An innovative heterogeneous modeling approach to build a cooling system for battery thermal management with common fluid properties involving FMI terminals

Karim Besbes

Siemens Digital Industry Software, France
karim.besbes@siemens.com

Abstract

In this paper, a new modeling approach combining native Simcenter Amesim submodels and Modelica submodels is presented. FMI terminals are used to enable the physical connection of fluids between Simcenter Amesim and Modelica. The innovation lies in the fluid properties that are computed in the causal and acausal worlds using the same technology. The architecture of a new Modelica AmeTpfMedia library developed to ensure continuity of fluid properties is presented, along with its accuracy and performance. Using this new approach, a demonstrator of a closed-loop heterogeneous cooling system for battery thermal management is built, opening the door to a new way of thinking complex multi-physics systems.
Keywords: Simcenter Amesim, Heterogeneous Modeling, FMI Terminals, Fluid Properties, Battery Thermal Management

1 Introduction

In multi-physics modeling, the choice between causal and acausal modeling paradigms, aligned with the needs of the modeling project, requires careful consideration.

The causal approach, commonly used in the Bond-Graph method (Hermien and Wiyatini 2021), consists of fixing the input and output status of variables of a model and each of its submodels, creating a causal relationship between inputs and outputs. This traditional approach makes it possible to build robust models. It allows precise control of system behavior, making it easier to interpret and understand system dynamics.

By contrast, acausal modeling, for which Modelica is particularly well suited, is based on equations (Fritzson 2014). In such a modeling paradigm, the only requirement is a well-balanced mathematical system defined by all declared equations, variables and data sets. Then, causality is naturally fixed by solving the mathematical system. This approach, more flexible, offers a convincing alternative when dealing with highly interconnected systems.

To meet the ever-increasing modeling challenges of tomorrow, with increasingly complex and interconnected systems, modeling techniques need to become ever more innovative and sophisticated. Combining causal and acausal paradigms, to build an heterogeneous system, can be one such innovative solutions.

Thanks to the Functional Mock-up Interface (FMI) standard and especially the 3.0 (Modelica Association 2022), such combined modeling approach is simplified, since it allows to regroup different signals in one connector. It then becomes conceivable to build an heterogeneous model involving fluids, as long as the mechanism used to calculate the thermodynamic properties in each submodel is the same.

This article aims to present an innovative heterogeneous modeling approach in Simcenter Amesim to build a cooling system for the battery of electrical vehicles. Simcenter Amesim, well known in the realm of system simulation, is natively causal and Bond-Graph based. It provides also a full-featured Modelica Editor including physical interfaces to connect both worlds "physically", with Terminals. To meet the challenge of ensuring perfect continuity of thermodynamic properties between Simcenter Amesim and Modelica's fluid libraries, a new Modelica media library, named AmeTpfMedia, has been developed. AmeTpfMedia takes advantage of the mechanism used in Simcenter Amesim to calculate fluid properties.

In the following article, the Modelica support in Simcenter Amesim is described as well as the definition of the physical interfaces. Then we will go over the AmeTpfMedia library developed to be fully consistent with Simcenter Amesim's Two Phase Flow library. Finally, the cooling system built as a proof-of-concept is presented and the simulation results are given.

2 Modelica support in Simcenter Amesim

The IMAGINE company, now Siemens Digital Industry Software has released its System Solution tool Simcenter Amesim back in 2012. This simulation platform gives access to a set of libraries of mechanical, electrical, hydraulic, pneumatic, fluid, and hybrid components created with the help of domain experts from both the industrial and the academic worlds. It enables the development of any new mechatronic product without having to write equations and offers numerous facilities for 1D system simulation (Vasiliu et al. 2018).

Among these facilities, Simcenter Amesim offers a full-

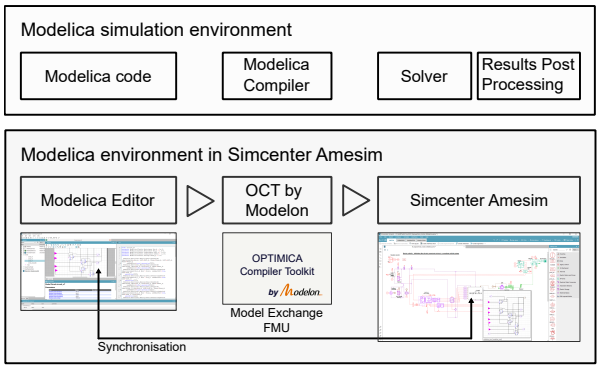

Figure 1. Modelica support and integration in Simcenter Amesim

featured editor dedicated to the Modelica language, as shown in the Figure 1. Since the version 17, the Modelica Editor embeds Modelon's OPTIMICA Compiler Toolkit (OCT), which is today compliant with the Modelica language specification 3.4 (Modelica Association 2017) and supporting both Modelica Standard Library (MSL) 3.2.3 as well as version 4.0.0, except Synchronous Language Elements and State Machines.

For the selected model in the Modelica Editor, the Modelica compiler generates a Model Exchange Functional Mock-up Units (FMU) version 2.0.4 extended with the FMI 3.0 terminals definition. It allows to define a group of variable to ease connecting compatible signals on the system level. The Model Exchange FMU is then treated as a native submodel by Simcenter Amesim, synchronized with the original Modelica code, removing pains that would be introduced due to cosimulation usage. This submodel can be connected to other native submodels and simulated with the internal solver. Thus, Simcenter Amesim can be used as a pure Modelica simulation environment made up by a Modelica Editor, a Modelica compiler, a solver and a bunch of post-processing features, or can be used to mix two modelling approaches to build a heterogeneous model.

3 Heterogeneous model with physical interfaces

In this article, we consider that a heterogeneous model is composed of at least two submodels built using different modeling tools or approaches.

In Simcenter Amesim, all the variables at the ports of the submodels have their causality fixed. Figure 2 illustrates four submodels taken from four different libraries. For example, the wall (c) has two ports where the heat flow rate is defined as an input and the temperature is defined as an output on both ports. Most of the time, several versions of the same submodel are available to modify causality. Such native submodels can only be connected to a Modelica submodel if three conditions are met. The first condition concerns the port variables at the interface, which must be defined as a group of variables. The second condition concerns the causality consistency and last one concerns unit consistency. The following sections describe how to deal with these conditions.

