electronics

Article

The DLR ThermoFluid Stream Library

Dirk Zimmer *, Michael Meißner † and Niels Weber *

Institute of System Dynamics and Control, German Aerospace Center (DLR), 82234 Wessling, Germany
* Correspondence: dirk.zimmer@dlr.de (D.Z.); niels.weber@dlr.de (N.W.)
† TWAICE Technologies GmbH, 80807 München, Germany.

Abstract: This paper introduces the DLR Thermofluid Stream Library: a free open-source library for the robust modeling of complex thermofluid architectures. Designed to be easy to use, easy to adapt, and enriched by a number of examples, this library contains the fundamental components for many different applications, such as thermal management of electric cars, power plants, or building physics. Different from many previous software implementations, the library exploits a new computational concept on the basis of the fluid inertance that derives the mass flow balance in such a manner that large non-linear equations systems in implicit form can be avoided. Hence, there is a reliable and efficient computational scheme for simulation and initialization. Although this scheme is explained in detail in previous publications, this paper focuses on the implementational aspects of the library, explaining its structure, the underlying equations of key components and providing basic examples. In addition to the practical value of the library, we also aim to display the underlying thought driving the design of the software.

Keywords: thermal fluids; thermal process systems; robust modeling; heat exchangers; open source software

Citation: Zimmer, D.; Meißner, M.; Weber, N. The DLR ThermoFluid Stream Library. *Electronics* **2022**, *11*, 3790. https://doi.org/10.3390/electronics11223790

Academic Editor: Martin Sjölund

Received: 30 September 2022
Accepted: 1 November 2022
Published: 18 November 2022

1. Introduction

Streams of thermofluids form the basis of many natural and technical systems. Power plants, environmental control systems, refrigerators can all be represented as thermodynamic processes where components, such as pumps, heat exchangers, or valves manipulate the stream of a working fluid flowing from the component inlet to its outlet. A computationally very attractive formulation is to express this coupling of inlets and outlets by using algebraic equations in the form:

$$\Theta_{\text{out}} = g(\Theta_{\text{in}}, \dot{m}, \mathbf{x}) \tag{1}$$

with Θ being a tuple representing the thermodynamic state of the medium, $\dot{m}$ being the mass-flow rate and $\mathbf{x}$ the internal state vector of the component (e.g., rotational speed of a pump).

Such an algebraic coupling (as simple as it may be) implies a dramatic idealization of the actual underlying physical system. For instance, we implicitly assume that upstream changes in pressure and enthalpy take immediate effect downstream. Not only may such an assumption be completely inadequate under certain circumstances also these idealizations give rise to highly non-linear equation systems where there might be multiple solutions or none at all. Even if there is a unique solution, it may be very difficult to retrieve.

For these reasons, the modeling of thermofluid streams is rarely performed in a purely algebraic fashion. The modeler may want to break the algebraic system down to feasible complexity (for instance by adding volume elements), the modeler may want to model thermal time-constants (for instance by adding volume elements) or transport delay (for instance by adding volume elements). However, for many applications, there is something more clever than just adding volume elements. The library we present in this paper implements such an alternative that is favorable for many applications. It enables the modeler to formulate a system of thermofluid streams in a robustly solvable form using only a few additional state variables.

Although robustness is one major design target for the library, the ability to adapt the library to specific needs is another one. The set of potential applications is very large and its elements may differ in their underlying assumptions. Hence, any concise set of components cannot realistically be expected to provide full coverage. Instead, the library shall provide solid base components that are easy to read and that can then be adapted to the specific needs that are raised by the user's application field.

Robustness and adaptiveness are, hence, the two main points what shall make the library competitive with already existing solution in this domain, such as other free, commercial, or proprietary Modelica libraries [1–6], or non-Modelica M&S tools [7–9] or even our own previous solutions [10]. These libraries mostly represent either an algebraic modeling style or an ODE modeling style. The paper [11] offers a comparison of our DAE-based approach to such modeling styles. We think that the DLR Thermofluid Stream Library excels especially for complex thermal architectures that undergo configurational changes during simulation.

Before we recapitulate the underlying robust computational scheme and take a look at the library, let us look at an introductory example.

Introductory Example

Figure 1 contains an example application of an automotive battery, drive-chain, and cabin thermal control system. This represents a fairly complex system with several loops and three different media. Amongst other items, there are several switches to change the flow topology and a vapor cycle with two parallel evaporators. Although the example itself is not part of the library, it is entirely build out of its components, some of which internally combine multiple valve models to realize switches for bypasses and loops. For the sake of clarity, we have removed the control elements from the diagram.

The model diagram depicts some important distinctions from the fluid library as part of the Modelica Standard Library (MSL) [3]. First of all, this library features dedicated models for junctions and splitters and abstains from (ab)using the Modelica connector for this purpose. Second, the default connector type is directional, clearly indicating the direction of the stream. The library offers a solution for undirected flows too but it is supposed to be used only when inevitable. Third, the modeler shall break loops (not merely bypasses, but actual loops) using volume elements. Mostly, this will happen anyway since actual loops need actual reservoirs but cases, such as the combined loops of liquid cooling with its two reservoirs require an attentive modeler.

The translated model contains of 49 states (18 of which are attributed to the discretized heat exchangers of the vapor cycle) and contains only 4 non-linear systems of size one that can each be locally attributed to one of the four volume components. The initialization problem consists of two linear systems that are manipulated to size zero and poses no problem. The simulation is robust and will run through various topology switches and changing heat-loads, unless one of the media models is driven out of its temperature or pressure ranges.

Although some parameterization of all components will have to be made to match measured data of an actual system, most components come with usable default parameters, therefore even complex topologies can be quickly built up and simulate successfully.

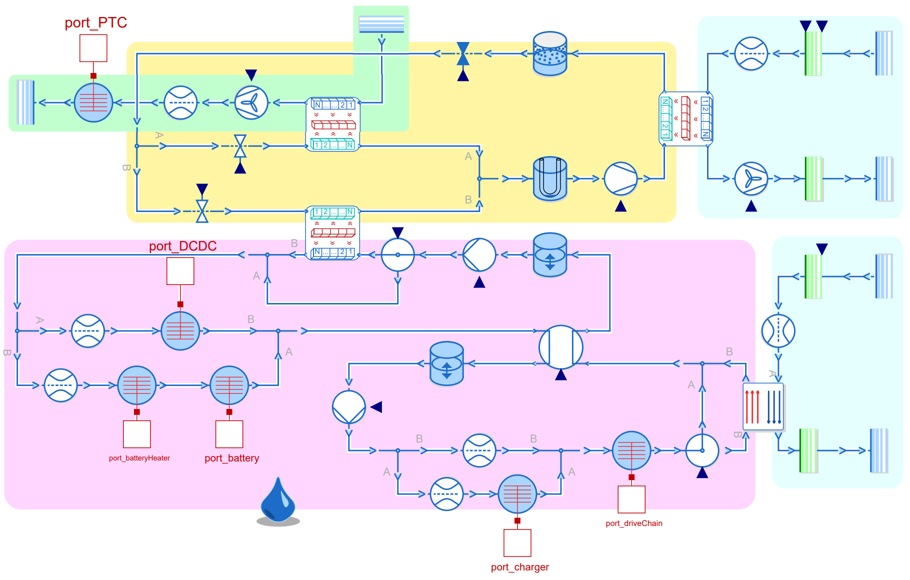

Figure 1. Example simulation of an automotive battery, drive-chain and cabin thermal control system. The glycol-water loop (magenta) is cooling battery, DC–DC converter, charger, and drive-chain. A four-way switch can split it into two separate loops instead of the combined loop that is shown. If needed, the loop is cooled by a radiator against fresh air (blue). Additionally it can be cooled with a R134a vapor-cycle (orange). The vapor-cycle is also cooling air going to the vehicle cabin (green) and, therefore, contains two parallel evaporators, that can independently be shut off by two valves when not in use. It is cooled against fresh air with a condenser. Furthermore, it contains receiver, accumulator, and an expansion valve. Battery and cabin-air can be heated with the battery heater and PTC, respectively. Both heat exchangers of the glycol loop can be bypassed if not in use. One of the fresh-air streams and the cabin-air stream is supported by a fan, while the other fresh-air stream is driven purely by dynamic pressure from the vehicle speed. See Figure 2 for a detailed shape explanation.

