MultiComponentMultiPhase – a framework for thermodynamic properties in Modelica

Johan Windahl[1] Katrin Prölss[1] Maarten Bosmans[2] Hubertus Tummescheit[1] Eli van Es[2] Awin Sewgobind[2]

[1]Modelon AB, Lund, Sweden,
{johan.windahl,katrin.prolss,hubertus.tummescheit}@modelon.com
[2]VORtech, Delft, Netherlands, {maarten.bosmans,eli.vanes,awin.sewgobind}@vortech.nl

Abstract

This paper describes the development and requirement specification of an open-source framework for multi-phase multi-component thermo properties in Modelica. The goal is to have a standardized interface to multi-component multi-phase fluids with access to external property packages in Modelica. This will make it easier to develop models for e.g. the process industry. The library uses a model based interface and implications of such a design are analyzed and compared with the traditional function based interface.

The work has been carried out in collaboration with Modelon AB and VORtech within the umbrella of *Methods and tools* as part of the CleanSky SGO project.

Keywords: CAPE-OPEN, FluidProp, RefProp, fluid properties, flash calculations

1 Introduction

Properties of working fluids define the achievable baseline accuracy for fluid system simulations. The availability of properties for steam and flue gases initiated the use of Modelica in the power industry, where it today is a well-established technology with several commercial and open source libraries available (Modelica Libraries, 2015). High quality fluid properties are laborious to produce and their non-availability is therefore a typical blocking argument for the use of a certain tool or technology.

Published work of modeling chemical process systems in Modelica exists, see (Tummescheit *et al*, 2002; Dietl *et al* 2011; Baharev *et al* 2012). But until today, the use of Modelica has not been widely spread to this area even though it would be well-suited to describe these processes. Modelica is equation based, similar to gPROMS, which is well established in the process industry. However, it is lacking a standardized interface for multi-component multi-phase fluid properties. For an overview of equation oriented methods for chemical and related process flowsheets, see (Morton, 2002).

In this project a Modelica library for multi-phase multi-component fluids has been developed together with an external C/C++ Modelica property interface with back ends to CAPE-OPEN, RefProp (Lemmon *et al*, 2013) and FluidProp (Fluid property library, 2015). The framework also contains a Modelica library for distillation processes for verification and testing of the media interface design.

2 Background

Modelica.Media is a freely available Modelica package contained in the Modelica standard library. It consists of property models from ideal gases up to high accuracy models of WaterIF97 and R134a. The current version 3.2.1 is restricted to pure two-phase or single phase mixtures. The goal is to extend the capabilities of Modelica.Media with support of multi-component multi-phase mixtures and to find an interface structure that is user-friendly both from an implementer and end-user perspective.

In order to collect input for the interface design, a meeting in Delft (Oct 2013) was held that gathered 17 people from academia, industry and members of the Modelica design group. During the meeting it became clear that technical challenges to implement such a property interface efficiently using the current structure of Modelica.Media are high. A large part of the project has therefore been focused on finding an interface structure both within and outside the limits of the Modelica specification 3.3.

2.1 Application overview

The framework developed in this project must cover a wide range of processes. The following types of processes have been identified where multi-phase multi-component fluids are used:

2.1.1 Thermodynamic cycle processes

Typical applications are refrigeration, heat pumps (vapor compression cycles) and Organic Rankine cycles using a blend of different working fluids, which may be available in RefProp. Models are usually characterized by a homogenous treatment of properties and flows, usually modelled with mass based units as is common in the energy and power industry. System may operate in overcritical conditions.

These applications fit well into the Modelica.Media structure as it was created with these processes in mind. Flexible models are required due to the number of present phases in a model can change during simulation.

2.1.2 Thermal separation processes (with and without chemical reactions)

Typical applications are rectification and absorption processes where the numbers of phases usually are limited to vapor and liquid and models use a mole basis. It is common to include chemical reactions and fluids are often taken from databases via CAPE-OPEN.

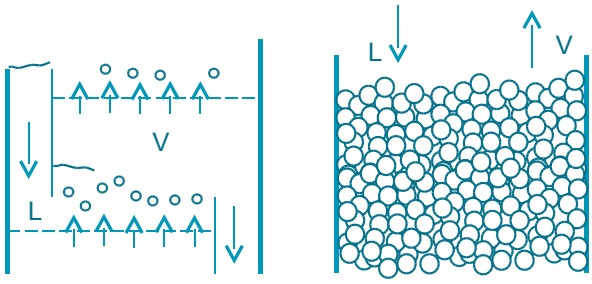

Figure 1. Example of thermal separation processes. Column tray (left) and column packing (right).

