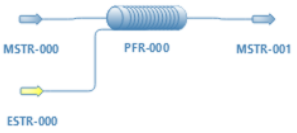



Figure 110: PFD setup

8. Configure the inlet stream (MSTR-000) as follows (enter the temperature, pressure, molar flow and composition in molar fractions):

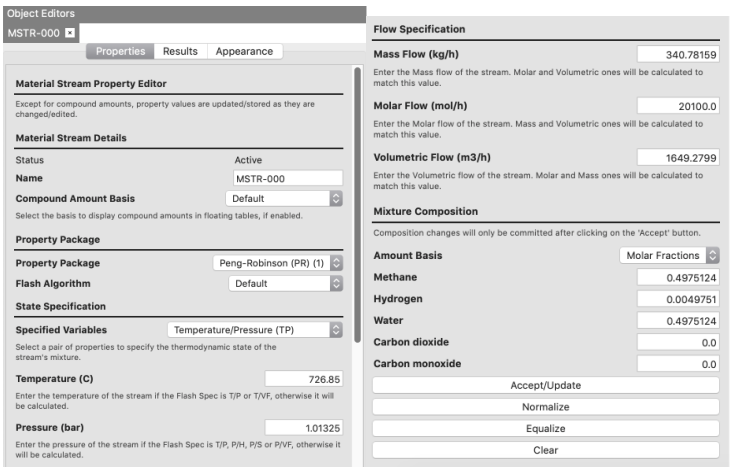



Figure 111: Inlet Stream setup

9. Configure the PFR as follows:

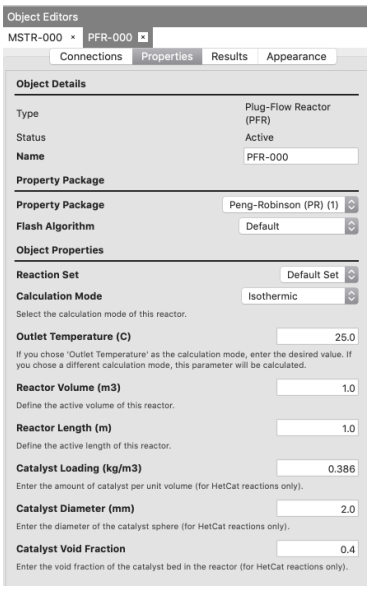



Figure 112: PFR setup

10. **Note:** when you create new reactions, they are automatically added to the **Default Reaction Set**. When you add new reactors to the flowsheet, they are automatically configured to use all supported and active reactions on the Default Reaction Set. You can create, edit and remove Reaction Sets at any time, and associate the individual reactors with different Reactions Sets too.

11. Run the simulation (press F5 or click on the **Solve Flowsheet** button on the toolbar). Wait for the calculation to finish.

12. Once finished, you should get the following results (methane conversion ~ **76.4%**):

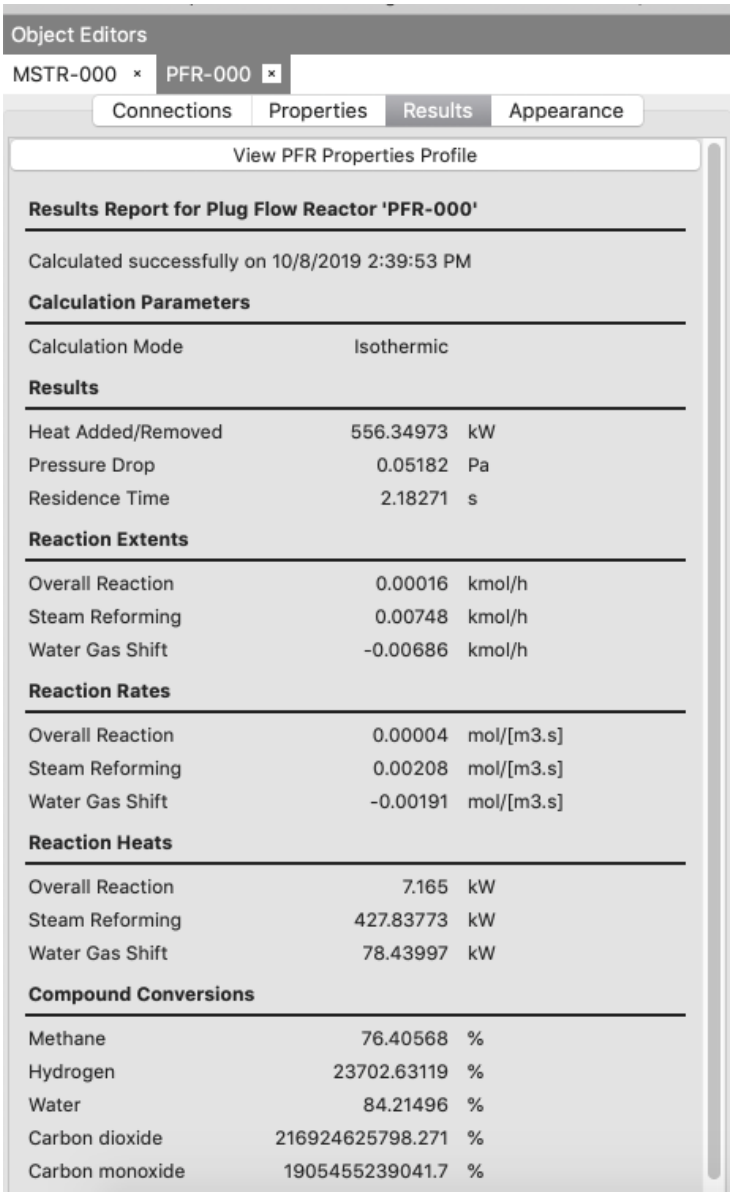



Figure 113: PFR calculation results

13. Create a new Sensitivity Analysis case to study the influence of the temperature on Methane conversion from 700 to 900 C. Go to **Tools** > **Sensitivity Analysis**.

14. Setup the Independent and Dependent Variables as follows:

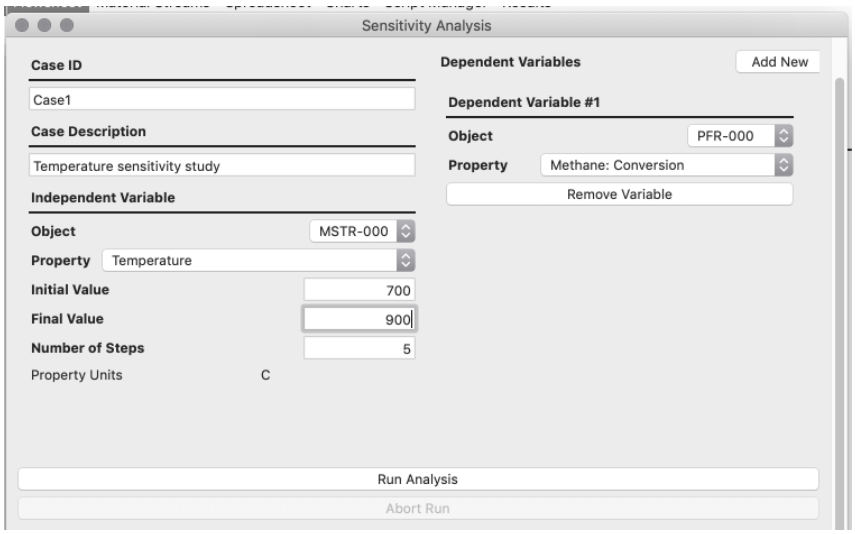



Figure 114: Sensitivity Analysis case setup

15. Run the Analysis. Wait for the calculations to finish (the **View Report** and **View Chart** buttons will become active).

16. Once finished, click on **View Chart**.

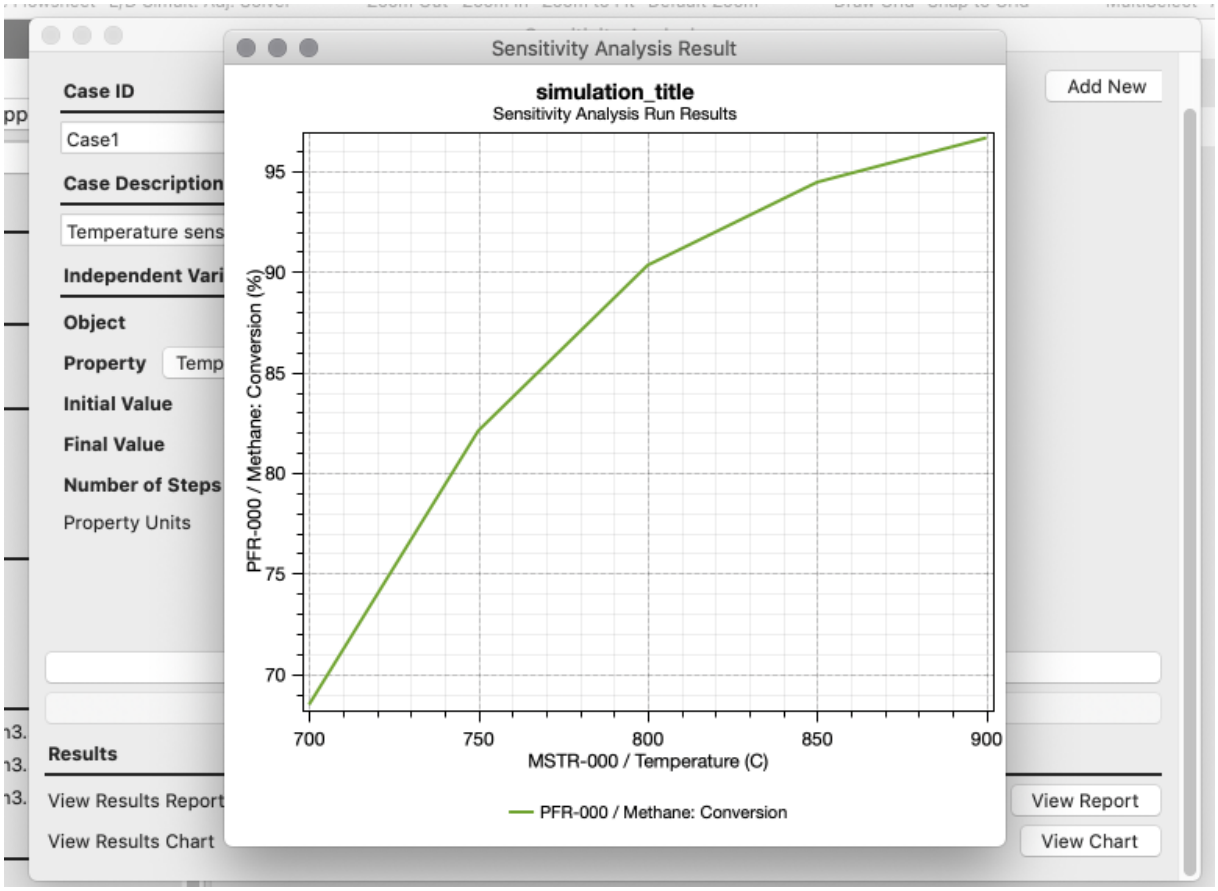



Figure 115: Sensitivity Analysis results

17. You can also view the concentration profile of the PFR using the Charts utility. Go to the **Charts tab**

on the main flowsheet window, click on **Add New 2D XY Chart**, select the PFR as the data source of the chart, select *Concentration Profile* as the **Chart Type** and click on **Update Chart Data:**

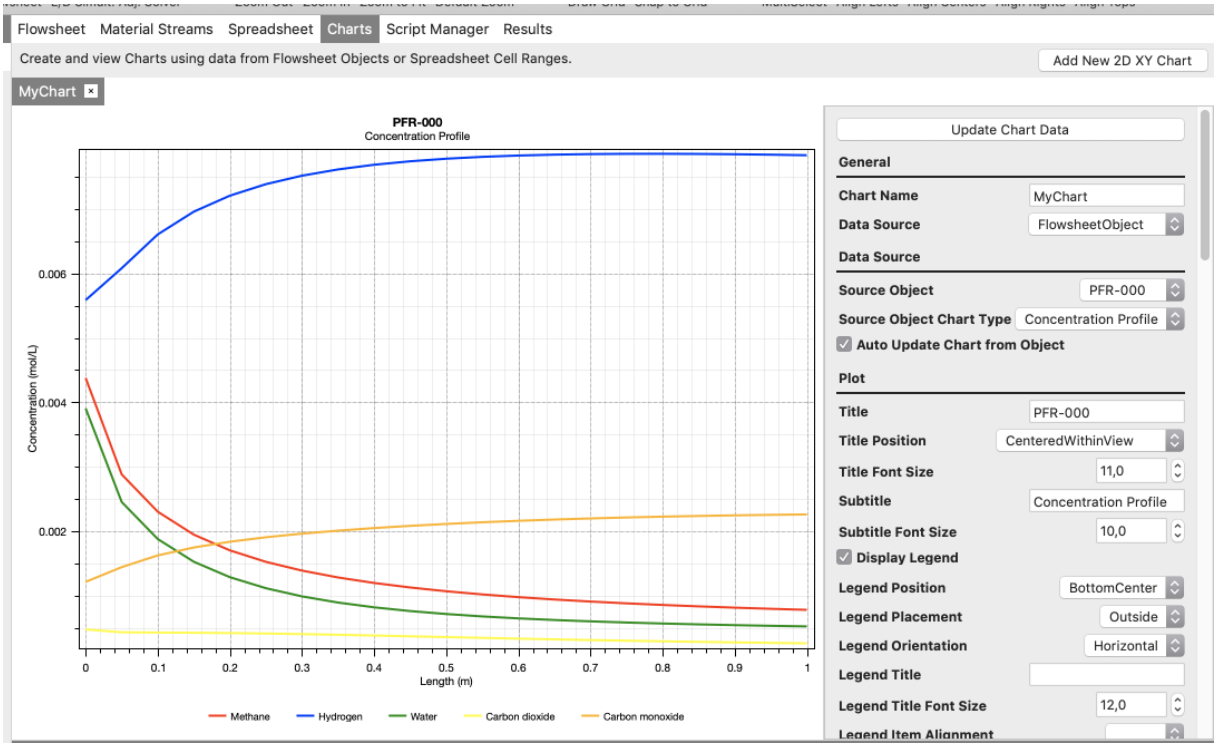



Figure 116: PFR concentration profile as visible on the Charts tool

# 28. Extractive Distillation

## 28.1. Introduction

This simulation is an example of Pressure Swing Azeotropic Distillation. Test case taken from COCO Simulator (link) (original author: Harry Kooijman - www.chemsep.org). Adapted from Luyben et al., Ind. Eng. Chem. Res. (2008) 47 pp. 2696-2707.

## 28.2. Background

Methanol and acetone form a minimum temperature azeotrope but the composition of this azeotrope is sensitive to the pressure. We can make use of this to separate the two components into pure products by operating two columns at different pressures.

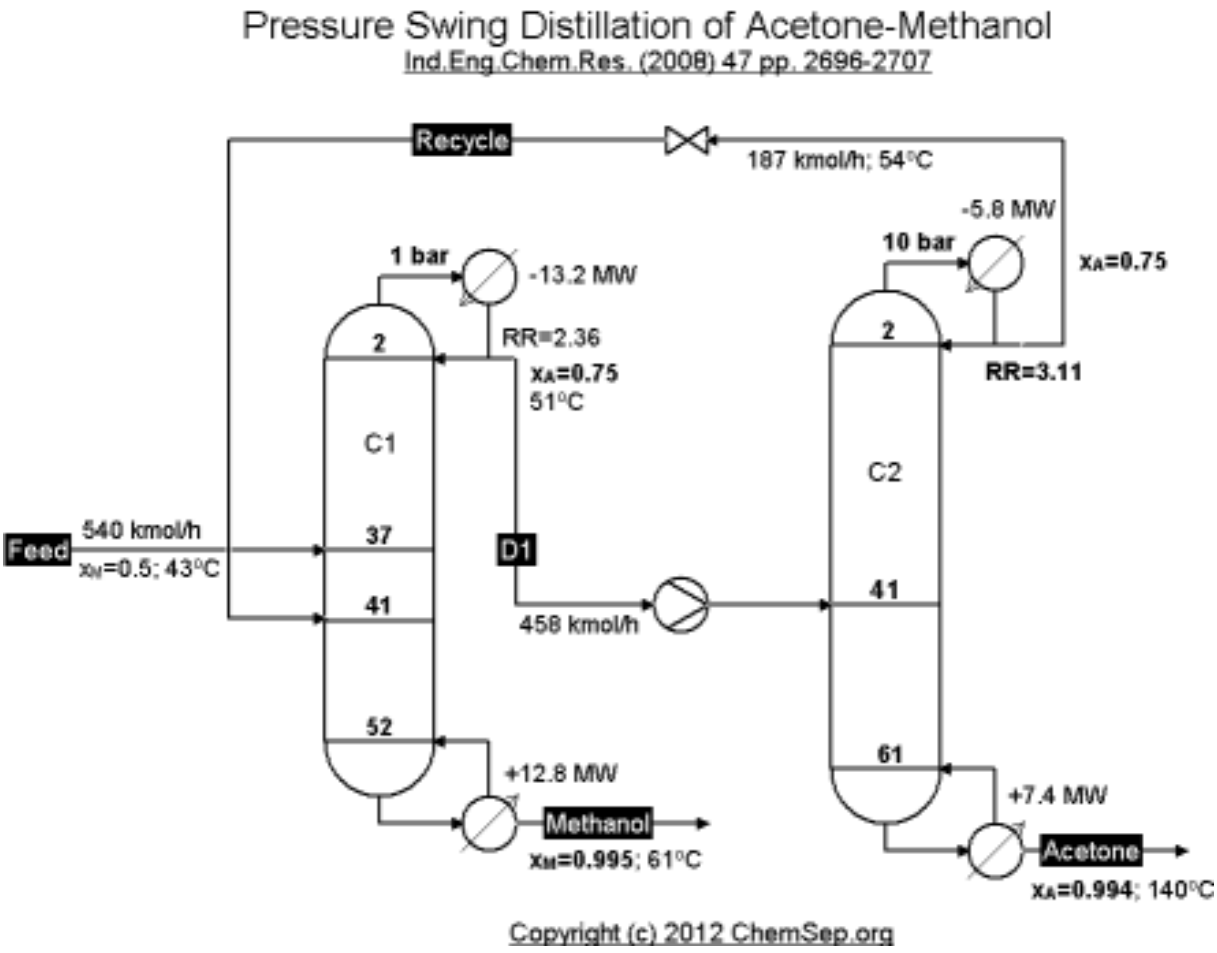



Figure 117: Process Flowsheet.

## 28.3. DWSIM Model (Classic UI)

1. Create a New Steady State Simulation. Close the Simulation Wizard.

> ***Remember to Save your simulation at the end of each step.***

2. Go to **Edit** > **Simulation Settings** > **Compounds**, and select Methanol and Acetone to add these compounds to the simulation.

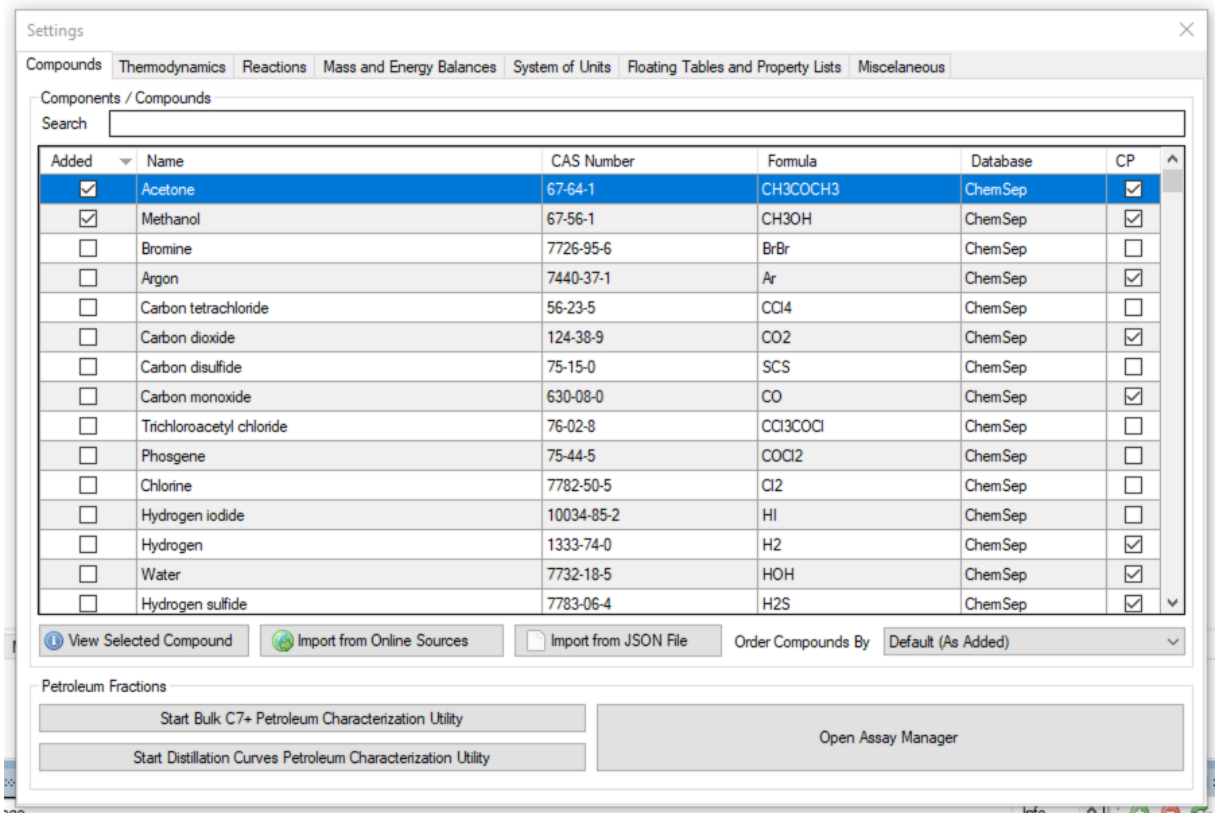



Figure 118: Compound Selection

3. Go to **Thermodynamics** tab, select **NRTL** on the Available Property Packages section and click **Add**. Add an instance of the Inside-Out VLE Flash Algorithm too.

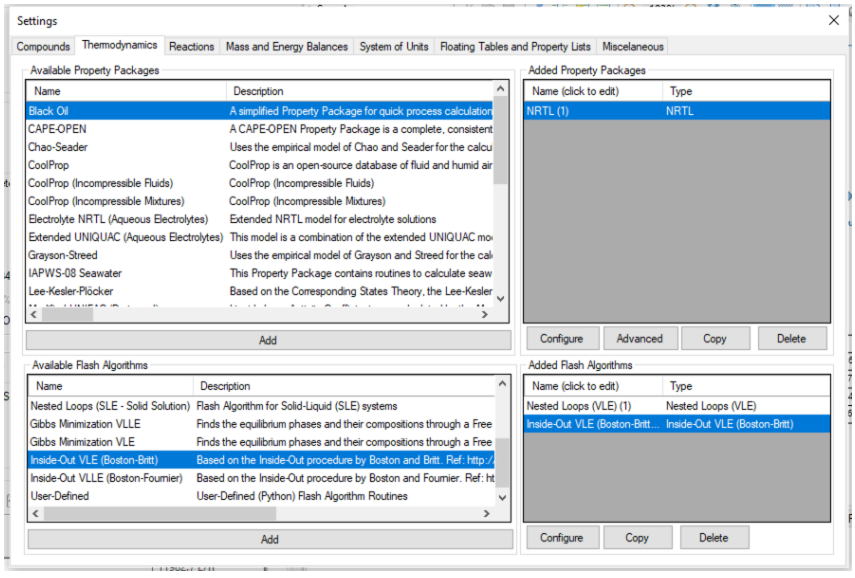



Figure 119: Property Package and Flash Algorithm Selection

4. Check if the NRTL Interaction Parameters are all set (click on **Configure** on the Added Property Packages section).

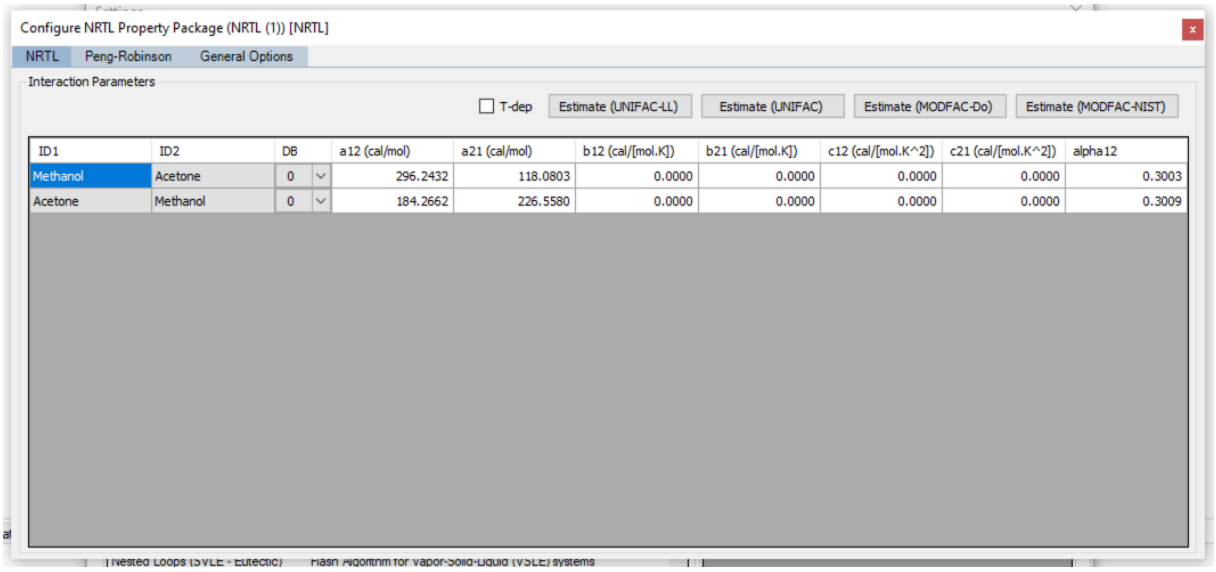



Figure 120: NRTL Interaction Parameters for Methanol/Acetone

5. Go to the **System of Units** tab and create a new System of Units, with the following setup:

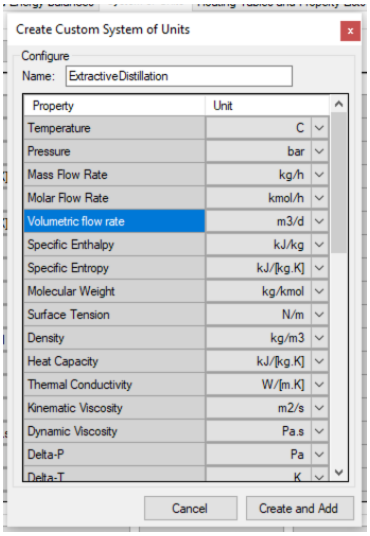



Figure 121: New System of Units

6. After creating this Units Set, select it on the System of Units combobox.

7. Add the objects to the flowsheet (streams, pump, valve, recycle and distillation columns) as depicted on the following figure, renaming them as required. You'll setup the connections between them later on this tutorial.

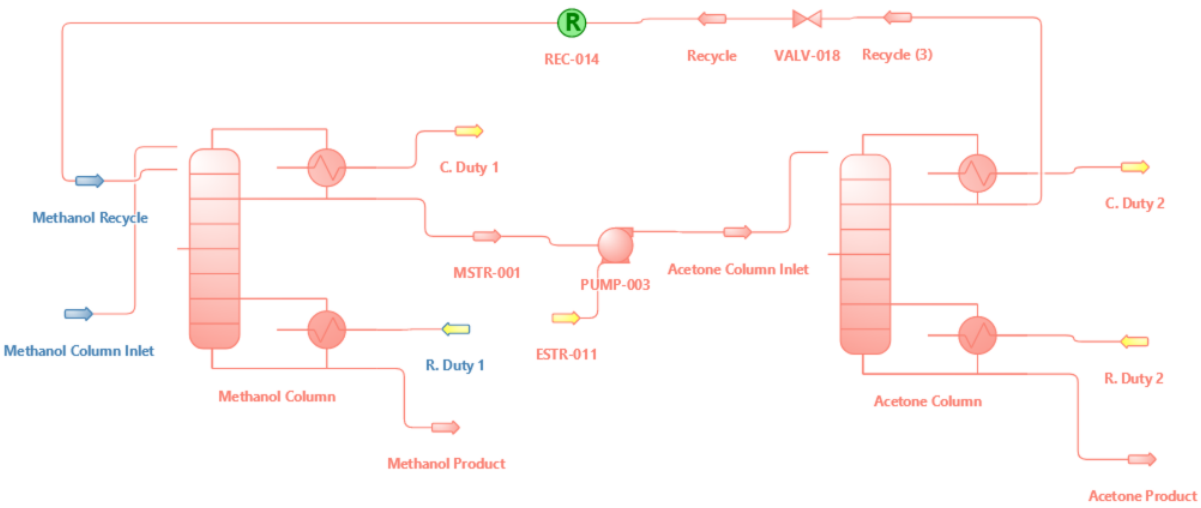



Figure 122: Process Flowsheet Diagram

8. Disable automatic calculation of the flowsheet.

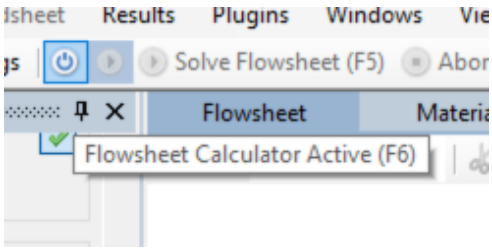



Figure 123: Enable/Disable Flowsheet Calculator/Solver

9. Setup the columns and their connections as follows:

   a) Methanol Column:

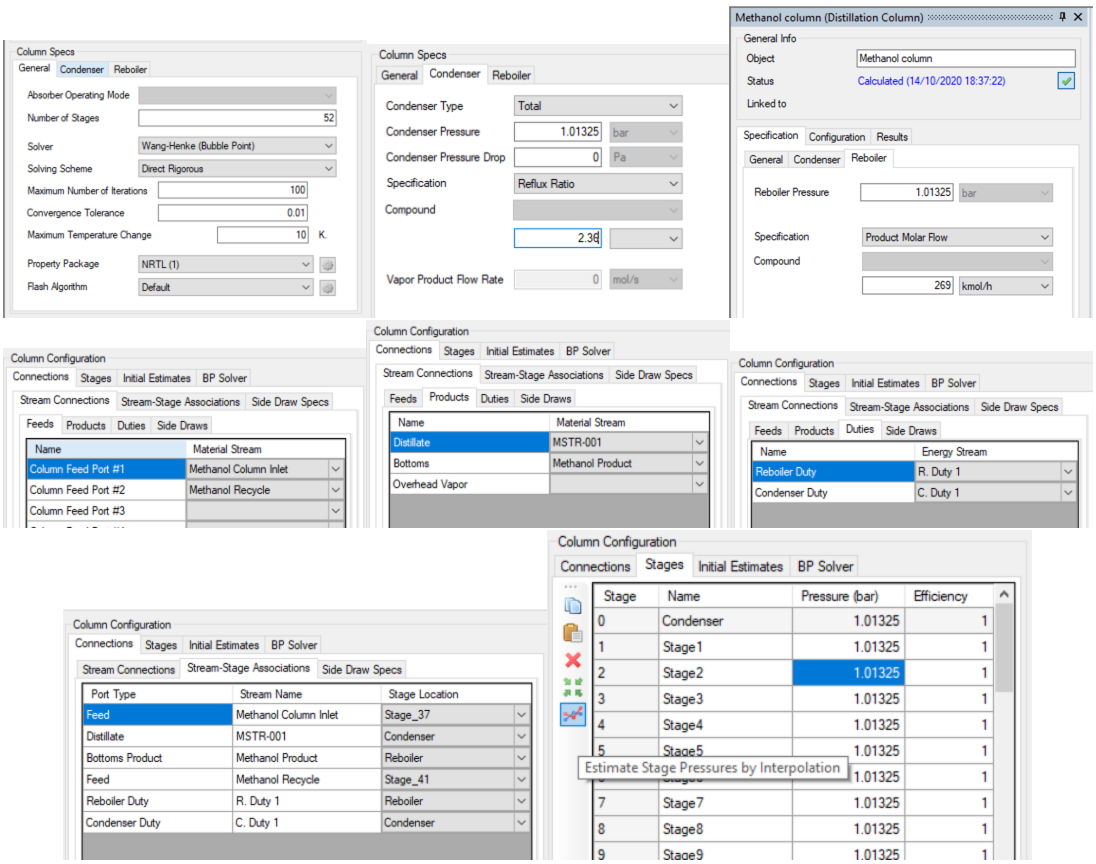



Figure 124: Methanol Column configuration

b) Acetone Column:

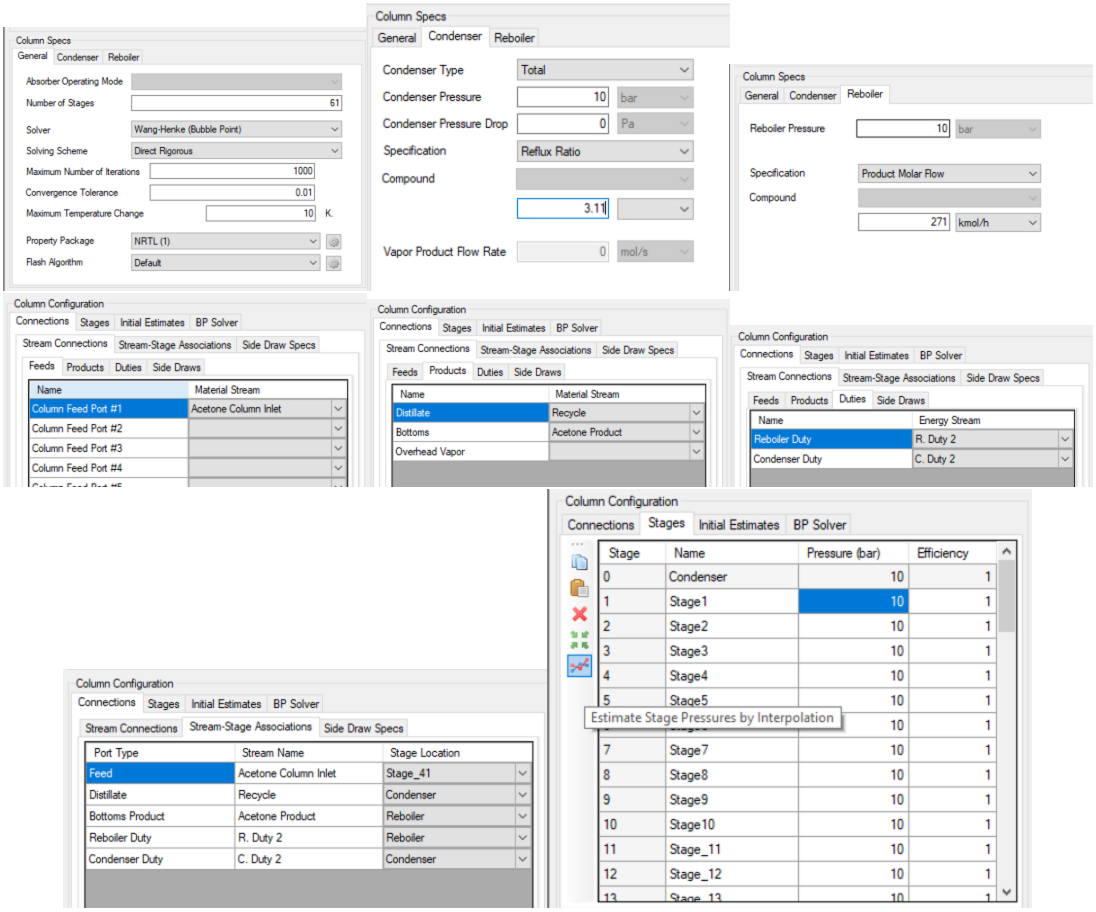



Figure 125: Acetone Column configuration

10. Enter initial estimates for the temperature profile of the Acetone Column, and check the **T** checkbox so DWSIM can use them. Insert only the boundary values (condenser and reboiler) and click on **Interpolate** to calculate the inner stage values.