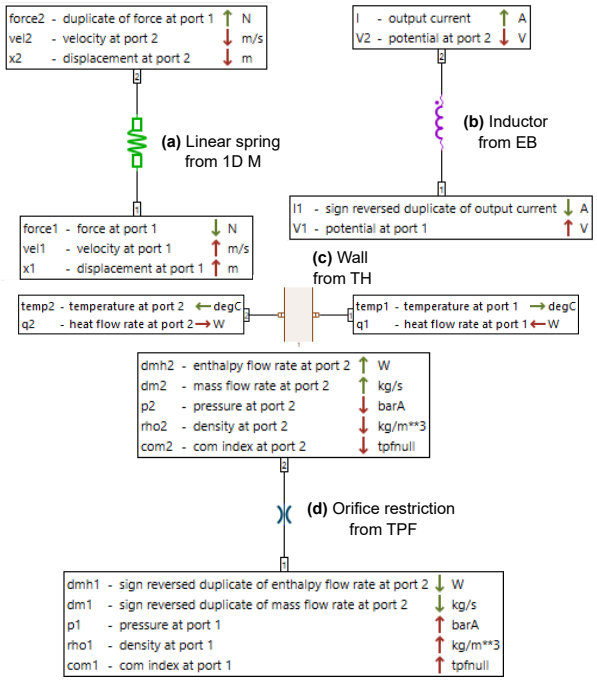

Figure 2. Example of Simcenter Amesim ports with fixed causality. The submodel (a) comes from the 1D Mechanical library, (b) from the Electrical Basics library, (c) from the Thermal library and (d) from the Two Phase Flow library

3.1 Terminals

In order to address new use cases, FMI 3.0 standard introduces the concept of layered standards adding optional features to the FMU, (Bertsch et al. 2023). It includes also an optional mechanism in the `icons/terminalsAndIcons.xml` to define the terminals and their graphical representation, as described in the section 2.4.9 of the specification. In Simcenter Amesim, the terminals definition has been backported to FMI 2.0.4 using the `/extra` directory. It is used to convert a Modelica connector into a physical Simcenter Amesim port.

Assuming that a Modelica connector is coded with terminals requirements, the compilation generates automatically an XML file where the root element is `fmiTerminalAndIcons` with *fmiversion="3.0"* attribute. The children elements are `GraphicalRepresentation` which is ignored in the current implementation and `Terminals` which is a sequence of `Terminal`. Each `Terminal` is described by a name, a mandatory matching rule (plug, bus or sequence) and an optional terminal kind used for the nominal typing. A `Terminal` is a sequence of terminal member variables, identified by their unique name among the list of scalar variables.

To illustrate a typical terminal definition, the Figure 3 shows an example of a XML tree from a compiled Mod-

DOI 10.3384/ecp218703

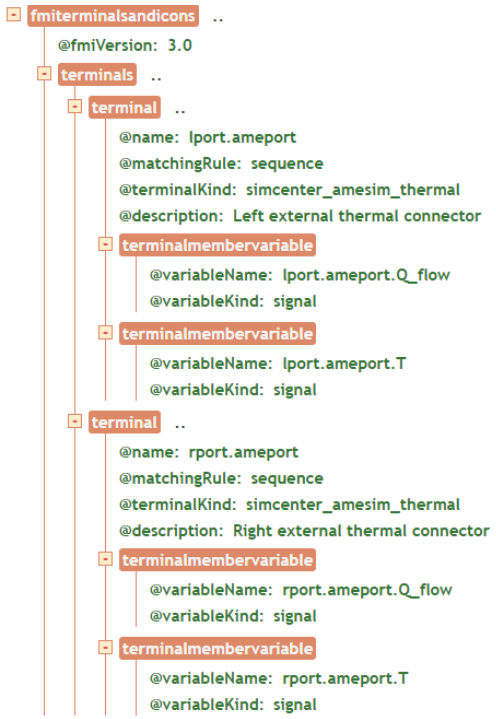

Figure 3. Example of a XML tree with terminals definition

elica submodel packaged into a FMU. The two terminals have a kind defined as `simcenter_amesim_terminals` and a **sequence** matching rule attribute. In Simcenter Amesim, a thermal port is declared with a group of two variables, heat flow rate and temperature, as shown in Figure 2(c). Given that `lport.ameport.Q_flow` and `lport.ameport.T` define a group of scalar variable already existing in Simcenter Amesim with the same causality, the corresponding port is automatically created. Figure 4 shows the external view of the corresponding submodel imported in Simcenter Amesim. Note that the two signal ports `TempOutput1` and `TempOutput2` are also created. But they refer to two other scalar variables with output causality that are not part of a terminal.

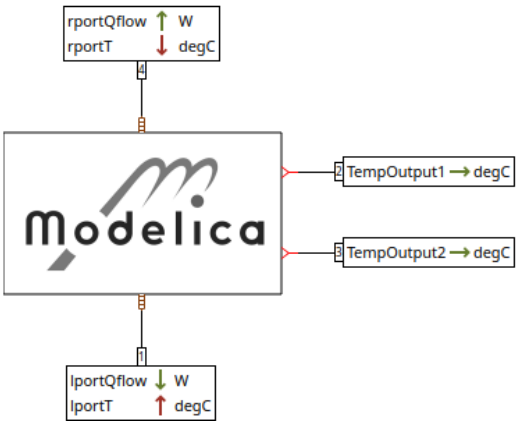

Figure 4. Example of a compiled and imported Modelica submodel using the terminals described in the XML tree

From Modelica point of view, in order to generate in the `modelDescription.xml` a terminal for a group of variables, Modelica compiler requires the following tool vendor annotation:

```
annotation(__simcenter(port(port_type="port
    type name")));
```

In Simcenter Amesim, there are 13 different port types classified per physical domain, listed in the Table 1.