2.1.3 Transport of multiphase flows

Typical applications are compositional pipe network simulations, which are computationally expensive. In this case multiple phases may coexist and a homogenous assumption is not valid.

2.2 Context

Following types of usage are possible:

- Dynamic simulation
- Steady-state simulation
- Optimization

From an interface perspective a difference is in the requirement of differentiation of properties. Solving an optimal control problem also involves the Hessian (Boyd *et al*) Even if these can be calculated numerically the performance and robustness increase if analytical derivatives can be provided (Åkesson *et al*, 2012) An interface should therefore support the usage of analytical derivatives if these can be provided.

2.3 Phase equilibrium calculation

A phase equilibrium calculation determines subject to specified constraint, e.g. fixed pressure and temperature, present phases and the composition and fraction of each present phase. It is an iterative calculation which often uses specialized algorithms; see (Parekh *et al*, 1998; Gernert *et al*, 2014).

Phase equilibrium calculations are time consuming and will dominate the total CPU usage, up to 95% according to (Trapp, 2014). Similar numbers have also been observed in this work.

To achieve competitive performance:

1. The number of phase equilibrium calculations should be minimized. This can be achieved by designing fluid and application library interfaces so that calculation result can be shared between components. This may require expanding the connector class with additional variables to avoid redundant calculations.
2. For each phase equilibrium calculation, the number of iterations inside these algorithms needs to be minimized. This can be achieved by providing good iteration guess values.

3 Modelica media interface

There are several possibilities to define an interface due to Modelica support both models and functions.

The first step in the design process was to analyze and list requirements.

3.1.1 General media requirements

1. User-friendliness and structure
   - The structure should be easy to understand and use. Implementation details such as external objects should be hidden from the user.
   - Interface that can be used by both a native Modelica and external C-code media implementation.
   - Calculation of parameters, preferable also structural parameters, from property functions that may have a dependency on external code/external object.
2. Possibility to create a media structure using inheritance. To easily create new media from existing templates and interfaces.
3. Performance
   - The interface should be designed with efficiency in mind. It should support differentiation of properties and caching through external objects.
   - Possibility to provide additional information about present phases that can be used to simplify or skip computational expensive phase equilibria calculations.

DOI 10.3384/ecp15118653

- Support to add initial guesses of start values.

4. Derivative functions
   - It should be possible to specify derivative functions needed for: state variable transformations, index reduction and generation of analytic Jacobians.
5. Additional
   - It should be possible to extend the interface with new functionality such as reaction properties.

3.1.2 Multi-component multi-phase requirements

There is a wide range of different properties that may be needed but here we consider the most basic usage.

- Calculation of phase equilibrium.
- Calculation of properties for a specified phase at phase equilibrium.
- Support for common properties such as fugacity and activity coefficients.
- Support of both molar and mass based properties. The chemical process industry usually works in mole while the energy industry works in mass based units.
- No restriction of the number of supported phases.

3.1.3 Additional requirement

- Take advantage of unit declaration. The possibility to declare units is a powerful feature in Modelica that should be used.
- Uniquely identify phases and compounds.

3.2 Design restrictions

When designing a property interface in Modelica following restrictions (Modelica Association, Specification, 2015) needs to be considered.

1. Functions need to be pure, i.e. they are not allowed to have any side effect. This is a fundamental assumption in Modelica that makes it possible for a tool to apply symbolic transformation and rearrange calculations.
2. Records are not suited to be used as function inputs. This is due to the fact that it is not allowed to specify a derivative function if the record contains a non-real variable. It is also not efficient to use a record with additional variables due to all variables needs to be considered for differentiation. A record is also not allowed to contain an external object.
3. It is not possible to access a previous value of a continuous variable. There is no such operator in Modelica.

3.2.1 Iterative algorithms

A consequence of the restrictions is that it is inefficient to implement explicit iterative algorithms in native Modelica. This is due to that functions are not allowed to have internal memory between function calls and there is no operator to access a previous value of a continuous variable. The start value of a variable in an algorithm is therefore equal to its start attribute during simulation. This is a drawback for function based media property calculations that need to be solved by an iteration process. If instead a model based interface is used and the algorithm is replaced with an implicit equation formulation, the tools non-linear solver can use the last solution point as a start for the next iteration (Olsson *et al*, 2005).

3.3 Model based interface structure

Based on the requirements and restrictions it was decided that the interface structure should be model based (this does not hinder the implementation to be function based).