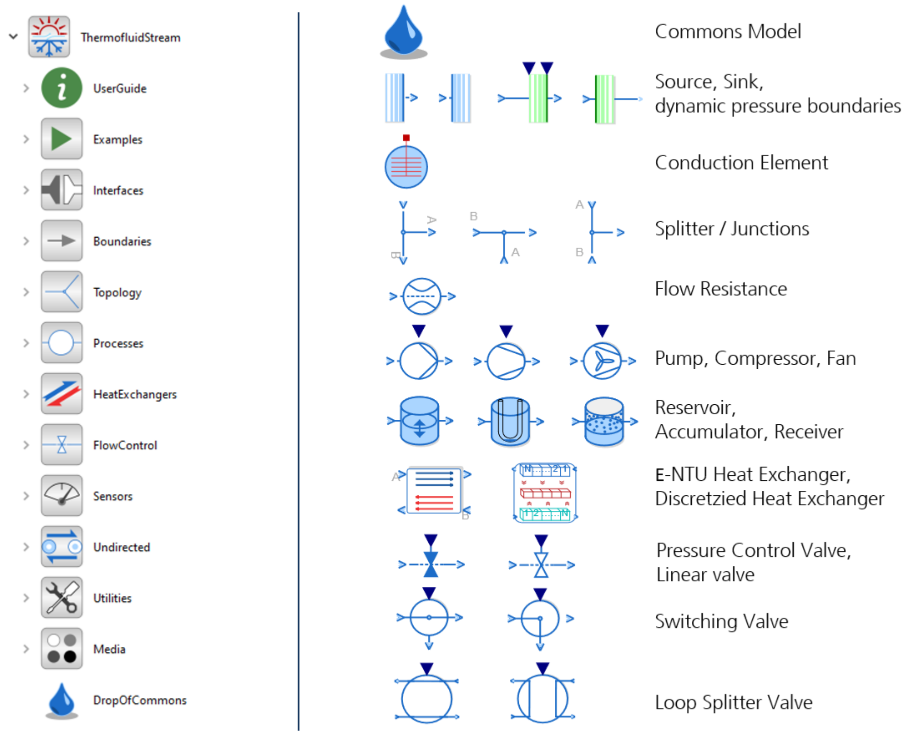

Figure 2. Overview of the library and its main packages on the left side and key components on the right side.

2. Robustness

2.1. Underlying Methodology and Assumptions

As suggested by its name a thermofluid stream is forming the central entity of this modeling approach. The stream is thereby bound either by dedicated boundary models such as sources or sinks or by elements that represent a volume of the medium.

Between these boundaries, the stream consists in a sequence of components, such as compressors, valves, etc., that may manipulate the thermodynamic state Θ. Shared among all these components is a common mass-flow rate. So all components alongside a stream uphold the mass-flow balance $\dot{m}_{\text{in}} = -\dot{m}_{\text{out}}$.

The mass-flow rate is always a state variable of the system. Each component of a stream expresses a differential equation of the form:

$$\frac{d\dot{m}}{dt} L = -\Delta r \tag{2}$$

Although r is denoted as inertial pressure and L is the inertance of the fluid. Please note that the inertance is solely defined by the geometry of the flow and independent of the thermodynamic state:

$$L = \int ds/A \tag{3}$$

with s being the length of the flow and A its cross-section area. We can make use of the inertial pressure r to decompose the general pressure gradient Δp into

$$\Delta p = \Delta\hat{p} + \Delta r \tag{4}$$

whereas $\hat{p}$ is introduced as steady-mass flow pressure (since $\hat{p} = p$ if $d\dot{m}/dt = 0$). Alongside a stream we approximate the thermodynamic state by using $\hat{p}$ instead of p, accepting a (mostly) small error for unsteady flow conditions. At each boundary we compensate the accumulated difference between $\hat{p}$ and p with the inertial pressure r and accelerate the corresponding mass flow with respect to r.

This leads to a very favorable structure of the equation system where all non-linear computations can be brought into explicit form and the only system of equations in implicit form is strictly linear. Hence, a robust solution of the system model can be reliably achieved supposing robust component and media models. More details on the approach and also the handling of junctions and splitters in [11].

Another way of looking at this approach is that we use different spatial resolutions for $\hat{p}$ and r. Although $\hat{p}$ may be resolved for each component, r is only resolved between boundaries of the stream. This is mostly fine because the impact of r on the thermodynamic state is typically low (or even zero for steady flow conditions). However, should the impact of r become vital (unsteady cavitation might be such a case), the modeler is advised to increase the spatial resolution by adding more volume elements at the place of concern.

2.2. Implementation in Modelica

To implement this decomposition, we use a connector that contains a pair of potential and flow variable (see Listing 1). The flow is naturally the mass flow rate and the corresponding potential is the inertial pressure since it is the potential variable that determines the dynamics of the flow. The thermodynamic state is then transferred as a signal. Since we are using the standard Modelica.Media library [2], using the state record of the media models seems a natural choice.

Listing 1. Connectors for directed thermofluid streams.

```
connector Inlet
  replaceable package Medium;
  SI.Pressure r;
  flow  SI.MassFlowRate m_flow;
  input Medium.ThermodynamicState state;
end Inlet;

connector Outlet
  replaceable package Medium;
  SI.Pressure r;
  flow  SI.MassFlowRate m_flow;
  output Medium.ThermodynamicState state;
end Outlet;
```

The approach chosen here is, thus, similar to [12] and a bit different to what has been described in [13]. Although the library is approximating the thermodynamic state on the steady mass-flow pressure, this is not made explicit. This means, in practice, that when one calls the function `Medium.pressure(inlet.state)`, the return value is the steady mass-flow pressure $\hat{p}$ and not p. An explicit denotation of the steady mass-flow pressure simply turns out to be too cumbersome and unpractical on the existing media model base. Yet it is important to have the underlying approximation in mind.

Additionally, a minor extension to the standard Media library is needed to support our interface: a function is added that retrieves the mass fraction for given thermodynamic state. For this reason, the DLR Thermofluid Stream Library currently uses a modified copy of Modelica.Media. Once the standard has been updated, the copy will be removed. Furthermore, XRG Simulation GmbH provided media models of refrigerants and further media that are compatible to our library.

A classic component which has a single stream from inlet to outlet can be built upon the following base model (Listing 2) that already contains the law for the inertial pressure gradient.

Listing 2. SISO basemodel.

```
partial model SISOFlow
  replaceable package Medium;
  parameter Utilities.Units.Inertance L = dropOfCommons.L;
  Inlet inlet(redeclare package Medium=Medium);
  Outlet outlet(redeclare package Medium=Medium);
protected
  outer DropOfCommons dropOfCommons;
equation
  inlet.m_flow + outlet.m_flow = 0;
  der(inlet.m_flow) * L  =  inlet.r - outlet.r;
end SISOFlow;
```

Although simplified w.r.t to the actual implementation, the code also displays the use of a global "DropOfCommons" model that is used to describe many generic parameters shared among many components. Among them is a default value for the inertance because often a replacement assumption is used and the corresponding dynamics is just seen as a way to reach the desired steady-state solution (or steady mass flow to be more precise).

2.3. Initialization

Per default, we use zero to initialize all mass flows. The occurring gradients in inertial pressure then direct the mass-flow rates toward their natural equilibrium. This approach is similar to ramping up a system from rest or plugging it in to a pressure source. We have successfully applied this method to a variety of systems so far and can confirm the applicability of this approach. The modeler needs to put much less thought into the initialization of its system. Nevertheless if desired, also other options are available.

2.4. Handling Zero-Mass Flow

The initialization approach alone stipulates the requirement that each component must be able to compute with zero-mass flow. Even stronger, we demand that each component can handle reverse flow in a well-natured manner. This means that the equations should not (unnecessarily) destabilize the system and if possible represent a physically plausible behavior. Care has been taken that each component fulfills these robustness requirements. This is especially relevant for active components, such as compressors and turbines that add or subtract power to a stream of fluid. Additionally, heat-exchangers require careful modeling in this respect.

2.5. Handling of Strong Pressure Gradients

Another issue that might lead the computation of the system to fail are strong pressure gradients that may lead to a negative steady-mass flow pressure. This may for instance happen when a valve is suddenly closed. Fortunately by attributing this strong pressure gradient to the inertial pressure this situation can be mitigated.