Table 1. List of native port types in Simcenter Amesim

Domain	*Native Port Type*
Mechanical	lshaft, rshaft, rope, mecha 2D
Fluid	hflow, pflow, geocom
Thermal	thermal
Magnetic	magnet
Electrical	elect, elect3p
Chemical	electroch
Signal	signal

3.2 Connectors

The equivalent of a Modelica connector in Simcenter Amesim is a port. From the moment a port is targeted, the rule to build the associated Modelica connector is always the same. The order of the declared variables in the connector must be exactly the same as the order of variables defined in the corresponding port. The same applies to variable causalities and types. And the port type name listed in the Table 1 must be specified in the tool vendor annotation. The following code uses a Modelica connector to generate the thermal port, shown in Figure 4:

```
connector PortTinQout
  output Q_flow(unit="W") "Output heat flow
      ";
  input Real T(unit="degC") "Input
      temperature";
annotation (__Modelon(isTopLevel=true),
    __simcenter(port(port_type="thermal")));
end PortTinQout;
```

As can be seen, order, causalities and units have been set to be consistent with the thermal port targeted as well as the port type name. Knowing these rules, a Modelica submodel can be coded to connect to any of Simcenter Amesim's native physical ports.

3.3 Interfaces

In this context, a Modelica interface is defined as a class where connectors from both worlds are already declared. It provides end-user capability to connect a Modelica library to a Simcenter Amesim's library for the same physical domain. In the Modelica Editor, in addition to the MSL, the library Simcenter Amesim Modelica Library (SAML) is provided. It is a standardized library based on the native submodels including several physical interfaces as illustrated in the Figure 5.

For most used physical domains and associated causalities there is a dedicated interface in the SAML. From

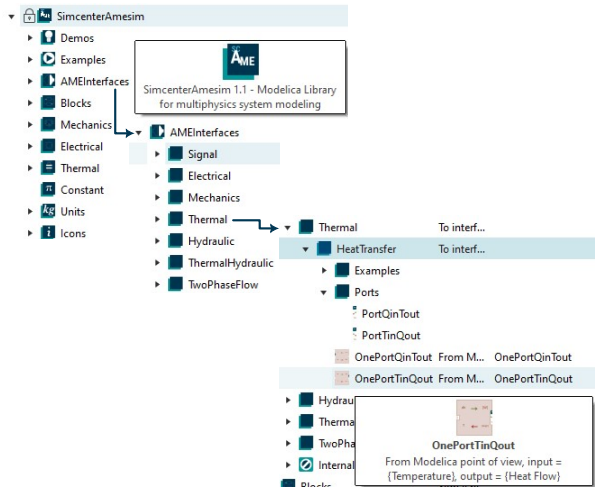

Figure 5. Simcenter Amesim Modelica Library 1.1 tree with the thermal connectors and interfaces

Modelica point of view, the MSL is the reference library of connector definition. For instance, the thermal interfaces in the SAML are built in order to connect any Modelica submodels containing a `HeatPort` connector from the MSL to any Simcenter Amesim submodels coming from the thermal library. In the MSL, the `HeatPort` connector contains the variables T and Q. From Simcenter Amesim point of view, thermal port contains also T and Q but with fixed opposite causalities. Then two causalities and two interfaces can be defined. Figure 6 shows both of them allowing to convert a convection submodel from the MSL dedicated to Simcenter Amesim. As illustrated, the interfaces contain two different connectors, one for Simcenter Amesim and the other to be connected to the convection submodel. Within the interfaces, declared connectors ensure causality consistencies for each variable, and equations ensure units consistency between the two environments.

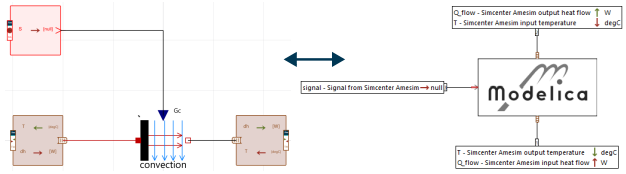

Figure 6. Usage of thermal interfaces to convert a Modelica convection submodel for Simcenter Amesim with the thermal ports

Note that thermal connectors in Figure 6 are not connected to anything, and the compilation is still proceeding correctly. This is due to a second tool vendor annotation telling that connector must be treated as a top-level connector, regardless of the layer in which it has been declared in the model. It allows to simplify the workflow for end-user. A simple drag and drop of the interface into the diagram and a connection with the Modelica submodel is sufficient. The following code is for the interface with heat flow rate as input and temperature as output:

```
model OnePortQinTout "From Modelica point
    of view, input = {Heat Flow}, output =
    {Temperature}"
  SimcenterAmesim.Thermal.Interfaces.
      HeatPort port;
  Ports.PortQinTout ameport;
equation
  port.T = ameport.T + SimcenterAmesim.
      Constant.celsius2kelvin;
  port.Q_flow = -ameport.Q_flow;
end OnePortQinTout;
```

And the thermal connector `ameport` including the two tool vendor annotations, one to treat the connector as a top-level connector and the second for the terminal definition, is coded as follows:

```
connector PortQinTout
  output Real T(unit="degC") "Simcenter
      Amesim output temperature";
  input Units.AMESIUnits.HeatFlowRate
      Q_flow "Simcenter Amesim input heat
      flow";
annotation (__Modelon(isTopLevel=true),
    __simcenter(port(port_type="thermal")))
    ;
end PortQinTout;
```

In this section, terminals, connectors and interfaces have been introduced, showing how to build a thermal interface as an example. The same strategy can be applied for other physical fields like mechanical or electrical and for other connectors than those from the MSL. However, as soon as a transport of a fluid is involved, the interface must also ensure mass and energy balance between both worlds. Such interface is described in the last section of this article where the demonstrator is presented. Finally, in this context, where Simcenter Amesim is the integration and simulation platform, the Modelica part must use the same mechanism than Simcenter Amesim to calculate the fluid properties to avoid physical and numerical discontinuities. This mechanism is introduced with AmeTpfMedia.