Main advantages with a model based structure are:

- Possible to implement a medium using a declarative approach as demonstrated in (Olsson *et al*, 2005). It makes it possible to quickly create a medium with good performance, see section 5.1
- Possible to share interface between an external code based media and a native Modelica based media.
- User friendly as implementation details can be hidden from the user and the possibility to work graphically and by that taking advantage of a tool support of e.g. unit conversion.
- Possible with a minimalistic interface as it is not necessary to create new models for new input combinations. The model based interface may not specify the causality of the variables.
- Avoid the need for a tool to support common-sub expression elimination as the result of an expensive calculation can be stored in a model.
- Possible to use block and models in an implementation, e.g. the Modelica Standard Library tables.

There are also disadvantages that should be considered:

- A model can't be instantiated inside a function which limits the scope where a media calculation can be executed.
- It is user unfriendly to calculate parameters from a media. It requires setting a parameters fixed attribute to false and an additional equation in the initial equation section.

- Not possible to calculate a property on demand as a model needs to be instantiated.

3.3.1 MultiPhaseMixture interface

The current interface structure is shown in **Figure 2**, it consists of a few models and helper functions.

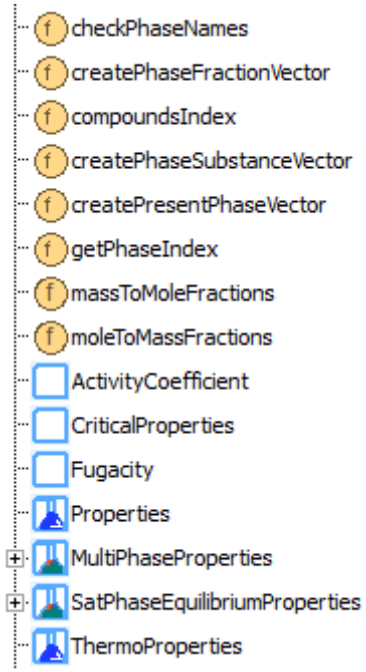

Figure 2. Screenshot of the current interface structure of MultiPhaseMixture. *Note that it is currently under development and is therefore subject to change.*

3.3.2 Example of usage

An example of how to use the ThermoProperties model is shown in Listing 1.

```
model ExampleOfUsage
  package Medium=MultiPhaseMixture.Air_PureModelica;
  Medium.ThermoProperties thermoProperties(
    inputs=MultiPhaseMixture.Interfaces.Inputs.pTY,
    p=100000,
    T=298.15,
    Zm=Medium.reference_Y);
 Density d;
equation
 d= thermoProperties.d;
end ExampleOfUsage;
```

Listing 1. Modelica code showing the usage of a property calculation using the model based interface.

4 External interface

Developing thermodynamic property models for multi-phase multi-component fluids is fairly complex and requires specialist knowledge. There already exist tools like MultiFlash and FluidProp that have been developed in the process industry and in academia over a long time. It is therefore useful for the new Media library to be able to interface with external fluid property tools and databases.

The overall structure of the external framework is illustrated in **Figure 3**.

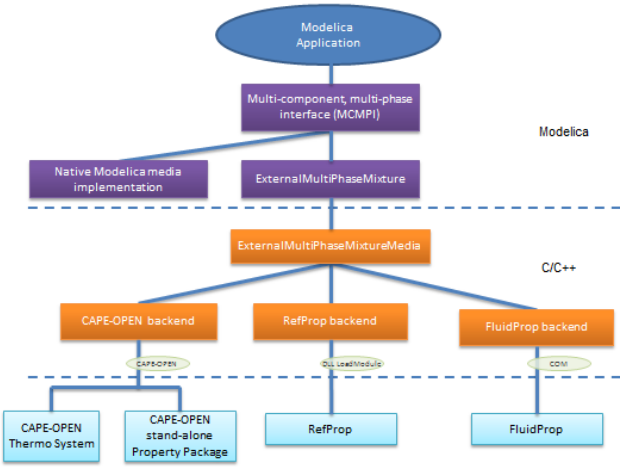

Figure 3. Overview of external framework structure.

4.1 Previous work in Modelica

There exists previous published work with interfacing external properties in Modelica, see (Tummescheit, 2002; Trapp 2014; Wellner 2014). There is also an open-source framework available, ExternalMedia (Casella, 2008), but it is limited to pure two-phase media.