If not taken care of, negative pressure values at the outlet of a given component might arise from different situations, for example a sudden pressure drop at the inlet of a flow resistance. Since the pressure difference between inlet and outlet Δp over the component typically is a function of mass-flow, that remains smooth, a sufficiently fast decrease in inlet pressure will push the outlet pressure (see Equation (5a)) below zero.

We counteract this by lower bounding the outlet pressure of the components (see Equation (5b)). The residual of the computed pressure drop, that was cut of by the lower limit, is then subtracted from the inertial pressure (see Equation (5c)).

$$p_{\text{out,naive}} = p_{\text{in}} - \Delta p \tag{5a}$$

$$p_{\text{out}} = \max(p_{\text{min}}, p_{\text{out,naive}}) \tag{5b}$$

$$r = r_{\text{naive}} - (p_{\text{out}} - p_{\text{out,naive}}) \tag{5c}$$

This limits the steady-state pressure p, representing the thermodynamic state, to the lower limit p_{min}, which is beneficial for media models and the overall stability of the simulation. Furthermore the total pressure $\hat{p} = r + p$ is still experiencing the whole computed pressure drop Δp. The additional drop in inertial pressure, results in a rapid deceleration of the fluid, which is a very stable and physically plausible behavior of the component.

3. Library Overview

3.1. Library Structure

The central entity of the library is the thermofluid stream as described by the connector and the previously listed base-class. There will be boundaries for the stream. These are typically inlets and outlets but note that also volume elements represent boundaries from the perspective of a stream. The inlet of a volume is an outlet of the stream and vice versa.

The topology of the architecture is formed by splitters and junctions. Different from the fluid library in the MSL, there are extra components for this purpose and it is not performed using connector equations. Finally, all those components that manipulate the working fluid of a stream are collected under the term processes. Heat exchangers and valves (or similar mechanism) for flow control are moved up to the highest level due to their significance. Hence, the structure as in Figure 2 results.

Each of the packages contains a sub-package with corresponding test cases. An "Examples" package holds several application examples for the library, that showcase the capabilities and can act as starting points for users. We use both the test cases, as well as the examples, for regression testing during development of the library.

In addition, there is a common model for global parameters and settings that can be used as default for the individual components. This model is denoted as "DropOfCommons".

The strategy of the implemented library is that we want to provide robust generic models with only a few parameters for a first iteration of modeling, such as cross- and counter-flow heat exchangers with the ϵ-NTU method (see Section 3.2.3). These kinds of models will be detailed enough to capture the overall system behavior of a wide range of applications and can act as placeholders for higher-fidelity models in later design stages. Since these high-fidelity models will vary for each specific application the modeler will have to implement them for their specific use-case.

3.2. On Specific Components

3.2.1. Volume Models

Volume models take a special role in the thermofluid-stream approach, as they should be used to break algebraic loops for circular fluid flow, isolating the non-linear equation systems locally between the different components [14]. Each closed fluid loop should contain at least one volume model.

We provide volume models with one or multiple inlets with or without flexible walls, as well as reservoir, accumulator, and receiver models, which are also volumes and can be used to break loops. All of these are derived from base classes of volumes making it easy to implement additional specialized volume models if required by the modeler. The base classes define three differential conservation equations $\mathbf{x}_{\text{volume}} = (M, U, M_{\text{i}})$ with M, U, and M_{i}, being the mass, internal energy, and mass of the different mass fractions in a mixture, respectively. For maximum flexibility, volumes offer a broad range of options, such as removing inlet or outlet, or adding a heat-port, as well as different initialization methods.

In order to avoid very fast, undampened oscillations between two or more directly coupled volumes or other boundaries, a damping term on the change of mass contained by the volume [15] is implemented in all volumes. Contrary to an artificial flow resistance on the inlet or outlet, the damping term does not affect the steady-state solution since it acts only on the change of mass in the volume. Nonetheless, the damping can be switched off by modifying the volumes parameters. With this damping, directly coupled boundaries may still result in very fast oscillations, but at least the dynamics are damped and should be well manageable for a stiff-system solver. If possible though, direct coupling of volumes to other volumes or boundaries should be avoided. For more details, see [16].

3.2.2. Turbo Components

All components that transfer work between a mechanical flange and the fluid share a common partial model "PartialTurboComponent" governed by Equation (6)

$$(\Delta p, \tau_{\text{s}}) = f(\dot{m}, \omega, \Theta_{\text{in}}) \tag{6a}$$

$$J\dot{\omega} = \tau - \tau_{\text{s}} \tag{6b}$$

$$\Delta h = \tau_{\text{s}} \cdot \omega / \dot{m} \tag{6c}$$

where Δp is the pressure gain over the component, ω is the angular velocity, J the moment of inertia, τ the torque applied to the flange, τ_{s} the torque needed to maintain static operation in the current conditions and Δh the specific enthalpy the fluid gains from inlet to outlet.

Equation (6a) represents the pressure/torque characteristic of the component and is not implemented in the partial class. Different implementations of f in child classes allow for pumps, compressors, turbines, fans, or other turbo components and for different levels of fidelity. Note that Equation (6) holds no assumption on the specific thermodynamic process of the turbo-component (e.g., compression/expansion with a fixed isentropic coefficient), since different processes can be implemented by different functions f in Equation (6a).

Equation (6b) can be replaced by a boundary condition on ω, when the angular dynamics of the component is not of interest.

Equation (6c) is additionally normalized for low mass-flow, effectively limiting $|\Delta h|$. If the fluids enthalpy is increased in the component, any work the fluid cannot take on is dumped onto a heat-port. If enthalpy is taken from the fluid (e.g., in a turbine) τ_s is reduced to still fulfill Equation (6c) in case of normalization, limiting the work that can be taken out of the fluid.

3.2.3. Heat Exchangers

For heat exchange between two fluids, two different approaches are used in this library. For applications with single-phase fluids on both sides, the ϵ-NTU method is implemented. For fluids with phase transition (for example refrigerants), the heat exchanger is discretized in multiple heat exchanging elements.

ϵ-NTU Method

When the outlet temperatures of the heat exchanger are not known a priori, the ϵ-NTU method is most convenient. It provides relatively simple correlations for different types of heat exchangers (counter-flow, cross-flow, parallel-flow, etc.). The effectiveness ϵ of a heat exchanger is defined as the ratio between the actual and the maximum heat flow rate:

$$\epsilon = \frac{\dot{Q}}{\dot{Q}_{\text{max}}} = \frac{\dot{Q}}{C_{\text{min}} \Delta T_{\text{max}}} \tag{7}$$

To obtain the maximum possible heat flow rate, the heat capacity rates $C = c_\text{p}\dot{m}$ on both sides (hot/cold) of the heat exchanger are compared. The side with the smaller heat capacity rate ($C = C_{\text{min}}$) needs less energy to experience the maximum temperature difference ($\Delta T_{\text{max}} = T_{\text{h,in}} - T_{\text{c,in}}$).

With those quantities, the so called Number of Transfer Units (NTU) can be calculated:

$$NTU = \frac{kA}{C_{\text{min}}} \tag{8}$$

where k is the overall heat transfer coefficient and A is the surface area for heat transfer. For any heat exchanger it can be shown that [17]:

$$\epsilon = f(NTU, C_\text{r}) \tag{9}$$

This means the effectiveness ϵ of the heat exchanger is a function of the dimensionless number of transfer units and the ratio of the minimal and maximal heat capacity rate $C_\text{r} = \frac{C_{\text{min}}}{C_{\text{max}}}$. For each type of heat exchanger, a specific relation can be found in literature, for example [18]. For cross-flow heat exchangers with both fluids unmixed, the effectiveness can be obtained by:

$$\epsilon = 1 - exp\left[\left(\frac{1}{C_\text{r}}\right) NTU^{0.22} exp[-C_\text{r} NTU^{0.78}] - 1\right] \tag{10}$$

This correlation and similar ones for counter-flow are implemented in the library.