4 AmeTpfMedia library

The AmeTpfMedia library has been developed in Modelica for calculation of fluid properties. It uses the same utilities than the Two-Phase Flow (TPF) library and developed for dynamic simulation and design of thermodynamic applications, such as automotive air-conditioning, heat pumps, Rankine cycles... These utilities have been recompiled specifically for AmeTpfMedia and allow to calculate the thermodynamic state properties and their partial derivatives for a set of refrigerants. The description quality of fluid properties depending on the type of equation of state (EoS) used to do the calculation can have a significant impact on the results, performance and robustness of the simulation. With AmeTpfMedia, when a refrigerant is selected, it is possible to find the best compromise for simulation requirements, providing the choice between

DOI
10.3384/ecp218703

Helmholtz, MBWR or Peng-Robinson EoS. Most refrigerants have at least two EoS choices including Helmholtz, and a small proportion have only one. The list of fluids available is presented in the Table 2.

4.1 Architecture

The structure of the library is illustrated in Figure 7. It contains the package AmeTpfMedium extending from PartialTwoPhaseMedium defined in the MSL media package. This strong choice of architecture is motivated by the number of well known libraries in the Modelica community already using the MSL media structure to call the function of fluids, like Buildings (Wetter et al. 2014) or ThermoCycle (Dechesne et al. 2014).

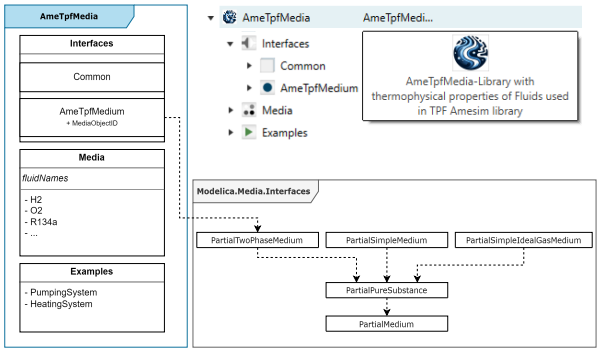

Figure 7. Structure of the AmeTpfMedia library

Inside the AmeTpfMedium package, the class MediaObjectID defines an ExternalObject instantiating the medium, set by an index of fluid named ID, as follows:

```
class MediaObjectID
  extends ExternalObject;
  function constructor
    input Integer ID;
    output MediaObjectID media;
external"C" media =
    create_modelica_media_as_predefined_fluid
    (ID)
    annotation (Include="#include <
        media_public.h>", Library={"
        AmeTpfMedia"});
  end constructor;
  function destructor
    input MediaObjectID media;
external"C" free_modelica_media(media)
    annotation (Include="#include <
    media_public.h>",
         Library={"AmeTpfMedia"});
  end destructor;
end MediaObjectID;
```

The ExternalObject is used as a pointer by all thermodynamic functions contained in the Internal package in order to call directly the compiled resources (headers and compiled objects). For instance, the function calculating the specific enthalpy according to the density and the temperature is coded as follows:

```
function h_dT
  input Density d;
  input Temperature T;
  input MediaObjectID media=mediaID;
  input Integer EoS=EoSInt;
  output SpecificEnthalpy h;
external"C" h =
    modelica_media_compute_h_from_rho_t(
    media,EoSInt,d,T)
    annotation (Include="#include <
        media_properties_from_rho_t_public.
        h>", Library{"AmeTpfMedia"});
end h_dT;
```

These internal functions allow to rebuild one by one the inherited functions from the MSL. The derivatives required for the inversions of temperature and density functions are implemented in Modelica as well as the time derivatives of the enthalpy, density, pressure and entropy to facilitate the numerical inversion.

4.2 Validation

The AmeTpfMedia library has been tested with a set of standard functions calculating the pressure and the specific enthalpy over a range of density and temperature. The utilities used by the AmeTpfMedia library have already been validated in the context of Simcenter Amesim. Nevertheless, to ensure the good integration in the Modelica world, a protocol of test has been defined for three very common fluids, which are Water, CO2, and R134a.

The protocol consists in comparing the results with reputed Modelica libraries of fluid properties: the MSL, the CoolProp library (Casella 2008) and the non-profit version of TilMedia from Clara library (XRG-Simulation GmbH TLK-Thermo GmbH 2023). Figure 8, 9 and 10 compare the results generated from AmeTpfMedia and those from CoolProp, TilMedia and/or MSL. These are results obtained in the form of $Y(T,\rho)$, where Y is the variable of interest, pressure p or specific enthalpy h, evaluated in the domain defined by framing the critical point of each fluid, with n the number of calculated points. To perceive the variations between the libraries, we define the average relative deviation *AvDev* between AmeTpfMedia and the other libraries as follows, where lib can be CoolProp, TilMedia or MSL:

$$AvDev_{lib}(Y) = \sum_{n}(Y_{AmeTpfMedia}/Y_{lib} - 1) * 100/n \quad (1)$$

Regarding the thermodynamic properties of the water calculated by AmeTpfMedia, Figure 8 shows a comparison with the three other libraries. For p and h, there is an average relative difference of 0.5% between AmeTpfMedia and the other libraries. This small difference can be explained by the fact that Simcenter Amesim uses a different standard for the thermodynamic formulation of water.

Regarding the CO2, Figure 9 shows an average relative deviation between AmeTpfMedia and CoolProp of 0.01% and between AmeTpfMedia and TilMedia of 1e-6%. Both are negligible. Note that the non-profit version

Table 2. List of fluids available in AmeTpfMedia

Family	*Fluid usage name*
Common fluids	pH2, H2, oH2, N2, O2, CO2, H2O, H2O2, NH3, Air
Refrigerants	R22, R32, R123, R125, R134a, R13b1, R13i1, R143a, R152a, R227ea R236ea, R236fa, R245fa, R1123, R1224yd(Z), R1234yf, R1234ze(E) R1243zf, R1336mzz(Z), Novec 649, Novec 7000, R404a, R407c R410a, R450a, R452b, R454c, R455a, R507a
Organic compounds	Ethylene, Methane, Ethane, Propane, Butane, Iso-Butane, Pentane Iso-Pentane, Cyclopentane, Heptane, Octanen, Iso-octane Dodecane, Methanol, Ethanol
Noble gas	He, Ne, Ar, Kr, Xe