4.2 Using external code in Modelica

Modelica has an external function interface to C which makes it possible to use external routines within a Modelica function. Following issues have been considered when building the new Modelica libraries around external code:

- Differentiation
- Error messages
- Use of external objects

An advantage of using native Modelica code over external code is that the Modelica compiler has access to structural information on the dependency between inputs and outputs. This makes it possible to automatically differentiate, create analytical Jacobians and explore sparsity patterns that will increase robustness and performance of a simulation.

For external functions, derivative information needs to be specified by the user. In the general case it would require full knowledge about the implementation. For thermodynamic properties the effort on providing this information can be decreased by taking advantage of thermodynamic properties definitions. But it is important that there are test cases as wrongly

DOI 10.3384/ecp15118653

calculated derivatives will lead to convergence failure which may be very hard to debug.

Another important aspect is implementation of appropriate error messages, as otherwise the simulation may crash without leaving any information to the user.

4.2.1 External object

With the use of external objects there is a defined way to allocate and de-allocate resources. It is also possible to store internal information between function calls which may be used to cache iteration start values. The C-interface is therefore based on external objects.

Disadvantages are restrictions on how they can be used in Modelica (Modelica Spec, 2015) and it is not clear how they work in combination with symbolic transformation such as inverse functions and sub-expression elimination. We have also encountered bugs related to the use of them, however it seems more stable with more recent versions of Modelica tools.

4.3 Modelica external media interface

MultiPhaseMixture.ExternalMixture is a template that implements the Modelica MultiPhaseMixture interface.

It consists mainly of:

- Functions that call the external C- code interface.
- Wrapper functions for various property and input combinations.
- Calculations of multi-phase properties and derivatives

With the wrapper functions it is possible to specify derivative annotations and support differentiation of external properties.

4.3.1 Derivatives

Neither CAPE-OPEN nor RefProp supports total overall properties derivatives, which may be needed for dynamic simulation, especially for state variable transformation. It was therefore decided that these types of calculations should be implemented on the Modelica side and not in the C-interface. For a pure fluid, the calculations are straight-forward and there are publications available (Thorade *et al*, 2013). For mixtures they are more complicated (Li, 1955). The difficult part is when several phases coexist. In that case they are currently calculated numerically.

```
function density_derh_p
  input Properties state;
  input MExternalMediaObject eo "External object";
  output Real ddhp "Density derivative wrt h at constant pressure";
protected
...
algorithm
  if (state.nbrOfPresentPhases == 1) then
    pd:=state.dpdd_TN_1ph[integer(state.presentPhaseIndex[1])];
    …
    ddhp:= -
state.d_overall*state.d_overall*pt/(state.d_overall*state.d_overall*pd*
cv + state.T_overall*pt*pt);
  elseif (nC == 1 and nP== 2) then
    …
    dpT :=  (vap_s - liq_s)*liq_d*vap_d/(max(liq_d -vap_d,eps));
    ddhp:=-state.d_overall*state.d_overall/(dpT*state.T_overall);
  else
    // multiple phases - calculate ddhp numerically
d_deltah:= Wrapper_phX.density_phX(p=state.p_overall, h=state.h_o
verall+deltah,X=X,
state=calcProperties_phX(p=state.p_overall,h=state.h_overall+deltah,
X=X,eo=eo),eo=eo);
    ddhp:=(d_deltah-state.d_overall)/deltah;
  end if;
```

Listing 2. Code snippet of a Modelica implementation of density derivative wrt to specific enthalpy at constant pressure.

```
protected
  Auxilary.Properties properties;
  final parameter  ExternalMediaObject eo=
ExternalMediaObject(setupInfo);
equation
  if (inputs == Inputs.pT) then
     properties =Auxilary.calcProperties_pTX(p=p,T=T,X=Z,eo=eo);
     d =Auxilary.Wrapper_pTX.density_pTX(p=p,T=T,X=Z,
       state=properties);
```

Listing 3 Code snippet of the MultiPhaseProperties model for the external media template.

4.3.2 Mole vs mole fractions

A recommendation was given to use mole numbers instead of mole fractions (Szczepanski, 2013). The sum of all mole fractions is equal to 1:

$$\sum_{i=1}^{N} x_i = 1 \qquad (1)$$

Mole fractions are not independent and are therefore more difficult to differentiate. This is further explained by (Molerup *et al*, 2002) where they state "*Derivatives with respect to mole fractions are best avoided, as they require a definition of the 'dependent' mole fraction and in addition lead to more complex expressions missing many important symmetry properties*."

What is not described is how this can be applied to a thermodynamic framework for dynamic simulation which may contain intensive continuous state variables or connector variables.