To determine the thermodynamic state at the outlet of the heat exchanger, the approach is slightly modified to our purpose. To avoid non-linear equation systems, especially when connecting multiple heat exchangers in series, it is beneficial to use the outlet enthalpy as a state. Therefore, the efficiency of the heat exchanger is formulated in terms of specific enthalpy:

$$\Delta h = \epsilon \Delta h_{\max} = \epsilon c_p \Delta T_{\max} \tag{11a}$$

$$h_{\text{out}} = h_{\text{in}} - \Delta h; \tag{11b}$$

Thus, the actual heat flow rate can be obtained from:

$$\dot{Q} = \dot{m}\Delta h \tag{12}$$

To become a state variable, the outlet enthalpy h_{out} is filtered with a first order term with time constant τ_{filter}:

$$\frac{\partial h_{\text{out}}}{\partial t}\tau_{\text{filter}} = h_{\text{in}} - \Delta h - h_{\text{out}} \tag{13}$$

The thermodynamic state is eventually retrieved from the specific outlet enthalpy h_{out} to prevent the creation of non-linear equation systems.

Discretized Heat Exchanger

When condensation and evaporation become relevant in a heat exchanger, a method is needed that is able to handle phase transition. Generally, there are two main approaches suitable for this application: the moving boundary approach and the discretization of the heat exchanger into a finite number of elements. In this library, the latter is implemented because it promises very robust behavior and can easily be understood.

The discretized heat exchanger consists of N heat conducting elements on each side of the fluid. They are connected via a thermal conductor from the MSL. The number of discretization elements can be set by the modeler. Figure 3 shows how the single elements are connected to each other. The fluid ports are arranged in terms of a counter-flow heat exchanger.

When dividing the heat exchanger in several elements, it is advantageous to model them in a way that no oscillations occur when multiple elements are connected to each other. To this end, the mass in each element is assumed to be quasistationary $(M = \rho V)$ and the inlet mass flow is coupled to the outlet mass flow $(\dot{m}_{\text{in}} = -\dot{m}_{\text{out}})$. Although this assumption neglects the change of enthalpy attributed to dM/dt for changing densities, in our experience it does not change the result drastically while reducing the problem's complexity. Now, the energy balance of each element is stated in the following form:

$$M\frac{\partial h}{\partial t} = \dot{Q} + \dot{m}(h_{\text{in}} - h) + V\frac{\partial p}{\partial t} \tag{14}$$

where h is the specific enthalpy at the outlet of the element. For the sake of robustness, the change of pressure in the fluid is neglected in the energy Equation (14) $(V\frac{\partial p}{\partial t} = 0)$ and, therefore, the pressure input is not required to be smooth.

The convective heat transfer from or to the fluid is calculated as follows:

$$\dot{Q} = \alpha A(T_{\text{heatPort}} - T_{\text{surface}}) \tag{15}$$

where α is the coefficient of heat transfer and A the surface area. The temperature difference is calculated between the fluid temperature T and the surface temperature $T_{heatPort}$. The discretization elements for single-phase and two-phase media are modeled the same way. The only difference lies in the estimation of the coefficient of heat transfer α. In general, a detailed calculation of the coefficient of heat transfer is not trivial. Hence, we offer a pragmatic approach exploiting that the coefficient of heat transfer α can be stated in terms of the Nusselt-Number [18] which, in turn, depends on the Reynolds number (presuming forced convection):

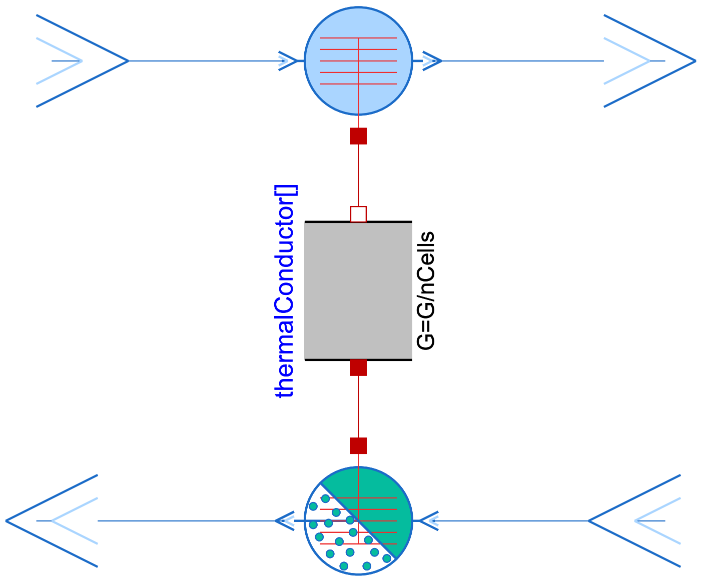

Figure 3. Cell model of discretized heat exchanger.

$$Nu = \frac{\alpha L}{\lambda} = C Re^m Pr^n \tag{16}$$

with the characteristic length L and the conductivity of the fluid λ. The values of the coefficient C and the exponents m and n are dependent on geometry and flow characteristics. Since the Reynolds number is proportional to the mass flow $\dot{m}$, we can derive a simple scaling law based on the Reynolds exponent m:

$$\alpha = \alpha_{\text{nom}} \left(\frac{|\dot{m}|}{\dot{m}_{\text{nom}}} \right)^m \tag{17}$$

For turbulent flow, the Reynolds exponent m is equal to 0.8. The coefficient of heat transfer for the multiphase element has to be estimated differently. It is determined by the actual phase in each discretization element, therefore it is dependent on the vapor quality. According to the single phase, for each phase (liquid, vapor, two-phase) a nominal coefficient of heat transfer can be set and the actual coefficient is calculated accordingly:

$$\alpha_{\text{liq}} = \alpha_{\text{liq,nom}} \left(\frac{|\dot{m}|}{\dot{m}_{\text{nom}}} \right)^{m_c} \tag{18a}$$

$$\alpha_{\text{vap}} = \alpha_{\text{vap,nom}} \left(\frac{|\dot{m}|}{\dot{m}_{\text{nom}}} \right)^{m_e} \tag{18b}$$

$$\alpha_{\text{tp}} = \alpha_{\text{tp,nom}} \tag{18c}$$

The Reynolds exponents for normalisation of the heat transfer coefficient for evaporation ($m_{\text{e}} = 0.5$) and condensation ($m_{\text{c}} = 0.4$) are taken from [19,20]. In the two-phase region, a constant coefficient of heat transfer is assumed. Furthermore, a minimum value for the coefficient of heat transfer α_{min} is introduced to ensure heat transfer at zero mass flow. The coefficient of heat transfer on the two-phase side of the heat exchanger depends on the actual phase. Therefore, the vapor quality χ has to be calculated in each element, using the dew and bubble enthalpies of the fluid:

$$\chi = \frac{h - h_{\text{bubble}}}{h_{\text{dew}} - h_{\text{bubble}}} \tag{19}$$

The coefficient of heat transfer used in the two-phase elements thus is formulated as a function of the vapor quality $\alpha(\chi)$. For smooth transition between different coefficients during phase change, an interpolation is applied. The definition of the vapor quality (Equation (19)) allows it to go below zero (when subcooled) and above one (when superheated). This allows the interpolation to be formulated across the phase boundaries and avoids jumping in those critical regions. The test models for a condenser and evaporator show robust and valid behavior of the discretized heat exchanger, although the performance is highly dependent on the number of discrete elements. Section 3.3 contains a corresponding application example.

3.2.4. Valve Models

As flow control is essential in thermofluid networks, the library provides various valve components to meet a wide range of modeling requirements. The package contains different types of valve models that either represent control valves or more functionally driven valve models. The latter are directly combined with splitter models to enable robust topology switching in complex architectures (Figure 1). The focus in this chapter lies on the description of the control valves and its different behavior.

Valve characteristics can be very important for the control design and control authority. The library, hence, features the most commonly used characteristic curves for pressure loss.