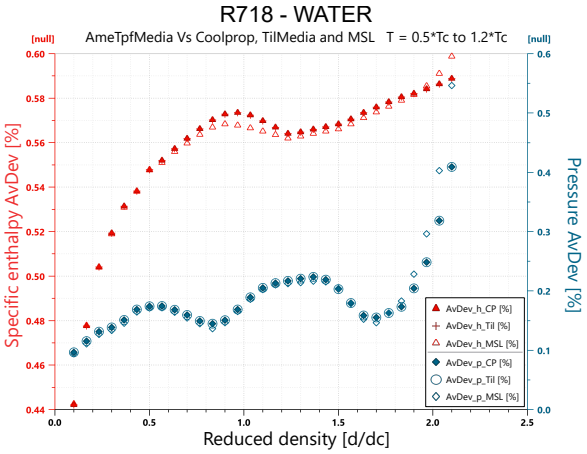

Figure 8. Comparison between AmeTpfMedia, Coolprop, TilMedia and the MSL for the Water

of TilMedia gives only an access to GERG CO2 built using the GERG EoS (Kunz and Wagner 2012) instead of Helmholtz, which is used by CoolProp and AmeTpfMedia. And yet, the biggest difference lies between AmeTpfMedia and Coolprop.

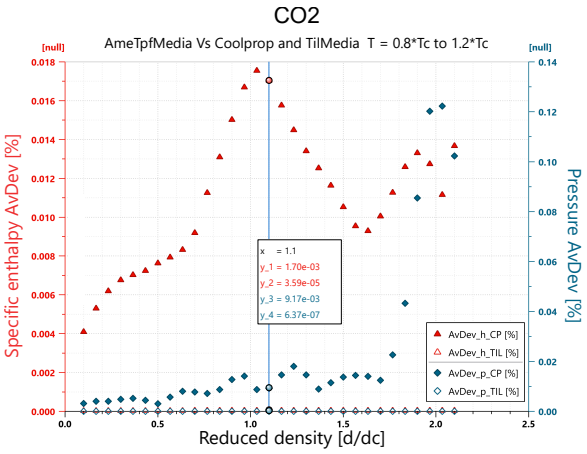

Figure 9. Comparison between AmeTpfMedia, Coolprop and TilMedia for the CO2

Finally, regarding the R134a, Figure 10 shows an average relative deviation of 5e-6% for h and of 1.5e-4% for p, between AmeTpfMedia and CoolProp or the MSL. As a result, the R134a is the fluid with the greatest similarity between the libraries.

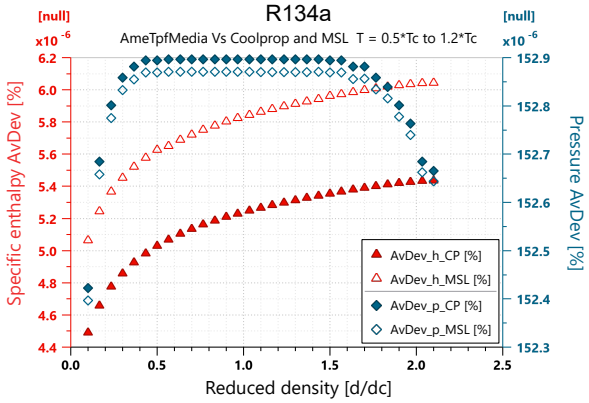

Figure 10. Comparison between AmeTpfMedia, Coolprop and the MSL for the R134a

In this section, thermodynamic properties calculated for three different fluids have been presented. Another article dealing specifically with TPF utilities in Simcenter Amesim would provide more results for more fluids, and address the slight differences observed. Nevertheless, it is considered that AmeTpfMedia reflects the expected behavior and the transition of TPF utilities to Modelica is validated.

4.3 Performance and Robustness

Performance and robustness are key when it comes to accurately calculating fluid properties for thermodynamic cycle simulation. This section introduces a benchmark consisting in simulating on the same environment and the Simcenter Amesim's solver an evaporator model taken from the ThermoCycle library. Water and carbon dioxide are chosen as refrigerants and CPU times of each library are compared. The aim is not to make a general statement on which library is the best performing. This would require a much larger number of tests with a wide selection of models and fluids. This is a qualitative reference to see

DOI 10.3384/ecp218703

if AmeTpfMedia performs comparably to other reputable libraries.

The model used for the test bench coming from ThermoCycle is named `Hx1DConst`. This is a discretized counter-current heat exchanger in which one of the two fluid has a constant heat capacity. The other fluid (Water or carbon dioxide) is described at each cell of the tube, where enthalpy, density, pressure and vapor quality is computed. The heat exchanger is discretized into 30 cells and the start values chosen are exactly the same for a selected fluid. The boundary conditions are defined in order to evaporate the selected fluid.

With regard to the water, as can be seen in Figure 11, CPU times of libraries using external objects for calculation are fairly similar, with AmeTpfMedia slightly ahead. As expected, the MSL gives the best CPU time. Indeed, as described in the MSL documentation, the water was explicitly designed to work well in dynamic simulations with a high focus on the performance. In the second graph, a comparison of the cooling power can be found. The results show a relative difference of 0.2% between AmeTpfMedia and the other libraries, which is consistent with the deviation observed in Figure 8.

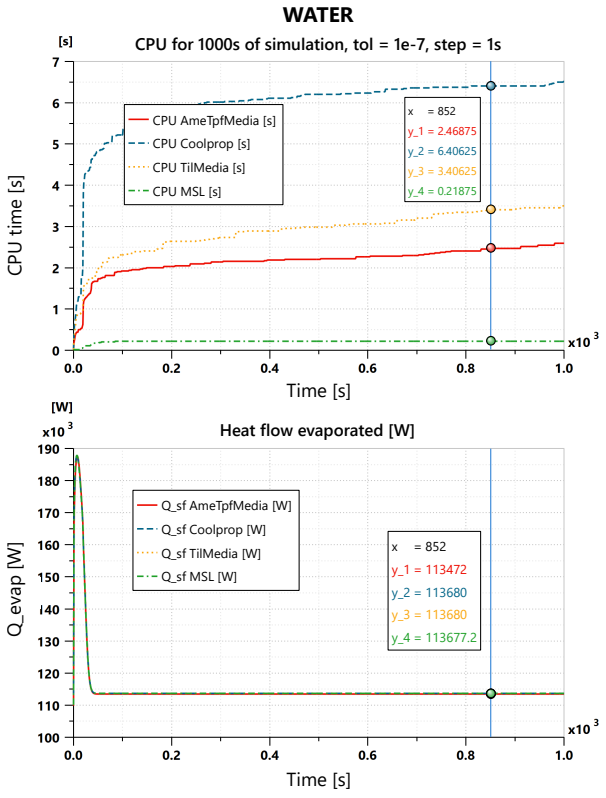

Figure 11. Comparison of CPU time and cooling power with water as working fluid

For carbon dioxide, as shown in Figure 11, AmeTpfMedia and TilMedia have similar CPU times, while CoolProp is 3 to 5 times slower. TilMedia is slightly ahead probably due to the GERG EoS used for the CO2. As for the cooling power, the result shows negligible differences.