Implications of changing the input mole fraction vector to a molar substance vector in a function interface are:

- A media model that is written in intensive form needs to add an extra conversion inside all functions. There may be cases where there are transformations back and forth. This affects the performance as it makes automatic differentiated code more complicated. This was seen in a simulation of the DistillationColumn model using the native Modelica media, where the total CPU time was increased with 7% when the input to the fugacity was changed from mole fraction to molar substance. Another disadvantage is that the code might get more verbose as it may require auxiliary variables with an appropriate unit when converting between fraction and substance.
- Advantages are that it will possible to support and calculate property models written in extensive variables and have better support of partial derivatives from external properties tools.

Currently the C-interface supports both mole fraction and molar substance by having an extra input that defines the unit of the fraction vector.

4.3.3 External object

The external object is a pointer to an instance of a Material class on the C++ side. It consists of:

- A pointer to a property calculator, i.e. an instance of e.g. RefProp or CAPE-OPEN where one calculator instance is shared between external objects with the same calculator key.
- An instance of a cache which may be used by a calculator to extract start values for iterative calculations.

On the Modelica side an external object should be associated with variables from one thermodynamic state set. An advantage with the model based approach is that these details can be hidden from the user which avoids the risk of a user breaking the rule and thereby mess up the caching.
An illustration of the structure is found in **Figure 4**.

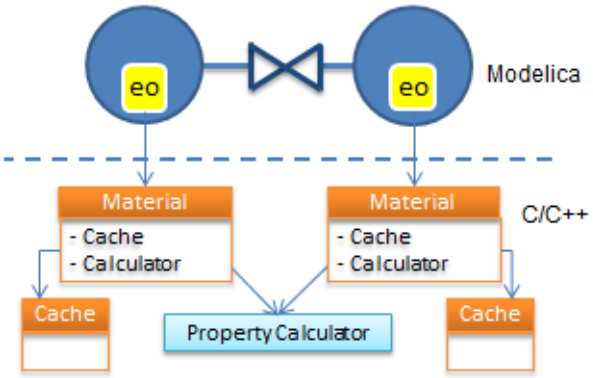

Figure 4. Illustration of the external object structure.

4.4 Challenges with external property code

Most of the available external property packages have not been designed to be used for dynamic simulations. General problems are:

- Error handling when calling properties outside their validity region.
- Limited support for partial derivatives.
- Lack of support to speed up iterative calculation by providing good start values.
- No access to the used tolerances, which may cause numerical problems when creating numerical derivatives.
- Non converging regions.

We have seen in this project that without any additional handling of the validity region issue, simulation will often crash during initialization or simulation. An explanation is that even if a simulation model is set-up to operate within the validity region, the solver might call property routines with invalid inputs when it tries to find a solution for a system of non-linear equations or when it test a large step-size.

In the external interface we decided that it should be the property calculator responsibility to handle this as different property types such as e.g. transport and equation of state based properties may have different validity regions and might be a function of composition.

4.5 CAPE-OPEN

"CAPE-Open standards are the uniform standards for interfacing process modelling software components developed specifically for the design and operation of chemical processes" (Colan, 2015).

The only currently widely adopted standard for thermodynamic property packages is the CAPE-OPEN Thermodynamic and Physical Properties. The backend that has been developed supports both the 1.0 and 1.1 version of the specification.

DOI 10.3384/ecp15118653

4.5.1 Disadvantages

Following disadvantages with the CAPE-OPEN thermo interface should be considered (Szczepanski, 2013).

- Missing calculations of: critical properties, phase boundaries, phase stability test.
- No support of flash derivatives (derivatives of flash outputs w.r.t flash specifications with phases in equilibrium)
- Single calculation, in some circumstances it would be useful to calculate properties for an array of inputs

Another disadvantage is that it contains several internal function calls which create an overhead in computation time. And although it was intended as a cross-platform specification, in practice CAPE-OPEN is only supported on Windows.

4.6 RefProp interface

A backend to RefProp has been developed. An early version of the interface was successfully tested on a full air-conditioning cycle model using the single component media R134a. The computational time of the simulation was in the same order as when a corresponding native Modelica implementation of R134a was used. But RefProp does not seem to be suited to be used for larger system simulations for mixtures due to the disadvantages mentioned in chapter 4.4 and that it by default use highly precise multi-parameter equation of state which is rarely used for mixtures due to the computational effort (Schultze, 2014). To overcome these limitations further analysis is needed.