To understand the specific valve behavior, lets revise the physical principles of valve modeling. Depending on the degree of opening u, valves are representing a respective flow resistance $\zeta(u)$. Assuming turbulent flow, the resulting pressure loss can be calculated relative to the dynamic pressure:

$$\Delta p = \zeta(u)\frac{\rho}{2}\bar{v}^2 \tag{20}$$

with ρ being the density of the medium and $\bar{v}$ the mean flow velocity. Substituting the velocity by the mass flow $\dot{m}$ and taking the cross-sectional area A of the valve into account, we can express Equation (20) as follows:

$$\Delta p(u) = \zeta(u)\frac{1}{2\rho}\left(\frac{\dot{m}}{A}\right)^2 \tag{21}$$

or

$$\dot{m}(u) = A\sqrt{\frac{2\rho}{\zeta(u)}\Delta p}. \tag{22}$$

In engineering, it is common practice to describe valve behavior against a reference. To this end, the mass flow or volume flow is measured for a fully opened valve ($\zeta_1 = \zeta(u = 1)$) at reference pressure Δp_0 and reference density ρ_0:

$$\dot{m}_0 = A\sqrt{\frac{2\rho_0}{\zeta_1}\Delta p_0} \tag{23}$$

$$\dot{V}_0 = A\sqrt{\frac{2}{\zeta_1}\frac{\Delta p_0}{\rho_0}} \tag{24}$$

Dividing the general behavior by the specific behavior ($\frac{\dot{V}(u)}{\dot{V}_0}$) yields:

$$\dot{V}(u) = \dot{V}_0\sqrt{\frac{\zeta_1}{\zeta(u)}\frac{\rho_0}{\Delta p_0}\frac{\Delta p}{\rho}}. \tag{25}$$

Usually, the specific reference values are chosen to be $\Delta p_0 = 1\,\text{bar}$ and $\rho_0 = 1000\,\text{kg/m}^3$ and the reference volume flow $\dot{V}_0$ is denoted as K_V-value in the unit of m^3/h. The desired behavior for control valves is a well-defined increase in the volume flow when the valve is opened:

$$\dot{V}(u) = \kappa(u)\dot{V}(u = 1). \tag{26}$$

The factor κ (as well as the degree of opening u) is a value between 0 and 1. With respective K_V-values for different valve openings, the characteristic curve is given as:

$$\kappa(u) = \frac{K_V(u)}{K_{VS}}. \tag{27}$$

The K_{VS}-value represents the K_V-value for a fully opened valve and is usually given in the data sheet by the valve manufacturer. It can be set in the parameter window of the *BasicControlValve* component of the library. In addition to the metric definition ($K_{VS}[\mathrm{m}^3/\mathrm{h}]$), other common unit definitions are also supported ($C_{VS}[\mathrm{gal}/\mathrm{min}]$).

The most commonly used characteristic curves are depicted in Figure 4. They describe the relation of the actual volume flow at a given valve opening (K_V) with respect to the volume flow of a fully opened valve K_{VS}. It can be set to be a linear, parabolic, or equal-percentage relation.

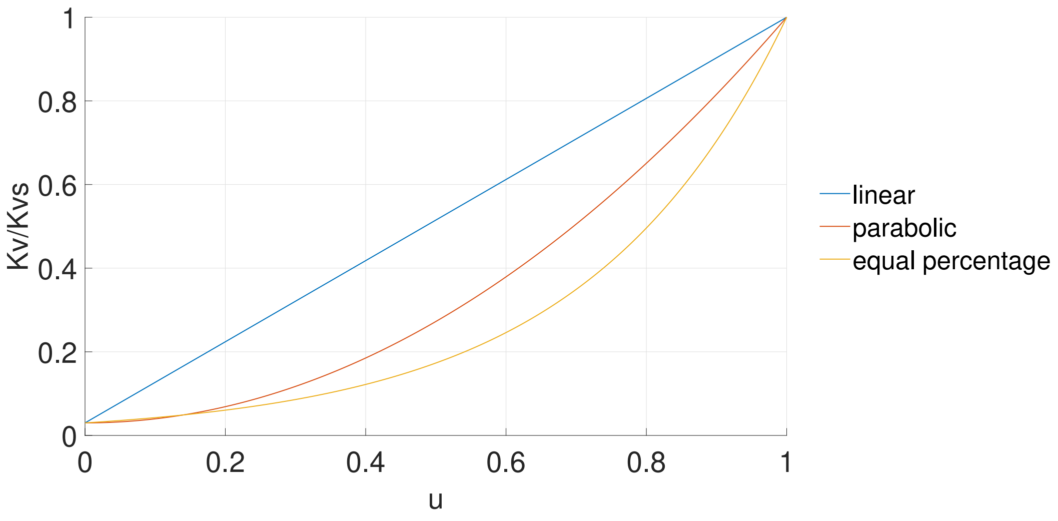

Figure 4. Characteristic valve curves.

With respect to controllability, the characteristic curve is limited to a minimum flow value k_{min}. It represents the remaining flow at a fully closed valve ($u = 0$) as a fraction of the maximum flow at $u = 1$ and can be set in the parameter window.

Depending on the type of valve, instead of the characteristic curve $\kappa(u)$ (Equation (27)), the ζ-values for different degrees of opening can be given. An example for a sliding valve is given in Table 1:

Table 1. ζ-values for different degrees of opening of a sliding valve with diameter d and height h of the slider.

h/d	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.12	0.1
ζ	0.00	0.06	0.17	0.44	0.98	2.06	4.60	10.00	35.00	97.80	100.00

The ζ-curve for the sliding valve of Table 1 is implemented as a start and can be chosen in the library component *SpecificValveType*. Further types of valves (e.g., throttle valve, ball valve) will be implemented in the future or can be added by the user.

In order to obtain a general equation for the pressure loss, regardless of the given curve, a relation between ζ-values and $\kappa(u)$ has to be found. It can be derived from Equations (25) and (26):

$$\kappa(u) = \sqrt{\frac{\zeta_1}{\zeta(u)}}. \tag{28}$$

This relation is eventually put into the general pressure loss function for all valve types:

$$\Delta p = \Delta p_0 \frac{\rho_0}{\rho} \left(\frac{\dot{m}}{\kappa(u)\dot{m}_0} \right)^2 . \tag{29}$$

All valves provide the feature to invert the control input. This results in a reversal of the actuation signal and, hence, $u = 1$ represents a fully closed valve. This feature is useful for switches that are represented as a combination of valves and splitters, as two flow paths can be controlled with only a single actuation input.

3.2.5. Sensor models

All implemented sensor models can be used in two ways. Firstly, they output the measured signal as a RealOutput as it is common for Modelica sensor models. Additionally, they display the current signal value during simulation using the DynamicSelect command (see Figure 5). This enables the user to obtain a fast, intuitive understanding of the current state of the simulation without the need for displaying any signal curve. We found the second use very practical and are using the sensors mostly in this way, which is why the actual signal output is conditionally removed by default.

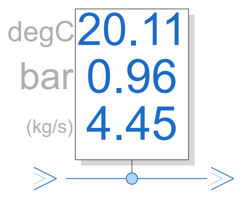

Figure 5. Exemplary sensor displaying three quantities.

The basic sensor types are sensors for measuring T and p or T, p and $\dot{m}$ with a selectable unit. Furthermore, we implemented sensor models, which are capable of sensing a wide range of easily expandable quantities that can be selected in the parameters of each sensor. The three sensors are for quantities related to the media state (e.g., p, $\hat{p}$, s, and c_v), quantities for two-phase media (e.g., steam-quantity or temperature over the saturation-point) and quantities related to mass-flow (e.g., $\dot{m}$, $\dot{C}_p$). For the first two, we also implemented sensors to sense the difference of the measured quantity between two points in the model. All sensor models implement an optional low-pass that can be used to model a generic sensor dynamic or to break loops of non-linear dependencies through the sensor without an additional PT1-block, leading to an overall cleaner looking model. All sensors that do not measure mass-flow can be connected directly to the network without needing a dedicated splitter, since they represent a boundary condition of zero mass-flow through them.

3.3. Specific Solution for Undirected Flows

Within the library, we provide a package containing components that can have undirected flows. It is similarly structured to the main library but contains less components. For these components however, the direction of mass-flow does not need to be known a priori and is determined dynamically during the simulation. The approach for undirected stream-dominated flow simulation is further detailed in [14]. Note that this approach is still stream-dominated and, while we continue to demand robustness for low and zero mass-flow, the results may not be valid for these conditions. The undirected simulations are, therefore, interesting for applications where non-zero mass-flow operating points are present for both flow directions while the switching dynamics between mass-flow directions are not of interest. In practice, this will be the case for many applications, such as a combined vapor-cycle/heat-pump.

When the direction of mass-flow is reversed, the flow of information is reversed with it. Therefore, a undirected connector carries information about the thermodynamic state in both directions: forward and rearward (see Listing 3).