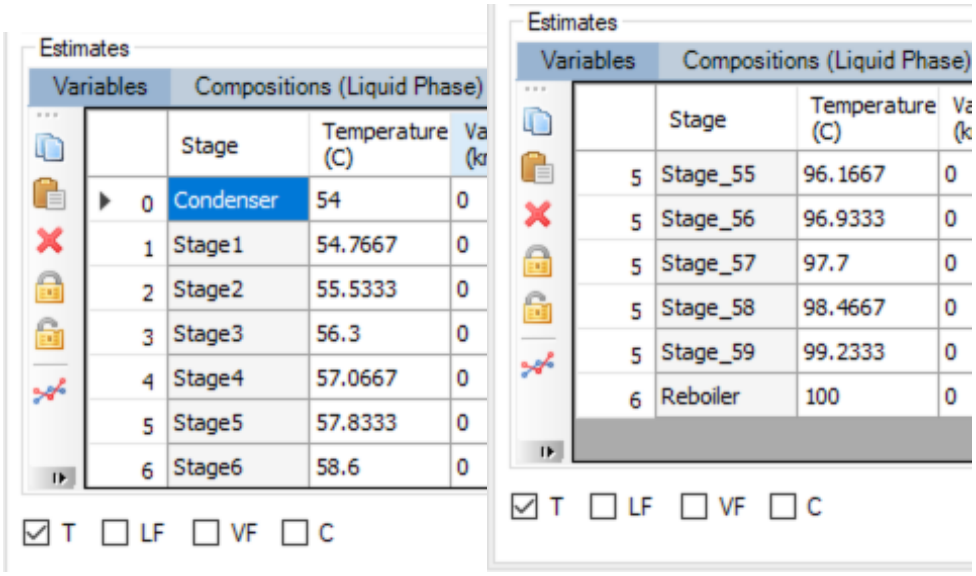



Figure 126: Acetone Column initial estimates for temperature profile

11. After the columns are correctly configured and connected to their associated streams, setup the pump, valve and recycle connections using their Editor Panels.

12. Setup the pump and valve properties as follows:

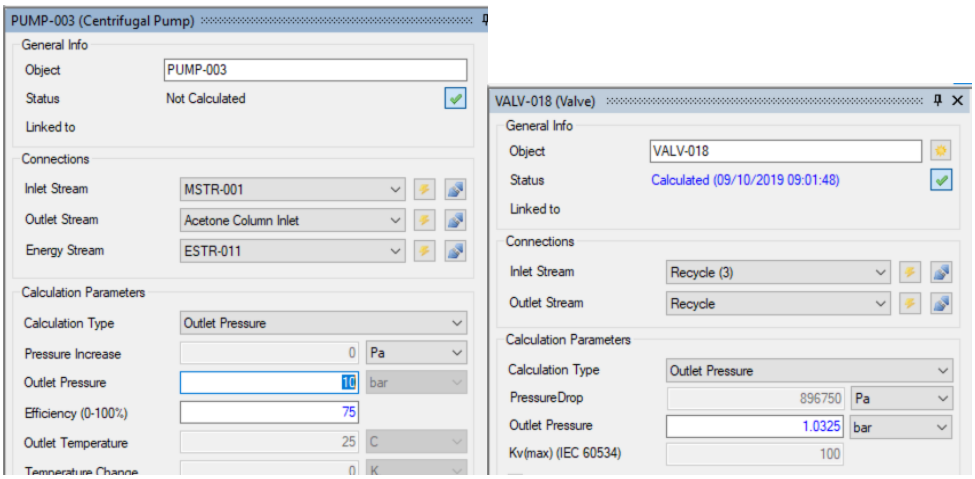



Figure 127: Pump and Valve properties

13. Configure the Methanol Inlet Stream as follows:

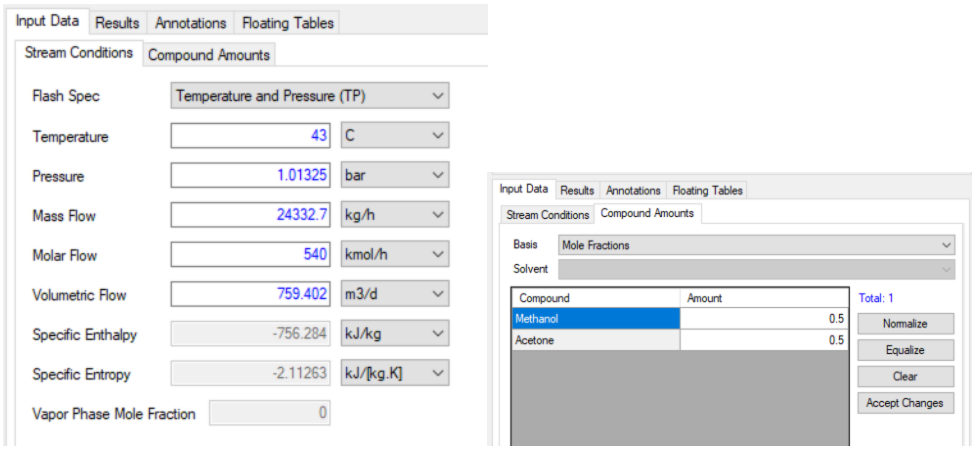



Figure 128: Methanol Inlet Stream configuration

14. Configure the Methanol Recycle Stream (initial estimates for the recycle stream) as follows:

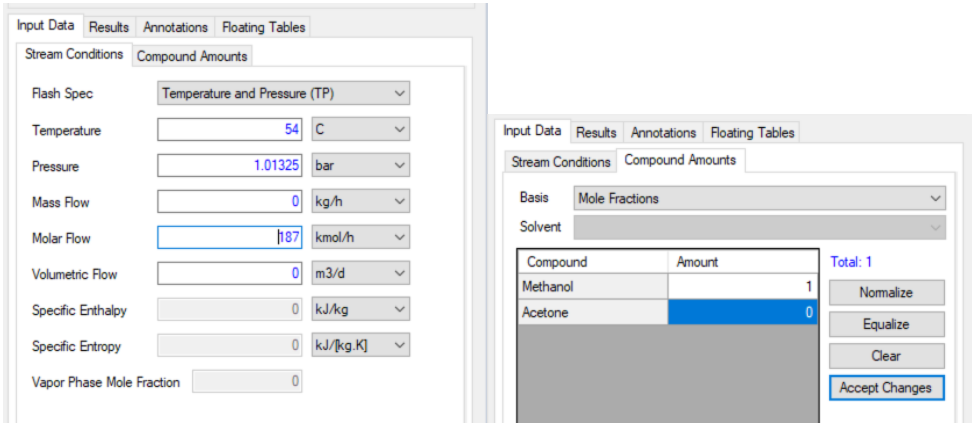



Figure 129: Methanol Recycle Stream configuration

15. Associate the Inside-Out Flash with the following streams: **MSTR-001**, **Methanol Product**, **Acetone Product** and **Recycle (3)**. This will prevent PH Flash errors from happening during the flowsheet calculation.

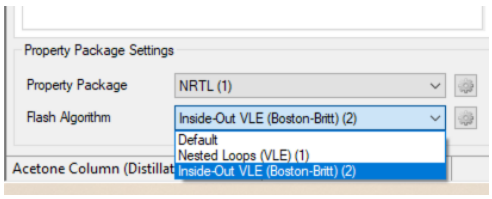



Figure 130: Associating the Inside-Out flash with the streams

16. Re-enable the solver (press F6) and calculate the flowsheet (press F5). Wait for the recycle to converge.

17. After the flowsheet solves, insert a new Property Table:

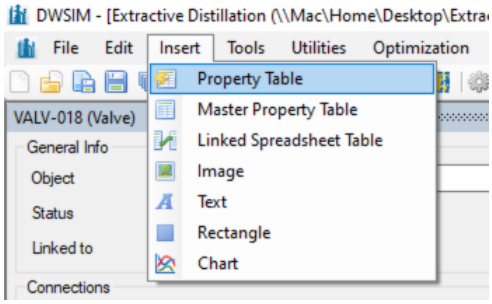



Figure 131: Inserting a Property Table

18. Double-click on the inserted table, search for the column energy streams and select Energy Flow for all of them, so these values can be shown on the Property Table.

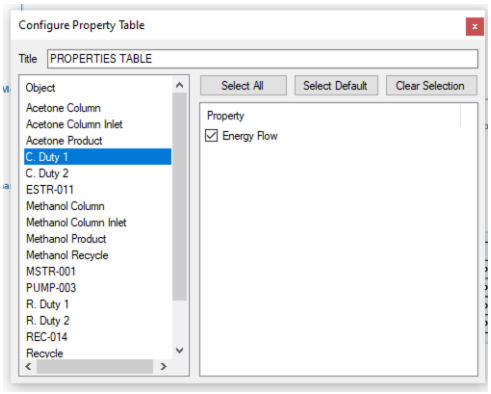



Figure 132: Setting up a Property Table

19. Compare the results obtained with the duties specified in the original problem.

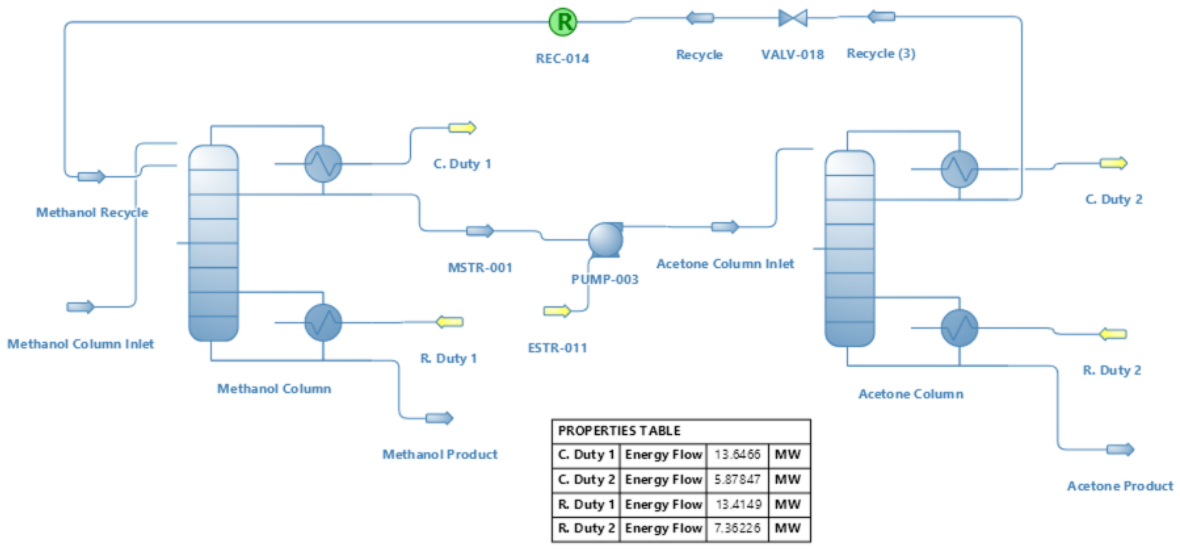



Figure 133: Final results

## 28.4. DWSIM Model (Cross-Platform UI)

1. Create a New Simulation. Close the Simulation Wizard.

> ***Remember to Save your simulation at the end of each step.***

2. Go to **Setup** > **General Settings**, **Interface** and disable the **Call Solver on Editor Property Update** option to avoid unnecessary calculations during the model building.

3. Go to **Setup** > **Compounds**, and select Methanol and Acetone to add these compounds to the simulation.

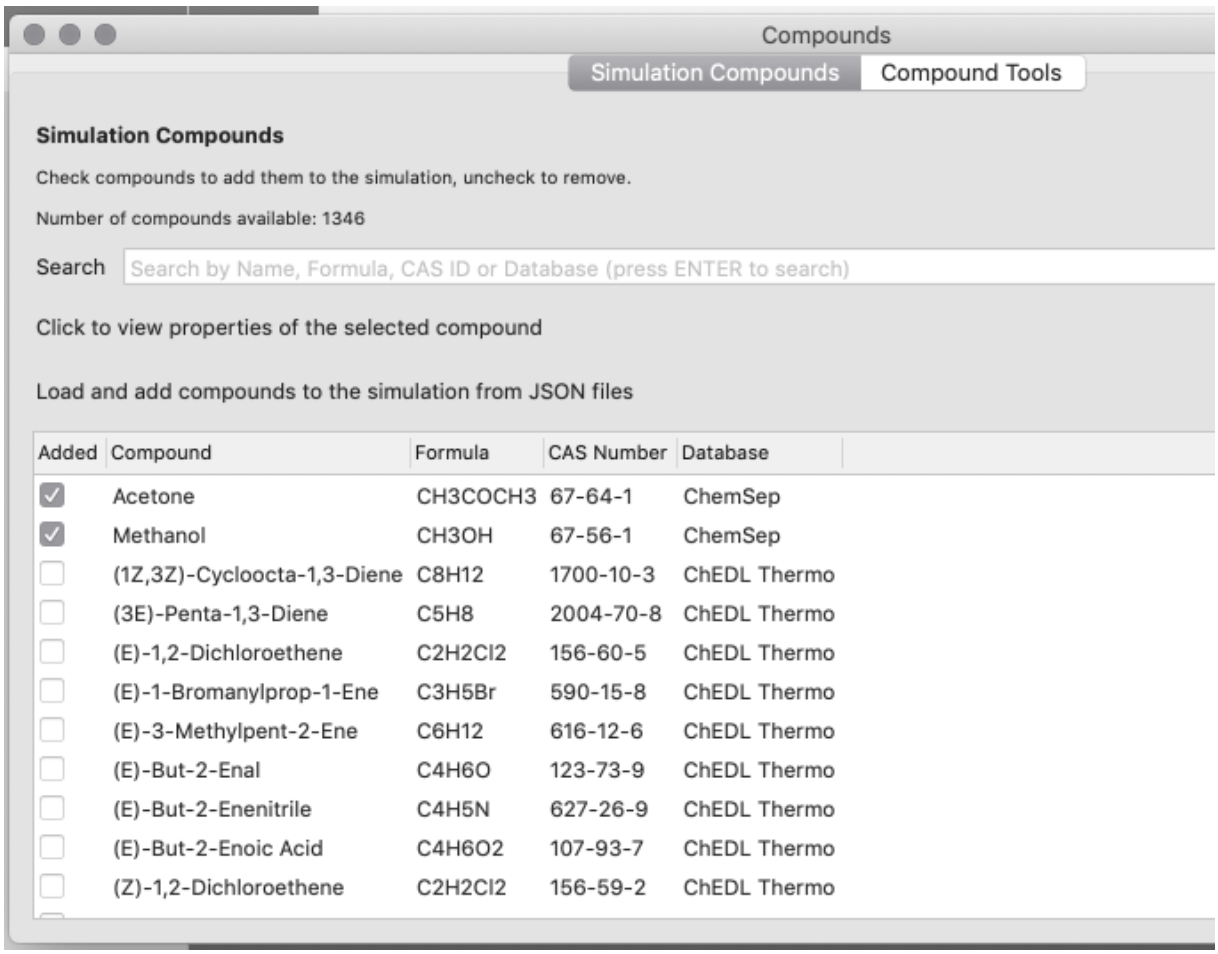



Figure 134: Compound Selection

4. Go to **Setup > Basis > Thermodynamics** tab, add a copy of the **NRTL** Property Package. Add an instance of the **Inside-Out VLE** Flash Algorithm too.

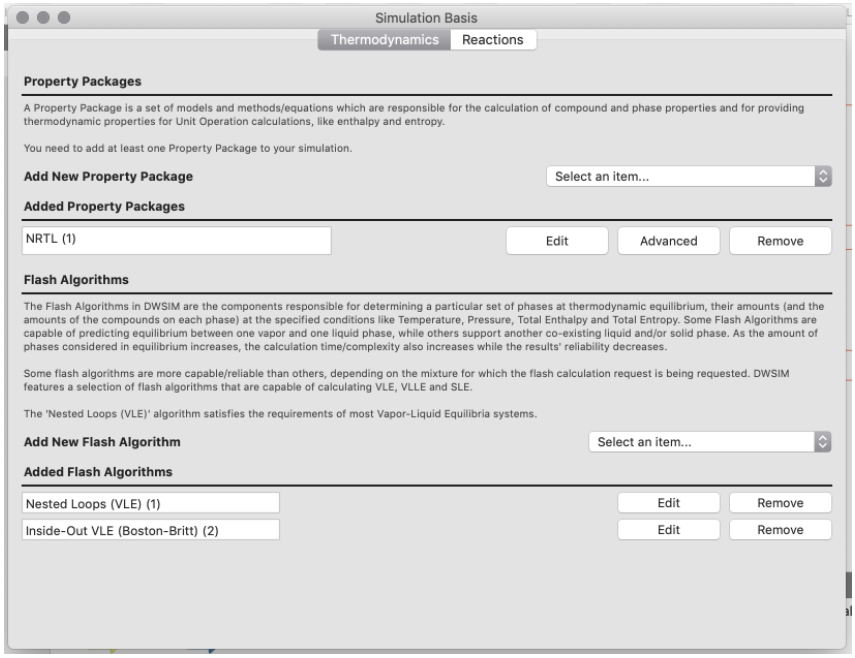



Figure 135: Property Package and Flash Algorithm Selection

5. Check if the NRTL Interaction Parameters are all set (click on **Edit** on the Added Property Packages section).

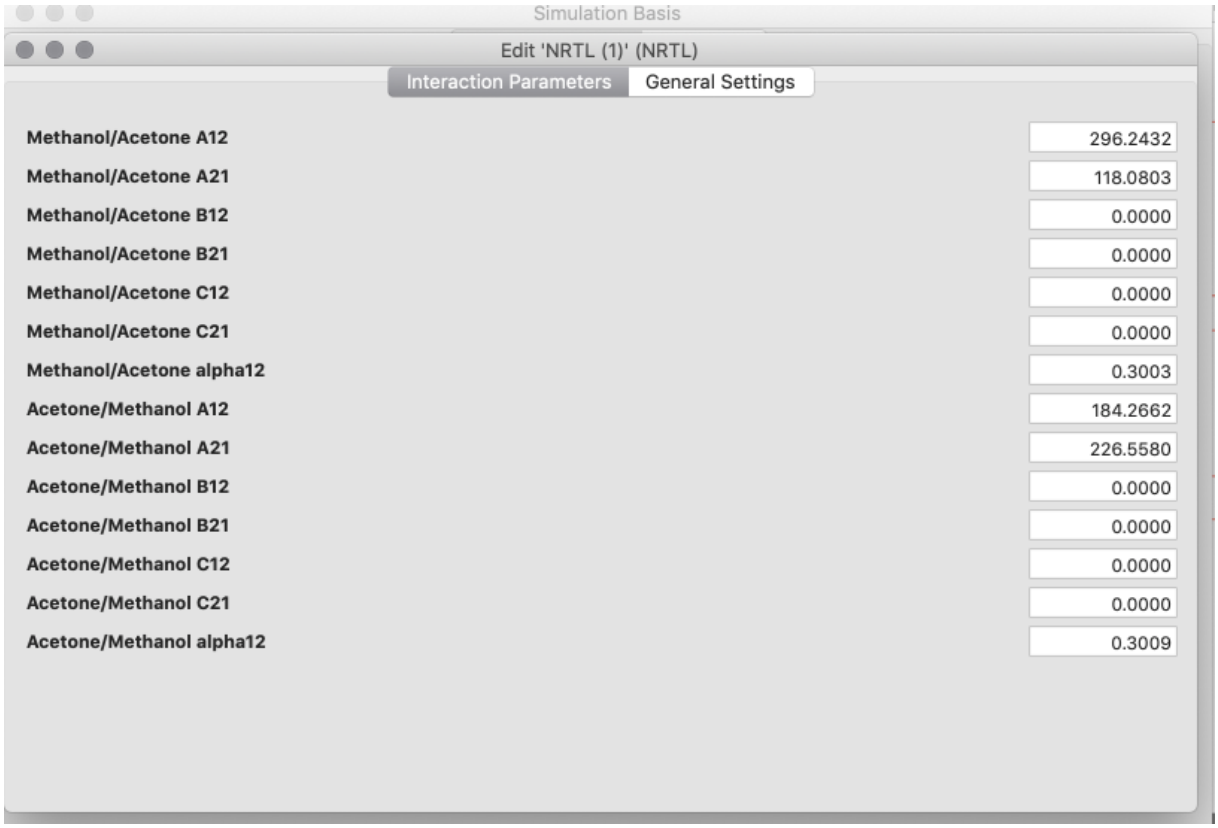



Figure 136: NRTL Interaction Parameters for Methanol/Acetone

6. Go to the **Setup > Flowsheet Settings > System of Units** section and create a new System of

Units, with the following setup:

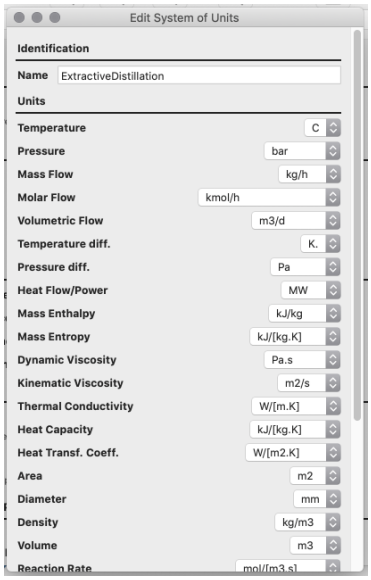



Figure 137: New System of Units

7. After creating this Units Set, select it on the System of Units combobox.

8. Add the objects to the flowsheet (streams, pump, valve, recycle and distillation columns) as depicted on the following figure, renaming them as required. You'll setup the connections between them later on this tutorial.

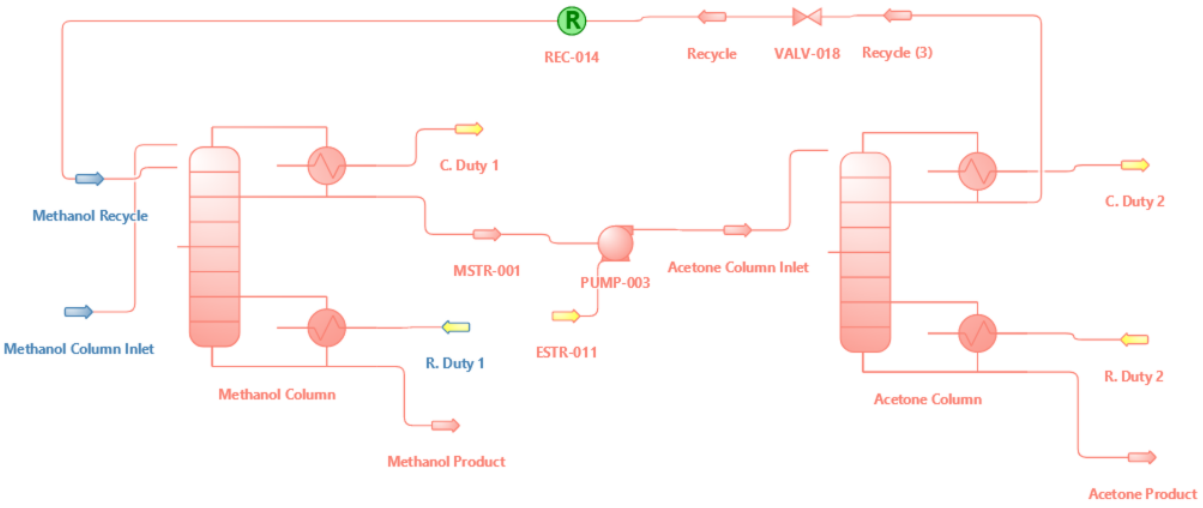



Figure 138: Process Flowsheet Diagram

9. Setup the columns and their connections as follows:

   a) Methanol Column (52 stages):

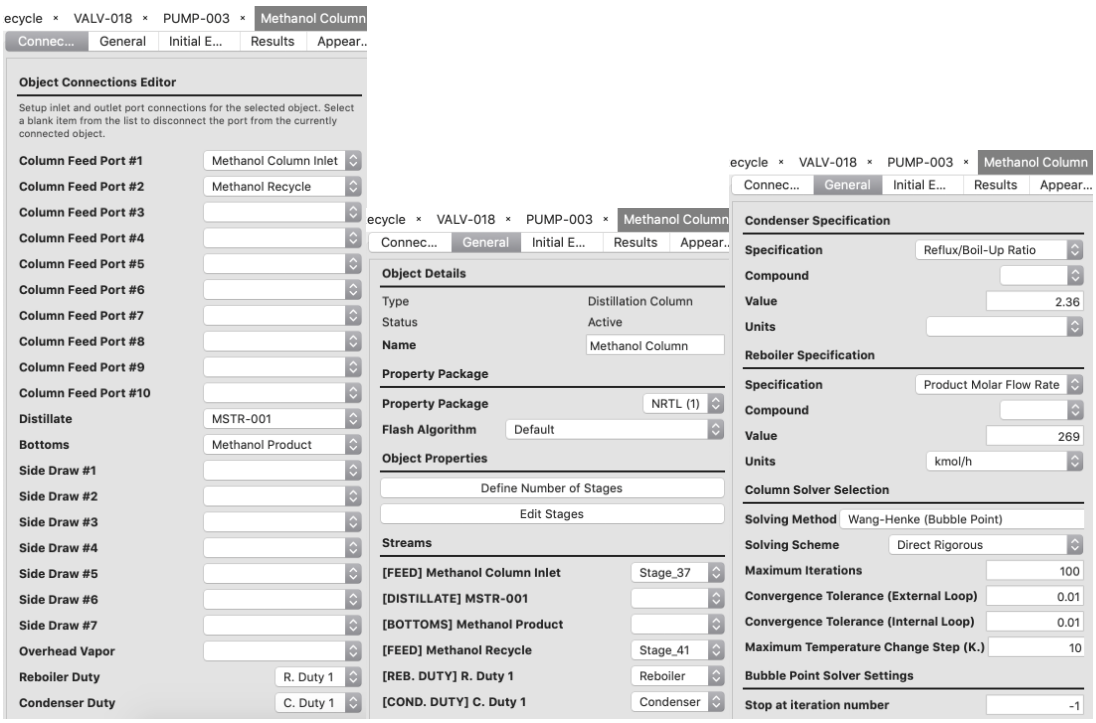



Figure 139: Methanol Column configuration

b) Acetone Column (61 stages):

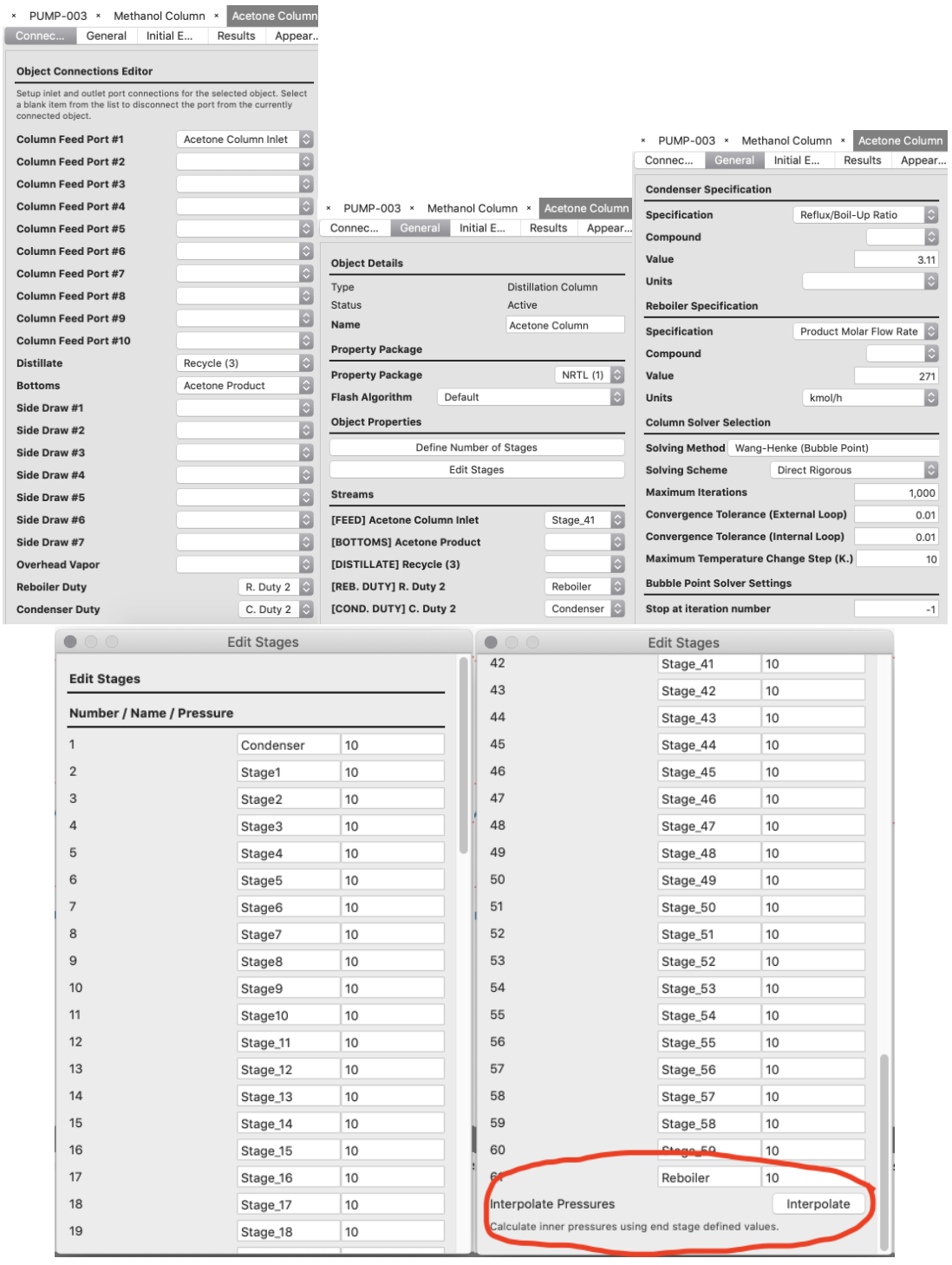



Figure 140: Acetone Column configuration

10. Enter initial estimates for the temperature profile of the Acetone Column, and check the **Override Temperature Estimates** checkbox so DWSIM can use them. Values must be tab separated, inserted as in the picture below:

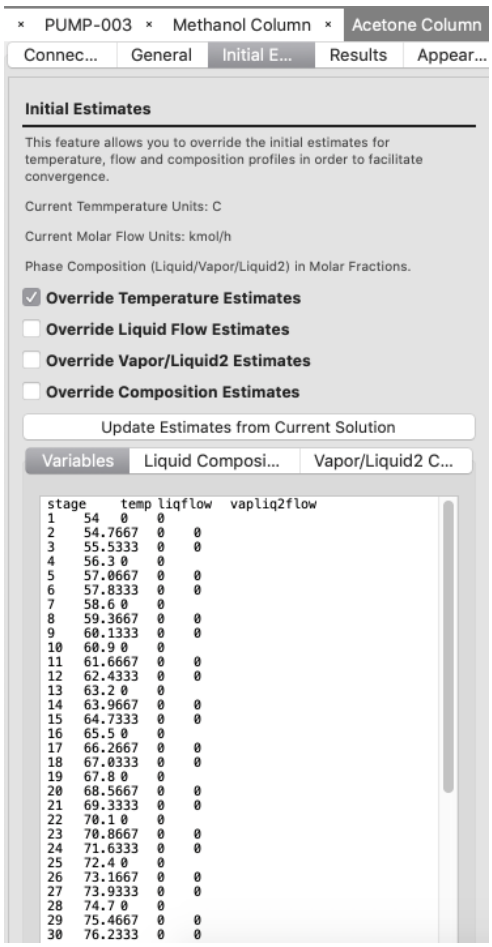



Figure 141: Acetone Column initial estimates for temperature profile

11. After the columns are correctly configured and connected to their associated streams, setup the pump, valve and recycle connections using their Editor Panels.