5 Application test case

To verify the overall interface structure a Modelica application library DistillationColumn was created based on work by (Yasaman, 2012). The library has been modified so it is easy to test different continuous state selections and property function inputs.

5.1 Native Modelica Air media

A native Modelica air media was implemented based on work by (Yasaman, 2012). It is a three component model were the phase equilibria conditions are described by the Rachford-Rice equation (Lämås, 2012) using a declarative approach. The equations are solved by the tool's non-linear solver. The vapor phase is described by an ideal gas volumetric equation of state, a linear polynomial for the heat capacity and polynomials adapted to experiment data of the fugacities. The liquid phase uses an incompressible assumption where density and specific heat capacity are constant and activity coefficients have been adapted to experimental data.

5.2 Air separation column

The lower pressure column in a cryogenic air separation unit was chosen to be used as a test case. Nitrogen and argon is separated from the liquid at atmospheric pressure and a temperature around 85-115K. Liquid with high concentration of oxygen is extracted from the bottom. The column is modeled by 40 equilibrium stages with a total of 164 continuous time states using a 3 component media (nitrogen, oxygen, argon)

5.2.1 Experiment description

Boundary conditions were set to fixed values except for the heat source which increase its value after 100 seconds. Initial transients are present due to the model is not initialized in steady-state

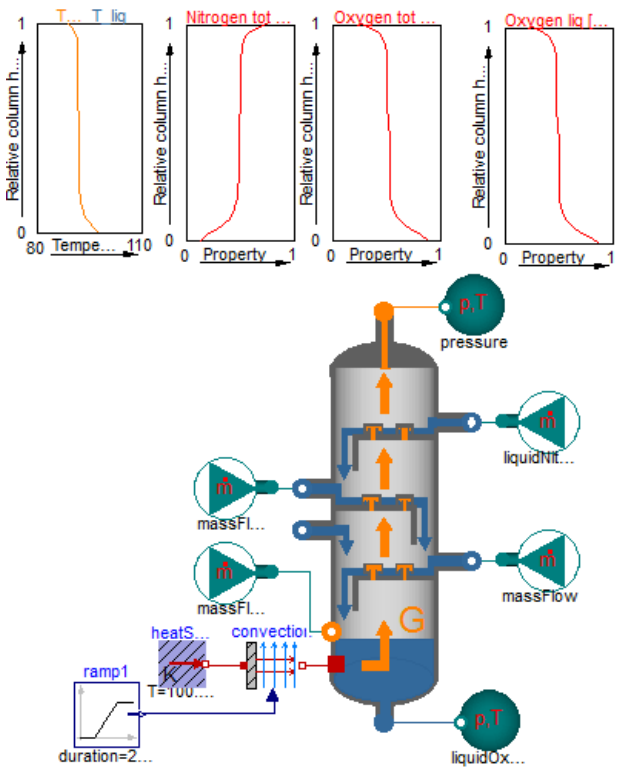

Figure 5 Model of the lower pressure column in a cryogenic air separation unit.

The model was simulated with Dymola 2015 FD01 using the Dassl solver with a relative tolerance of 1e-5 and a non-equidistant time grid. A standard desktop computer (Intel i7, 8GB Ram and 64-bit Windows operating system) was used for the simulation.

5.2.2 Case 1: Simulation with native Air Modelica media

Result, the simulation finished in 9.46s using 383 successful time steps. The result agreed with result presented in (Yasaman, 2012).

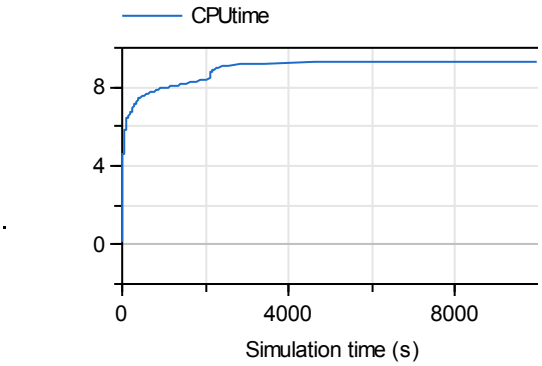

Figure 6. CPU-time with native Modelica media.

The result shows that it is possible to implement an efficient media with the proposed interface using a declarative approach. But good start values are required to succeed with the initialization.

5.2.3 Case 2: Simulation with RefProp Air media

The same model was simulated with a 3 component air media from RefProp. Several different states selection and property input combinations were tested but the tests were unsuccessful due to solver failure when calling properties outside their validity region or due to the solver getting stuck.