Listing 3. Two connectors for undirected flows.

```
connector Thermalplug_fore
  replaceable package Medium;
  SI.Pressure r;
  flow SI.MassFlowRate m_flow;
  input Medium.State state_rearwards;
  output Medium.State state_forewards;
end Thermalplug_fore;

connector Thermalplug_rear
  replaceable package Medium;
  SI.Pressure r;
  flow SI.MassFlowRate m_flow;
  input Medium.State state_forewards;
  output Medium.State state_rearwards;
end Thermalplug_rear;
```

Each component also computes its influence on the state in both the forward and backward direction (see Equation (30)).

$$\begin{aligned} \Theta_{\text{out,fw}} &= g_{\text{fw}}(\Theta_{\text{in,fw}}, \dot{m}, \mathbf{x}) \\ \Theta_{\text{out,bw}} &= g_{\text{bw}}(\Theta_{\text{in,bw}}, \dot{m}, \mathbf{x}) \end{aligned} \tag{30}$$

When connecting these undirected components in a complex topology, junctions and splitters cannot be distinguished and become generic nodes. The implementation of a node has to avoid cyclic dependencies of the signal flow, as well as to provide a regularization around the zero-mass flow regime. It is, thus, almost a re-implementation of the Modelica stream connector [21]. Unfortunately, the regularization of the stream connector semantics is numerically vague and also not subject to a regularization parameter. The width of the regularization scheme may however play an important role in situations of flow reversal and also influences the eigenvalues of the system. Hence, a proper option for parametrization is needed and the Modelica stream connector itself could, thus, not be used for this library. Here, the default value of the regularization width is specified by the dropOfCommons.

It is strongly advisable to use undirected components only when the flow-direction is really unknown. Knowing the direction a priori is a too valuable piece of information to throw away. For instance, using directed components helps avoiding spurious loops (those which appear in the connection graph but are never realized by the fluid flow) and the need to cut those loops. Additionally, creating models that work in both directions is not always reasonable and certainly creates often code that is needlessly complex. Please note that we provide also adapters between directed and undirected stream networks. These can be used to isolate parts of the network that require undirected flow, and keep the rest directed.

An example architecture that contains undirected components is given in Figure 6. It shows a reversible heat pump as it can be used for residential air conditioning. The speciality of this system is that the direction of the refrigerant flow can be reversed. The heat exchangers can, thus, act as evaporator or condenser according to the current cycle operation. In cooling mode (blue arrows), the indoor unit acts as an evaporator and the outdoor unit (blue) acts as a condenser. Thus, the heat is absorbed from the inside air and rejected to the outside. In heating mode (red arrows), the cycle is reversed which makes the indoor unit the condenser and the outdoor unit the evaporator. Hence, the heat is absorbed from the outside and rejected to the inside. The system is built out of a combination of undirected and directed components. It consists of an undirected phase separator (receiver) and two separate metering devices (magenta). This allows us to control the superheating temperature after the evaporator in both operating modes. In practice, the change of flow direction is carried out by a reversing valve. In our example, we control the flow direction by a system of valves and undirected junctions (yellow). The example simulates robustly with fixed boundary conditions for the air side and the compressor and the cycle can be reversed during simulation.

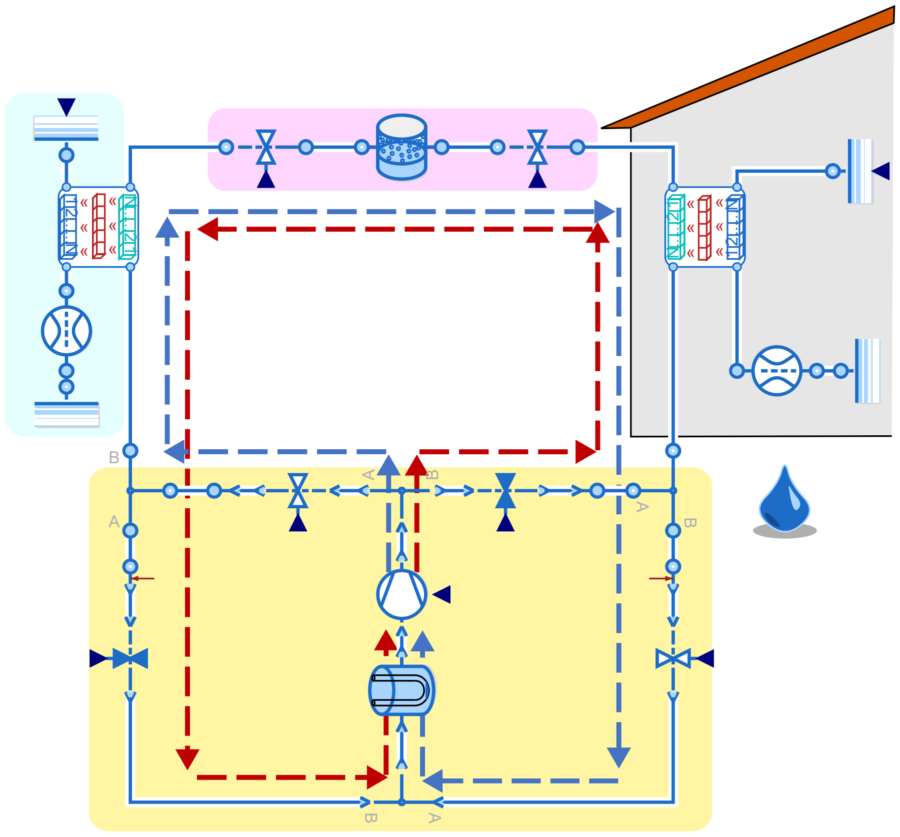

Figure 6. Example of a reversible heat pump that contains two undirected heat exchangers. The operating mode of the cycle can be switched during simulation.

3.4. Specific Solution for Dynamic Pressure

The dynamic pressure arises from the macroscopic motion of the fluid. Its computation for a given mass flow rate hence also requires information on the geometry. Typically, the cross-section area is used to compute the velocity:

$$v = \frac{\dot{m}}{A\rho}$$

The dynamic pressure q is then

$$q = \rho/2v^2 = \frac{\dot{m}^2}{2\rho A^2}$$

If the dynamic pressure is vital for each part of the system, it is a natural choice to include the cross-section area in the connector and specify the geometry at each component. However, we do not think that this represents the majority of use-cases for our library. Many thermodynamic processes are adequately described without needing the dynamic pressure. Its occurrence is mostly confined to special sub-systems, such as ram-air inlets, venturi pumps, or diffusers. Under this presumption, the need to describe the geometry in all components does more harm than good. Providing a localized solution for the dynamic pressure seems to be the most useful approach. In this way, it can be considered when needed and ignored otherwise. Hence, we introduce special boundaries for entering and exiting a zone where dynamic pressure is considered. One side of this boundary expresses the static pressure for a specified velocity assumption whereas the other side computes the velocity resulting from mass flow-rate, density, and cross-section area. The boundary then assigns a pressure difference so that the total pressure balance is upheld. A typical use of such a boundary is its use of ram-air going through a heat exchanger as in a car, as shown in Figure 1.

A typical component that uses dynamic pressure is a nozzle that accelerates (or decelerates) a fluid. In case the fluid is diffused, an increase in mass-flow rate will increase the outlet pressure. Hence, a correct formulation of the boundary conditions is needed because otherwise the inertial dynamics of the fluid may destabilize the system. A corresponding example of the Venturi-effect is part of the library.

4. Easy-to-Read, Easy-to-Adapt

4.1. General Modeling Style of Components

In general, the library is structured by a flat hierarchy, limiting the use of partial models, functions, and packages to only those cases with high usefulness. This is done intentionally to increase the readability and understandability of the individual components while still utilizing the might of object-oriented modeling when it is of benefit.

Components are implemented to provide a formulation of Equation (1) in explicit form, whenever feasible. Nonetheless, sometimes certain quantities that are required to compute the outlet state Θ_{out} depend not only on the inlet state Θ_{in} mass-flow $\dot{m}$ and component state $\mathbf{x}$, but also on the outlet state itself. An example is the change in dynamic pressure, that depends on the outlet density. This typically results in systems of non-linear equations within the component. Although the stream-dominated approach still manages to keep the non-linear systems separated in small local ones, for general simulation speed and the real-time application of the library, even small non-linear systems are undesirable. Therefore, we decided to avoid non-linear equation systems larger than size 1. This can be achieved by reformulating the equations in a manner and/or applying simplifications until the components' non-linear equation systems drop to the desired size, or by introducing the problematic quantity as an artificial state within the component $\tilde{x}$, effectively implementing a fast low-pass on it, as it was done in Section 3.2.3. Although the latter solution avoids simplifications, it introduces additional artificial states, whose time-constants have to be chosen with care. Although this is not an ideal solution, when done right the result is a fast running and accurate simulation, with only local non-linear systems of maximum size 1. For hard real-time applications the remaining non-linear systems can still be resolved by adding additional states in the corresponding components.