12. Setup the pump and valve properties as follows:

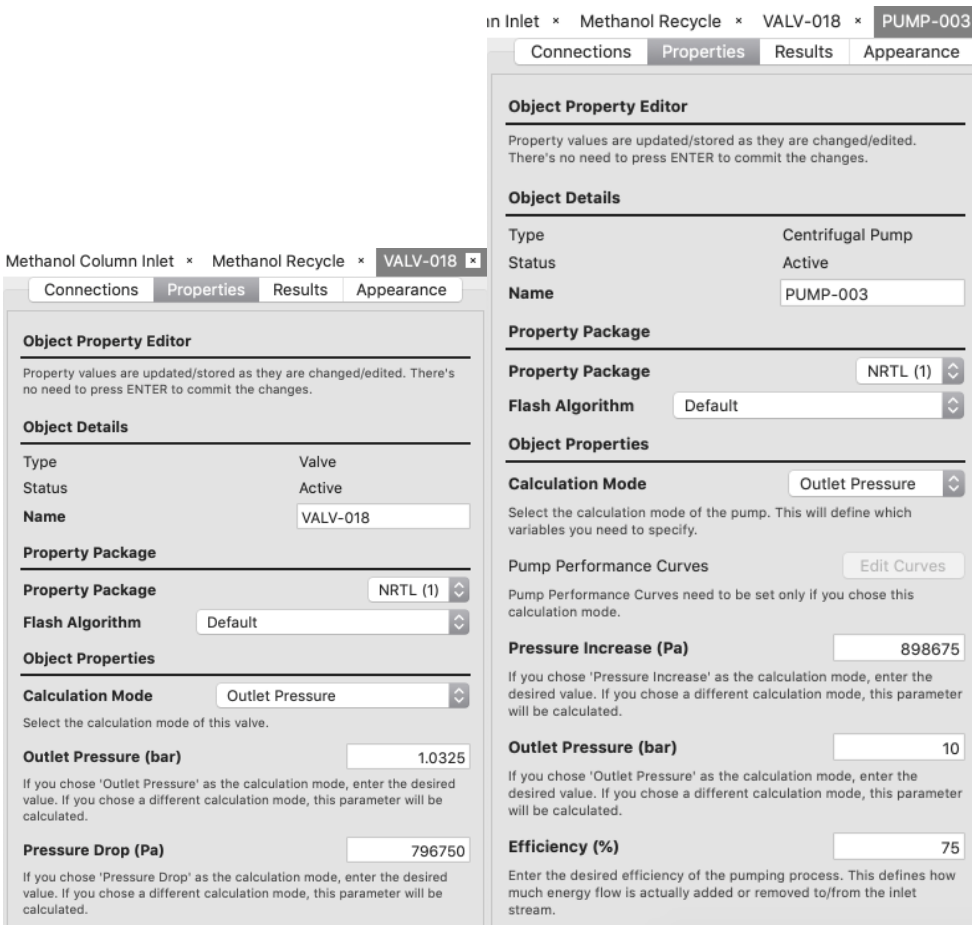



Figure 142: Pump and Valve properties

13. Configure the Methanol Inlet Stream as follows:

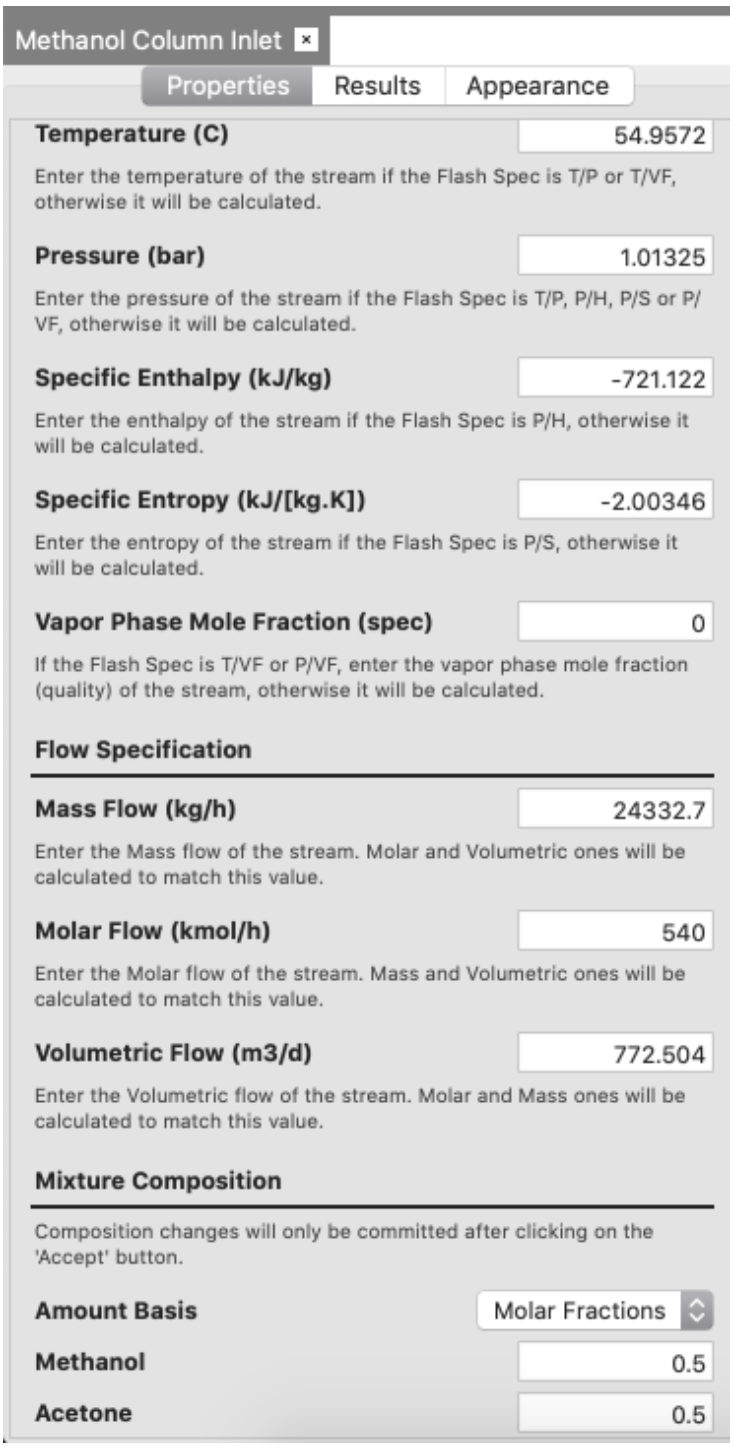



Figure 143: Methanol Inlet Stream configuration

14. Configure the Methanol Recycle Stream (initial estimates for the recycle stream) as follows:

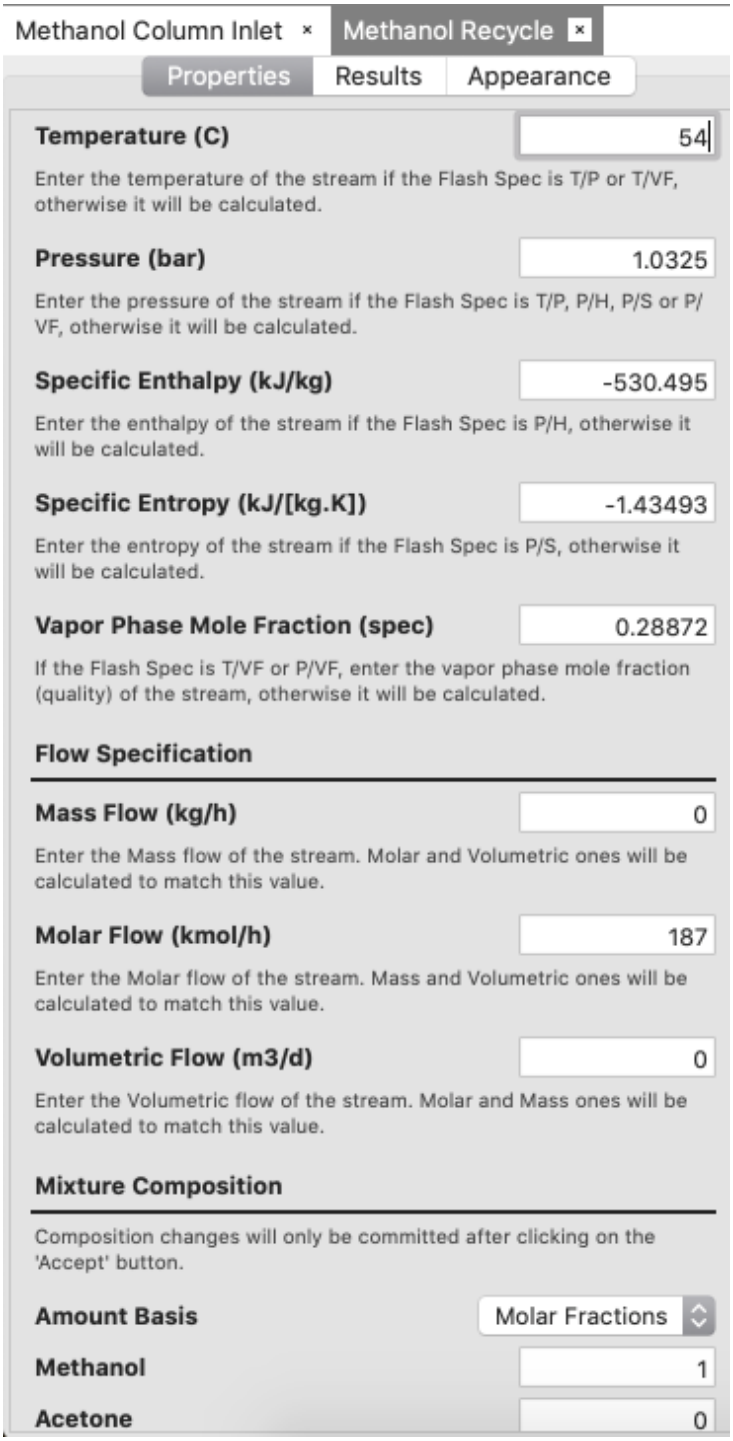



Figure 144: Methanol Recycle Stream configuration

15. Associate the Inside-Out Flash with the following streams: **MSTR-001**, **Methanol Product**, **Acetone Product** and **Recycle (3)**. This will prevent PH Flash errors from happening during the flowsheet calculation.

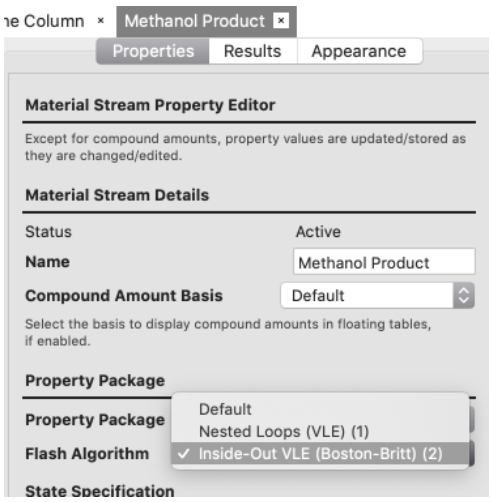



Figure 145: Associating the Inside-Out flash with the streams

16. Calculate the flowsheet (press F5). Wait for the recycle to converge.

17. After the flowsheet solves, insert a new Property Table:

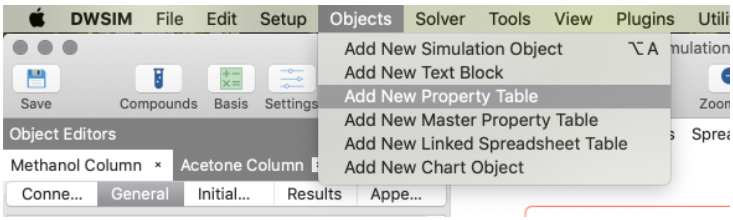



Figure 146: Inserting a Property Table

18. Double-click on the inserted table, search for the column energy streams and select Energy Flow for all of them, so these values can be shown on the Property Table.

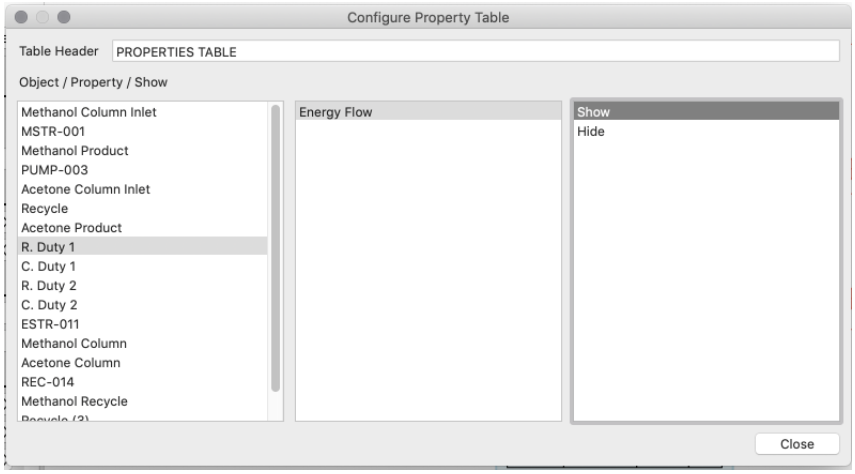



Figure 147: Setting up a Property Table

19. Compare the results obtained with the duties specified in the original problem.

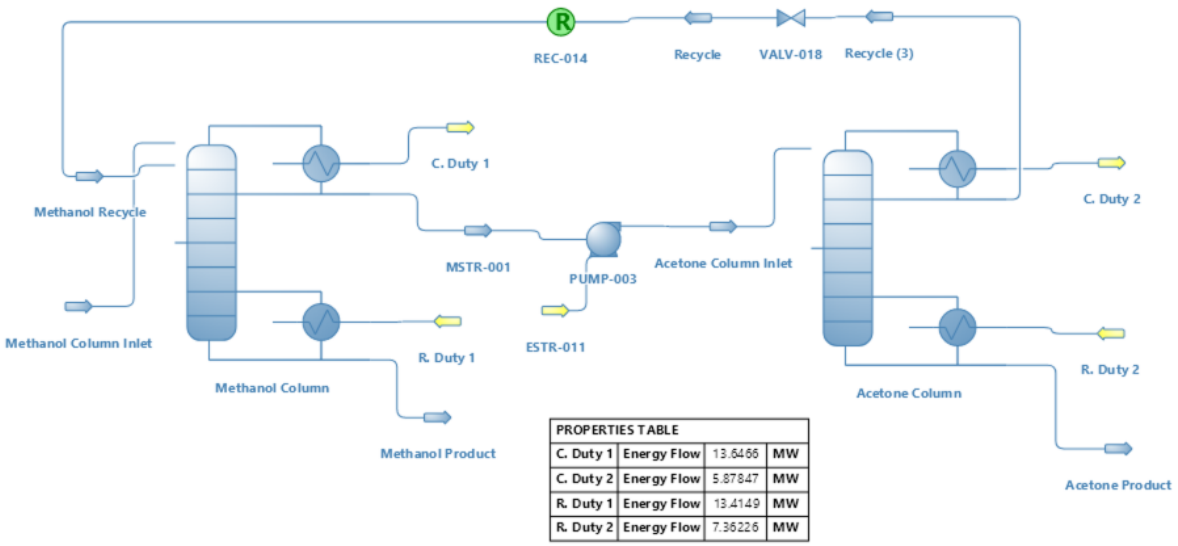



Figure 148: Final results

# 29. Flowsheet Control with Python Scripts

## 29.1. Introduction

Let us study the effect of the pressure on the temperature profile of the Acetone Column created on the previous tutorial. We will use the **IronPython Scripting**, **Spreadsheet** and **Charts** features available in DWSIM to generate, organize and analyze the results.

## 29.2. DWSIM Model (Classic UI)

1. Save the previous simulation with a different file name and remove everything from the flowsheet except the objects depicted on the following picture:

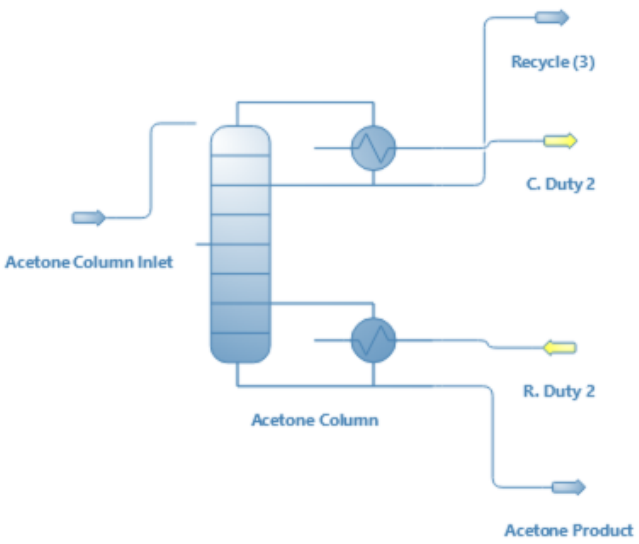



Figure 149: Process Flowsheet Diagram

2. Go to the **Script Manager** and enter the following script:

```python
# Get Acetone Column object reference from the Flowsheet

column = Flowsheet.GetFlowsheetSimulationObject("Acetone Column")

# define the list of column pressures in bar

Plevels = [8, 9, 10, 11, 12]

# setup spreadsheet table headers

Spreadsheet.Worksheets[0].Cells["A1"].Data = "Stage"
Spreadsheet.Worksheets[0].Cells["B1"].Data = "P = 8 bar"
Spreadsheet.Worksheets[0].Cells["C1"].Data = "P = 9 bar"
Spreadsheet.Worksheets[0].Cells["D1"].Data = "P = 10 bar"
Spreadsheet.Worksheets[0].Cells["E1"].Data = "P = 11 bar"
Spreadsheet.Worksheets[0].Cells["F1"].Data = "P = 12 bar"

# add column of stage numbers

j = 1

for stage in column.Stages:

    Spreadsheet.Worksheets[0].Cells[j, 0].Data = j

    j += 1

# loop through the pressure values, set them and run the simulation, collecting the results

i = 1

for Pnew in Plevels:

    for stage in column.Stages:

        stage.P = Pnew * 100000 # set new stage pressures in Pa

    Flowsheet.SolveFlowsheet2() # request a flowsheet calculation

    j = 1

    for t in column.Tf: # column.Tf is the vector of final stage temperatures in K

        Spreadsheet.Worksheets[0].Cells[j, i].Data = t # write the temperature values in the corresponding column

        j += 1

    i += 1
```

Figure 150: Python Script

3. Run the script asynchronously (this prevents DWSIM from freezing until the calculation finishes):


Figure 151: Run Python Script (Async)

4. Go to the **Spreadsheet**, select the entire data range, click with the right mouse button and select **Create 2D XY Chart from Selection**.

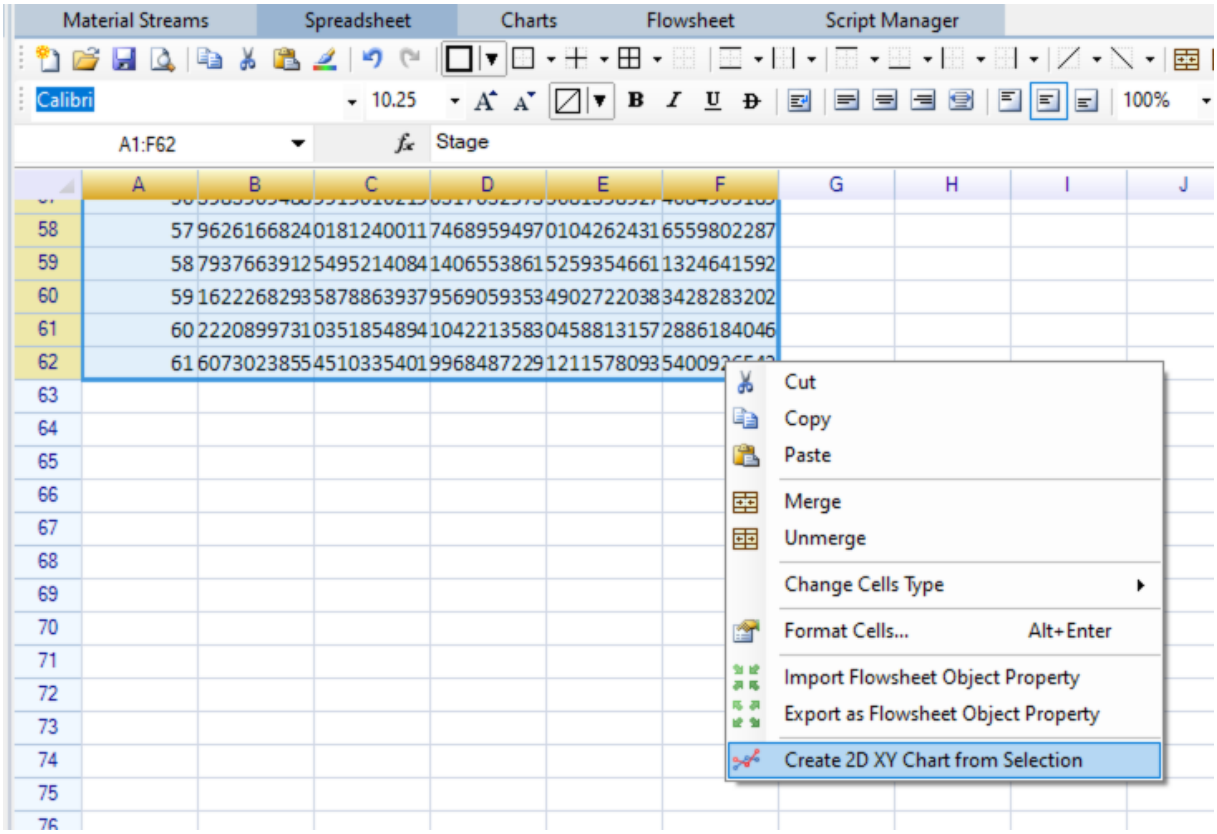



Figure 152: Create new chart from selected spreadsheet data range

5. View and configure the newly created chart as in the following picture:

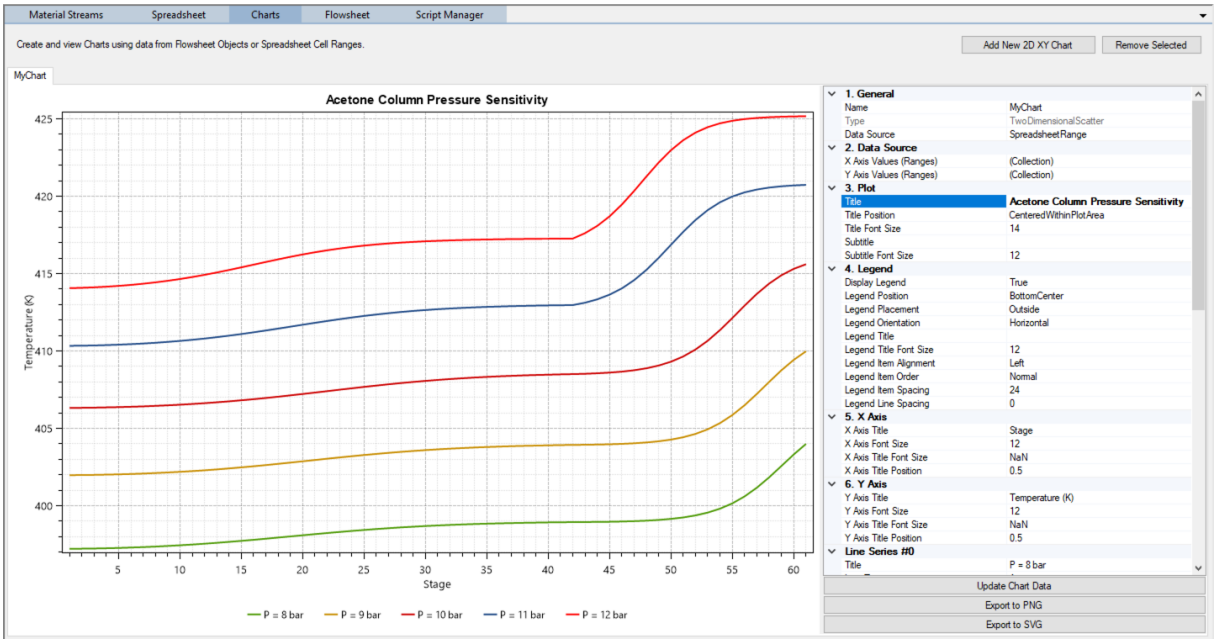



Figure 153: Pressure-Temperature dependence of the Acetone Column

6. Analyze the results obtained and discuss them with your colleagues.

## 29.3. DWSIM Model (Cross-Platform UI)

1. Save the previous simulation with a different file name and remove everything from the flowsheet except the objects depicted on the following picture:

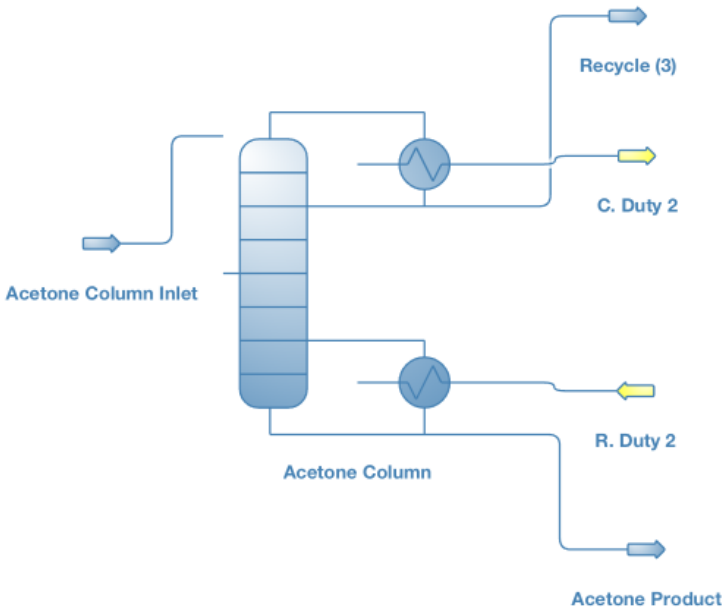



Figure 154: Process Flowsheet Diagram

2. Go to the **Script Manager,** create a New Script, select it on the list and enter the following code:

```python
# Get Acetone Column object reference from the Flowsheet

column = Flowsheet.GetFlowsheetSimulationObject("Acetone Column")

# define the list of column pressures in bar

Plevels = [8, 9, 10, 11, 12]

# setup spreadsheet table headers

Spreadsheet.Worksheets[0].Cells["A1"].Data = "Stage"
Spreadsheet.Worksheets[0].Cells["B1"].Data = "P = 8 bar"
Spreadsheet.Worksheets[0].Cells["C1"].Data = "P = 9 bar"
Spreadsheet.Worksheets[0].Cells["D1"].Data = "P = 10 bar"
Spreadsheet.Worksheets[0].Cells["E1"].Data = "P = 11 bar"
Spreadsheet.Worksheets[0].Cells["F1"].Data = "P = 12 bar"

# add column of stage numbers

j = 1

for stage in column.Stages:

    Spreadsheet.Worksheets[0].Cells[j, 0].Data = j

    j += 1

# loop through the pressure values, set them and run the simulation, collecting the results

i = 1

for Pnew in Plevels:

    for stage in column.Stages:

        stage.P = Pnew * 100000 # set new stage pressures in Pa

    Flowsheet.SolveFlowsheet2() # request a flowsheet calculation

    j = 1

    for t in column.Tf: # column.Tf is the vector of final stage temperatures in K

        Spreadsheet.Worksheets[0].Cells[j, i].Data = t # write the temperature values in the corresponding column

        j += 1

    i += 1
```

Figure 155: Python Script

3. Run the script asynchronously (this prevents DWSIM from freezing until the calculation finishes):

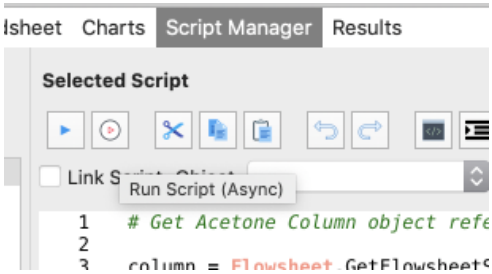



Figure 156: Run Python Script (Async)

4. Go to the **Spreadsheet**, select the entire data range, click with the right mouse button and select **Create Chart from Selected Range**.