Explanation for the slow simulation might be that RefProp uses very high accurate media models that are computational expensive to calculate and that each new flash calculation restarts from scratch every time it is called. For the general flash routines there is no possibility to provide start values. Currently the Modelica-C interface does not support differentiation of all properties which creates numerical Jacobians which are more computational expensive. There might also be other explanations why the simulation performance is not satisfactory. This has to be analyzed further.

6 Limitations

Currently there are restrictions from the used tool and in the Modelica language which makes it harder to use the model based media structure.

6.1 Modelica tools

Following limitations have been observed for different tools:

- Not possible to calculate iteration start values from a property model.
- Not possible to calculate structural parameters from a function using an external object.

The first limitation is severe if a model contains iteration variables that are not equal to a model's start value parameters. If the specific enthalpy is an iteration variable it should be calculated from the given start value parameters as illustrated in Listing 4.

```
parameter  SpecificEnthalpy h_start (fixed=false)
annotation(Evaluate=true);
SpecificEnthalpy h(start=h_start);

Medium.MultiPhaseProperties
   flash_init(Z=Z_start,p=p_start,T=T_start,
   presentPhases=presentPhases,
   presentPhasesStatus=presentPhasesStatus,
   init(p=p_start, x=fill(Z_start, Medium.nP)),
   inputs=MultiPhaseMixture.Interfaces.Inputs.pTX)
initial equation
  h_start=flash.h;
```

Listing 4. Calculation of parameters from a model.

The second limitation requires that the user manually specify the number of phases and compounds in the property declaration.

6.2 Modelica specification

Currently it is inconvenient to use a model or block based structure to calculate parameters as illustrated in Listing 4. It would be more user friendly if a model or block could be used in a similar way as a function to calculate parameters.

6.2.1 Solver callback interface

The external interface ExternalMixtureMedia has been designed with a structure that supports caching. The idea is to cache result from a calculation and use it as start values in a next coming calculation to decrease the number of internal iterations and increase robustness. A problem with this approach is that it is not possible to distinguish a function call during normal continuous simulation from one where the steady-state solver desperately tries to find a solution.

During continuous simulation a good strategy would be to use values from the last accepted step. For the steady-state case it might be an idea to let the non-linear solver update the starting values of the iteration variables hidden in these algorithms, when the solver makes good progress.

A solution would be to have the possibility to register a solver callback interface, which could be used to update iteration start values in a controlled way.

DOI 10.3384/ecp15118653

A suggestion:

- onSolverAcceptedStep() - called by solver/simulation environment when an accepted step has occurred. Place to implement updates of iteration start variables.
- onSolverSteadyStateProgress() - called by solver/simulation environment when progress in steady-state solver. Place to update iteration start values variables.

Advantages with introducing callback methods are that the iteration start values can be updated in a controlled way and thereby avoiding the risk of an update during a bad steady-state iteration step.

7 Conclusions

A new framework for thermodynamic properties with support for multi-component multi-phase has been presented. It is the authors hope that this work will initiate a similar development in the process industry as those that have already taken place in the automotive and power industries, where innovative companies have built their innovation processes for systems engineering around the Modelica technology.

The developed Modelica thermo property library use a model based interface which is in line with the Modelica spirit of equation based modelling. The model based interface makes it possible to implement a thermo property model using a declarative approach and the concept was demonstrated by simulating a column in a cryogenic air separation unit.

This work should be seen as a starting point for a model based framework for multi-component multi-phase thermo properties in Modelica. It is possible to improve the framework in following directions:

- Implement an infrastructure for native Modelica implementation of fluids with support of various equations of states and mixing rules including phase equilibrium solvers. The later could be an interesting research topic on how to best formulate these algorithms in a declarative way. A difficulty with the equation based approach is the initialization part, where it would be interesting to see how property models can be formulated to better support initialization. For example by using the homotopy operator.
- Extend the C-interface back end to support more property packages such as MultiFlash.
- Adding additional functionality such as reaction properties.

It would also be interesting to create use cases for the other application mentioned in section 2.1. We encourage people to take part of continuing the development.

Acknowledgements

The work has been partially funded by the Seventh Framework Programme of the European Union (project MODELICAPROP, Clean Sky number 325975). The financial support from the European Union is highly appreciated.

References

Ali Baharev and Arnold Neumaier. Chemical Process Modeling in Modelica, *Proceedings of the 9th International Modelica 2012 Conference*, Munich, Germany, September 3-5 2012.