During the benchmark, no significant difference was observed concerning the robustness. The libraries behaved in the same way at each simulation.

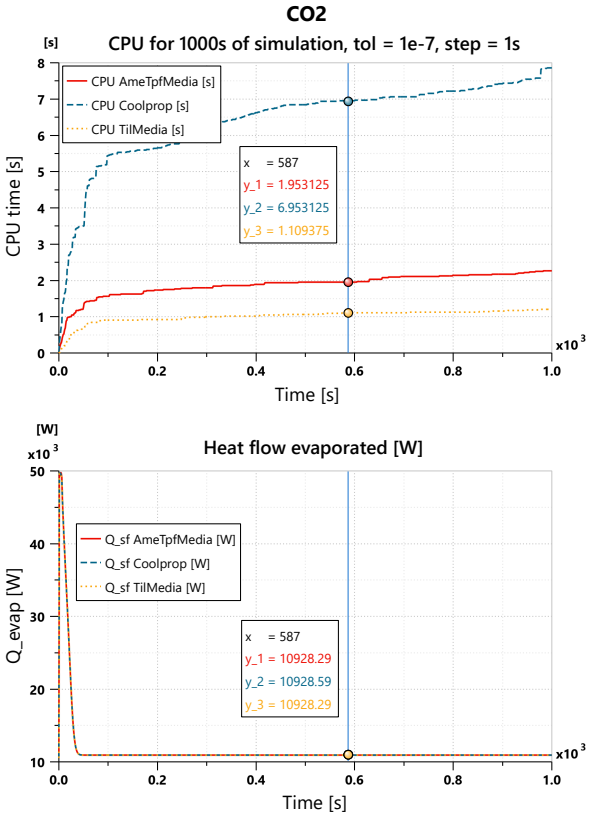

Figure 12. Comparison of CPU time and cooling power with carbon dioxide as working fluid

5 Use case: heterogeneous cooling system

Imagine two teams tasked with modeling a complex system like battery thermal management. One team employs Simcenter Amesim, while the other develops in-house submodels using the Modelica language. Nowadays, this scenario is becoming increasingly common, particularly among large industrial groups with multiple R&D departments. The following demonstrator aims to answer this need.

5.1 Description

The first team has developed the complete system in Simcenter Amesim, with the exception of the compressor, condenser and valve, which were developed by the Modelica team. Thanks to the Terminals and AmeTpfMedia, it becomes possible to integrate each development made by each team to build the full system presented in the Figure 13. As can be seen, the system consists of the charger, the battery in purple, the coolant system in orange, the refrigerant system in blue, part of which is made in Modelica, and the control in red.

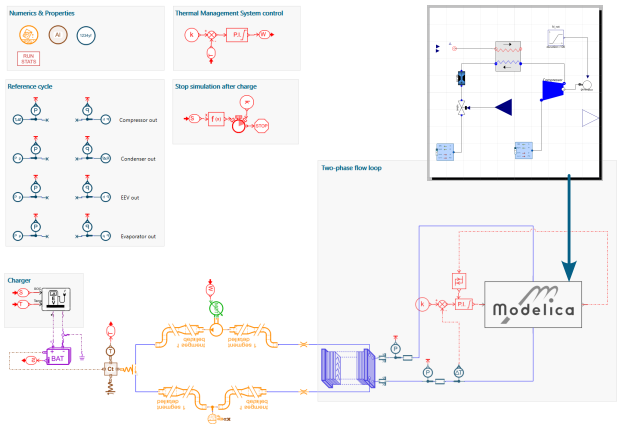

Figure 13. Heterogeneous chiller loop design for battery thermal management

For this use case, the focus is on the design of the cooling loop, composed of an evaporator from Simcenter Amesim and a compressor, condenser and expansion valve from Modelica's ThermoCycle library. The evaporator is a discretized brazed plate heat exchanger with 5 cells. The condenser from ThermoCycle is a discretized heat exchanger with 6 cells. The required cooling power is 7 kW. The evaporation temperature is set at 5°C with a superheat of 5°C. The condensation temperature is set at 55°C with a subcooling target of 7.5°C. Finally, the working fluid is R1234yf.

5.2 TPF physical interface

At evaporator boundaries, the causalities for both ports are defined as shown in the Figure 14(a). These boundaries impose the causality to the Modelica submodel. The Figure 14(b) is a representation of the Modelica TPF interface where pressure and density are given by Simcenter Amesim and specific enthalpy and mass flow are given by Modelica. The Modelica part built with ThermoCycle's submodels use the MSL connectors for the fluid. Thus, the TPF interface must be defined with a MSL fluid connector and a TPF connector.