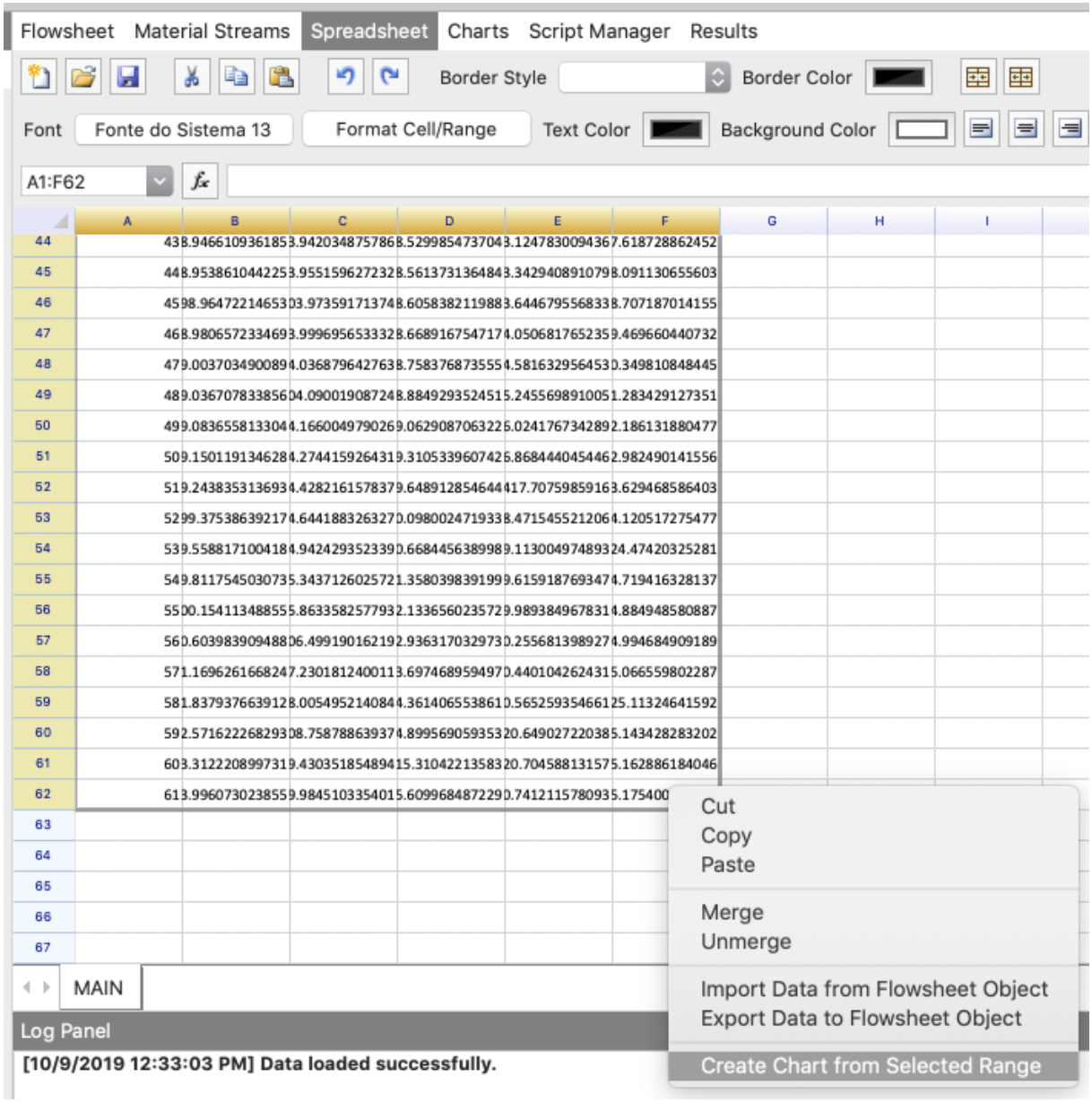



Figure 157: Create new chart from selected spreadsheet data range

5. View and configure the newly created chart as in the following picture:

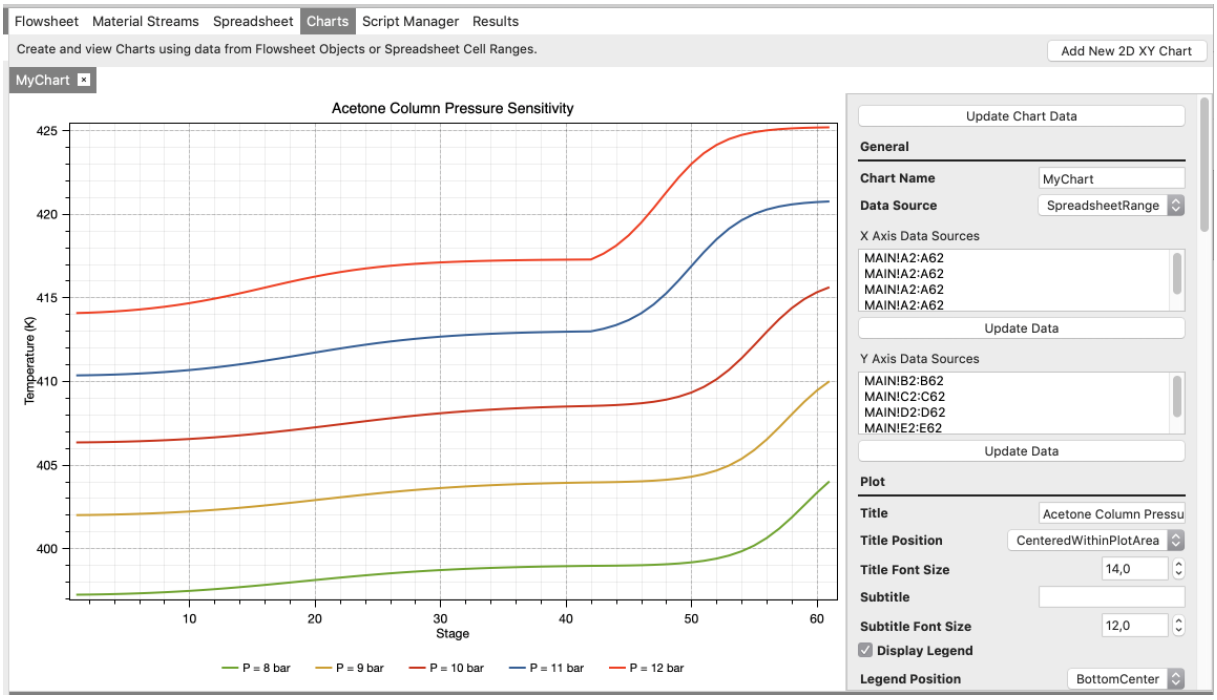



Figure 158: Pressure-Temperature dependence of the Acetone Column

6. Analyze the results obtained and discuss them with your colleagues.

# 30. Basic Dynamic Simulation Tutorial

## 30.1. Introduction

In this tutorial, we will learn how to do a dynamic simulation of a water storage tank, adding a PID Controller to keep the liquid level inside the tank around a desired value.

## 30.2. DWSIM Model (Classic UI)

Create and configure a new simulation. Add Water as the only compound and use the Steam Tables Property Package.

### 30.2.1. Model Building

1. Build your model as in the following picture:

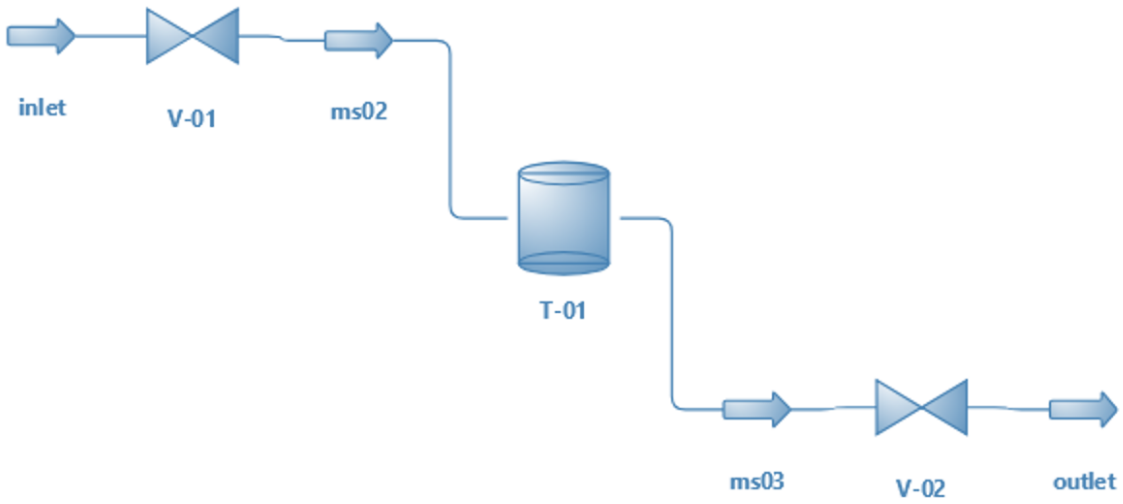



Figure 159: Water Tank model

2. Enable/Activate **Dynamic Mode**.

3. Set the inlet stream Pressure to 130000 Pa and Mass Flow to 10 kg/s. Set its Dynamic Spec to 'Flow'.

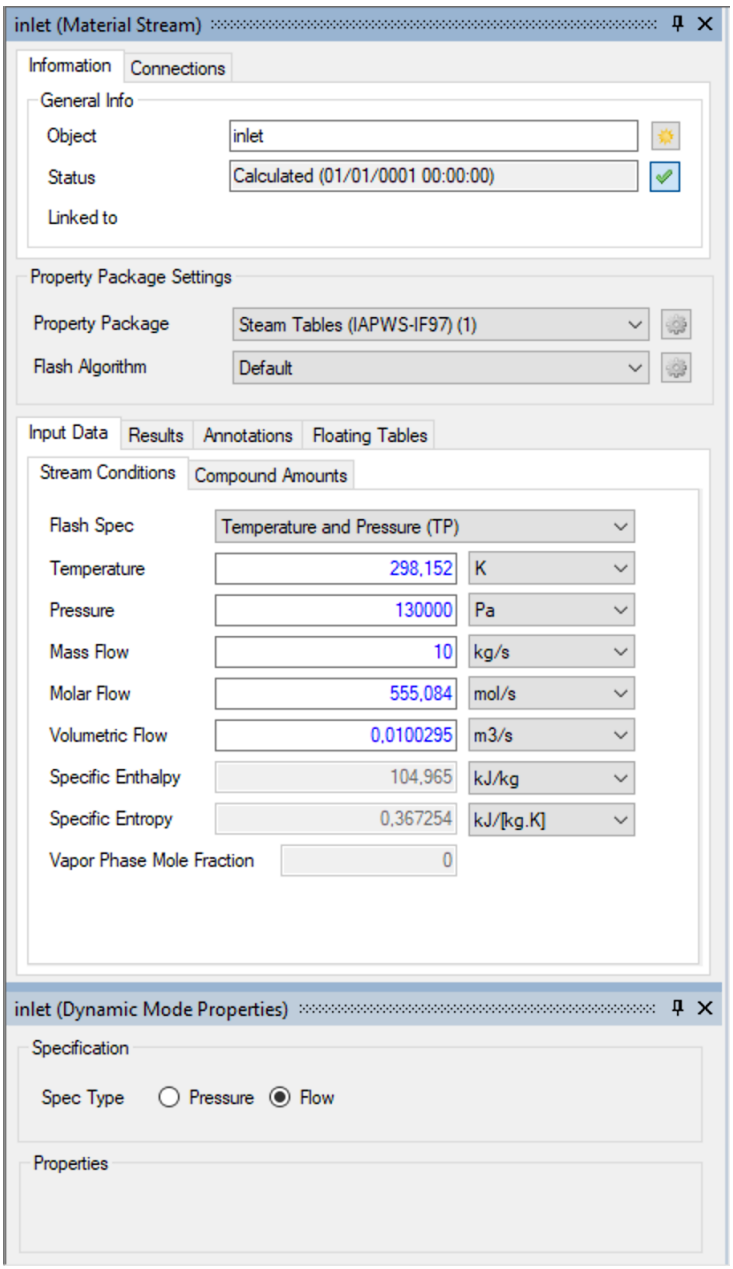



Figure 160: Inlet stream properties.

4. Set V-01 calculation type to Liquid Kv, Kvmax to 100, check the Opening vs Kv/Kvmax box and set the valve opening to 50%.

5. Set T-01 volume to 2 m3 and available liquid height to 2 m. Set the "Reset Content" property to 1.

6. Set V-02 calculation type to Liquid Kv, Kvmax to 400, check the Opening vs Kv/Kvmax box and set the valve opening to 50%.

7. Set the outlet stream Pressure to 101325 Pa. Set its Dynamic Spec to 'Pressure'.

With the above settings, the flow rate of water entering the tank will be fixed at 10 kg/s. The dynamic model for the Tank considers the liquid height contribution (static pressure) for the pressure of the tank's outlet stream. Since V-02's outlet stream pressure is fixed, the actual outlet flow will be calculated by the

valve using the current opening and the connected stream pressures. As a result, the liquid level inside the tank will vary according to the difference between inlet and outlet flow rates.

### 30.2.2. Dynamic Simulation

1. Add a Level Gauge and associate it with the Tank's Liquid Level property. Set its maximum value to 3 m.
2. Save the current flowsheet state as **NewState1**.
3. Go to the **Dynamics Manager** and create a new Integrator (Int1) with Integration Step equal to 5 seconds and Duration equal to 10 minutes. Add the Tank's liquid level and the openings of the two valves as monitored variables for this integrator.
4. Create a new Schedule (Sch1) and associate the previously created Integrator and Flowsheet State with it.
5. Open the Integrator Controls Panel and run the dynamic simulation. The liquid level on the tank should stabilize around 0.33 meters after 7 minutes.
6. On the Integrator Controls Panel, click on the **View Results** button. On the created Spreadsheet, select the A and B columns and create a new chart from the selection. View the generated chart.

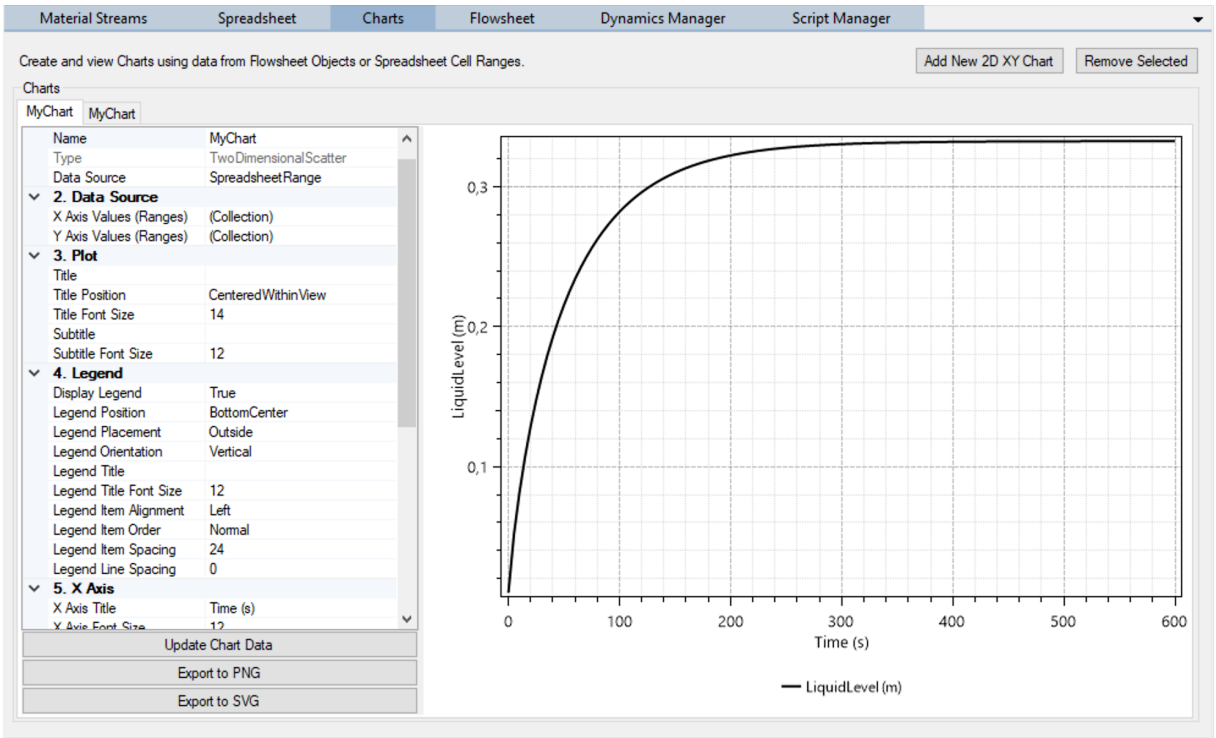



Figure 161: Liquid Level versus time.

### 30.2.3. Adding a PID Controller

1. Add a PID Controller to the flowsheet and set the V-02's opening as the manipulated variable and the Tank's liquid level as the controlled one. The set-point should be equal to 1 (m). Set Kp to 100

and enable Reverse Acting.

2. Add a new Chart to the flowsheet and associate it with the PID's History item.

3. Open the Integrator Controls Panel and run the dynamic simulation. The liquid level on the tank should stabilize around 1.2 meters after 3 minutes.

4. Now set the controller's Ki to 10 and rerun the dynamic simulation. The liquid level on the tank should stabilize around 0.95 meters.

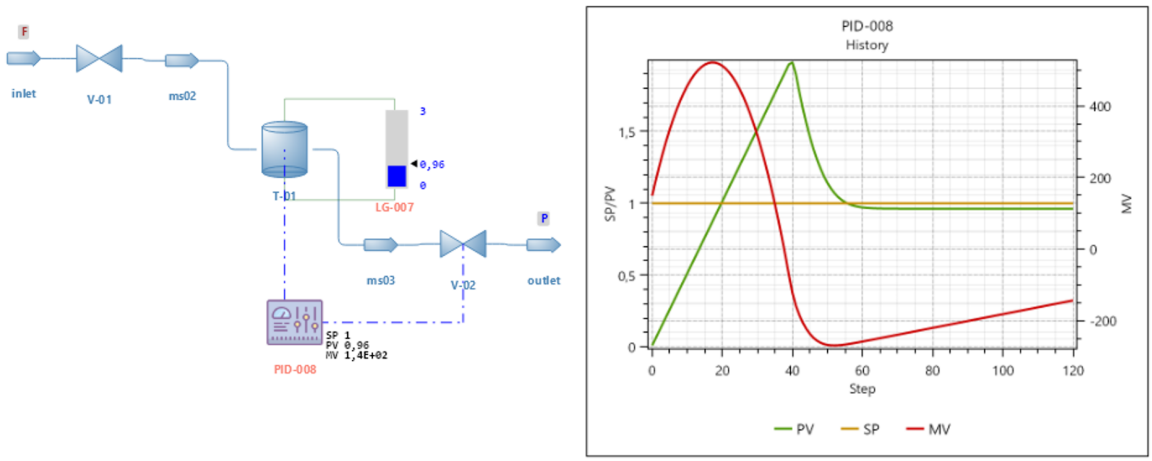



Figure 162: Liquid Level versus time with controller engaged.

### 30.2.4. PID Controller Tuning

1. Open the PID Controller Tuning Tool, select the controller added to the simulation and run the tuning.

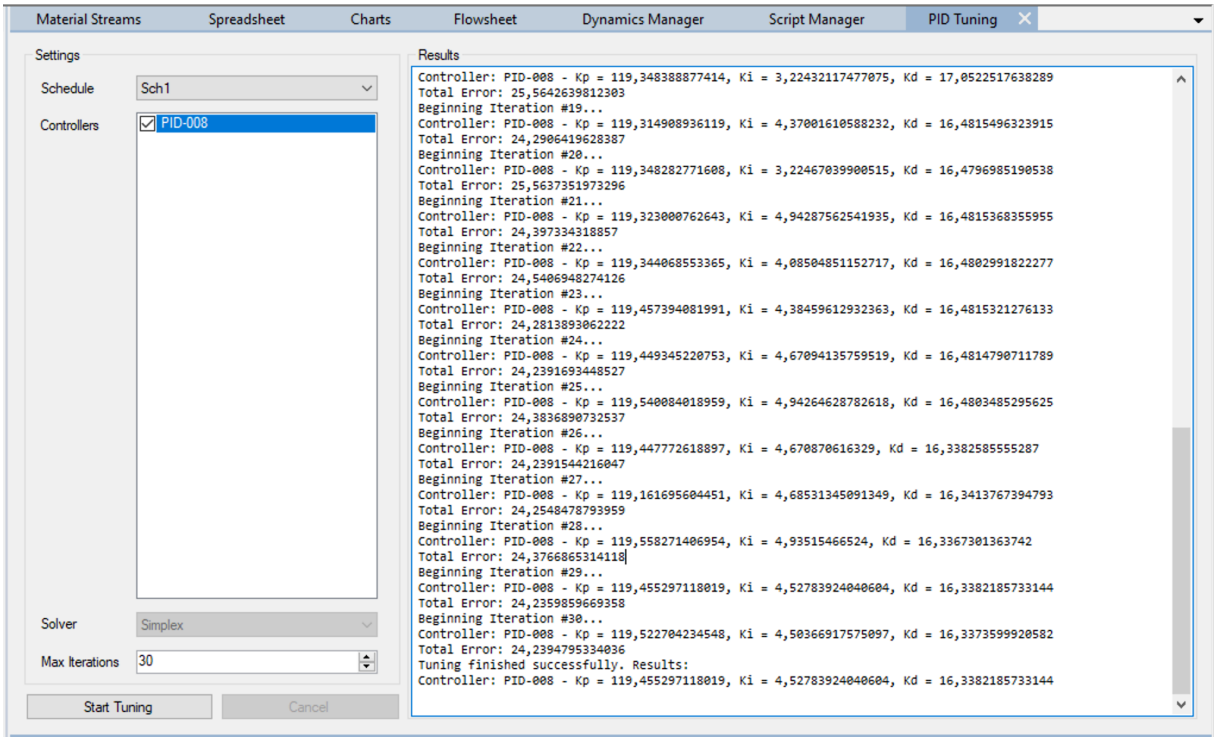



Figure 163: PID Controller Tuning Tool.

2. Open the Integrator Controls Panel and run the dynamic simulation. The liquid level on the tank should stabilize around 1 meter after 5 minutes. Notice that there is still a very high overshoot on the liquid level, even after the PID tuning. Perhaps you can try tuning it again with different initial values for Kp, Ki and Kd and/or increase the number of optimizer runs.

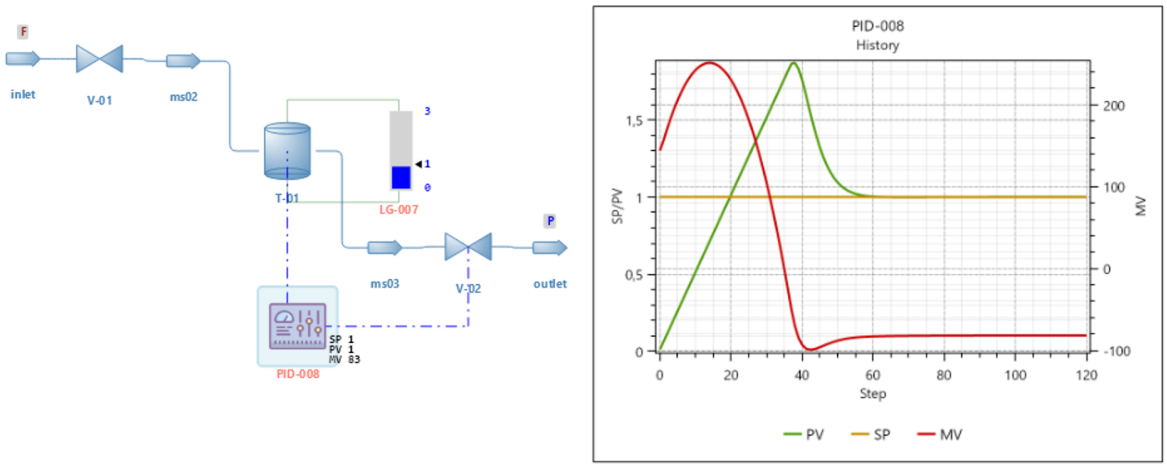



Figure 164: Liquid Level versus time with controller engaged.

## 30.3. Real-Time Mode

1. Change the flowsheet to **Control Panel Mode**. It should become dark and read-only, i.e. you cannot drag and/or add new objects.

2. Run the dynamic simulation in real-time mode by clicking on the 'clock' button on the Integrator Controls Panel.

3. After some time, click on the PID Controller and set the SP value to 1.7 (m). Watch how the system reacts to this change.

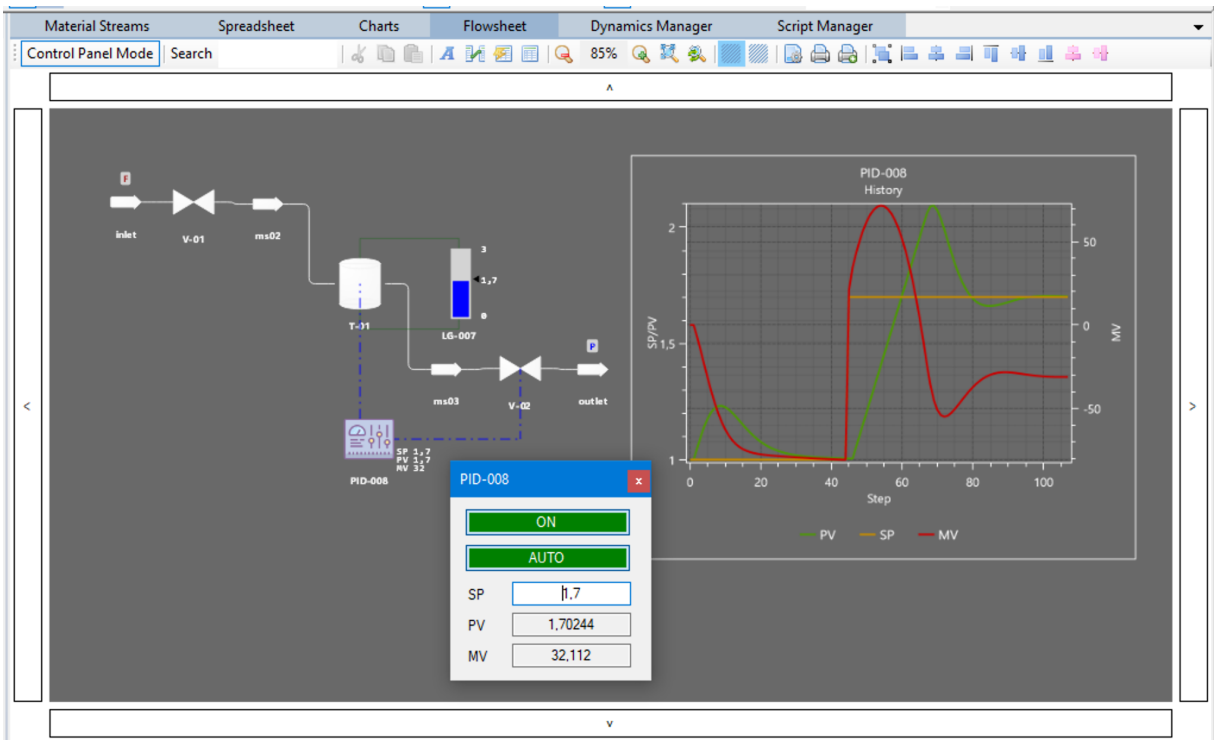



Figure 165: Control Panel mode with changes to PID parameters.

4. Remember that, after each integrator run, you can click on **View Results** and inspect the values of the monitored variables on that ru

# Part VIII.

# Advanced Topics

## 31. Python Scripting

The scripting capability in DWSIM allows the user to execute additional tasks in the simulation through the use of Python scripts, which have access to all objects in the flowsheet, the solver and the simulation itself.

You can play with the objects in the flowsheet, do calculations using property packages, get data from external sources, run multiple cases while changing properties between runs and much more.

The script blocks can be associated with events in the flowsheet, i.e. they can run when a specific object is calculated or when an error occurs in the calculation. A script can also run when the simulation is opened, closed and saved.

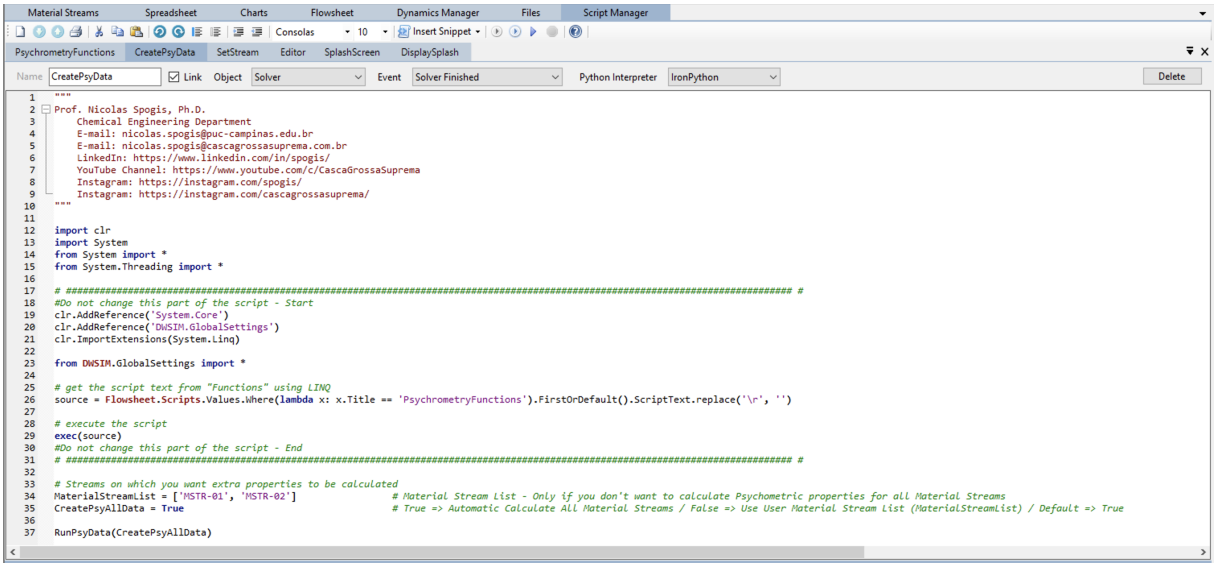



Figure 166: Python Script Manager (Classic UI)

### 31.1. Python Interpreters

DWSIM uses two Python interpreters: **IronPython** and **Python.NET**.

IronPython is a .NET implementation of the python scripting language, so there is no relation with an specific Python version - it is "embedded" in DWSIM and contains a Python Standard Library corresponding to Python v2.7, though it is always better to use .NET equivalent of the classes and functions, which are directly accessible from the scripts. For instance, the System.IO namespace, which deals with files, directories and paths on your system.

Python.NET links DWSIM to a Python installation on your machine. It runs your scripts using that external Python environment.

Unless you need to use a specific Python external module like numpy, scipy, tensorflow, matplotlib, etc, which is the main reason why the Python.NET interpreter exists in DWSIM, it is preferable to use

IronPython, which it is much faster and has much better integration with other DWSIM features.

## 31.2. IronPython Interactive Console (Classic UI)

The IronPython Interactive Console enables you to interact with the flowsheet in real-time, changing variables on the fly and requesting flowsheet calculations, among other features. The interactive console is specially useful when used in dynamic simulations, as it lets you change variables during an integrator run and observing the impact of such changes in other flowsheet variables.

**Intellisense** is available in the IronPython Interactive Console.


Figure 167: IronPython Interactive Console (Classic UI)

### 31.2.1. Available Functions

**solve()**

Requests a Flowsheet calculation.

**save()**

Saves the current flowsheet to its currently associated file.

**apihelp()**

Opens the API Help page in a new browser window.

#### 31.2.2. Changing object properties

When accessing flowsheet objects from the console, remove white spaces and underlines from the object name, i.e., to change the value of the temperature of a Material Stream named *'MSTR-01*', do the following:

```
MSTR01.SetTemperature(400)
```

The above will set the temperature of MSTR-01 to 400 K.

You can also set a stream variable with specific units, i.e.

```
MSTR01.SetPressure('10␣bar')
```

The above will set the pressure of MSTR-01 to 10 bar or 1000000 Pa.

For more Python/IronPython programming help, see **Section 40**.

# 32. Model Customization

## 32.1. Introduction

You can override various internal model calculation routines with custom python code using the Script Manager. As of DWSIM v5.6 Update 12, you can override calculation routines from Unit Operations on the flowsheet, Fugacity, Enthalpy and Entropy calculation routines from Property Packages added to the flowsheet and use custom Flash calculation routines through User-Defined Flash Algorithm instances added to the flowsheet.

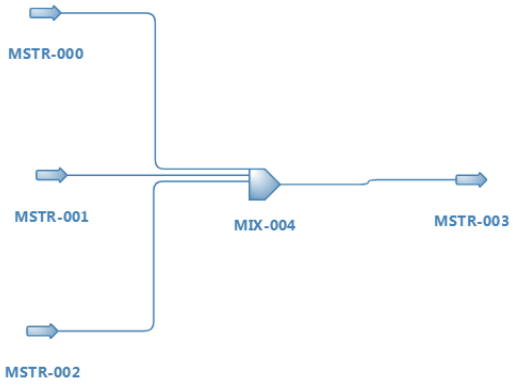
The custom functions must be defined and set only once before working with your simulation. For this reason it is better to associate your code with the Simulation Opened event after it is ready for production, so it can be automatically executed on every simulation file opening/loading.

If you choose to make any modifications to the code, don't forget to update it by running the script using the 'Run Script' button on the Script Manager.