Stephen Boyd and Lieven VandenBerghe. Convex Optimization, *Cambridge University Press*.

CAPE OPEN, Thermodynamic and Physical Properties v1.1, May 2011, Downloaded from http://www.colan.org (accessed 2015-05-17).

Francesco Casella and Christoph Richter, ExternalMedia: A Library for Easy Re-Use of External Fluid Property Code in Modelica, *Modelica Conference Proceedings*, 2008.

Colan, http://www.colan.org/index-16.html, accessed 2015-05-17.

Karin Dietl, Kilian Link and Gerard Schmitz. Thermal Separation Library: Examples of Use, *Proceedings of the 8th International Modelica 2011 Conference*, Dresden, Germany, March 20-22 2011.

Fluid property library; a common interface to various state-of-the-art thermodynamic and transport property models. http://www.asimptote.nl/software/fluidprop/ (accessed 2015-05-17).

Johannes Gernert, Andreas Jäger and Roland Span. Calculation of phase equilibria for multi-component mixtures using highly accurate Helmholtz energy equations of state, *Fluid Phase Equilibria 375 (2014) 209–218.*

Lemmon, E.W., Huber, M.L., McLinden, M.O. NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 9.1, *National Institute of Standards and Technology, Standard Reference Data Program*, Gaithersburg, 2013.

James C. M. Li, Clapeyron Equation for MultiComponent Systems, *The Journal of Chemical Physics volume 25. number 3 september*. 1956.

Hans Lämås, Algorithms for Multi-component Phase Equilibrium Models in Modelica, *MSc Thesis*, Chalmers University of Technology, Gothenburg, Sweden, 2012.

Yasaman Mirsadraee. Dynamic modeling and simulation of a cryogenic air separation plant, *Msc Thesis*, Linköping, Sweden, 2012

J.M. Mollerup and M.L. Michelsen. Calculation of Thermodynamic Equilibrium Properties, *Fluid Phase Equilibria 74 (1992) 1–15. 1992*

W Morton. Equation oriented simulation and optimization, *Proc Indian Natn Sci Acad 69, (2003) pp. 317-357. 2003*

Hans Olsson, Hubertus Tummescheit and Hilding Elmqvist. Using Automatic Differentiation for Partial Derivatives of Functions in Modelica, *Proceedings of the 4th International Modelica 2005 Conference*, Hamburg, Germany, March 7-8 2005.

Vipul Parekh and Paul Mathias, Efficient flash calculations for chemical process design – extension to the Boston-Britt Inside-Out flash algorithm to extreme conditions and new flash types, *Computers and chemical engineering*, vol 22 pp 1371-1380 (1998)

Modelica Association, Modelica Language Specification, Version 3.3, 2015 https://www.modelica.org/documents/ModelicaSpec33.pdf , accessed 2015-05-17.

Modelica Association Libraries. Available at https://www.modelica.org/libraries, accessed 2015-05-17.

C. Schultze, A Contribution to Numerically Efficient Modelling of Thermodynamic Systems, PhD thesis, Technische Universität Braunschweig, Fakultät für Maschinenbau., (2014)

Richard Szczepanski, Physical Property Modelling – MultiFlash and CAPE-OPEN. *Presentation for ModelicaProp Workshop 9-10 Oct*, Delft, 2013.

Mathis Thorade and Ali Saadat, Partial derivatives of thermodynamic state properties for dynamic simulation, *Environ Earth Sci* 70:3497–3503. 2013

C. Trapp, F.Casella, T. Stelt, P. Colonna. Use of External Fluid property Code in Modelica of a Pre-combustion Co2 Capture Process Involving Multi-Component, Two-Phase Fluids, *Proceedings of the 10th International Modelica 2014 Conference*, Lund, Sweden, March 10-12 2014.

Hubertus Tummescheit, Jonas Eborn, Chemical Reaction Modeling with Thermofluid/MF and MultiFlash, *Proceedings of the 2nd International Modelica Conference*, Munich, 2002.

K. Wellner, C. Trapp, G. Schmitz and F.Casella. Interfacing Models for Thermal Separation Processes with Fluid Property Data from External Sources, *Proceedings of the 10th International Modelica 2014 Conference*, Lund, Sweden, March 10-12 2014.

J.Åkesson, W. Braun, P.Lindholm, B.Bachmann, , Generation of Sparse Jacobians for the Function Mock-Up Interface 2.0, *Proceedings of the 9th International Modelica Conference*, Munich, 2012.

 DOI 10.3384/ecp15118653