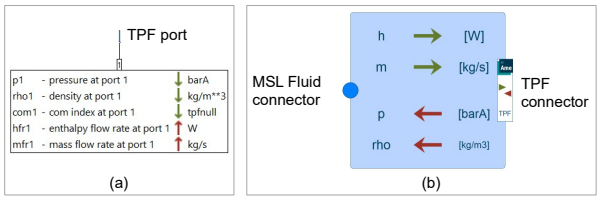

Figure 14. TPF port definition with a fixed causality. (a) Represents external TPF ports and (b) represents the corresponding TPF interface in the Modelica world

The TPF connector is coded as follows:

```
connector Portpinmout "p,rho in / m,Q
    out"
output SimcenterAmesim.Units.AMESIUnits.
    HeatFlowRate Q_flow
  "Enthalpy flow rate";
output SimcenterAmesim.Units.AMESIUnits.
    MassFlowRate m_flow "Mass flow rate";
input AmeTpfMedia.AMEUnits.
    AbsolutePressure_bar p "Pressure";
input SimcenterAmesim.Units.AMESIUnits.
    Density rho "Density";
input Real ComId(unit="tpfnull") "Com
    index";
annotation (
  __Modelon(isTopLevel=true),
  __simcenter(port(port_type="hflow")));
end Portpinmout;
```

The causalities are fixed in the good order, as well as the units and the annotations with the port type name related to the fluid domain. Note that the ComId is a communication variable specifically used in Simcenter Amesim context. This variable avoids redundant calculations and evaluates the total volume of a system by counting the number of capacitive (volume). In this demonstrator, the Modelica part receives this variable and therefore does not interfere with Simcenter Amesim.

The interface between the Modelica fluid connector and the TPF connector is coded as follows:

```
model OnePortpinmout "(R) TPF Interface"
  // Medium package
  replaceable package Medium = AmeTpfMedia.
      Media.partialMedium "Medium model";
    "Mass flow rate";
  // Connectors
  Modelica.Fluid.Interfaces.FluidPort_a
      fluidport(
    redeclare package Medium = Medium,
    C_outflow=fill(1/Medium.nC, Medium.nC),
    Xi_outflow=fill(1/Medium.nX, Medium.nX
        - 1));
  SimcenterAmesim.TPF.Interfaces.Ports.
      Portpinmout ameport(
    p(start=1),
    rho(start=500));
  // Medium state
  Medium.ThermodynamicState state=Medium.
      setState_pdX(ameport.p*1e5, ameport.
      rho);
equation
  fluidport.p = ameport.p*1e5;
  ameport.m_flow - fluidport.m_flow = 0;
  if ameport.m_flow >= 0 then
    ameport.Q_flow = ameport.m_flow*
        inStream(fluidport.h_outflow);
  else
    ameport.Q_flow = ameport.m_flow*state.h
        ;
  end if;
  state.h = fluidport.h_outflow;
  fluidport.Xi_outflow = inStream(fluidport
      .Xi_outflow);
  fluidport.C_outflow = inStream(fluidport.
      C_outflow);
end OnePortpinmout;
```

This TPF interface is made up of a replaceable package for defining the fluid, the required connectors, and a pressure- and density-dependent state functions (inputs from Simcenter Amesim). It contains also the equations

ensuring the unit consistency, the mass and energy balance and the sign consistency taking into account the Modelica `flow` and `stream` definition.

With such interface, the end-user only needs to drag and drop this class to the diagram, define the fluid and connect it to the the Modelica models as shown in the Figure 15.

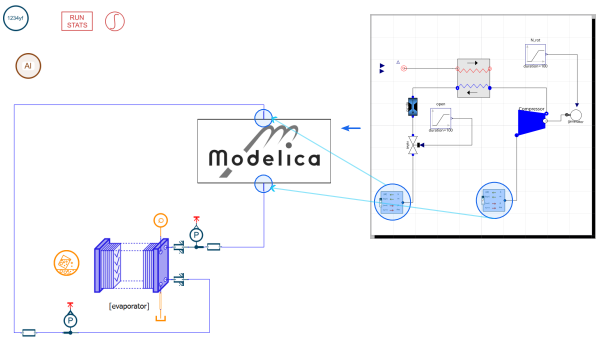

Figure 15. Heterogeneous cooling system with TPF evaporator, TPF Modelica interfaces and ThermoCycle submodels. The R1234yf is set in both worlds

5.3 Results

All submodels have been designed step by step to match the pre-design cycle calculated on the basis of the requirements listed in the descriptive section. The heterogeneous model was compared with the original homogeneous model built using only Simcenter Amesim's native submodels. Simulation time is 1000s with a tolerance of 1e-7 for both models. As shown in Figure 16, some differences in dynamics can be observed, as well as in the starting points. This is acceptable given that the compressor, the condenser, and the expansion valve found in the Modelica ThermoCycle library are not defined and built like the Simcenter Amesim's submodels replaced in this use case. Looking at the steady state, the cooling power of the evaporator sticks well to the requirement of 7 kW. The mass flow rate, pressures, and specific enthalpies are very similar in both models and are also aligned with the requirements. These observations validate the proof-of-concept that combines two modeling paradigms to build a closed-loop cooling system with its working fluid.

6 Discussion and Outlooks

In this use case, we intentionally selected the causality in order to obtain the ComId variable from Simcenter Amesim, rather than the other way around. This decision was made because our current OCT integration lacks internal communication to ensure a consistent value of the ComId from the Modelica's perspective. However, we can propose a workaround by introducing a small volume, such as a pipe, in the Simcenter Amesim component before the connection. This will reverse the causality and allow us to achieve the same conditions as demonstrated.

Furthermore, it's important to note that the connections between Modelica and Simcenter Amesim in our current