## 32.2. Unit Operation / Material Stream Calculation Routine Override

### 32.2.1. Unit Operations

The following example shows how to override the calculation routine of a simple mixer. The Python code is a direct conversion from DWSIM's Mixer calculation routine, just for illustration purposes. It works with this sample flowsheet:




First, you must define a Python function which takes no arguments and does the calculation and sets properties of outlet material streams. This function doesn't need to return anything. Then you set the **OverrideCalculationRoutine** to true and the **CalculationRoutineOverride** property to your function's delegate (= name).

```
1  import clr
2  clr.AddReference('DWSIM.MathOps')
3  clr.AddReference('DWSIM.UnitOperations')
4  clr.AddReference('DWSIM.Interfaces')
5
6  from DWSIM import *
7  from DWSIM.Thermodynamics.Streams import *
8  from DWSIM.UnitOperations import *
9  from DWSIM.MathOps.MathEx import *
10 from System import *
11 from System.Collections.Generic import *
12
13 # gets the mixer object
14 mixer = Flowsheet.GetFlowsheetSimulationObject('MIX-004')
15
16 def CalcMixer():
17
18     ms = MaterialStream()
19
20     P = 0.0
21     W = 0.0
22     H = 0.0
23     i = 0
24     for cp in mixer.GraphicObject.InputConnectors:
25         if cp.IsAttached:
26             ms = Flowsheet.SimulationObjects[cp.AttachedConnector.AttachedFrom.Name]
27             ms.Validate()
28             if mixer.PressureCalculation == UnitOperations.Mixer.PressureBehavior.Minimum:
29                 print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixer Mode: Outlet Pressure = Minimum Inlet
                       Pressure'
30                 if ms.Phases[0].Properties.pressure < P:
31                     P = ms.Phases[0].Properties.pressure
32                 elif P == 0.0:
33                     P = ms.Phases[0].Properties.pressure
34             elif mixer.PressureCalculation == UnitOperations.Mixer.PressureBehavior.Maximum:
35                 print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixer Mode: Outlet Pressure = Maximum Inlet
                       Pressure'
36                 if ms.Phases[0].Properties.pressure > P:
```

```
37                  P = ms.Phases[0].Properties.pressure
38              elif P == 0:
39                  P = ms.Phases[0].Properties.pressure
40          else:
41              print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixer Mode: Outlet Pressure = Inlet Average'
42              P += ms.Phases[0].Properties.pressure
43              i += 1
44          We = ms.Phases[0].Properties.massflow
45          W += We
46          if not Double.IsNaN(ms.Phases[0].Properties.enthalpy): H += We * ms.Phases[0].Properties.
                enthalpy
47
48  if W != 0.0:
49      Hs = H / W
50  else:
51      Hs = 0.0
52
53  if mixer.PressureCalculation == UnitOperations.Mixer.PressureBehavior.Average: P = P / (i - 1)
54
55  print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixture Pressure (Pa): ' + str(P)
56  print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixture Mass Flow (kg/s): ' + str(W)
57  print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixture Enthalpy (kJ/kg): ' + str(Hs)
58
59  T = 0.0
60  n = Flowsheet.SelectedCompounds.Count
61  Vw = Dictionary[String, Double]()
62  for cp in mixer.GraphicObject.InputConnectors:
63      if cp.IsAttached:
64          ms = Flowsheet.SimulationObjects[cp.AttachedConnector.AttachedFrom.Name]
65          for comp in ms.Phases[0].Compounds.Values:
66              if not Vw.ContainsKey(comp.Name):
67                  Vw.Add(comp.Name, 0)
68              Vw[comp.Name] += comp.MassFraction * ms.Phases[0].Properties.massflow
69          if W != 0.0: T += ms.Phases[0].Properties.massflow / W * ms.Phases[0].Properties.temperature
70
71  if W == 0.0: T = 273.15
72  print '[' + mixer.GraphicObject.Tag + '] ' + 'Mixture Temperature Estimate (K): ' + str(T)
73
74  omstr = Flowsheet.SimulationObjects[mixer.GraphicObject.OutputConnectors[0].AttachedConnector.
        AttachedTo.Name]
75  omstr.Clear()
76  omstr.ClearAllProps()
77  if W != 0.0: omstr.Phases[0].Properties.enthalpy = Hs
78  omstr.Phases[0].Properties.pressure = P
79  omstr.Phases[0].Properties.massflow = W
80  omstr.Phases[0].Properties.molarfraction = 1
81  omstr.Phases[0].Properties.massfraction = 1
82  for comp in omstr.Phases[0].Compounds.Values:
83      if W != 0.0: comp.MassFraction = Vw[comp.Name] / W
84  mass_div_mm = 0.0
85  for sub1 in omstr.Phases[0].Compounds.Values:
86      mass_div_mm += sub1.MassFraction / sub1.ConstantProperties.Molar_Weight
87  for sub1 in omstr.Phases[0].Compounds.Values:
88      if W != 0.0:
89          sub1.MoleFraction = sub1.MassFraction / sub1.ConstantProperties.Molar_Weight / mass_div_mm
90      else:
91          sub1.MoleFraction = 0.0
92      print '[' + mixer.GraphicObject.Tag + '] ' + sub1.Name + ' outlet molar fraction: ' + str(sub1.
            MoleFraction)
93  omstr.Phases[0].Properties.temperature = T
94  omstr.SpecType = Interfaces.Enums.StreamSpec.Pressure_and_Enthalpy
```

```
95
96      print '[' + mixer.GraphicObject.Tag + '] ' + 'Outlet Stream variables set successfully.'
97
98      return None
99
100 mixer.OverrideCalculationRoutine = True
101 mixer.CalculationRoutineOverride = CalcMixer
```

After updating the script and running the simulation, you should get the following output:

| Type | Message |
|---|---|
| Message | Last run execution time: 0:00:00,340034 |
| Message | The flowsheet was calculated successfully. |
| Message | [MIX-004] Outlet Stream variables set successfully. |
| Message | [MIX-004] Propane outlet molar fraction: 0.428456587139 |
| Message | [MIX-004] Ethane outlet molar fraction: 0.286977541469 |
| Message | [MIX-004] Methane outlet molar fraction: 0.284565871392 |
| Message | [MIX-004] Mixture Temperature Estimate (K): 330.0 |
| Message | [MIX-004] Mixture Enthalpy (kJ/kg): -64.3536871804 |
| Message | [MIX-004] Mixture Mass Flow (kg/s): 5.0 |
| Message | [MIX-004] Mixture Pressure (Pa): 932175.0 |
| Message | [MIX-004] Mixer Mode: Outlet Pressure = Inlet Average |
| Message | [MIX-004] Mixer Mode: Outlet Pressure = Inlet Average |
| Message | [MIX-004] Mixer Mode: Outlet Pressure = Inlet Average |
| Message | |
| Message | The flowsheet is being calculated, please wait... |

### 32.2.2. Material Streams

Usually you don't need to override the calculation routines of material streams as they are very structured and involve a lot of internal checks and property updates.

## 32.3. Property Package Fugacity Coefficients / Enthalpy / Entropy Calculation Routine Override

### 32.3.1. Fugacity Coefficients

The following code overrides the fugacity coefficient calculation routines (called by the K-values calculation routine which is, by its turn, called by the flash algorithms to do the equilibrium calculations) from a Peng-Robinson Property Package instance added to the simulation. The code reproduces the internal DWSIM's PR EOS routines, just for illustration purposes.

```
1 import clr
2 clr.AddReference('DWSIM.MathOps')
3
4 from DWSIM import *
5 from DWSIM.MathOps.MathEx import *
6 from DWSIM.Thermodynamics.PropertyPackages import *
```

```
7  from System import *
8
9  # gets the first Property Package added to the the simulation
10
11 pp = Flowsheet.PropertyPackagesArray[0]
12
13 def calcroots(coeffs):
14
15     # auxiliary function
16     # calculates the roots of of a cubic polynomial and returns only the real ones
17
18     a = coeffs[0]
19     b = coeffs[1]
20     c = coeffs[2]
21     d = coeffs[3]
22
23     # uses DWSIM's internal 'CalcRoots' function to calculate roots
24     # https://github.com/DanWBR/dwsim5/blob/windows/DWSIM.Math/MATRIX2.vb#L29
25
26     res = PolySolve.CalcRoots(a, b, c, d)
27
28     roots = [[0] * 2 for i in range(3)]
29
30     roots[0][0] = res[0, 0]
31     roots[0][1] = res[0, 1]
32     roots[1][0] = res[1, 0]
33     roots[1][1] = res[1, 1]
34     roots[2][0] = res[2, 0]
35     roots[2][1] = res[2, 1]
36
37     # orders the roots
38
39     if roots[0][0] > roots[1][0]:
40         tv = roots[1][0]
41         roots[1][0] = roots[0][0]
42         roots[0][0] = tv
43         tv2 = roots[1][1]
44         roots[1][1] = roots[0][1]
45         roots[0][1] = tv2
46     if roots[0][0] > roots[2][0]:
47         tv = roots[2][0]
48         roots[2][0] = roots[0][0]
49         roots[0][0] = tv
50         tv2 = roots[2][1]
51         roots[2][1] = roots[0][1]
52         roots[0][1] = tv2
53     if roots[1][0] > roots[2][0]:
54         tv = roots[2][0]
55         roots[2][0] = roots[1][0]
56         roots[1][0] = tv
57         tv2 = roots[2][1]
58         roots[2][1] = roots[1][1]
59         roots[1][1] = tv2
60
61     validroots = []
62
63     if roots[0][1] == 0 and roots[0][0] > 0.0: validroots.append(roots[0][0])
64     if roots[1][1] == 0 and roots[1][0] > 0.0: validroots.append(roots[1][0])
65     if roots[2][1] == 0 and roots[2][0] > 0.0: validroots.append(roots[2][0])
66
67     # returns only valid real roots
```

```
68
69      return validroots
70
71  def fugcoeff(Vz, T, P, state):
72
73      # calculates fugacity coefficients using PR EOS
74      # Vx = composition vector in molar fractions
75      # T = temperature in K
76      # P = Pressure in Pa
77      # state = mixture state (Liquid, Vapor or Solid)
78
79      R = 8.314
80
81      n = len(Vz)
82
83      Tc = pp.RET_VTC() # critical temperatures
84      Pc = pp.RET_VPC() # critical pressures
85      w = pp.RET_VW() # acentric factors
86
87      alpha = [0] * n
88      ai = [0] * n
89      bi = [0] * n
90
91      for i in range(n):
92          alpha[i] = (1 + (0.37464 + 1.54226 * w[i] - 0.26992 * w[i] ** 2) * (1 - (T / Tc[i]) ** 0.5)) ** 2
93          ai[i] = 0.45724 * alpha[i] * R ** 2 * Tc[i] ** 2 / Pc[i]
94          bi[i] = 0.0778 * R * Tc[i] / Pc[i]
95
96      a = [[0] * n for i in range(n)]
97
98      # get binary interaction parameters (BIPs/kijs) from PR Property Package
99
100     kij = [[0] * n for i in range(n)]
101
102     vkij = pp.RET_VKij()
103
104     for i in range(n):
105         for j in range(n):
106             kij[i][j] = vkij[i, j]
107             a[i][j] = (ai[i] * ai[j]) ** 0.5 * (1 - kij[i][j]) # <- default mixing rule for amix
108
109     amix = 0.0
110     bmix = 0.0
111     amix2 = [0] * n
112
113     for i in range(n):
114         for j in range(n):
115             amix += Vz[i] * Vz[j] * a[i][j] # <- default mixing rule for amix
116             amix2[i] += Vz[j] * a[j][i]
117
118     for i in range(n):
119         bmix += Vz[i] * bi[i] # <- default mixing rule - no interaction parameters for bmix
120
121     AG = amix * P / (R * T) ** 2
122     BG = bmix * P / (R * T)
123
124     coeff = [0] * 4
125
126     coeff[0] = 1
127     coeff[1] = BG - 1
128     coeff[2] = AG - 3 * BG ** 2 - 2 * BG
```

```
129     coeff[3] = -AG * BG + BG ** 2 + BG ** 3
130
131     roots = calcroots(coeff) # <- get the real roots of the cubic equation
132
133     # compressibility factor = cubic equation's root
134
135     if state == State.Liquid:
136         # liquid
137         Z = min(roots)
138     else:
139         # vapor
140         Z = max(roots)
141
142     # gets a special zeroed vector from the property package because DWSIM requires a
143     # .NET array as the returning value, not a Python one
144
145     fugcoeff = pp.RET_NullVector()
146
147     for i in range(n):
148         t1 = bi[i] * (Z - 1) / bmix
149         t2 = -Math.Log(Z - BG)
150         t3 = AG * (2 * amix2[i] / amix - bi[i] / bmix)
151         t4 = Math.Log((Z + (1 + 2 ** 0.5) * BG) / (Z + (1 - 2 ** 0.5) * BG))
152         t5 = 2 * 2 ** 0.5 * BG
153         fugcoeff[i] = Math.Exp(t1 + t2 - (t3 * t4 / t5))
154
155     # returns calculated fugacity coefficients
156
157     print 'calculated fugacities = ' + str(fugcoeff) + ' (' + str(state) + ')'
158
159     return fugcoeff
160
161 # activate fugacity calculation override on PR Property Package
162
163 pp.OverrideKvalFugCoeff = True
164
165 # set the function that calculates the fugacity coefficient
166
167 pp.KvalFugacityCoefficientOverride = fugcoeff
```

### 32.3.2. Enthalpy/Entropy

The following code overrides the enthalpy calculation routine (called by the Material Stream's Phase Property and by Pressure-Enthalpy flash routines) from a Peng-Robinson Property Package instance added to the simulation. The code reproduces the internal DWSIM's PR routines, just for illustration purposes.

```
1  import clr
2  clr.AddReference('DWSIM.MathOps')
3
4  from DWSIM import *
5  from DWSIM.MathOps.MathEx import *
6  from DWSIM.Thermodynamics.PropertyPackages import *
7  from System import *
8
9  # gets the first Property Package added to the the simulation
10
11 pp = Flowsheet.PropertyPackagesArray[0]
12
```

```
13  def calcroots(coeffs):
14
15      # auxiliary function
16      # calculates the roots of of a cubic polynomial and returns only the real ones
17
18      a = coeffs[0]
19      b = coeffs[1]
20      c = coeffs[2]
21      d = coeffs[3]
22
23      # uses DWSIM's internal 'CalcRoots' function to calculate roots
24      # https://github.com/DanWBR/dwsim5/blob/windows/DWSIM.Math/MATRIX2.vb#L29
25
26      res = PolySolve.CalcRoots(a, b, c, d)
27
28      roots = [[0] * 2 for i in range(3)]
29
30      roots[0][0] = res[0, 0]
31      roots[0][1] = res[0, 1]
32      roots[1][0] = res[1, 0]
33      roots[1][1] = res[1, 1]
34      roots[2][0] = res[2, 0]
35      roots[2][1] = res[2, 1]
36
37      # orders the roots
38
39      if roots[0][0] > roots[1][0]:
40          tv = roots[1][0]
41          roots[1][0] = roots[0][0]
42          roots[0][0] = tv
43          tv2 = roots[1][1]
44          roots[1][1] = roots[0][1]
45          roots[0][1] = tv2
46      if roots[0][0] > roots[2][0]:
47          tv = roots[2][0]
48          roots[2][0] = roots[0][0]
49          roots[0][0] = tv
50          tv2 = roots[2][1]
51          roots[2][1] = roots[0][1]
52          roots[0][1] = tv2
53      if roots[1][0] > roots[2][0]:
54          tv = roots[2][0]
55          roots[2][0] = roots[1][0]
56          roots[1][0] = tv
57          tv2 = roots[2][1]
58          roots[2][1] = roots[1][1]
59          roots[1][1] = tv2
60
61      validroots = []
62
63      if roots[0][1] == 0 and roots[0][0] > 0.0: validroots.append(roots[0][0])
64      if roots[1][1] == 0 and roots[1][0] > 0.0: validroots.append(roots[1][0])
65      if roots[2][1] == 0 and roots[2][0] > 0.0: validroots.append(roots[2][0])
66
67      # returns only valid real roots
68
69      return validroots
70
71  def enthalpy(Vz, T, P, state):
72
73      # calculates enthalpy using PR EOS
```

```
74      # Vx = composition vector in molar fractions
75      # T = temperature in K
76      # P = Pressure in Pa
77      # state = mixture state (Liquid, Vapor or Solid)
78
79      # ideal gas enthalpy contribution
80
81      Hid = pp.RET_Hid(298.15, T, Vz)
82
83      R = 8.314
84
85      n = len(Vz)
86
87      Tc = pp.RET_VTC() # critical temperatures
88      Pc = pp.RET_VPC() # critical pressures
89      w = pp.RET_VW() # acentric factors
90
91      alpha = [0] * n
92      ai = [0] * n
93      bi = [0] * n
94      ci = [0] * n
95
96      for i in range(n):
97          alpha[i] = (1 + (0.37464 + 1.54226 * w[i] - 0.26992 * w[i] ** 2) * (1 - (T / Tc[i]) ** 0.5)) ** 2
98          ai[i] = 0.45724 * alpha[i] * R ** 2 * Tc[i] ** 2 / Pc[i]
99          bi[i] = 0.0778 * R * Tc[i] / Pc[i]
100         ci[i] = 0.37464 + 1.54226 * w[i] - 0.26992 * w[i] ** 2
101
102     a = [[0] * n for i in range(n)]
103
104     # get binary interaction parameters (BIPs/kijs) from PR Property Package
105
106     kij = [[0] * n for i in range(n)]
107
108     vkij = pp.RET_VKij()
109
110     for i in range(n):
111         for j in range(n):
112             kij[i][j] = vkij[i, j]
113             a[i][j] = (ai[i] * ai[j]) ** 0.5 * (1 - kij[i][j]) # <- default mixing rule for amix
114
115     amix = 0.0
116     bmix = 0.0
117     amix2 = [0] * n
118
119     for i in range(n):
120         for j in range(n):
121             amix += Vz[i] * Vz[j] * a[i][j] # <- default mixing rule for amix
122             amix2[i] += Vz[j] * a[j][i]
123
124     for i in range(n):
125         bmix += Vz[i] * bi[i] # <- default mixing rule - no interaction parameters for bmix
126
127     AG = amix * P / (R * T) ** 2
128     BG = bmix * P / (R * T)
129
130     coeff = [0] * 4
131
132     coeff[0] = 1
133     coeff[1] = BG - 1
134     coeff[2] = AG - 3 * BG ** 2 - 2 * BG
```

```
135     coeff[3] = -AG * BG + BG ** 2 + BG ** 3
136
137     roots = calcroots(coeff) # <- get the real roots of the cubic equation
138
139     # compressibility factor = cubic equation's root
140
141     if state == State.Liquid:
142         # liquid
143         Z = min(roots)
144     else:
145         # vapor
146         Z = max(roots)
147
148     # amix temperature derivative
149
150     dadT1 = -8.314 / 2 * (0.45724 / T) ** 0.5
151     dadT2 = 0.0#
152     for i in range(n):
153         j = 0
154         for j in range(n):
155             dadT2 += Vz[i] * Vz[j] * (1 - kij[i][j]) * (ci[j] * (ai[i] * Tc[j] / Pc[j]) ** 0.5 + ci[i] * (ai
                    [j] * Tc[i] / Pc[i]) ** 0.5)
156
157     dadT = dadT1 * dadT2
158
159     uu = 2
160     ww = -1
161
162     DAres = amix / (bmix * (uu ** 2 - 4 * ww) ** 0.5) * Math.Log((2 * Z + BG * (uu - (uu ** 2 - 4 * ww) **
          0.5)) / (2 * Z + BG * (uu + (uu ** 2 - 4 * ww) ** 0.5))) - R * T * Math.Log((Z - BG) / Z) - R * T
          * Math.Log(Z)
163     DSres = R * Math.Log((Z - BG) / Z) + R * Math.Log(Z) - 1 / (8 ** 0.5 * bmix) * dadT * Math.Log((2 * Z +
           BG * (2 - 8 ** 0.5)) / (2 * Z + BG * (2 + 8 ** 0.5)))
164     DHres = DAres + T * (DSres) + R * T * (Z - 1)
165
166     # mixture molar weight (MW)
167
168     MW = pp.AUX_MMM(Vz)
169
170     print 'calculated enthalpy = ' + str(Hid + DHres/ MW) + ' kJ/kg (' + str(state) + ')'
171
172     return Hid + DHres / MW # kJ/kg
173
174 # activate enthalpy calculation override on PR Property Package
175
176 pp.OverrideEnthalpyCalculation = True
177
178 # set the function that calculates the enthalpy
179
180 pp.EnthalpyCalculationOverride = enthalpy
```

The Entropy override works the same way by defining the **OverrideEntropyCalculation**/**EntropyCalculationOverride** properties from the Property Package instance.

## 32.4. Overriding Flash Algorithms

You can define custom Python functions to calculate the Pressure/Temperature, Pressure/Enthalpy, Pressure/Entropy, Pressure/VaporFraction and Temperature/VaporFraction flashes (phase equilibria calculation routines).

To override the default Flash algorithms, create a new UserDefinedFlash() object and set the UserDefinedFlash property of the UniversalFlash object to the created object:

```
udflash = UserDefinedFlash()
UniversalFlash.UserDefinedFlash = udflash
```

After creating the actual overriding functions, set them to their respective placeholders in the udflash object:

```
def PTFlash(Vz, P, T, pp):
    (...)

def PHFlash(Vz, P, H, Test, pp):
    (...)

def PSFlash(Vz, P, H, Test, pp):
    (...)

udflash.PTFlash = PTFlash
udflash.PHFlash = PHFlash
udflash.PSFlash = PSFlash
```

Full Python code:

```
import clr

clr.AddReference('DWSIM.MathOps')

from DWSIM import *
from DWSIM.MathOps.MathEx import *
from DWSIM.Thermodynamics.PropertyPackages import *
from DWSIM.Thermodynamics.PropertyPackages.Auxiliary.FlashAlgorithms import *
from System import *
from System.Collections.Generic import *

udflash = UserDefinedFlash()

UniversalFlash.UserDefinedFlash = udflash

def PTFlash(Vz, P, T, pp):

    n = len(Vz)

    L = 0.0
    V = 0.0

    Vy = [0] * n
    Vx = [0] * n

    Ki = [0] * n

    Vp = [0] * n

    count = 0

    alreadysolved = False

    if T > Common.Max(pp.RET_VTC(), Array[Double](Vz)):
        # supercritical mixture
        alreadysolved = True
        Vy = list(Vz)
        Vx = list(Vz)
```

```
        V = 1.0
        L = 0.0

    for i in range(n):
        Vp[i] = pp.AUX_PVAPi(i, T)
        Ki[i] = Vp[i] / P

    Px = 0.0
    for i in range(n):
        if Vp[i] != 0.0: Px += Vz[i] / Vp[i]

    Px = 1 / Px
    Pmin = Px

    Px = 0.0
    for i in range(n):
        Px += Vz[i] * Vp[i]

    Pmax = Px

    Pb = Pmax
    Pd = Pmin

    if Math.Abs(Pb - Pd) / Pb < 0.0000001:
        # one comp only
        alreadysolved = True
        Px = sum(Vp * Vz)
        if Px <= P:
            L = 1.0
            V = 0.0
            Vx = list(Vz)
            for i in range(n):
                Vy[i] = Vx[i] * Ki[i]
        else:
            L = 0.0
            V = 1.0
            Vy = list(Vz)
            for i in range(n):
                Vx[i] = Vy[i] / Ki[i]

    if alreadysolved == False:

        Vmin = 1.0#
        Vmax = 0.0#

        for i in range(n):
            if (Ki[i] * Vz[i] - 1) / (Ki[i] - 1) < Vmin: Vmin = (Ki[i] * Vz[i] - 1) / (Ki[i] - 1)
            if (1 - Vz[i]) / (1 - Ki[i]) > Vmax: Vmax = (1 - Vz[i]) / (1 - Ki[i])

        if Vmin < 0.0: Vmin = 0.0
        if Vmin == 1.0: Vmin = 0.0
        if Vmax == 0.0: Vmax = 1.0
        if Vmax > 1.0: Vmax = 1.0

        V = (Vmin + Vmax) / 2

        g = 0.0
        for i in range(n):
            g += Vz[i] * (Ki[i] - 1) / (V + (1 - V) * Ki[i])

        if g > 0.0:
```

```
100             Vmin = V
101         else:
102             Vmax = V
103 
104         V = Vmin + (Vmax - Vmin) / 2
105         L = 1.0 - V
106 
107         for i in range(n):
108             if Vz[i] != 0.0:
109                 Vy[i] = Vz[i] * Ki[i] / ((Ki[i] - 1) * V + 1)
110                 if Ki[i] != 0.0:
111                     Vx[i] = Vy[i] / Ki[i]
112                 else:
113                     Vx[i] = Vz[i]
114                 if Vy[i] < 0.0: Vy[i] = 0.0
115                 if Vx[i] < 0.0: Vx[i] = 0.0
116             else:
117                 Vy[i] = 0.0
118                 Vx[i] = 0.0
119 
120         error = 1E10
121 
122         Vy_prev = [0] * n
123         Vx_prev = [0] * n
124         Ki_prev = [0] * n
125         V_ant = V
126 
127         while error > 1E-6:
128 
129             Ki_prev = list(Ki)
130             Ki = pp.DW_CalcKvalue(Array[Double](Vx), Array[Double](Vy), T, P)
131 
132             Vy_prev = list(Vy)
133             Vx_prev = list(Vx)
134 
135             if V == 1.0:
136                 Vy = list(Vz)
137                 for i in range(n):
138                     Vx[i] = Vy[i] / Ki[i]
139             elif V == 0.0:
140                 Vx = list(Vz)
141                 for i in range(n):
142                     Vy[i] = Ki[i] * Vx[i]
143             else:
144                 for i in range(n):
145                     Vy[i] = Vz[i] * Ki[i] / ((Ki[i] - 1) * V + 1)
146                     Vx[i] = Vy[i] / Ki[i]
147 
148             error = 0.0
149             for i in range(n):
150                 error += Math.Abs(Vx[i] - Vx_prev[i]) + Math.Abs(Vy[i] - Vy_prev[i])
151 
152             V_prev = V
153 
154             F = 0.0
155             dF = 0.0
156             for i in range(n):
157                 if Vz[i] > 0.0:
158                     F += Vz[i] * (Ki[i] - 1) / (1 + V * (Ki[i] - 1))
159                     dF -= Vz[i] * (Ki[i] - 1) ** 2 / (1 + V * (Ki[i] - 1)) ** 2
160 
```

```
161             if Math.Abs(F) < 1E-6: break
162
163             V = -F / dF + V_prev
164
165             L = 1.0 - V
166
167             if V <= 0.0:
168                 V = 0.0
169                 L = 1.0
170                 Vx = list(Vz)
171                 for i in range(n):
172                     Vy[i] = Ki[i] * Vx[i]
173             if V >= 1.0:
174                 V = 1.0
175                 L = 0.0
176                 Vy = list(Vz)
177                 for i in range(n):
178                     Vx[i] = Vy[i] / Ki[i]
179
180             if Math.Abs(V - V_prev) < 1E-6: break
181
182             count += 1
183
184             if count > 100: raise Exception("PT Flash: Maximum Iterations Reached")
185
186             print 'PT Flash iteration #' + str(count) + ' (V = ' + str(V) + ', convergence error = ' + str(
                    error) + ')'
187
188     Vyn = [0] * n
189     Vxn = [0] * n
190
191     for i in range(n):
192         Vyn[i] = V * Vy[i]
193         Vxn[i] = L * Vx[i]
194
195     results = FlashCalculationResult(pp.DW_GetConstantProperties())
196
197     results.MixtureMoleAmounts = List[Double](Vz)
198
199     zerovec = pp.RET_NullVector()
200
201     results.VaporPhaseMoleAmounts = List[Double](Vyn)
202     results.LiquidPhase1MoleAmounts = List[Double](Vxn)
203     results.LiquidPhase2MoleAmounts = List[Double](zerovec)
204     results.SolidPhaseMoleAmounts = List[Double](zerovec)
205     results.Kvalues = List[Double](Ki)
206
207     results.IterationsTaken = count
208
209     return results
210
211 udflash.PTFlash = PTFlash
212
213 def Herror(Href, Vz, P, Test, pp):
214
215     tresult = PTFlash(Vz, P, Test, pp)
216
217     Vw = tresult.GetVaporPhaseMassFraction()
218     Lw = tresult.GetLiquidPhase1MassFraction()
219
220     Vy = tresult.GetVaporPhaseMoleFractions()
```

```
221     Vx = tresult.GetLiquidPhase1MoleFractions()
222
223     Hv = 0.0
224     Hl = 0.0
225
226     if Vw > 0.0: Hv = pp.DW_CalcEnthalpy(Array[Double](Vy), Test, P, State.Vapor)
227     if Lw > 0.0: Hl = pp.DW_CalcEnthalpy(Array[Double](Vx), Test, P, State.Liquid)
228
229     # calculate mixture enthalpy in kJ/kg
230
231     Hest = Vw * Hv + Lw * Hl
232
233     return (Href - Hest)
234
235 def PHFlash(Vz, P, H, Test, pp):
236
237     if Test == 0.0: Test = 273.15
238
239     T = Test
240
241     fx = 1E6
242
243     count = 0
244
245     while Math.Abs(fx) > 1E-4:
246         fx = Herror(H, Vz, P, T, pp)
247         fx2 = Herror(H, Vz, P, T + 0.1, pp)
248         dfdx = (fx2 - fx) / 0.1
249         dx = fx / dfdx
250         T = T - 0.7 * dx
251         count += 1
252         if count > 25: raise Exception("PH Flash: Maximum Iterations Reached")
253         print 'PH Flash iteration #' + str(count) + ' (T = ' + str(T) + ', current enthalpy error = ' + str
                (fx) + ' kJ/kg)'
254
255     result = PTFlash(Vz, P, T, pp)
256
257     result.CalculatedTemperature = T
258
259     return result
260
261 udflash.PHFlash = PHFlash
262
263 def Serror(Sref, Vz, P, Test, pp):
264
265     tresult = PTFlash(Vz, P, Test, pp)
266
267     Vw = tresult.GetVaporPhaseMassFraction()
268     Lw = tresult.GetLiquidPhase1MassFraction()
269
270     Vy = tresult.GetVaporPhaseMoleFractions()
271     Vx = tresult.GetLiquidPhase1MoleFractions()
272
273     Sv = 0.0
274     Sl = 0.0
275
276     if Vw > 0.0: Sv = pp.DW_CalcEntropy(Array[Double](Vy), Test, P, State.Vapor)
277     if Lw > 0.0: Sl = pp.DW_CalcEntropy(Array[Double](Vx), Test, P, State.Liquid)
278
279     # calculate mixture entropy in kJ/[kg.K]
280
```

```
281     Sest = Vw * Sv + Lw * Sl
282
283     return (Sref - Sest)
284
285 def PSFlash(Vz, P, S, Test, pp):
286
287     if Test == 0.0: Test = 273.15
288
289     T = Test
290
291     fx = 1E6
292
293     count = 0
294
295     while Math.Abs(fx) > 1E-4:
296         fx = Serror(S, Vz, P, T, pp)
297         fx2 = Serror(S, Vz, P, T + 0.1, pp)
298         dfdx = (fx2 - fx) / 0.1
299         dx = fx / dfdx
300         T = T - 0.7 * dx
301         count += 1
302         if count > 25: raise Exception("PS Flash: Maximum Iterations Reached")
303         print 'PS Flash iteration #' + str(count) + ' (T = ' + str(T) + ', current entropy error = ' + str(
                fx) + ' kJ/[kg.K])'
304
305     result = PTFlash(Vz, P, T, pp)
306
307     result.CalculatedTemperature = T
308
309     return result
310
311 udflash.PSFlash = PSFlash
```

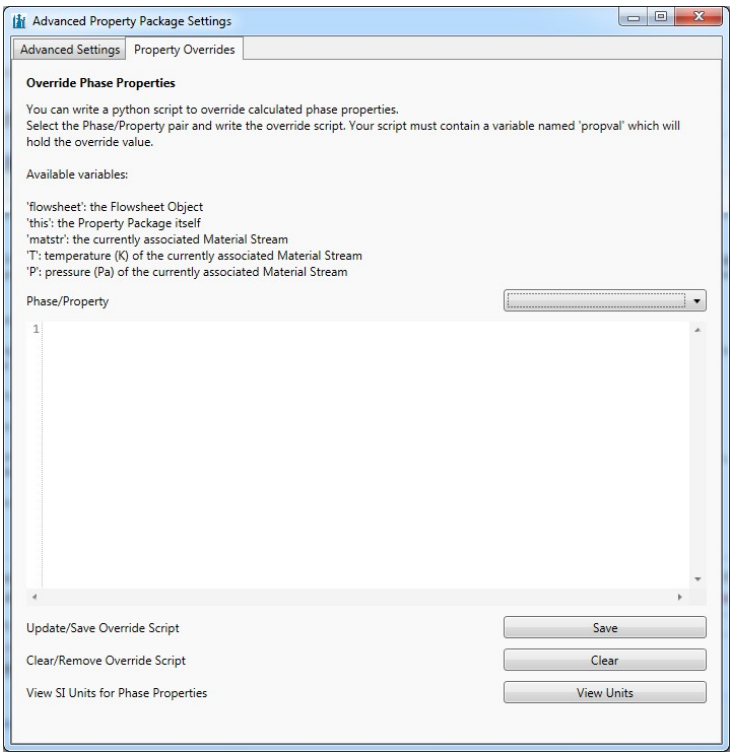
# 33. Overriding Calculated Properties

Introduction Starting from DWSIM Version 5.1, you can override the calculated phase properties through Python scripts. This can be useful if the calculated property is far from the expected value, or if you need to include advanced mixing rules when calculating mixed phase properties.

## 33.1. Accessing the feature

**Classic UI** Click on 'Edit' menu item > 'Simulation Settings' > 'Basis' and, on the Added Property Packages section, select an added Property Package, click on 'Settings' and go to 'Property Overrides'.

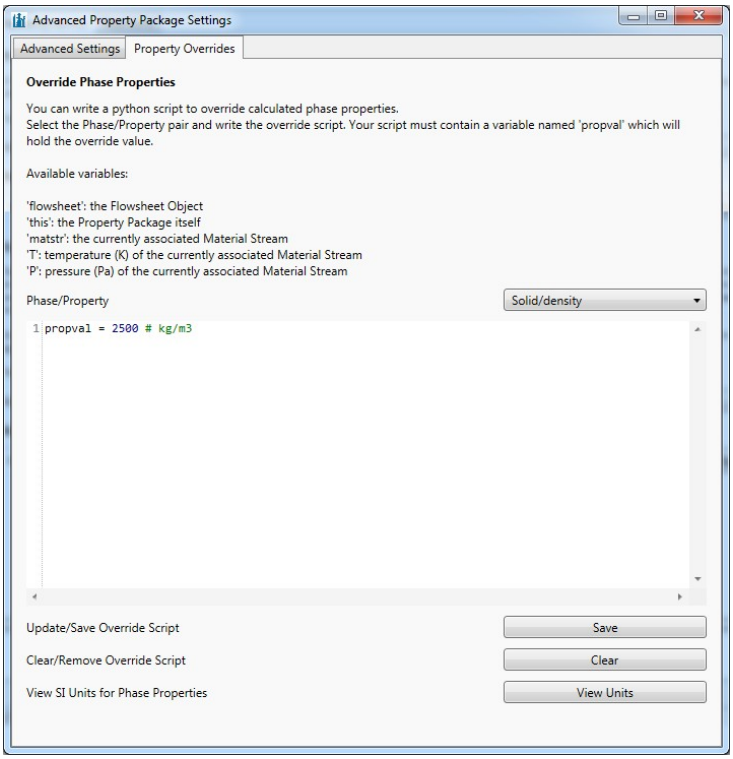
**Cross-Platform UI** Click on 'Setup' menu item > 'Basis' and, on the Added Property Packages section, select an added Property Package, click on 'Advanced' and go to 'Property Overrides'.