Although implementing the models, we documented the sources of equations, as well as the simplifications and mental models for the components in code and the documentation annotation. Furthermore, intuitive icons provide a good readability of the high-level models.

4.2. Adapt to Your Own Needs: A Use Case

Although many use-cases can be simulated by the library's standard components, many others will require specialized components. The relative flat implementation style enables the modeler to easily implement own components and adapt the library for their needs. An example for this is provided in Figure 7. It depicts a simulation model for an espresso machine.

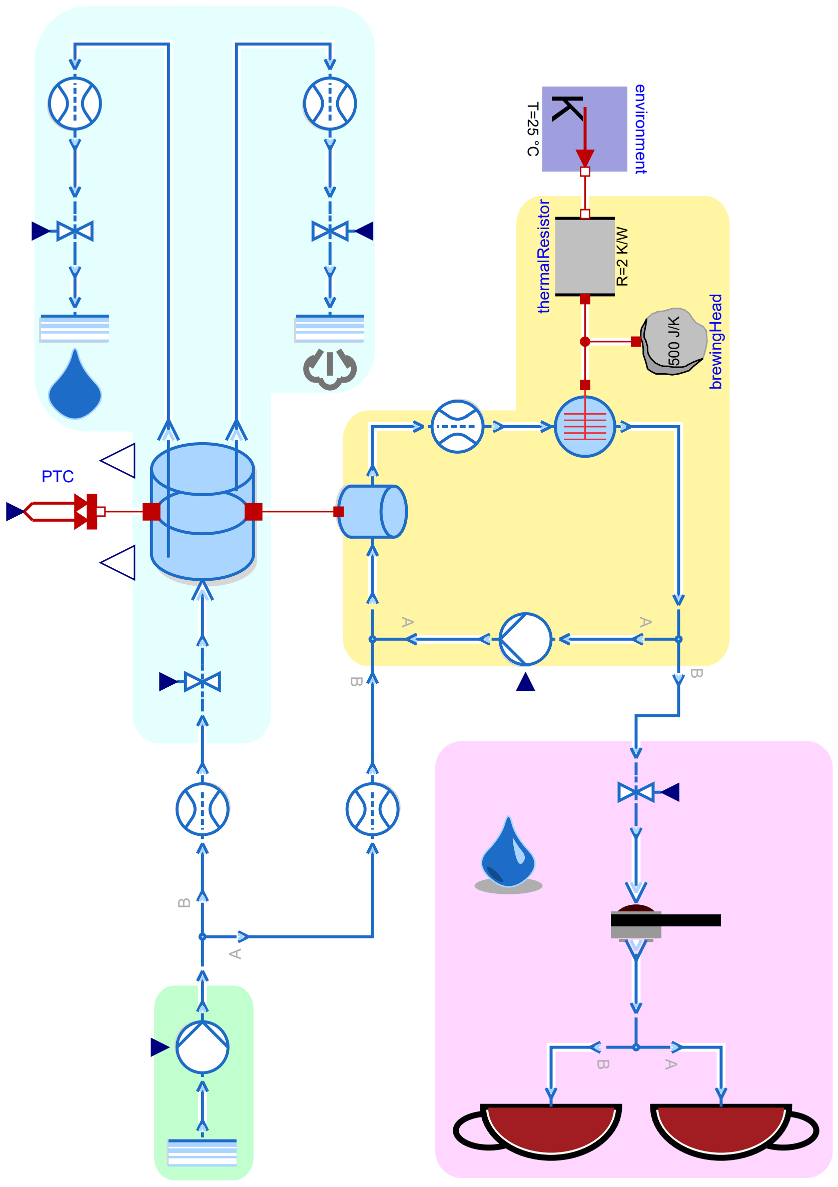

Figure 7. Example of a model with individualized component models. This model simulates an espresso machine composed of water source and pump (green); boiler, water, and steam outlets (blue); brewing head with cycling water (yellow); and valve, coffee strainer, and cups (magenta). Cold water enters the machine through a source at atmospheric pressure and becomes pressurized by a pump to 8.5 bar. Part of the water goes through a control valve into a boiler, where the pressure is regulated by heating through a PTC and the water level is controlled by the valve. For this purpose, the boiler outputs both of these quantities. From the boiler, steam or boiling water can exit if the corresponding valve is opened. The boiler is in thermal contact to a secondary small volume from which water cycles to the brewing head and back, heating up the brewing head. If the machine is at temperature, the valve in front of the coffee strainer is opened and water flows from the pump into the cups, picking up heat in the secondary volume and the brewing head.

Although many standard components are used, the sinks for steam and boiling water, the coffee strainer, the cups, and the boiler are specialized components. All except the boiler are internally composed of standard components and only implement a specialized icon. The coffee strainer consists of a flow resistance and each cup contains a flow resistance and a sink, as well as a variable to track the level of coffee in the cup. The two sinks for steam and water only update the image of the original sink.

Contrastingly, the boiler, while derived from the standard volume, is a fully individual component. It has one inlet and implements the conservation equations for mass and energy similar to a normal volume, but contains two heat-ports and two specialized outlets with states on the bubble- and dew-line, respectively, instead of one normal outlet. Additionally, liquid level and pressure are output as real numbers and a different icon is implemented. The relevant code for the boiler is depict in Listing 4.

Listing 4. Relevant code for the individual component boiler.

```
der(m) =inlet.m_flow + steam_out.m_flow +water_out.m_flow;
der(U) =heatport_HX.Q_flow + heatport_heat.Q_flow +
    inlet.m_flow * Medium.specificEnthalpy(inlet.state) +
    steam_out.m_flow * Medium.specificEnthalpy(
    steam_out.state) + water_out.m_flow *
    Medium.specificEnthalpy(water_out.state);

steam_out.state = Medium.setDewState(Medium.setSat_T(medium.T
    ));
water_out.state = Medium.setBubbleState(Medium.setSat_T(
    medium.T));

steam_out.r = 0;
water_out.r = 0;
inlet.r = medium.p - Medium.pressure(inlet.state);

y_out = m*(1 - x)/Medium.bubbleDensity(Medium.setSat_T(
    medium.T))/V;
p_out = medium.p;

heatport_heat.Q_flow = UA_heat*(heatport_heat.T - medium.T);
heatport_HX.Q_flow = UA_HX*(heatport_HX.T - medium.T);
```

5. Concluding Remarks

5.1. Library Source and Terms of Use

The library is available on GitHub: github.com/DLR-SR/ThermofluidStream (accessed on 16 November 2022).

It is available under the 3-Clause BSD License. It has been developed using Dymola and is based on Modelica 3.2.3. Pedantic checking has been applied to all components in order to improve cross-tool compatibility. Compatibility with Open Modelica and Modelon Impact has been achieved to a large degree and is documented in corresponding issues on GitHub.

If you publish work that is based on this library, please cite this paper and [11]. We also welcome feedback in the form of issues raised on GitHub. Additionally, when you have positive feedback, you can feel free to raise an issue to share your experience. Have fun!

5.2. Remarks for the Modelica Association

The development of a library also helps to identify (or remind about) certain potential improvements of the Modelica standard. Hence, the following sections shall be understood as public feedback for the Modelica Association.

5.2.1. Information-Complete Media Models

To use the thermodynamic state record of a medium as signal, the implementation of the medium must be information complete. This means all relevant data for its recreation must be obtained from the abstract data structure. Unfortunately this has not been the case: it was forgotten to include a function to obtain the independent mass fractions from the abstract medium interface. This function has been added together with boilerplate implementation for single-phase and multi-phase mediums. With this modification, all media models now can be used with the new library.

5.2.2. Propagation of Media Models

That media models have to be set for each component individually can be an annoying source of error. This is a reminder to tackle a long lasting issue and since there are already good proposals out there and there is a good implementation in Modia [12], we do not revisit this discussion in this paper.

5.2.3. Name Bindings for Bipartite Connector Types

Although an automatic binding of connector variables by their local name is meaningful for physical variables, such as potential and flow variables, this is not merely the case for bipartite connectors such as input and output variables. Here, offering an explicit binding choice is meaningful. A formulation as in the following example would enable us to provide a more elegant solution for undirected flow components (see Listing 5).