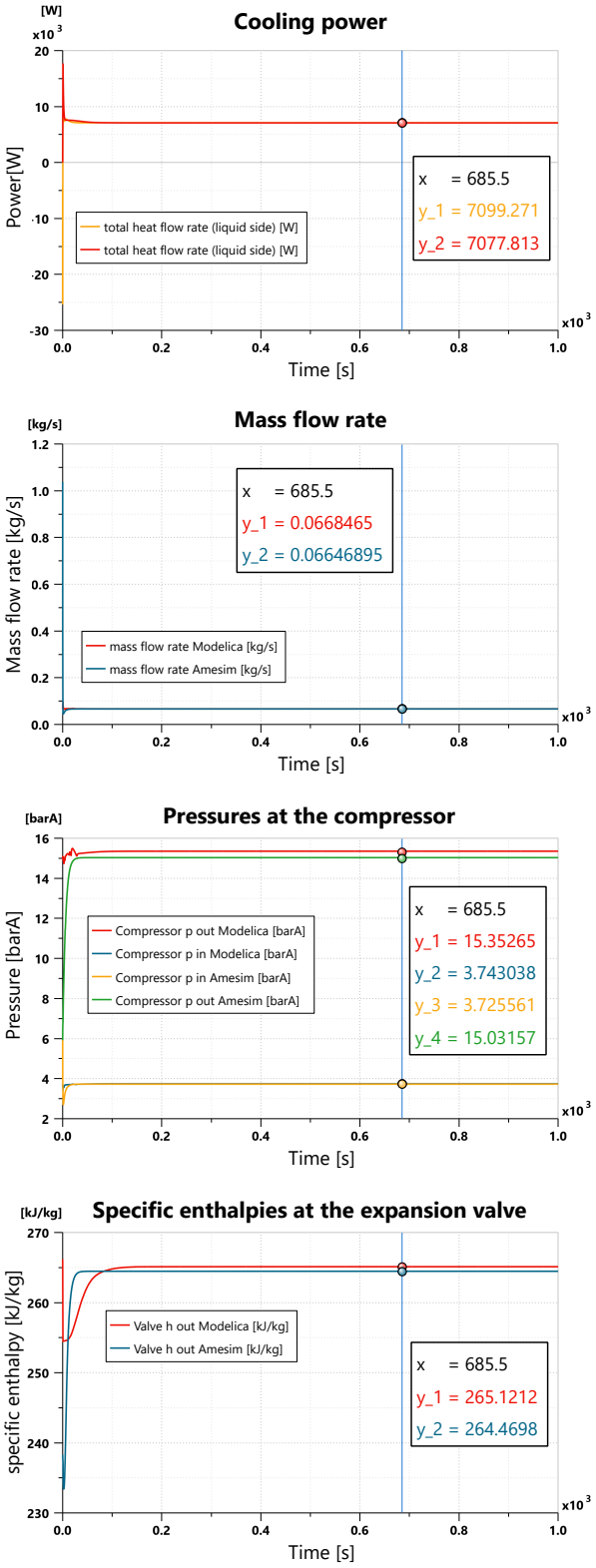

Figure 16. Comparison of simulation results between a homogeneous and a heterogeneous cooling system

demonstration are purely physical. Unfortunately, there is no capability to transmit information from Simcenter Amesim to an inner class in the Modelica world. There is currently no possibility to exchange the volume of the

system or the mass of the refrigerant between the two worlds, which could potentially impact the initialization process or certain internal variables, such as total pressure loss. To address this limitation, a workaround is to manually incorporate the correct mass, volume, or pressure loss values, calculated or fixed, in each world. However, this workaround assumes that the selected or developed Modelica library takes these factors into consideration. As a suggestion for future work, it would be beneficial to explore methods of communicating information outside the FMI standard to the compiler.

Another important milestone that has been identified pertains to AmeTpfMedia. Currently, the library only includes methods for calculating properties of a single fluid. However, TPF utilities already have the capability to calculate properties of fluid mixtures. Therefore, the next step would involve adding functionalities specifically dedicated to fluid mixtures, allowing for configuration of the number of fluids and their respective mass fractions.

7 Conclusions

This paper presented how the FMI terminals and the library AmeTpfMedia offer a new way to build heterogeneous closed loop fluid systems in Simcenter Amesim. This new solution meets needs coming from the industry where Simcenter Amesim and Modelica are most often used in separate departments. It can

A series of test cases demonstrated that AmeTpfMedia is well suited to the simulation of fluid systems, thanks to its accuracy and performance. Its architecture inherited from the MSL saves a lot of time when it comes to use open-source libraries also compatible with the MSL.

Finally, by following the steps described in the article, it becomes possible to recreate manually a Modelica physical interface to connect Simcenter Amesim to an in-house connector.

References

Bertsch, Christian et al. (2023-12). "Beyond FMI - Towards New Applications with Layered Standards". In: *Modelica Conferences* 204, pp. 381–388. ISSN: 1650-3740. DOI: 10.3384/ECP204381. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/947.

Casella, Francesco (2008-01). *ExternalMedia: a Library for Easy Re-Use of External Fluid Property Code in Modelica*. URL: http://coolprop.org/.

Dechesne, Bertrand et al. (2014). *ThermoCycle 2.3 Library*. University of Liege Thermodynamic laboratory - Belgium. URL: https://thermocycle-docs.readthedocs.io/en/latest/index.html.

Fritzson, Peter (2014-11). *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Vol. 9781118859124. Wiley-IEEE Press, pp. 1–1223. ISBN: 9781118989166. DOI: 10.1002/9781118989166. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118989166.

Hermien, Nugraheni and Tri Wiyatini (2021-10). *Engineering Systems Dynamics, Modelling, Simulation, and Design*. British Columbia Institute of Technology, p. 252. URL: https://open.umn.edu/opentextbooks/textbooks/1603.

Kunz, O. and W. Wagner (2012-11). "The GERG-2008 wide-range equation of state for natural gases and other mixtures: An expansion of GERG-2004". In: *Journal of Chemical and Engineering Data* 57 (11), pp. 3032–3091. ISSN: 00219568. DOI: 10.1021/JE300655B/ASSET/IMAGES/MEDIUM/JE-2012-00655B_0029.GIF. URL: https://pubs.acs.org/doi/abs/10.1021/je300655b.

Modelica Association (2017-04). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4*. Tech. rep. Linköping: Modelica Association. URL: https://www.modelica.org/documents/ModelicaSpec34.pdf.

Modelica Association (2022-04). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 3.0.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Vasiliu, Nicolae et al. (2018-04). *Simulation of Fluid Power Systems with Simcenter Amesim*. CRC Press. ISBN: 9781315118888. DOI: 10.1201/9781315118888. URL: https://www.taylorfrancis.com/books/mono/10.1201/9781315118888/simulation-fluid-power-systems-simcenter-amesim-nicolae-vasiliu-daniela-vasiliu-constantin-c%C4%83linoiu-radu-puhalschi.

Wetter, Michael et al. (2014-07). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7 (4), pp. 253–270. ISSN: 19401493. DOI: 10.1080/19401493.2013.765506.

XRG-Simulation GmbH TLK-Thermo GmbH, and (2023). *ClaRa 1.8.1*. URL: https://www.claralib.com/.