## 33.2. Override Script Samples

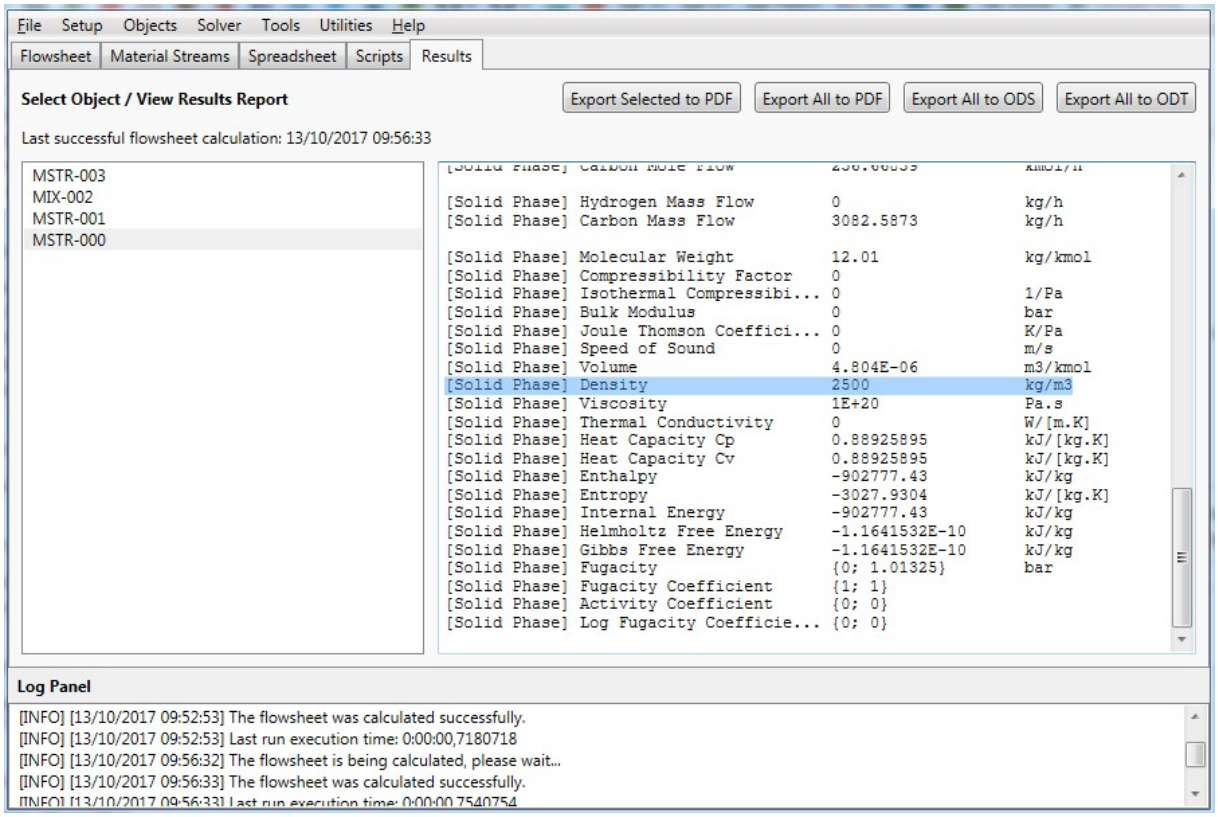
### 33.2.1. Solid Density

If you have a solid phase in your simulation, try the following:




Click 'Save' to store the override script on the Property Package. Solve the simulation and check the calculated value. This override will work on all Material Streams which are associated with the modified

Property Package.




## 33.3. Water/Hydrocarbon Emulsion Viscosity

In this example, we will use a script to update the viscosity calculation method of the overall liquid phase when there are two liquid phases on the mixture, in order to take into account the emulsion effects in a pipe flow simulation.

We will replace the current method for liquid mixture viscosity (a simple mass fraction weighted average) by this emulsion equation:

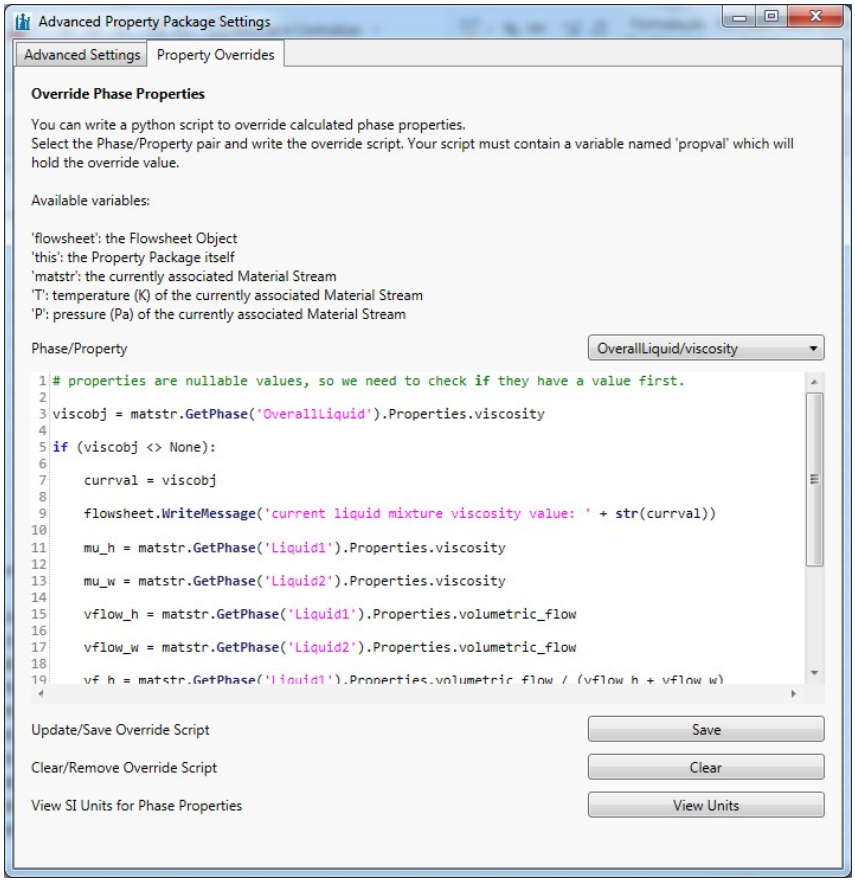
$\mu = (\mu_w/\delta_w)[1 + 1.5\mu_h\delta_h(\mu_w + \mu_h)]$

where $\mu$ is viscosity and $\delta$ is volumetric fraction. Subscripts $w$ and $h$ refer to water and hydrocarbon phase, respectively.

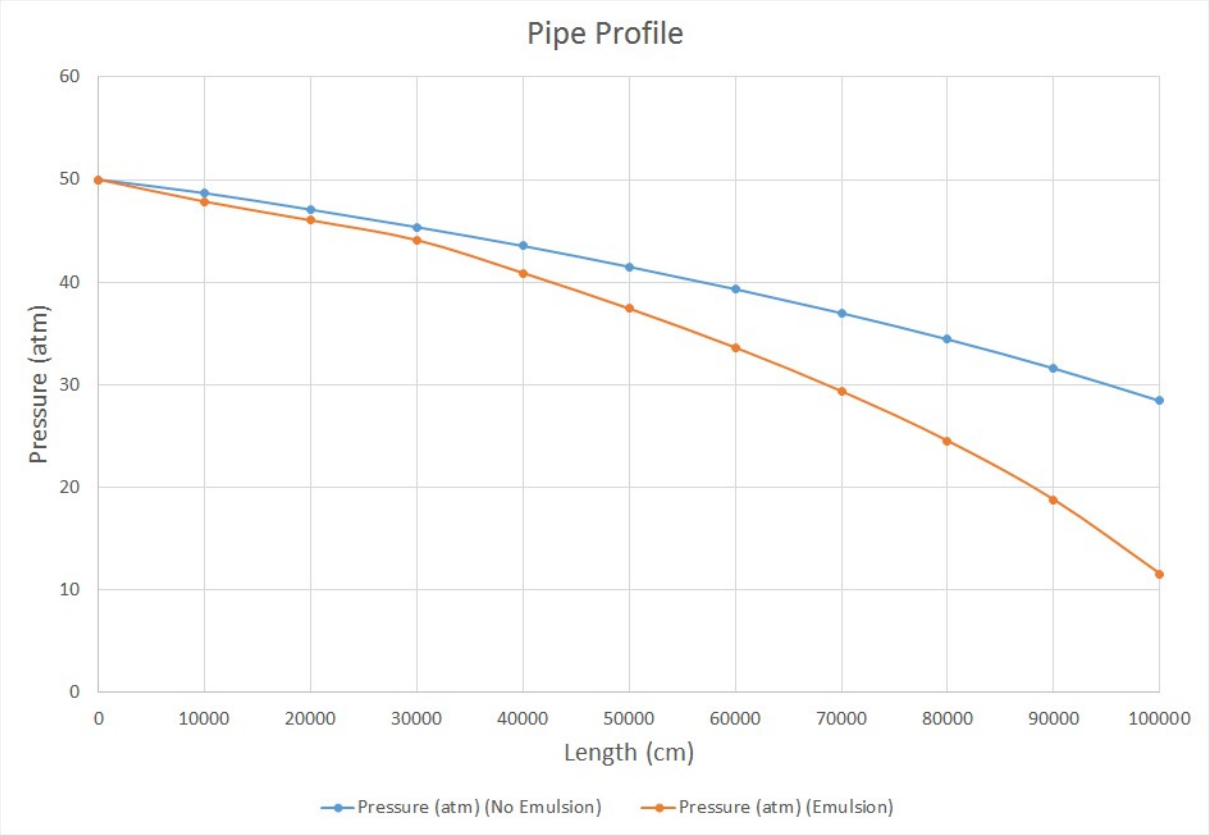
Open the Property Override editor and enter the following script for the "OverallLiquid/viscosity" property:

```
# properties are nullable values, so we need to check if they have a value first.

viscobj = matstr.GetPhase('OverallLiquid').Properties.viscosity

if (viscobj <> None):

    currval = viscobj

    # write current value to the flowsheet (Pa.s)
    flowsheet.WriteMessage('current liquid mixture viscosity value (Pa.s): ' + str(currval))

    # get viscosity of the liquid hydrocarbon phase
    mu_h = matstr.GetPhase('Liquid1').Properties.viscosity

```

```
    # get viscosity of the liquid water phase
    mu_w = matstr.GetPhase('Liquid2').Properties.viscosity

    # get volumetric flow of the liquid hydrocarbon phase
    vflow_h = matstr.GetPhase('Liquid1').Properties.volumetric_flow

    # get volumetric flow of the liquid water phase
    vflow_w = matstr.GetPhase('Liquid2').Properties.volumetric_flow

    # calculate volumetric fraction of the liquid hydrocarbon phase
    vf_h = matstr.GetPhase('Liquid1').Properties.volumetric_flow / (vflow_h + vflow_w)

    # calculate volumetric fraction of the liquid water phase
    vf_w = matstr.GetPhase('Liquid2').Properties.volumetric_flow / (vflow_h + vflow_w)

    # update liquid mixture viscosity with the result from emulsion equation (Pa.s)
    propval = (mu_w/vf_w)*(1.0+1.5*mu_h*vf_h/(mu_w+mu_h))

    flowsheet.WriteMessage('updated liquid mixture (emulsion) viscosity value (Pa.s): ' + str(propval))

else:

    propval = 0.0
```




Run the simulation again and store the results. You can view the effect of the updated viscosity on the pipe pressure profile:




# 34. Automation

Automation enables software packages to expose their unique features to scripting tools and other applications. Using Automation, you can:

- Create applications and programming tools that expose objects.
- Create and manipulate objects exposed in one application from another application.
- Create tools that access and manipulate objects.

These tools can include embedded macro languages, external programming tools, object browsers, and compilers. On a Windows environment, the objects an application or programming tool exposes are called ActiveX objects. Applications and programming tools that access those objects are called ActiveX clients. ActiveX objects and clients interact as follows: Applications and other software packages that support ActiveX technology define and expose objects which can be acted on by ActiveX components. ActiveX components are physical files (for example .exe and .dll files) that contain classes, which are definitions of objects. Type information describes the exposed objects, and can be used by ActiveX components at either compile time or at run time.

## 34.1. Automation support in DWSIM

Starting from version 4.2, DWSIM exposes its main Classes and Interfaces to Automation via COM/.NET. Automating DWSIM enables you to manipulate flowsheets and run sensitivity/optimization studies directly from Microsoft Excel, for instance, without the need of opening DWSIM directly. Interfacing DWSIM with

Excel through VBA macros results in a powerful tool in process engineering activities, such as design, optimization and process evaluation.

The simulation results may be output to an Excel spreadsheet in the development of the Heat and Material Balance for the process design, enabling quick manipulation of the resulting data through a well-known tool for every Chemical Engineer.

## 34.2. Registering DLLs for COM Automation

You can register DWSIM DLLs for automation during the installation process. You can also run the **automation_reg.bat** batch file (located in DWSIM's current installation directory) with admin privileges to register. To de-register, run **automation_unreg.bat** also as admin. When you uninstall DWSIM, the DLLs are automatically deregistered.

If your automation project is based on a .NET language, there's no need to register the DLLs. You'll only need to add a reference to them.

Automating DWSIM through COM is limited to Windows, though .NET is recommended as the default mechanism. On a Linux environment, you can use Mono to create and/or run an automation project in C#.

Introduction to Interfaces Before proceeding, read this text to get used to Interfaces and their implementation in actual Classes: Interfaces in Object-Oriented Programming

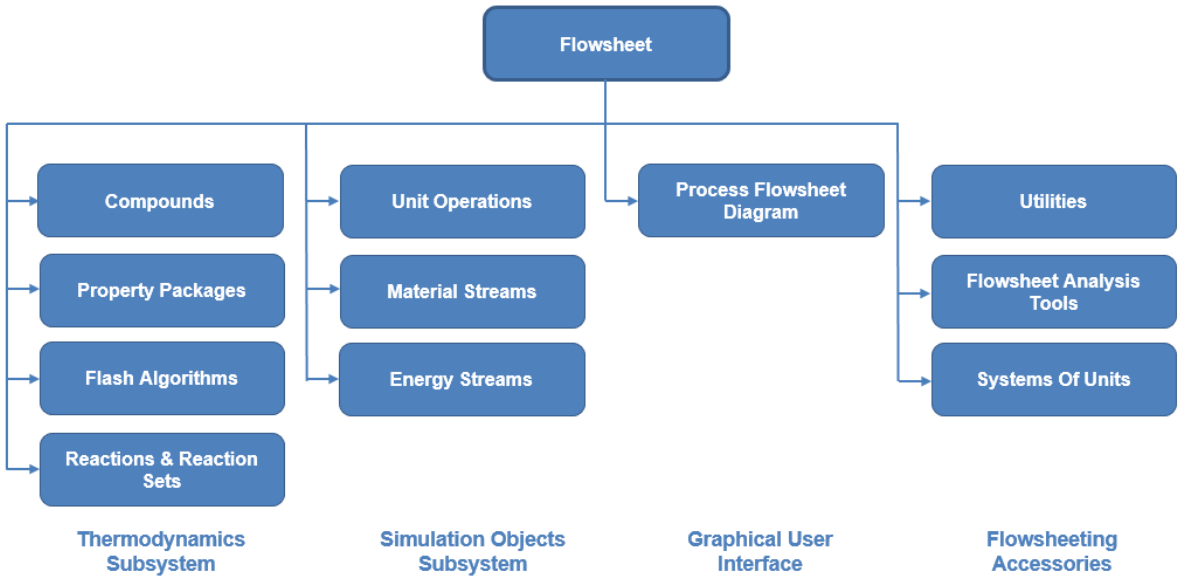
## 34.3. API Reference Documentation

Automation Class: http://dwsim.inforside.com.br/api_help60/html/N_DWSIM_Automation.htm
Interface Definitions: http://dwsim.inforside.com.br/api_help60/html/G_DWSIM_Interfaces.htm
Unit Operations: http://dwsim.inforside.com.br/api_help60/html/G_DWSIM_UnitOperations.htm
Thermodynamics: http://dwsim.inforside.com.br/api_help60/html/G_DWSIM_Thermodynamics.htm
Base Class Shared Library: http://dwsim.inforside.com.br/api_help60/html/G_DWSIM_SharedClasses.htm

Flowsheet GUI and DWSIM main executable: http://dwsim.inforside.com.br/api_help60/html/N_DWSIM.htm

CAPE-OPEN Reference: http://www.colan.org/specifications/

## 34.4. DWSIM Flowsheet Class Structure




The Flowsheet class in DWSIM provides access to all objects in the simulation:

- **Thermodynamics Subsystem:** includes Compounds, Property Packages, Flash Algorithms and Reactions/Reaction Sets collections.
- **Simulation Objects Subsystem:** includes Material & Energy Streams and Unit Operation blocks.
- **Graphical User Interface:** provides access to the displayed objects in the flowsheet and the connections between them.
- **Accessories:** includes added utilities, sensitivity & optimization studies, system of units definitions and other simulation definitions.

The Flowsheet object in DWSIM implements various interfaces, including **IFlowsheet** and **IFlowsheetOptions**.

- **IFlowsheet:** this is the main interface implemented by the Flowsheet class. It provides direct access to the various flowsheet components and helper functions to manipulate objects. http://dwsim.inforside.com.br/ap
- **IFlowsheetOptions:** this interface defines the flowsheet settings and other properties. http://dwsim.inforside.com

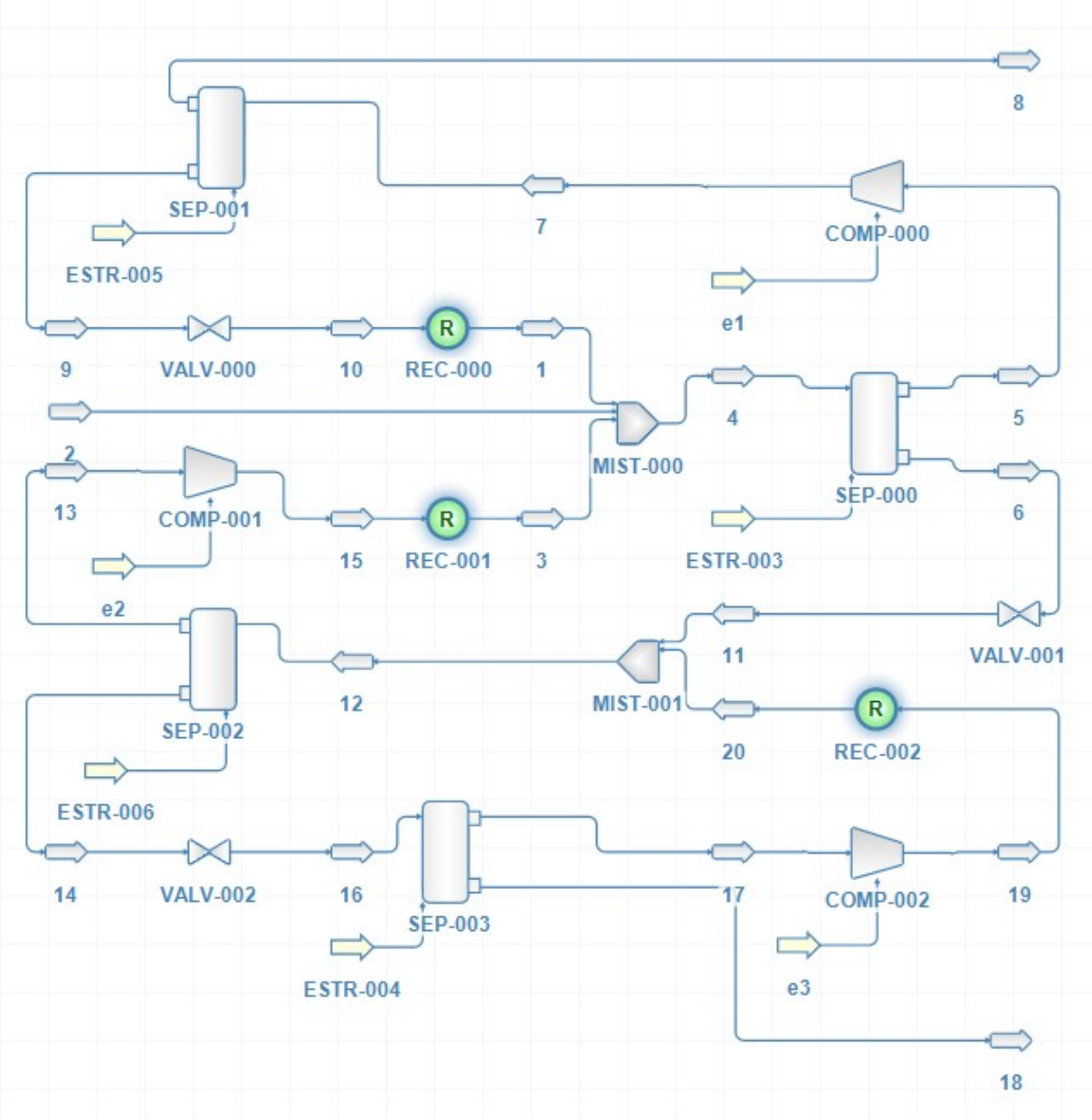
When you use the Automation class to load a simulation, an IFlowsheet object is returned, which is actually an instance of the Flowsheet class. You can cast the returned object to any of the interfaces implemented by the Flowsheet class to access all available functions, properties and procedures.

## 34.5. Sample Automation

This sample automation code will run Cavett's Problem (simulation file located in the samples folder) with four different feed mass flow values, check outlet mass flows and calculate the mass balance of the flowsheet, displaying the results to the user.

### 34.5.1. About Cavett's Problem

A simulation problem proposed by Cavett (1963) has been used to test various chemical engineering simulation programs. It provides a useful benchmark to compare and contrast various tear stream locations and convergence algorithms. The process is equivalent to a four theoretical stage near isothermal distillation flash tanks.




- Feed Stream: 2
- Vapor Outlet Stream: 8
- Liquid Outlet Stream: 18

### 34.5.2. Excel VBA

To run this sample, create a new Excel VBA project and add a reference to **CAPE-OPEN 1.1 Type Library** (http://www.colan.org/software-tools/cape-open-type-libraries-and-primary-interop-assemblies/), **DWSIM Simulator Automation Interface** and **DWSIM Simulator Interface Definitions Library**.

```
1  Public Sub Sub1()
2
3      'create automation manager
4      Dim interf As DWSIM_Automation.Automation
5      Set interf = New DWSIM_Automation.Automation
6
7      'declare the flowsheet variable
8      Dim sim As DWSIM_Interfaces.IFlowsheet
9
10     'load Cavett's Problem simulation file
11     Set sim = interf.LoadFlowsheet(Application.ActiveWorkbook.Path & "\Cavett's Problem.dwxml")
12
13     'use CAPE-OPEN interfaces to manipulate objects
14     Dim feed As CAPEOPEN110.ICapeThermoMaterialObject
15     Dim vap_out As CAPEOPEN110.ICapeThermoMaterialObject
16     Dim liq_out As CAPEOPEN110.ICapeThermoMaterialObject
17
18     Set feed = sim.GetFlowsheetSimulationObject("2")
19     Set vap_out = sim.GetFlowsheetSimulationObject("8")
20     Set liq_out = sim.GetFlowsheetSimulationObject("18")
21
22     'mass flow rate values in kg/s
23     Dim flows(4) As Variant
24
25     flows(0) = 170#
26     flows(1) = 180#
27     flows(2) = 190#
28     flows(3) = 200#
29
30     'vapor and liquid flows
31     Dim vflow, lflow As Double
32
33     For i = 0 To 3
34         'set feed mass flow
35         Call feed.SetProp("totalflow", "overall", Nothing, "", "mass", Array(flows(i)))
36         'calculate the flowsheet (run the simulation)
37         MsgBox "Running simulation with F = " & flows(i) & " kg/s, please wait..."
38         Call interf.CalculateFlowsheet(sim, Nothing)
39         'check for errors during the last run
40         If sim.Solved = False Then
41             MsgBox "Error solving flowsheet: " & sim.ErrorMessage
42         End If
43         'get vapor outlet mass flow value
44         vflow = vap_out.GetProp("totalflow", "overall", Nothing, "", "mass")(0)
45         'get liquid outlet mass flow value
46         lflow = liq_out.GetProp("totalflow", "overall", Nothing, "", "mass")(0)
47         'display results
48         MsgBox "Simulation run #" & (i + 1) & " results:" & vbCrLf & "Feed: " & flows(i) & ", Vapor: " &
               vflow & ", Liquid: " & lflow & " kg/s" & vbCrLf & "Mass balance error: " & (flows(i) - vflow -
               lflow) & " kg/s"
49     Next
50
51     MsgBox "Finished OK!"
52
```

```
53 End Sub
```

### 34.5.3. VB

To run this sample, create a new VB.NET Console Application project and add a reference to DWSIM.Automation.dll, DWSIM.Interfaces.dll and CapeOpen.dll.

```
1  Module Module1
2
3      Sub Main()
4
5          System.IO.Directory.SetCurrentDirectory("C:/Program Files/DWSIM6") ' replace with DWSIM's
               installation directory on your computer
6
7          'create automation manager
8          Dim interf As New DWSIM.Automation.Automation
9
10         Dim sim As Interfaces.IFlowsheet
11
12         'load Cavett's Problem simulation file
13         sim = interf.LoadFlowsheet("samples" & IO.Path.DirectorySeparatorChar & "Cavett's Problem.dwxml")
14
15         '(optional) set a listener to catch solver messages
16         sim.SetMessageListener(Sub(msg As String)
17                                    Console.WriteLine(msg)
18                                End Sub)
19
20         'use CAPE-OPEN interfaces to manipulate objects
21         Dim feed, vap_out, liq_out As CapeOpen.ICapeThermoMaterialObject
22
23         feed = sim.GetFlowsheetSimulationObject1("2")
24         vap_out = sim.GetFlowsheetSimulationObject1("8")
25         liq_out = sim.GetFlowsheetSimulationObject1("18")
26
27         'mass flow rate values in kg/s
28         Dim flows(3) As Double
29
30         flows(0) = 170.0#
31         flows(1) = 180.0#
32         flows(2) = 190.0#
33         flows(3) = 200.0#
34
35         'vapor and liquid flows
36         Dim vflow, lflow As Double
37
38         For i = 0 To flows.Length - 1
39             'set feed mass flow
40             feed.SetProp("totalflow", "overall", Nothing, "", "mass", New Double() {flows(i)})
41             'calculate the flowsheet (run the simulation)
42             Console.WriteLine("Running simulation with F = " & flows(i) & " kg/s, please wait...")
43             interf.CalculateFlowsheet(sim, Nothing)
44             'check for errors during the last run
45             If sim.Solved = False Then
46                 Console.WriteLine("Error solving flowsheet: " & sim.ErrorMessage)
47             End If
48             'get vapor outlet mass flow value
49             vflow = vap_out.GetProp("totalflow", "overall", Nothing, "", "mass")(0)
50             'get liquid outlet mass flow value
51             lflow = liq_out.GetProp("totalflow", "overall", Nothing, "", "mass")(0)
```

```
52              'display results
53              Console.WriteLine("Simulation run #" & (i + 1) & " results:" & vbCrLf & "Feed: " & flows(i) & ",
                     Vapor: " & vflow & ", Liquid: " & lflow & " kg/s" & vbCrLf & "Mass balance error: " & (
                    flows(i) - vflow - lflow) & " kg/s")
54          Next
55
56          Console.WriteLine("Finished OK! Press any key to close.")
57          Console.ReadKey()
58
59      End Sub
60
61  End Module
```

### 34.5.4. C#

To run this sample, create a new C# Console Application project and add a reference to **DWSIM.Automation.dll**, **DWSIM.Interfaces.dll** and **CapeOpen.dll**.

```
1   using System;
2
3   static class Module1
4   {
5
6       public static void Main()
7       {
8
9           System.IO.Directory.SetCurrentDirectory("C:/Program Files/DWSIM6"); // replace with DWSIM's
                installation directory on your computer
10
11          //create automation manager
12          DWSIM.Automation.Automation interf = new DWSIM.Automation.Automation();
13
14          DWSIM.Interfaces.IFlowsheet sim;
15
16          //load Cavett's Problem simulation file
17          sim = interf.LoadFlowsheet("samples" + System.IO.Path.DirectorySeparatorChar + "Cavett's Problem.
                dwxml");
18
19          //use CAPE-OPEN interfaces to manipulate objects
20          CapeOpen.ICapeThermoMaterialObject feed, vap_out, liq_out;
21
22          feed = (CapeOpen.ICapeThermoMaterialObject)sim.GetFlowsheetSimulationObject("2");
23          vap_out = (CapeOpen.ICapeThermoMaterialObject)sim.GetFlowsheetSimulationObject("8");
24          liq_out = (CapeOpen.ICapeThermoMaterialObject)sim.GetFlowsheetSimulationObject("18");
25
26          //mass flow rate values in kg/s
27          double[] flows = new double[4];
28
29          flows[0] = 170.0;
30          flows[1] = 180.0;
31          flows[2] = 190.0;
32          flows[3] = 200.0;
33
34          //vapor and liquid flows
35          double vflow = 0;
36          double lflow = 0;
37
38          for (var i = 0; i <= flows.Length - 1; i++)
39          {
```

```
40            //set feed mass flow
41            feed.SetProp("totalflow", "overall", null, "", "mass", new double[] { flows[i] });
42            //calculate the flowsheet (run the simulation)
43            Console.WriteLine("Running simulation with F = " + flows[i] + " kg/s, please wait...");
44            interf.CalculateFlowsheet(sim, null);
45            //check for errors during the last run
46            if (sim.Solved == false)
47            {
48                Console.WriteLine("Error solving flowsheet: " + sim.ErrorMessage);
49            }
50            //get vapor outlet mass flow value
51            vflow = ((double[])vap_out.GetProp("totalflow", "overall", null, "", "mass"))[0];
52            //get liquid outlet mass flow value
53            lflow = ((double[])liq_out.GetProp("totalflow", "overall", null, "", "mass"))[0];
54            //display results
55            Console.WriteLine("Simulation run #" + (i + 1) + " results:\nFeed: " + flows[i] + ", Vapor: " +
                  vflow + ", Liquid: " + lflow + " kg/s\nMass balance error: " + (flows[i] - vflow - lflow) +
                   " kg/s");
56        }
57
58        Console.WriteLine("Finished OK! Press any key to close.");
59        Console.ReadKey();
60
61    }
62
63 }
```

### 34.5.5. Python

```
1  import pythoncom
2  pythoncom.CoInitialize()
3
4  import clr
5
6  from System.IO import Directory, Path, File
7  from System import String, Environment
8
9  dwsimpath = "C:\\Program Files\\DWSIM6\\"
10
11 clr.AddReference(dwsimpath + "CapeOpen.dll")
12 clr.AddReference(dwsimpath + "DWSIM.Automation.dll")
13 clr.AddReference(dwsimpath + "DWSIM.Interfaces.dll")
14 clr.AddReference(dwsimpath + "DWSIM.GlobalSettings.dll")
15 clr.AddReference(dwsimpath + "DWSIM.SharedClasses.dll")
16 clr.AddReference(dwsimpath + "DWSIM.Thermodynamics.dll")
17 clr.AddReference(dwsimpath + "DWSIM.UnitOperations.dll")
18
19 clr.AddReference(dwsimpath + "DWSIM.Inspector.dll")
20 clr.AddReference(dwsimpath + "DWSIM.MathOps.dll")
21 clr.AddReference(dwsimpath + "TcpComm.dll")
22 clr.AddReference(dwsimpath + "Microsoft.ServiceBus.dll")
23
24 from DWSIM.Interfaces.Enums.GraphicObjects import ObjectType
25 from DWSIM.Thermodynamics import Streams, PropertyPackages
26 from DWSIM.UnitOperations import UnitOperations
27 from DWSIM.Automation import Automation2
28 from DWSIM.GlobalSettings import Settings
29
30 Directory.SetCurrentDirectory(dwsimpath)
```

```
31
32  # create automation manager
33
34  interf = Automation2()
35
36  sim = interf.CreateFlowsheet()
37
38  # add water
39
40  water = sim.AvailableCompounds["Water"]
41
42  sim.SelectedCompounds.Add(water.Name, water)
43
44  # create and connect objects
45
46  m1 = sim.AddObject(ObjectType.MaterialStream, 50, 50, "inlet")
47  m2 = sim.AddObject(ObjectType.MaterialStream, 150, 50, "outlet")
48  e1 = sim.AddObject(ObjectType.EnergyStream, 100, 50, "power")
49  h1 = sim.AddObject(ObjectType.Heater, 100, 50, "heater")
50
51  sim.ConnectObjects(m1.GraphicObject, h1.GraphicObject, -1, -1)
52  sim.ConnectObjects(h1.GraphicObject, m2.GraphicObject, -1, -1)
53  sim.ConnectObjects(e1.GraphicObject, h1.GraphicObject, -1, -1)
54
55  sim.AutoLayout()
56
57  # steam tables property package
58
59  stables = PropertyPackages.SteamTablesPropertyPackage()
60
61  sim.AddPropertyPackage(stables)
62
63  # set inlet stream temperature
64  # default properties: T = 298.15 K, P = 101325 Pa, Mass Flow = 1 kg/s
65
66  m1.SetTemperature(300) # K
67  m1.SetMassFlow(100) # kg/s
68
69  # set heater outlet temperature
70
71  h1.CalcMode = UnitOperations.Heater.CalculationMode.OutletTemperature
72  h1.OutletTemperature = 400 # K
73
74  # request a calculation
75
76  Settings.SolverMode = 0
77
78  errors = interf.CalculateFlowsheet2(sim)
79
80  print(String.Format("Heater Heat Load: {0} kW", h1.DeltaQ))
81
82  # save file
83
84  fileNameToSave = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "heatersample.
        dwxmz")
85
86  interf.SaveFlowsheet(sim, fileNameToSave, True)
87
88  # save the pfd to an image and display it
89
90  clr.AddReference(dwsimpath + "SkiaSharp.dll")
```

```
clr.AddReference("System.Drawing")

from SkiaSharp import SKBitmap, SKImage, SKCanvas, SKEncodedImageFormat
from System.IO import MemoryStream
from System.Drawing import Image
from System.Drawing.Imaging import ImageFormat

PFDSurface = sim.GetSurface()

bmp = SKBitmap(1024, 768)
canvas = SKCanvas(bmp)
canvas.Scale(1.0)
PFDSurface.UpdateCanvas(canvas)
d = SKImage.FromBitmap(bmp).Encode(SKEncodedImageFormat.Png, 100)
str = MemoryStream()
d.SaveTo(str)
image = Image.FromStream(str)
imgPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "pfd.png")
image.Save(imgPath, ImageFormat.Png)
str.Dispose()
canvas.Dispose()
bmp.Dispose()

from PIL import Image

im = Image.open(imgPath)
im.show()
```

# 35. Excel Add-In for Thermo Calculations

## 35.1. Introduction

The DWSIM Excel Add-In exposes some of the internal Thermodynamic Property Calculation Routines to Microsoft Excel, including:

- Single Compound Properties (i.e. Boiling Point, Heat Capacity, Viscosity...)
- Single Phase Mixture Properties (i.e. Enthalpy, Entropy, Molar Weight, Thermal Conductivity, Viscosity...)
- Pressure-Temperature, Pressure-Enthalpy, Pressure-Entropy, Pressure-VaporFraction and Temperature-VaporFraction Flash Calculators, using an algorithm of your choice
- Other auxiliary functions

Property and Equilibrium calculation functionality is now available to Excel just as any other add-in function.