Listing 5. Two connectors for undirected flows.

```
connector Thermalplug_bidirectional
  replaceable package Medium;
  SI.Pressure r;
  flow SI.MassFlowRate m_flow;
  input Medium.State state_inflow from state_outflow;
  output Medium.State state_outflow;
end Thermalplug_bidirectional;
```

The new keyword `from` would allow us to indicate for each input the desired output signal identifier to connect to. In comparison to Section 3.3, we only need one connector type and can then consequently simplify the corresponding components. Additionally, fewer topological components would be needed since using just a single connector enables more combinations to connect components to be feasible.

5.3. Future Work

We will keep maintaining this library and improving its components. Apart from having robust models, hard real-time capability, suitable control design as well as various forms of health monitoring are our main research and development interests.

Author Contributions: Software, all authors with special effort by M.M.; writing—original draft preparation, all authors; writing—review and editing, all authors with special effort by N.W.; Validation and Quality Control, M.M. and N.W.; Conceptualization and funding acquisition, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the German Helmholtz Gemeinschaft in frame of a research project KW-BOSCH-1 for on-board diagnosis.

Data Availability Statement: Not applicable.

Acknowledgments: During the development of this library, we profited from the feedback of experienced engineers. Hence we would like to thank the team of Robert Bosch GmbH for feedback and motivating use-case scenarios. Further thanks goes to Chistopher Laughman from Mitsubishi Electronic Research Center for providing encouraging feedback. Additionally, the work of Tobias Krenz form TU Darmstadt offered a valuable perspective. We are still grateful to Airbus and our former colleagues Alexander Pollok and Daniel Bender who were pioneering the new methodology. Additionally, we like to thank our colleague Peter Eschenbacher for his valuable modeling experience on valves and pumps. Thankfully, the Open Modelica team supported us with respect to tool compatibility and Stefan Wischhusen from XRG generously provided additional media models.

Conflicts of Interest: The authors declare no conflict of interest.

Symbols

Symbol	Explanation	Unit
$\boldsymbol{\Theta}$	thermodynamic state vector	
$\dot{m}$	mass flow rate	[kg/s]
$\mathbf{x}$	vector representing internal states	
L	fluid inertance	[1/m]
s	length	[m]
A	area	[m^2]
r	inertial pressure	[Pa]
p	pressure	[Pa]
$\hat{p}$	steady mass-flow pressure	[Pa]
M	mass	[kg]
U	internal energy	[J]
τ	torque	[Nm]
ω	angular velocity	[1/s]
ϵ	effectiveness	[1]
C	heat capacity	[J/K]
NTU	number of (heat) transfer units	[1]
J	rotational inertia	[Nm/s^2]
$\dot{Q}$	heat flux	[J/s]
T	temperature	[K]
k	overall heat transfer coefficient	[J/Kms]
h	specific enthalpy	[J/kg]
V	volume	[m^3]
α	coefficient of heat transfer	[W/(m^{2}K)]
Nu	Nusselt number	[1]
Re	Reynolds number	[1]
Pr	Prandtl number	[1]
χ	vapor quality	[1]
u	valve opening degree	[1]
ζ	zeta value for flow resistance	[Pa m s^2/kg]
ρ	density	[kg/m^3]
$\dot{V}$	volume flow rate	[m^3/s]
K_V	reference volume flow rate	[m^3/h]
κ	valve opening factor	[1]
v	velocity	[m/s]
q	dynamic pressure	[Pa]

References

1. Casella, F. Object-Oriented Modelling & Simulation of Power Plants with Modelica. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 15 December 2005; pp. 7597–7602.
2. Casella, F.; Otter, M.; Proelss, K.; Richter, C.; Tummescheit, H. The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks. In Proceedings of the 5th International Modelica Conference, Vienna, Austria, 4–5 September 2006; pp. 559–568.
3. Franke, R.; Casella, F.; Sielemann, M.; Proelss, K.; Otter, M.; Wetter, M. Standardization of Thermo-Fluid Modeling in Modelica. Fluid. In Proceedings of the 7th Modelica Conference, Como, Italy, 20–22 September 2009; pp. 122–131.

4. El Hefni, B.; Bouskela, D. Dynamic modelling of a Condenser with the Thermo SysPro Library. In Proceedings of the 10th International Modelica Conference, Lund, Sweden, 10–12 March 2014; pp. 1113–1122.
5. Westhäuser, J.; Albrecht, J.; Lemke, J.C.; Tegethoff, N.; Köhler, W. Development of a cyclic vehicle heat pump frosting and defrosting operation strategy. In *SAE Thermal Management Systems Digital Summit*; SAE: Warrendale, PA, USA, 2022.
6. Wetter, M.; Benne, K.; Ravache, B. Software Architecture and Implementation of Modelica Buildings Library Coupling for Spawn of EnergyPlus. In Proceedings of the 14th International Modelica Conference, Linköping, Sweden, 20–24 September 2021; pp. 325–334. [CrossRef]
7. EcosimPro. FLUIDAPRO. Available online: https://www.ecosimpro.com/products/fluidapro/ (accessed on 3 August 2021).
8. Process Systems Enterprise. gPROMS. Available online: www.psenterprise.com/products/gproms (accessed on 3 August 2021).
9. Gamma Technologies LLC. GT-SUITE. Available online: https://www.gtisoft.com/gt-suite/ (accessed on 23 October 2022).
10. Sielemann, M.; Giese, T.; Oehler, B.; Gräber, M. Optimization of an unconventional environmental control system architecture. *SAE Int. J. Aerosp.* **2011**, *4*. [CrossRef]
11. Zimmer, D. Robust object-oriented formulation of directed thermofluid stream networks. *Math. Comput. Model. Dyn. Syst.* **2020**, *26*, 204–233. [CrossRef]
12. Otter, M.; Elmqvist, H.; Zimmer, D.; Laughman, C. Thermodynamic Property and Fluid Modeling with Modern Programming Language Constructs. In Proceedings of the 13th International Modelica Conference, Regensburg, Germany, 4–6 March 2019.
13. Zimmer, D.; Bender, D.; Pollok, A. Robust Modeling of Directed Thermofluid Flows in Complex Networks. In Proceedings of the 2nd Japanese Modelica Conference, Tokyo, Japan, 17–18 May 2018; pp. 39–48.
14. Zimmer, D.; Niels Weber, M.M. Robust Simulation of Stream-Dominated Thermo-Fluid Systems: From Directed to Non-Directed Flows. *Simul. News Eur.* **2021**, *31*, 177–184. [CrossRef]
15. Zimmer, D. Towards hard real-time simulation of complex fluid networks. In Proceedings of the 13th International Modelica Conference, Regensburg, Germany, 4–6 March 2019; pp. 579–587.
16. Meißner, M.; Zimmer, D. Robust Modeling of Volumes for Dynamic Simulations of Thermo-Fluid Stream Networks. *IFACPapersOnLine* **2022**, *55*, 265–270. [CrossRef]
17. Schlünder, E.U.; Gnielinski, V.; Martin, H.; Mewes, D.; Stephan, K.; Steiner, D. *VDI-Wärmeatlas: Berechnungsblätter für den Wärmeübergang*; Springer: Berlin/Heidelberg, Germany, 1997.
18. Incropera, F.; Lavine, A.; Bergman, T.; DeWitt, D. *Fundamentals of Heat and Mass Transfer*; Wiley: New York, NY, USA, 2007.
19. Yan, Y.Y.; Lin, T.F. Evaporation heat transfer and pressure drop of refrigerant R-134a in a plate heat exchanger. *J. Heat Transf.* **1999**, *121*, 118–127. [CrossRef]
20. Yan, Y.Y.; Lin, T.F. Condensation heat transfer and pressure drop of refrigerant R-134a in a small pipe. *Int. J. Heat Mass Transf.* **1999**, *42*, 697–708. [CrossRef]
21. Franke, R.; Casella, F.; Otter, M.; Sielemann, M.; Elmqvist, H.; Mattson, S.E.; Olsson, H. Stream Connectors—An Extension of Modelica for Device-Oriented Modeling of Convective Transport Phenomena. In Proceedings of the 7th Modelica Conference, Como, Italy, 20–22 September 2009; pp. 108–121.