## 35.2. Installation

The Excel Add-In is part of DWSIM Simulator for Windows Desktop - you must install it first.

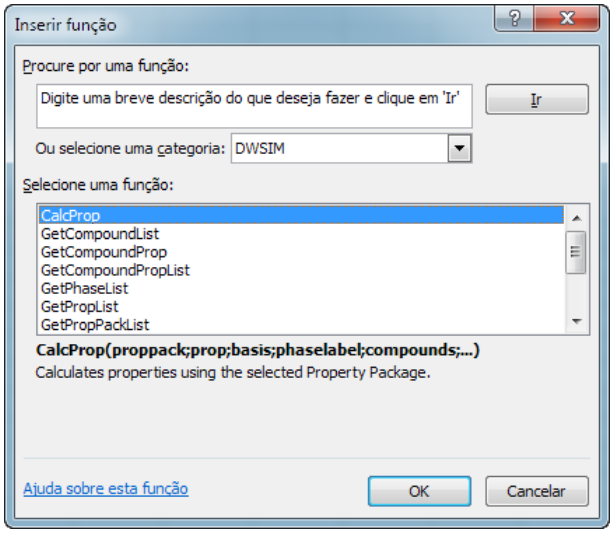
Remember the location where DWSIM was installed, as you'll use this location to find the Add-In XLL file.

After installing DWSIM, open Excel and go to File > Options > Add-Ins > Manage (Excel Add-Ins) > Go > Browse. More information: Add or remove add-ins in Excel

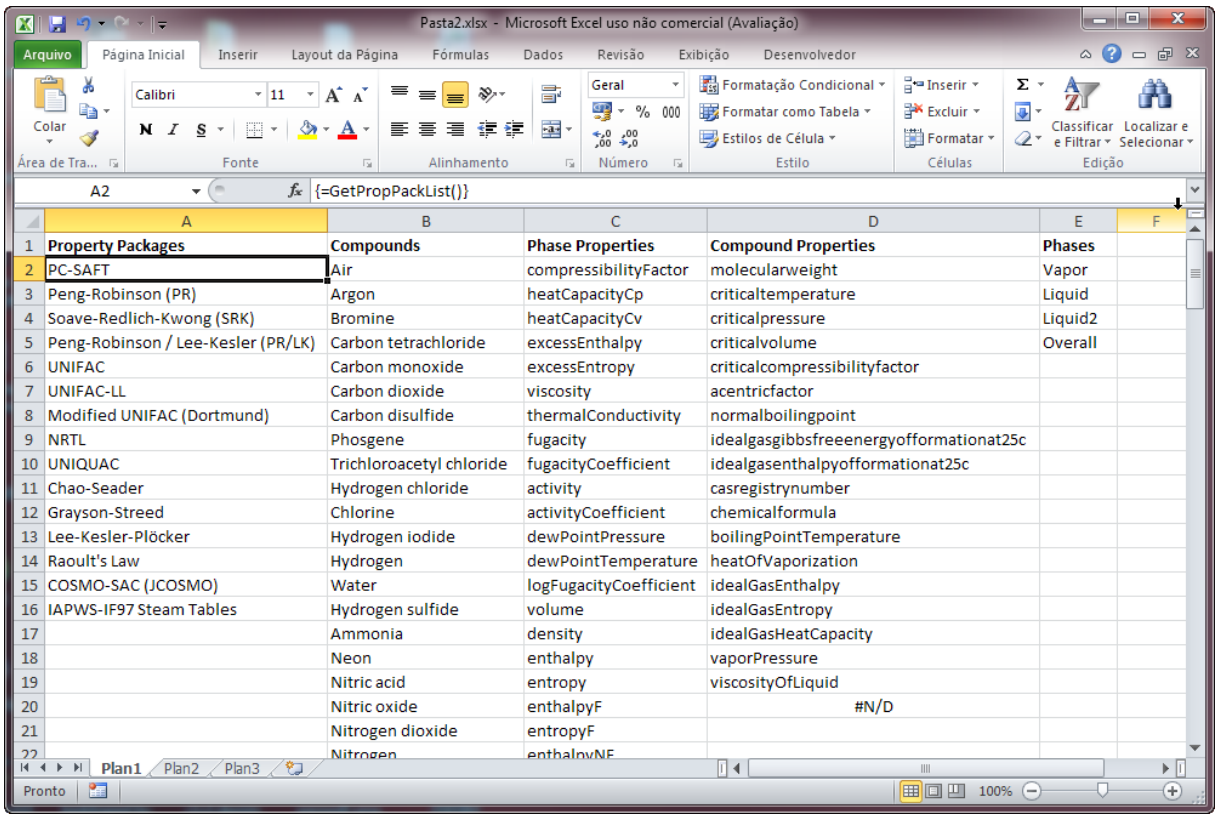
Look for **DWSIM.xll** in DWSIM's installation directory if you're running the 32-bit Excel version, otherwise look for for **DWSIM_64.xll** if you're running the 64-bit version.

## 35.3. Usage

Functions exposed by this add-in will be grouped in a category named DWSIM:

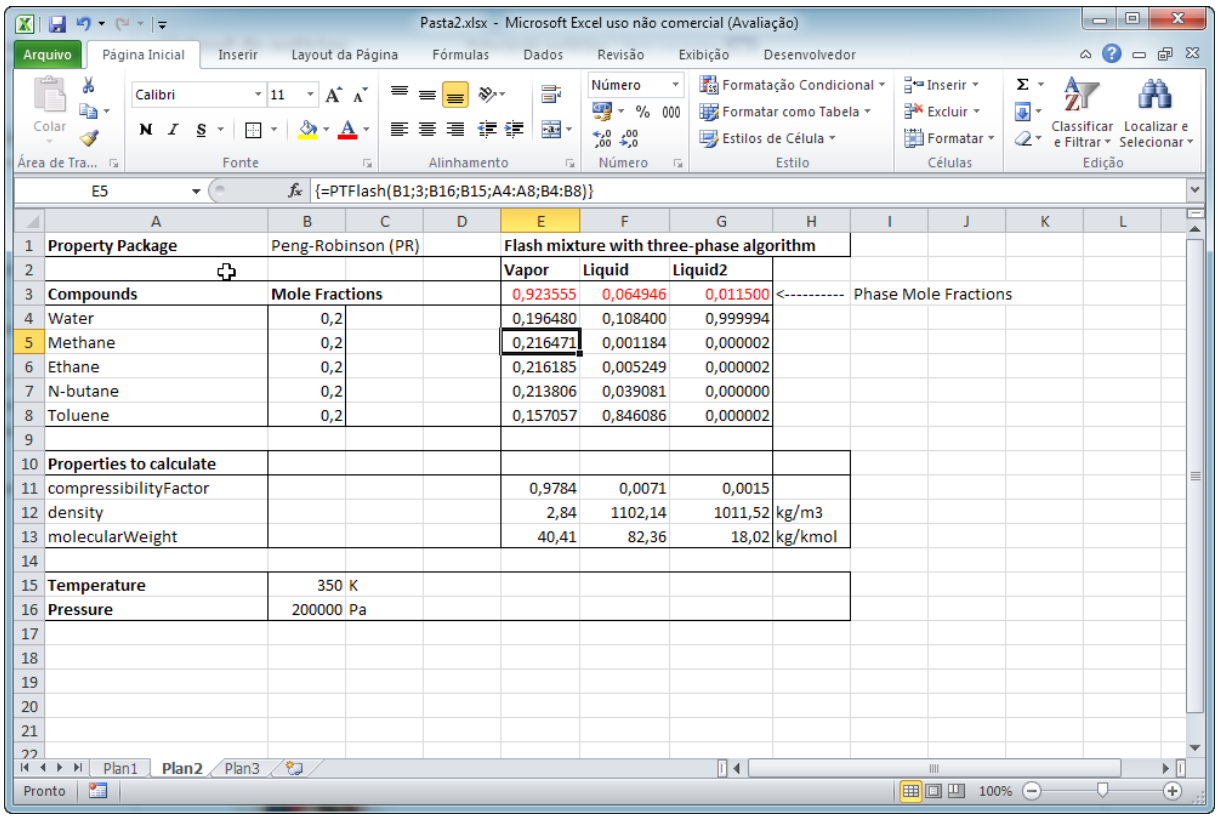



Property and Equilibrium calculation functions require parameters that must be one or more values returned by **GetPropPackList**, **GetCompoundList**, **GetPropList**, **GetCompoundPropList** and **GetPhaseList**. They are self-explanatory, and will return values listed in a single column, so you probably will have to select some cells in a single column and call the functions using Ctrl+Shift+Enter:




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Property Packages** | **Compounds** | **Phase Properties** | **Compound Properties** | **Phases** |
| 2 | PC-SAFT | Air | compressibilityFactor | molecularweight | Vapor |
| 3 | Peng-Robinson (PR) | Argon | heatCapacityCp | criticaltemperature | Liquid |
| 4 | Soave-Redlich-Kwong (SRK) | Bromine | heatCapacityCv | criticalpressure | Liquid2 |
| 5 | Peng-Robinson / Lee-Kesler (PR/LK) | Carbon tetrachloride | excessEnthalpy | criticalvolume | Overall |
| 6 | UNIFAC | Carbon monoxide | excessEntropy | criticalcompressibilityfactor | |
| 7 | UNIFAC-LL | Carbon dioxide | viscosity | acentricfactor | |
| 8 | Modified UNIFAC (Dortmund) | Carbon disulfide | thermalConductivity | normalboilingpoint | |
| 9 | NRTL | Phosgene | fugacity | idealgasgibbsfreeenergyofformationat25c | |
| 10 | UNIQUAC | Trichloroacetyl chloride | fugacityCoefficient | idealgasenthalpyofformationat25c | |
| 11 | Chao-Seader | Hydrogen chloride | activity | casregistrynumber | |
| 12 | Grayson-Streed | Chlorine | activityCoefficient | chemicalformula | |
| 13 | Lee-Kesler-Plöcker | Hydrogen iodide | dewPointPressure | boilingPointTemperature | |
| 14 | Raoult's Law | Hydrogen | dewPointTemperature | heatOfVaporization | |
| 15 | COSMO-SAC (JCOSMO) | Water | logFugacityCoefficient | idealGasEnthalpy | |
| 16 | IAPWS-IF97 Steam Tables | Hydrogen sulfide | volume | idealGasEntropy | |
| 17 | | Ammonia | density | idealGasHeatCapacity | |
| 18 | | Neon | enthalpy | vaporPressure | |
| 19 | | Nitric acid | entropy | viscosityOfLiquid | |
| 20 | | Nitric oxide | enthalpyF | #N/D | |
| 21 | | Nitrogen dioxide | entropyF | | |
| 22 | | Nitrogen | enthalpyNF | | |

For example, the **PTFlash** function requires the name of the Property Package to use, the compound names and mole fractions, temperature in K, pressure in Pa and you may optionally provide new interaction parameters that will override the ones used internally by DWSIM. The calculation results will be returned as a (n+2) x (4) matrix, where n is the number of compounds. First row will contain the phase names (Vapor, Liquid, Liquid2 and Solid, in this order), the second will contain the phase mole fractions and the other lines will contain the compound mole fractions in the corresponding phases:




| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **Property Package** | Peng-Robinson (PR) | | | **Flash mixture with three-phase algorithm** | | | | | | | |
| 2 | | | | | **Vapor** | **Liquid** | **Liquid2** | | | | | |
| 3 | **Compounds** | **Mole Fractions** | | | 0,923555 | 0,064946 | 0,011500 | <---------- | Phase Mole Fractions | | | |
| 4 | Water | 0,2 | | | 0,196480 | 0,108400 | 0,999994 | | | | | |
| 5 | Methane | 0,2 | | | 0,216471 | 0,001184 | 0,000002 | | | | | |
| 6 | Ethane | 0,2 | | | 0,216185 | 0,005249 | 0,000002 | | | | | |
| 7 | N-butane | 0,2 | | | 0,213806 | 0,039081 | 0,000000 | | | | | |
| 8 | Toluene | 0,2 | | | 0,157057 | 0,846086 | 0,000002 | | | | | |
| 9 | | | | | | | | | | | | |
| 10 | **Properties to calculate** | | | | | | | | | | | |
| 11 | compressibilityFactor | | | | 0,9784 | 0,0071 | 0,0015 | | | | | |
| 12 | density | | | | 2,84 | 1102,14 | 1011,52 | kg/m3 | | | | |
| 13 | molecularWeight | | | | 40,41 | 82,36 | 18,02 | kg/kmol | | | | |
| 14 | | | | | | | | | | | | |
| 15 | **Temperature** | 350 | K | | | | | | | | | |
| 16 | **Pressure** | 200000 | Pa | | | | | | | | | |
| 17 | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | |

For PH, PS, TVF and PVF flash calculation functions, and additional line is returned that will contain the temperature in K or pressure in Pa in the first column.

## 35.4. Overriding Interaction Parameters

You can directly override the interaction parameters used by Property Packages when calling calculations from Excel by providing n x n matrices containing the values, where n is the number of compounds. This feature is optional and should be used only when you know exactly what you are doing.

The following table shows the user-definable interaction parameters for each Property Package:

| Property Package | IP Set #1 | IP Set #2 | IP Set #3 | IP Set #4 | IP Set #5 | IP Set #6 | IP Set #7 | IP Set #8 |
|---|---|---|---|---|---|---|---|---|
| **PC-SAFT** | PC-SAFT kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Peng-Robinson (PR)** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Soave-Redlich-Kwong (SRK)** | SRK kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Peng-Robinson-Stryjek-Vera 2 (PRSV2)** | PRSV2-M kij | PRSV2-M kji | Not used | Not used | Not used | Not used | Not used | Not used |
| **Peng-Robinson / Lee-Kesler (PR/LK)** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **UNIFAC** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **UNIFAC-LL** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Modified UNIFAC (Dortmund)** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **NRTL** | PR kij | NRTL A12 (cal/mol) | NRTL A21 (cal/mol) | NRTL Alpha | NRTL B12 (cal/mol.K) | NRTL B21 (cal/mol.K) | NRTL C12 (cal/mol.K²) | NRTL C21 (cal/mol.K²) |
| **UNIQUAC** | PR kij | UNIQUAC A12 (cal/mol) | UNIQUAC A21 (cal/mol) | UNIQUAC B12 (cal/mol.K) | UNIQUAC B21 (cal/mol.K) | UNIQUAC C12 (cal/mol.K²) | UNIQUAC C21 (cal/mol.K²) | Not used |
| **Chao-Seader** | Not used | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Grayson-Streed** | Not used | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Lee-Kesler-Plöcker** | LKP kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **Raoult's Law** | Not used | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **COSMO-SAC (JCOSMO)** | PR kij | Not used | Not used | Not used | Not used | Not used | Not used | Not used |
| **IAPWS-IF97 Steam Tables** | Not used | Not used | Not used | Not used | Not used | Not used | Not used | Not used |

# 36. Property Package Methods and Correlation Profiles

The following table lists the methods and correlations used by the Property Packages to do the requested calculations. Thermal and transport properties are calculated using experimental data when available.

| Property Package | Vapor Phase | | | Liquid Phases | | | | |
|---|---|---|---|---|---|---|---|---|
| | Fugacity | Enthalpy / Entropy / Cp/Cv | Thermal Conductivity * | Viscosity | Density | Fugacity | Enthalpy / Entropy / Cp/Cv | Thermal Conductivity * | Viscosity * | Density * |
| **Peng-Robinson (PR)** | EOS | EOS | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | EOS | Latini | Letsou-Stiel | EOS / Rackett |
| **Peng-Robinson-Stryjek-Vera 2 (PRSV2)** | EOS | EOS | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | EOS | Latini | Letsou-Stiel | EOS / Rackett |
| **Soave-Redlich-Kwong (SRK)** | EOS | EOS | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | EOS | Latini | Letsou-Stiel | EOS / Rackett |
| **Peng-Robinson / Lee-Kesler (PR/Lee-Kesler)** | EOS | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **UNIFAC** | PR EOS / Ideal | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | PR EOS / Ideal | UNIFAC | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **UNIFAC-LL** | PR EOS / Ideal | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | PR EOS / Ideal | UNIFAC-LL | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **Modified UNIFAC (Dortmund)** | PR EOS / Ideal | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | PR EOS / Ideal | MODFAC | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **NRTL** | PR EOS / Ideal | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | PR EOS / Ideal | NRTL | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **UNIQUAC** | PR EOS / Ideal | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | PR EOS / Ideal | UNIQUAC | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **Chao-Seader** | EOS | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | Lee-Kesler | Latini | Letsou-Stiel | EOS / Rackett |
| **Grayson-Streed** | EOS | Lee-Kesler | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | Lee-Kesler | Latini | Letsou-Stiel | Rackett |
| **Lee-Kesler-Plöcker** | EOS | EOS | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | EOS | EOS | EOS | Latini | Letsou-Stiel | Rackett |
| **Raoult's Law** | Ideal | Ideal | Ely-Hanley | Lucas / Jossi-Stiel-Thodos | Ideal | Ideal | Ideal | Latini | Letsou-Stiel | Rackett |
| **IAPWS-IF97 Steam Tables** | IAPWS-IF97 Water/Steam Formulation | | | | | | | | | |
| **IAPWS-08 Seawater** | IAPWS-08 Seawater Properties Formulation | | | | | | | | | |
| **Black-Oil** | Black-Oil Correlations (Lee-Kesler, Riazi, Standing, etc.) | | | | | | | | | |

• can also be calculated using experimental data if available on the compound database.

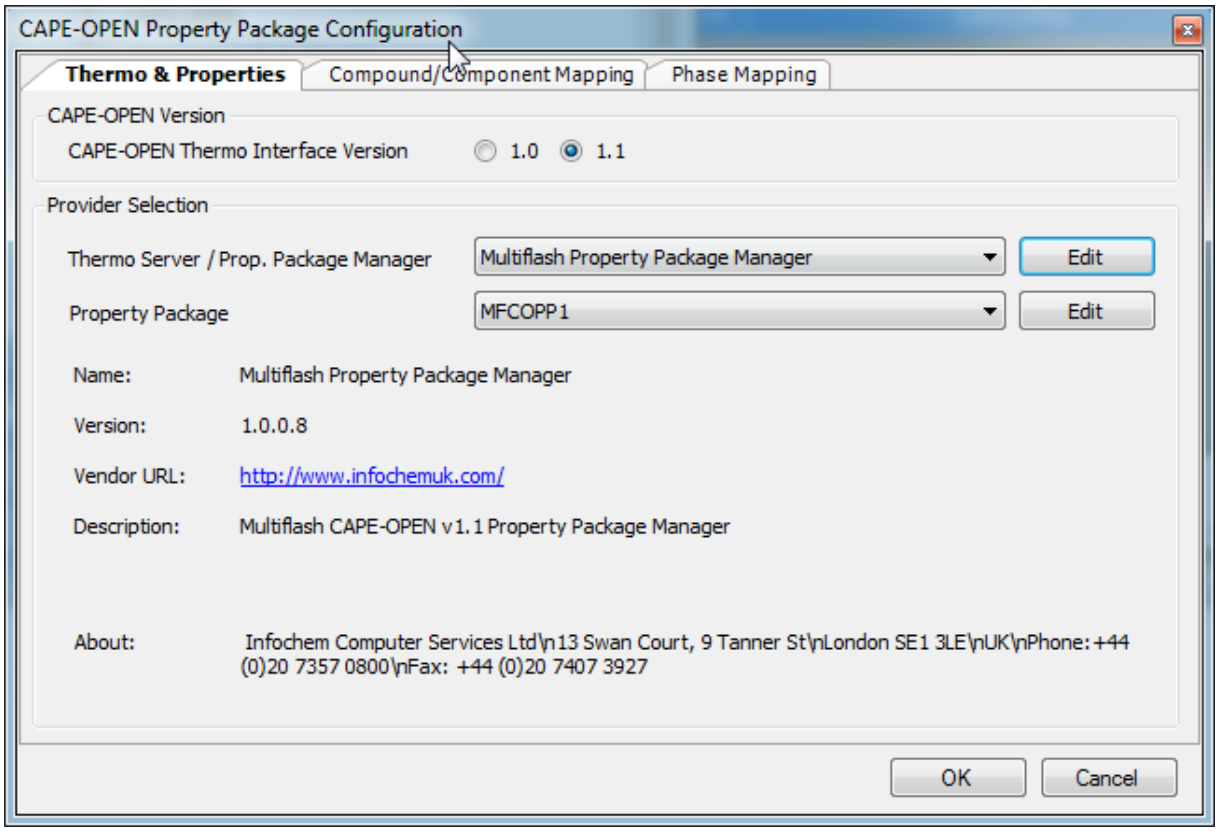
# 37. CAPE-OPEN Subsystem

## 37.1. Introduction

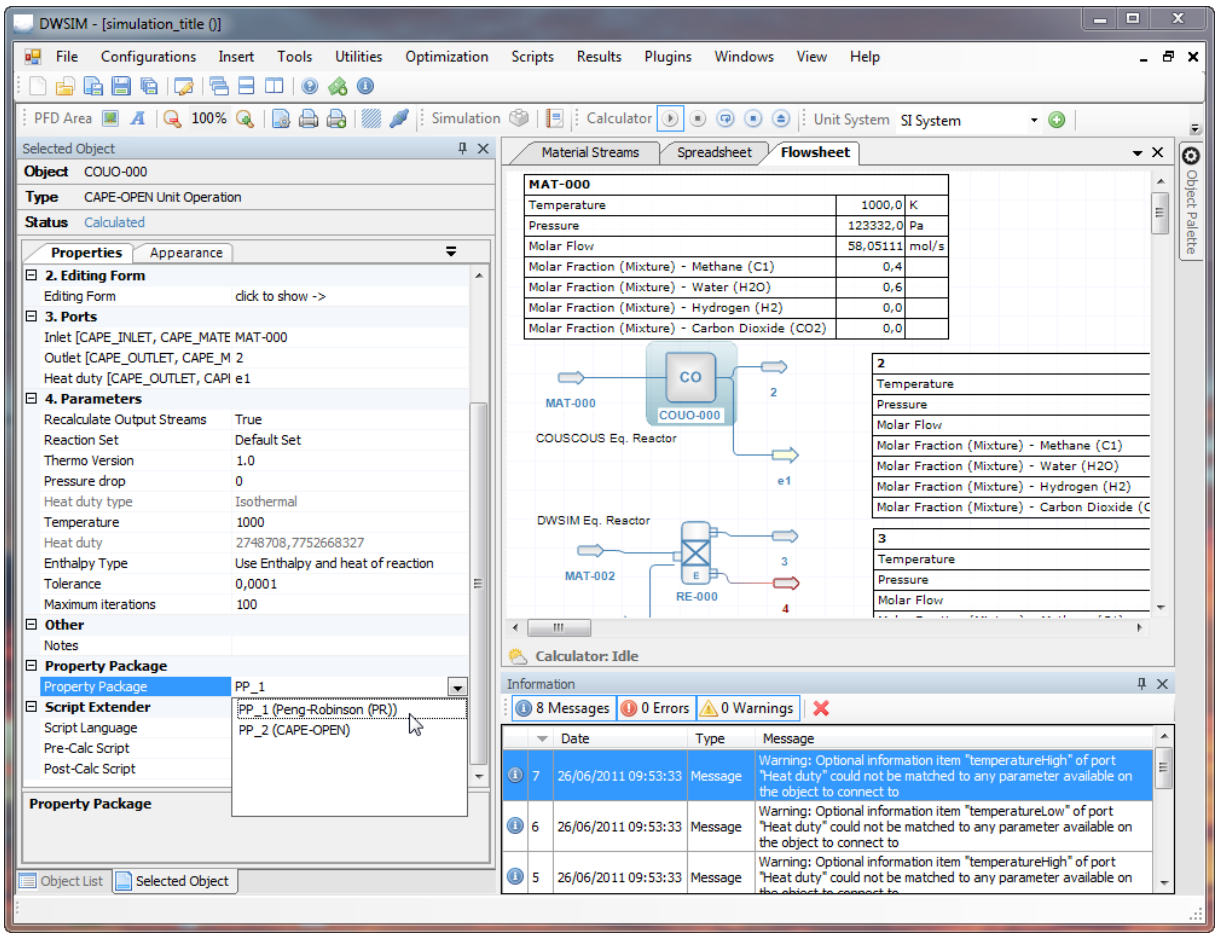
CAPE-OPEN standards are the uniform standards for interfacing process modelling software components developed specifically for the design and operation of chemical processes. They are based on universally recognized software technologies such as COM and CORBA. CAPE-OPEN standards are open, multiplatform, uniform and available free of charge. They are described in a formal documentation set.

DWSIM supports a number of CAPE-OPEN features, including:

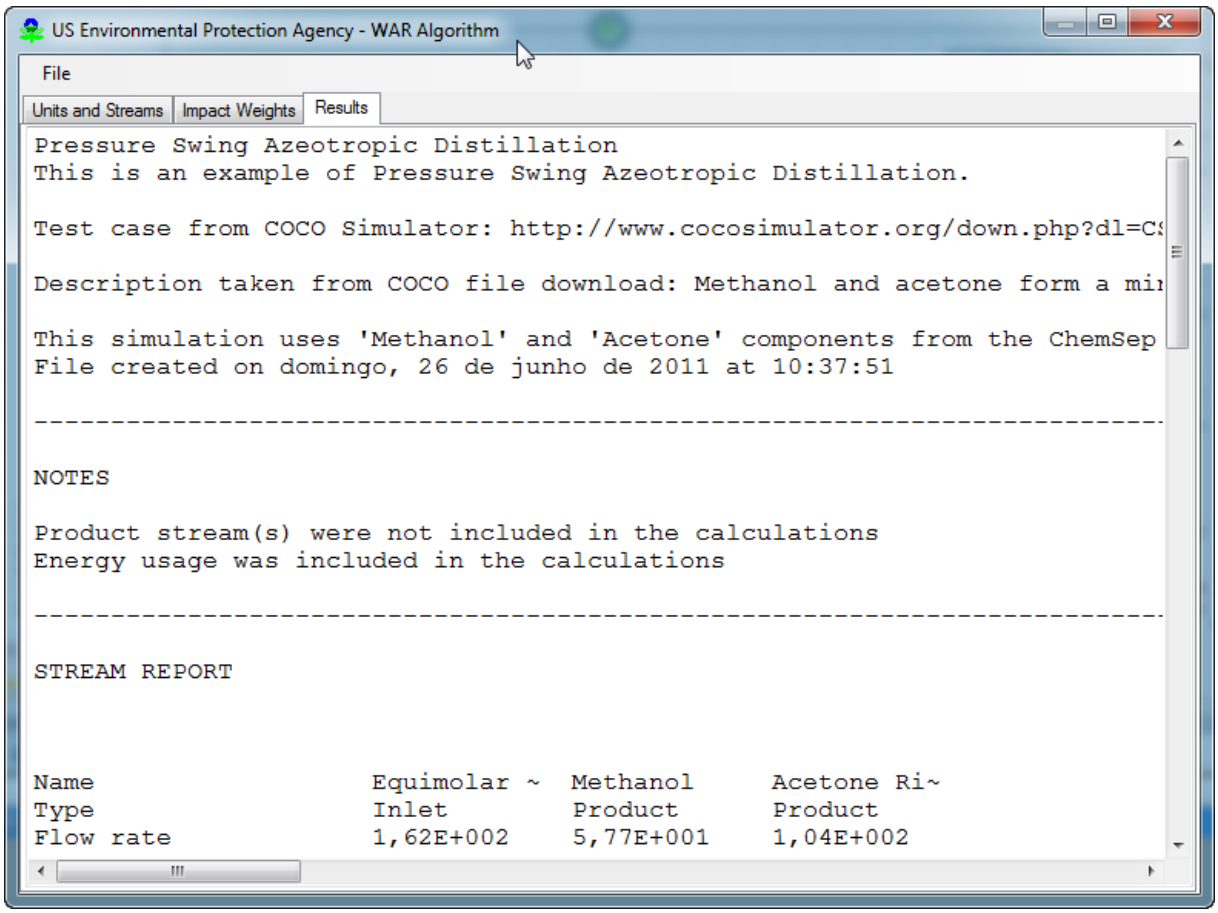
**Property Packages (Thermo Specs 1.0 and 1.1)** You can use external CAPE-OPEN thermodynamic equilibrium and property calculators as Property Packages in DWSIM. Integration is done transparently. You'll only have to map the external property package components and phases to the ones in the internal DWSIM databases.




**Unit Operations** CAPE-OPEN Unit Operations can be added to DWSIM flowsheets and connected to/from energy and material streams just as normal DWSIM Unit Operations. DWSIM also implements the CAPE-OPEN Reaction interfaces so you can use your CAPE-OPEN Reactor Model together with DWSIM and manage your reactions using the Reactions Manager as usual.




**Flowsheet Monitoring Objects** DWSIM support CAPE-OPEN plugins, 'aka' Flowsheet Monitoring Objects, such as the WAR Add-in created by William Barrett, USEPA:

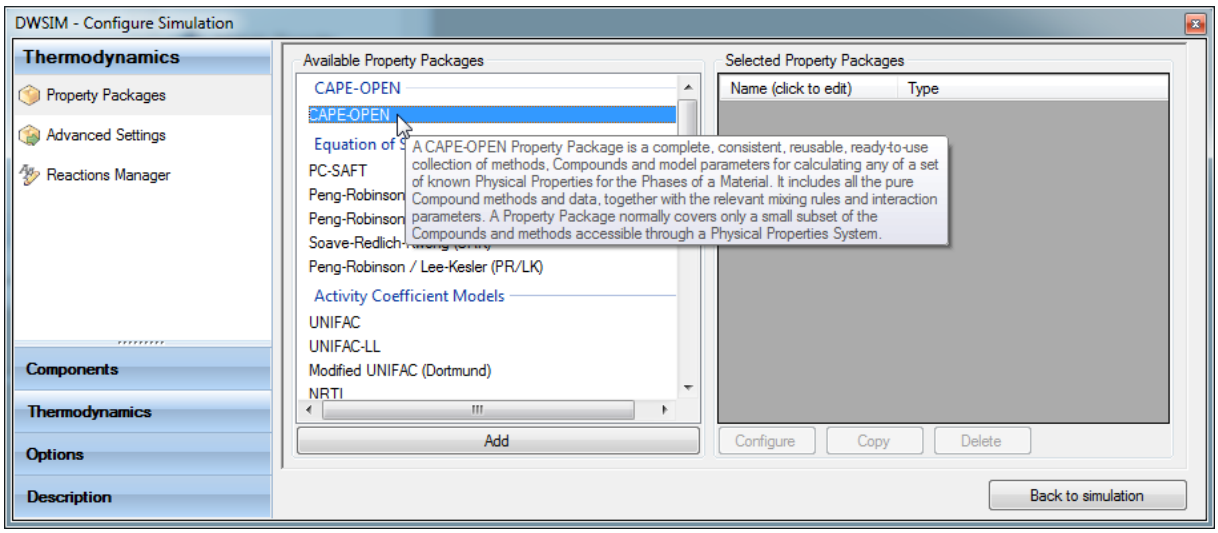



CAPE-OPEN Property Packages, CAPE-OPEN Unit Operations, DWSIM Property Packages and DWSIM Unit Operations can work together on any possible combination. For instance, you can use a DWSIM Property Package as the thermodynamics provider for a CAPE-OPEN Unit Operation in the same way you can use a CAPE-OPEN Property Package as the thermodynamics provider for a DWSIM Unit Operation.

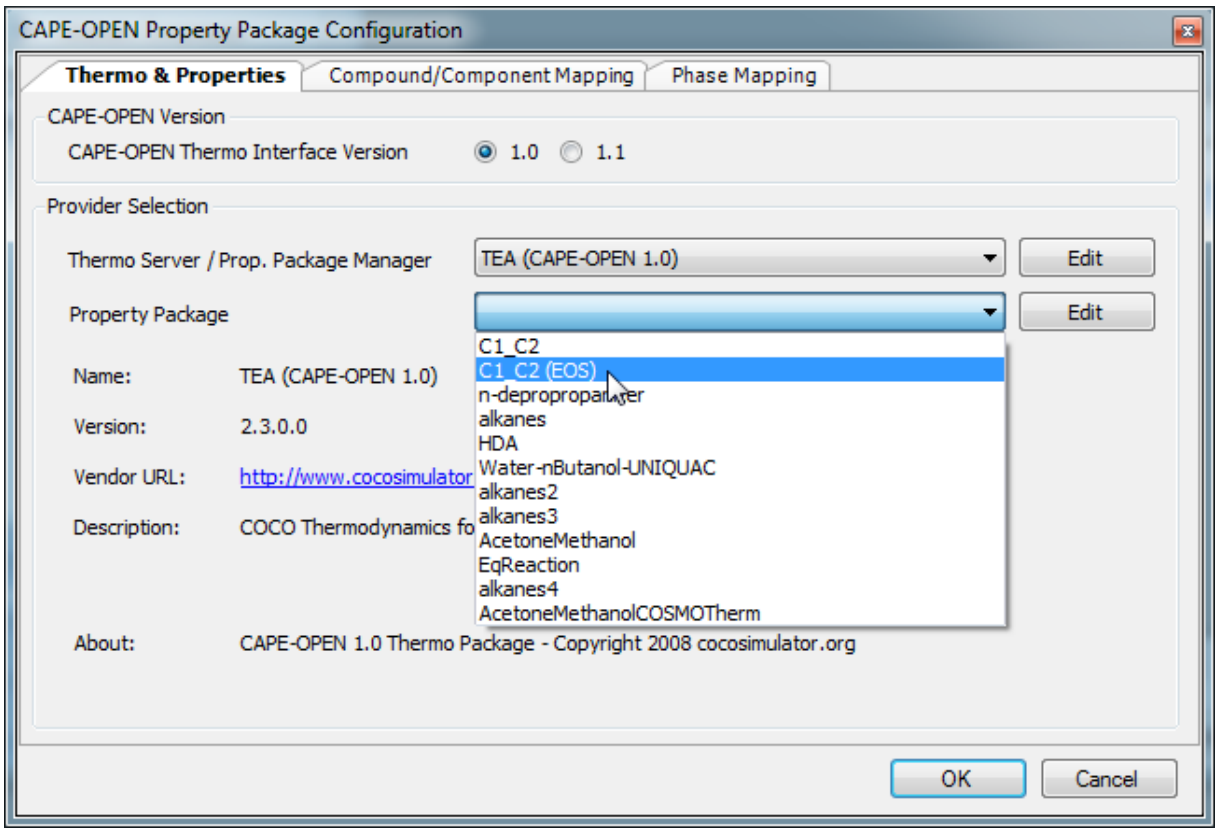
## 37.2. Using external components

### 37.2.1. Property Packages

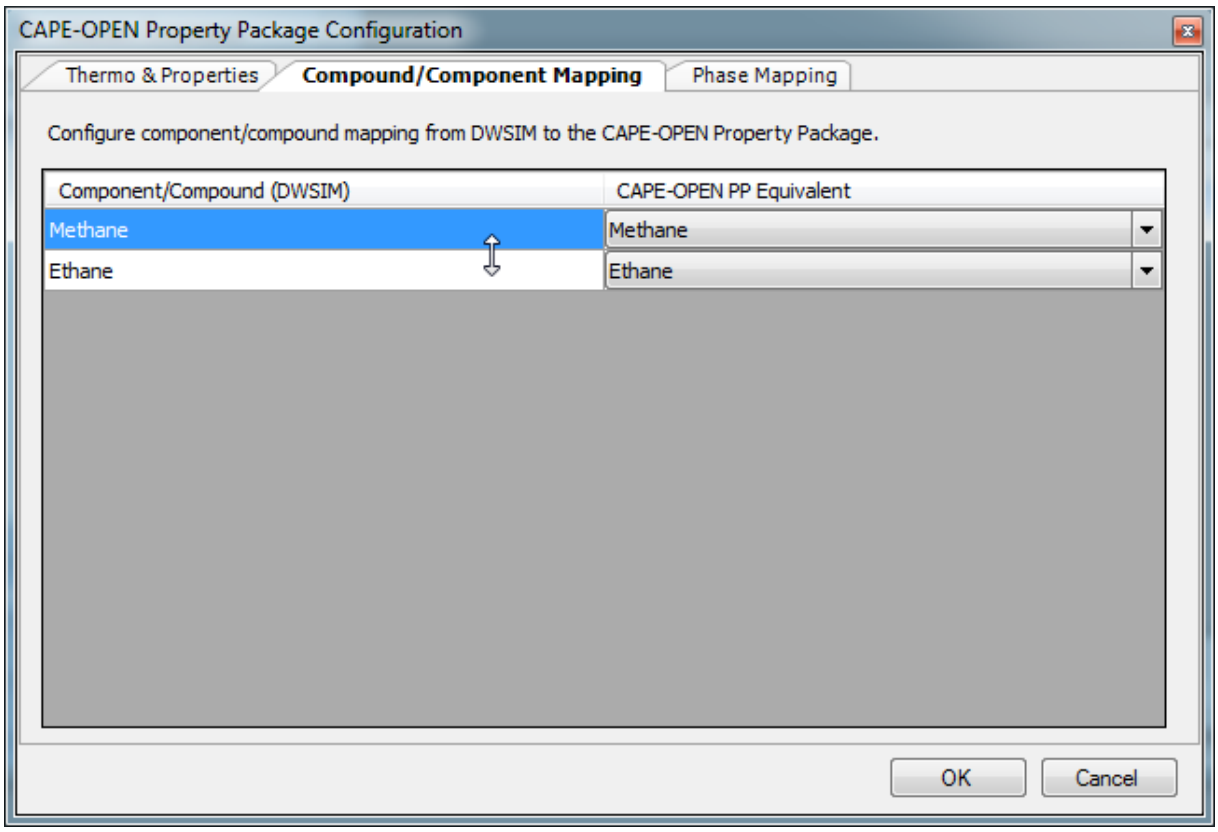
To use external CAPE-OPEN Property Packages in DWSIM, add a Property Package of the “CAPE-OPEN” type to the flowsheet:

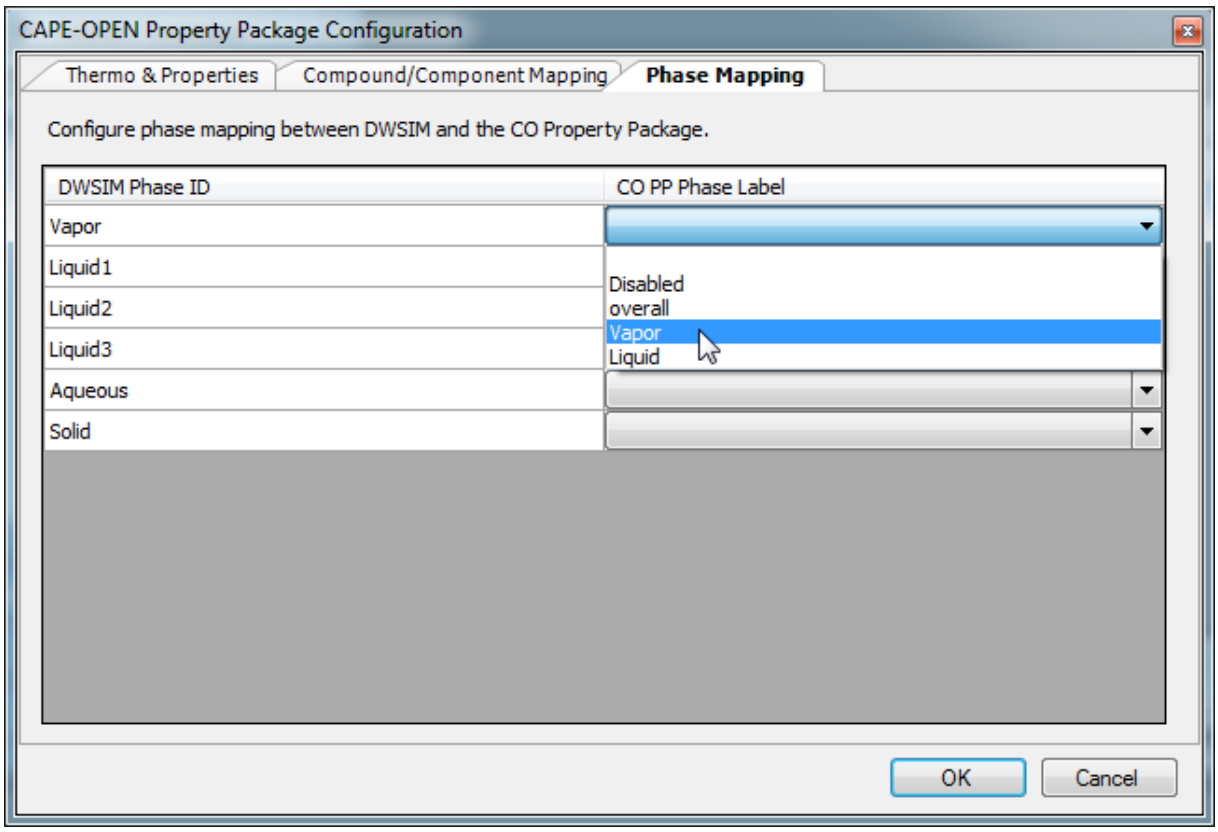



After that, click on “Configure” to setup your property package. On the window that appears, select a Thermo Server or Property Package Manager, depending on the CAPE version you chose. After selecting the server, a list of available Property Packages for that server/provider should be available for selection on the PP combo box:




After selecting your Property Package, you can edit it by clicking on the “Edit” button. If the package was just selected or you’ve done changes to its compounds or phases, you MUST map compounds and phases to DWSIM equivalents on the “Compound/Component Mapping” and “Phase Mapping” tabs:



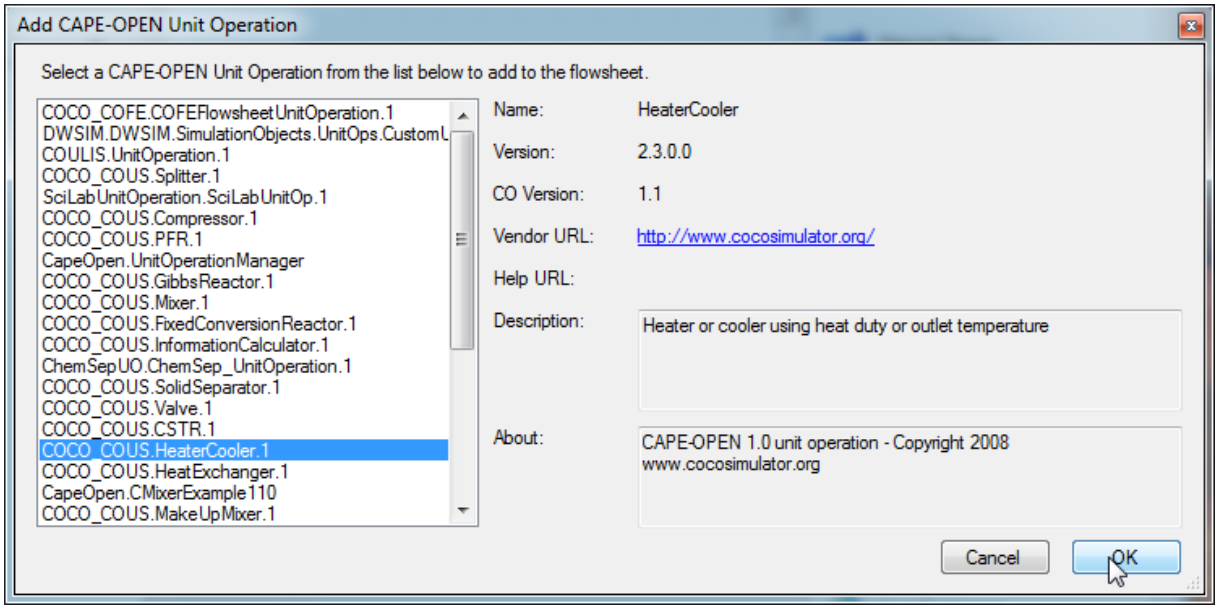




**Attention:** If your PP lists “Overall” or “Mixture” as a phase, you should not associate it with any of DWSIM phases. If a DWSIM phase doesn’t exist in the PP, select “Disable” as its label - DO NOT leave it blank! DWSIM and the CAPE-OPEN PP should have exactly the same number of compounds, even if

some of them aren't used by DWSIM or by the Property Package. After the compound and phase mapping steps, click "OK" and you're ready to go.

### 37.2.2. Unit Operations

To add CAPE-OPEN Unit Operations to DWSIM flowsheets, drag and drop the "CAPEOPEN Unit Operation icon to the flowsheet. A selection window will appear where you should choose which Operation will be added:

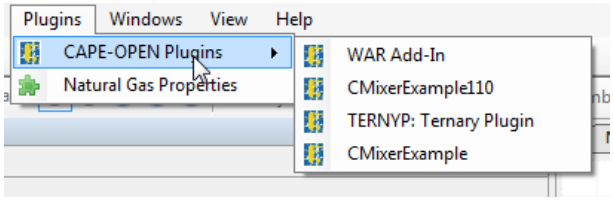



After the Unit is added, it works the same way as a DWSIM Unit Operation. You can edit its connections to Material and Energy Streams, parameters and access its 'Edit' window through the Property Editor.

**Attention:** The "Property Package" setting for the CAPE-OPEN Unit Operation has no effect. It accesses the property packages linked to Material Streams to do calculations. It is recommended that all streams connected to a CAPE-OPEN UO have the same associated Property Package to ensure consistency of the results obtained after flowsheet calculation.

### 37.2.3. Flowsheet Monitoring Objects

CAPE-OPEN Add-ins can be accessed through the "Plugins" menu item:

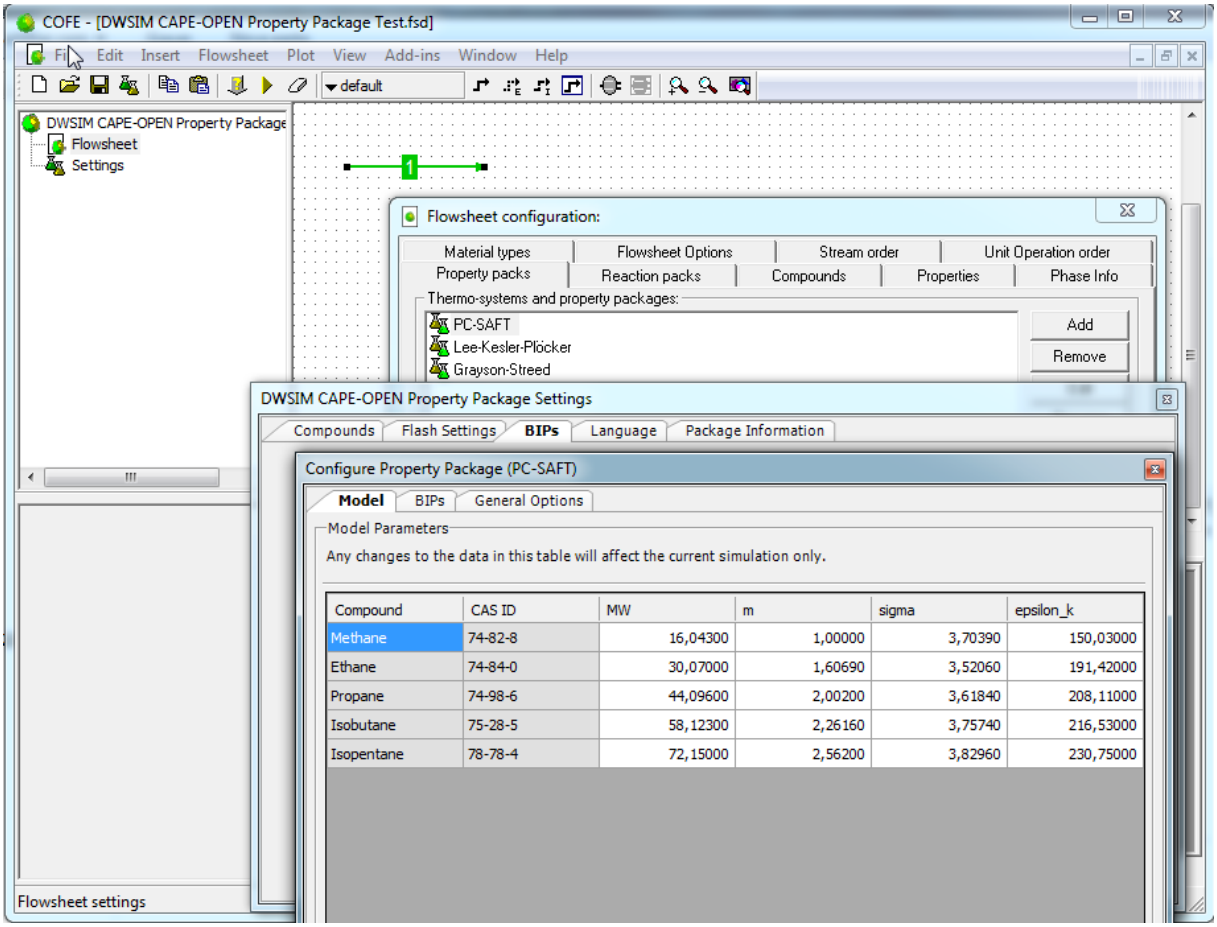



## 37.3. Other features

### 37.3.1. Using DWSIM as a CAPE-OPEN Property Package Manager (Thermo 1.1)

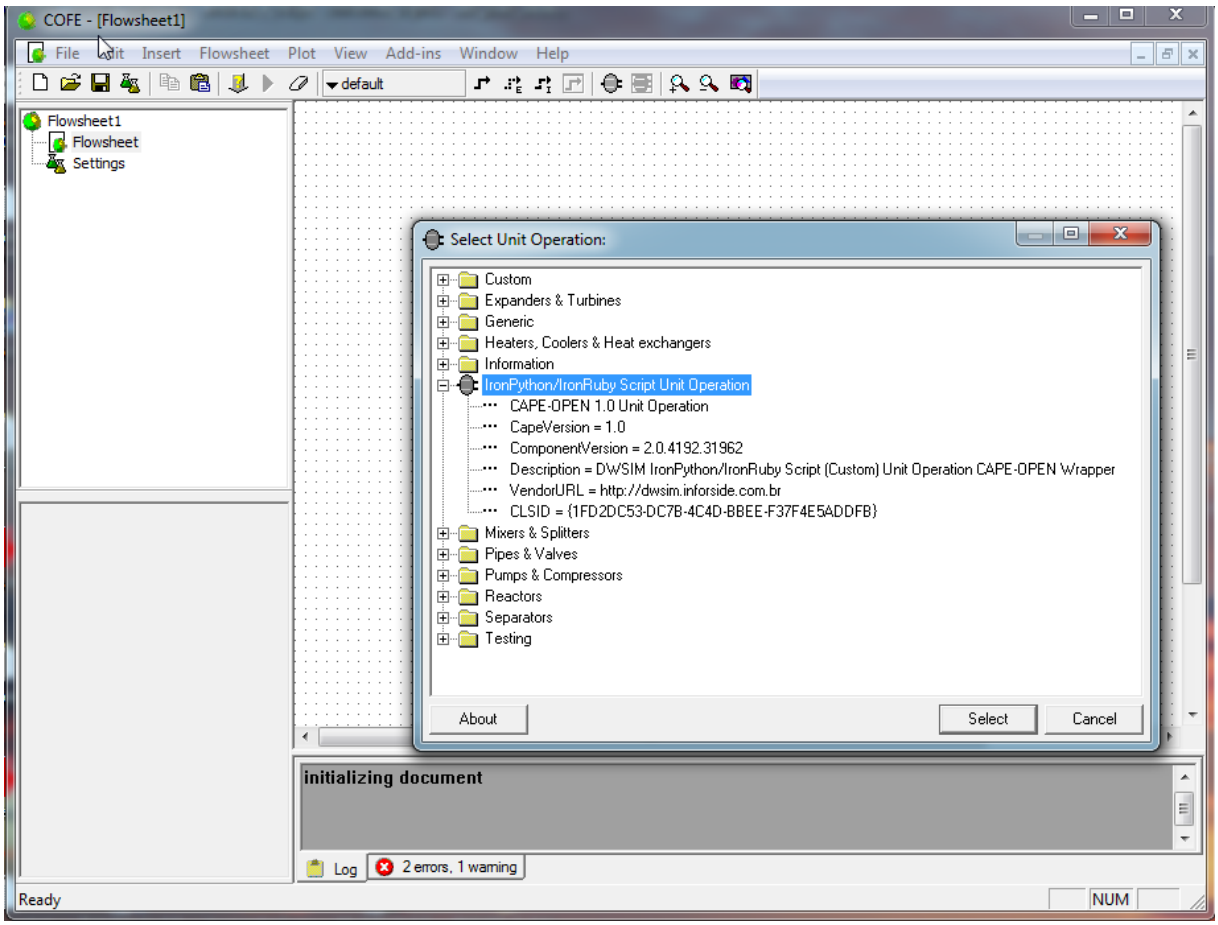
If you registered DWSIM types during the installation process, DWSIM will expose its Property Packages to CAPE-OPEN compliant simulators through Thermo 1.1 Interfaces (if your simulator is only Thermo

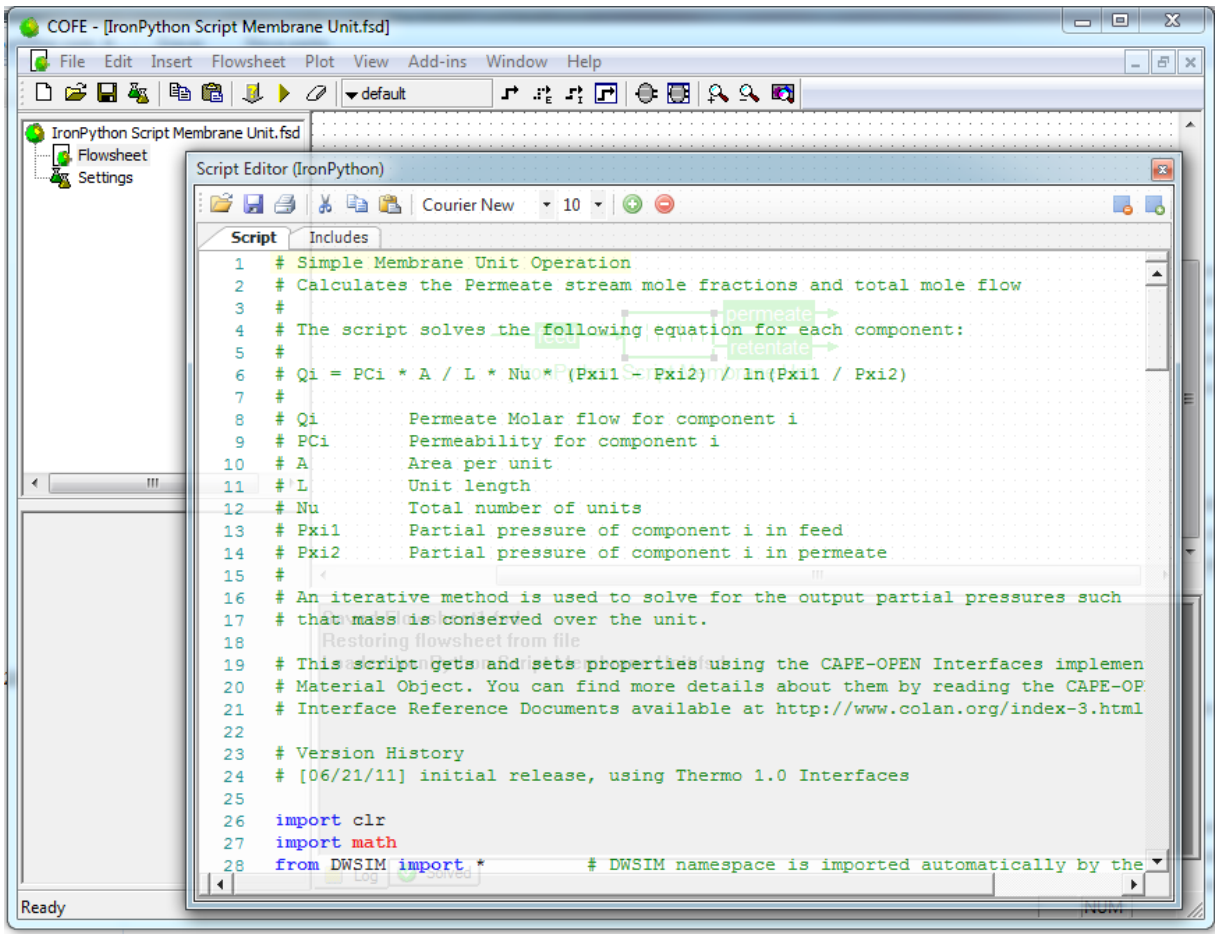
1.0 compliant, DWSIM Thermo Server will not be selectable). The configuration window allows you to add/remove compounds, configure flash settings, set the GUI language and edit model parameters and binary interaction parameters (BIPs). You can download an example of DWSIM Property Packages used as property and equilibrium calculators in COCO/COFE on SourceForge.




### 37.3.2. Using the Script Unit Operation in CAPE-OPEN compliant simulators

If you registered DWSIM types during the installation process, the Custom Unit Operation will be exposed to CAPE-OPEN compliant simulators as “IronPython Script Unit Operation”. You can download an example of the Script UO being used to model a simple membrane separation process in COCO/COFE on SourceForge.


COFE - [Flowsheet1]
File Edit Insert Flowsheet Plot View Add-ins Window Help
default
Flowsheet1
Flowsheet
Settings
Select Unit Operation:
Custom
Expanders & Turbines
Generic
Heaters, Coolers & Heat exchangers
Information
IronPython/IronRuby Script Unit Operation
CAPE-OPEN 1.0 Unit Operation
CapeVersion = 1.0
ComponentVersion = 2.0.4192.31962
Description = DWSIM IronPython/IronRuby Script (Custom) Unit Operation CAPE-OPEN Wrapper
VendorURL = http://dwsim.inforside.com.br
CLSID = {1FD2DC53-DC7B-4C4D-BBEE-F37F4E5ADDFB}
Mixers & Splitters
Pipes & Valves
Pumps & Compressors
Reactors
Separators
Testing
About
Select
Cancel
initializing document
Log
2 errors, 1 warning
Ready
NUM


COFE - [IronPython Script Membrane Unit.fsd]
File Edit Insert Flowsheet Plot View Add-ins Window Help
default
IronPython Script Membrane Unit.fsd
Flowsheet
Settings
Script Editor (IronPython)
Courier New
10
Script
Includes
1 # Simple Membrane Unit Operation
2 # Calculates the Permeate stream mole fractions and total mole flow
3 #
4 # The script solves the following equation for each component:
5 #
6 # Qi = PCi * A / L * Nu * (Pxi1 - Pxi2) / ln(Pxi1 / Pxi2)
7 #
8 # Qi Permeate Molar flow for component i
9 # PCi Permeability for component i
10 # A Area per unit
11 # L Unit length
12 # Nu Total number of units
13 # Pxi1 Partial pressure of component i in feed
14 # Pxi2 Partial pressure of component i in permeate
15 #
16 # An iterative method is used to solve for the output partial pressures such
17 # that mass is conserved over the unit.
18
19 # This script gets and sets properties using the CAPE-OPEN Interfaces implemen
20 # Material Object. You can find more details about them by reading the CAPE-OP
21 # Interface Reference Documents available at http://www.colan.org/index-3.html
22
23 # Version History
24 # [06/21/11] initial release, using Thermo 1.0 Interfaces
25
26 import clr
27 import math
28 from DWSIM import * # DWSIM namespace is imported automatically by the
permeate
retentate
Restoring flowsheet from file
Ready

# Part IX.

# Additional Resources

## 38. Online Courses and Videos

- Official YouTube Channel: Link
- FOSSEE Spoken Tutorials: Link
- Nicolas Spogis' YouTube Channel: Link
- Vídeo-Tutoriais em Português pelo Prof. Delano (UFBA): Link
- Videotutoriales en Español: Link

## 39. Learning Resources

- FOSSEE Resources: Link
- Download Solved Problems from Completed Books: Link
- Minicurso: Introdução ao DWSIM: Link
- Tutorial em Português: Link
- Curso de DWSIM Básico: Link
- Introducción a DWSIM (en Español): Link
- Basic Process Modeling and Simulation with DWSIM (New UI) (Unofficial Udemy Course): Link
- Process Simulation using DWSIM by Prof. P. R. Naren, SASTRA University: Link

## 40. Programming Help

- API Documentation: Link
- IronPython Script Snippets: Link
- Python Scripting Support Forums: Link

# List of Figures

# List of Tables

## References


[1] M. Michelsen, J. Mollerup, *Thermodynamic Models: Fundamentals and Computational Aspects*, Tie-Line Publications, Denmark, 2007.

[2] H. Z. Kister, *Distillation Design*, McGraw-Hill Professional, 1992.

[3] J. D. Seader, E. J. Henley, *Separation Process Principles*, Wiley, 2005.

[4] T. Tinker, *Trans. ASME*, 1958, **36**, 80.

[5] W. McCabe, J. Smith, P. Harriott, *Unit Operations of Chemical Engineering*, of *Chemical Engineering Series*, McGraw-Hill Education, 2005.

[6] J. Smith, *Intro to Chemical Engineering Thermodynamics*, McGraw-Hill Companies, New York, 1996.

[7] D.-Y. Peng, D. B. Robinson, *Industrial and Engineering Chemistry Fundamentals*, 1976, **15**, 59–64.

[8] S. Vitu, J.-N. Jaubert, F. Mutelet, *Fluid Phase Equilibria*, 2006, **243**, 9 – 28.

[9] G. Soave, *Chemical Engineering Science*, 1972, **27**, 1197–1203.

[10] C. H. Whitson, M. R. Brule, *Phase Behavior (SPE Monograph Series Vol. 20)*, S. of Petroleum Engineers (Ed.), Society of Petroleum Engineers, 2000.

[11] R. Stryjek, J. H. Vera, *The Canadian Journal of Chemical Engineering*, 1986, **64**, 820–826.

[12] K. C. Chao, J. D. Seader, *AICHE Journal*, 1961, **7**, 599–605.

[13] H. G. Grayson, C. W. Streed, *N/A*, 1963, **VII**.

[14] J. W. Kang, V. Diky, M. Frenkel, *Fluid Phase Equilibria*, 2015, **388**, 128 – 141.

[15] H. Renon, J. M. Prausnitz, *AICHE Journal*, 1968, **14**, 135–144.

[16] R. Reid, *The Properties of Gases and Liquids*, McGraw-Hill, New York, 1987.

[17] B. I. Lee, M. G. Kesler, *AIChE Journal*, 1975, **21**, 510–527.

[18] I. H. Bell, J. Wronski, S. Quoilin, V. Lemort, *Industrial & Engineering Chemistry Research*, 2014, **53**, 2498–2508.

[19] M. R. Riazi, H. A. Al-Adwani, A. Bishara, *Journal of Petroleum Science and Engineering*, 2004, **42**, 195–207.

[20] M. R. Riazi, *Characterization and properties of petroleum fractions*, ASTM, 2005.

[21] B. K. Harrison, W. H. Seaton, *Industrial and Engineering Chemistry Research*, 1988, **27**, 1536–1540.

[22] J. Marrero, R. Gani, *Fluid Phase Equilibria*, 2001, **183-184**, 183–208.

[23] R. A. Heidemann, A. M. Khalil, *AIChE Journal*, 1980, **26**, 769–779.