Standardization of Thermo-Fluid Modeling in Modelica.Fluid

Rüdiger Franke, ABB AG, Germany – Ruediger.Franke@de.abb.com,
Francesco Casella, Politecnico di Milano, Italy – Casella@elet.polimi.it,
Michael Sielemann, DLR Institute for Robotics and Mechatronics – Michael.Sielemann@dlr.de,
Katrin Proelss, TU Hamburg-Harburg, Germany – K.Proelss@tu-harburg.de,
Martin Otter, DLR Institute for Robotics and Mechatronics, Germany – Martin.Otter@dlr.de,
Michael Wetter, LBNL, USA – MWetter@lbl.gov

Abstract

This article discusses the Modelica.Fluid library that has been included in the Modelica Standard Library 3.1. Modelica.Fluid provides interfaces and basic components for the device-oriented modeling of one-dimensional thermo-fluid flow in networks containing vessels, pipes, fluid machines, valves and fittings.

A unique feature of Modelica.Fluid is that the component equations and the media models as well as pressure loss and heat transfer correlations are decoupled from each other. All components are implemented such that they can be used for media from the Modelica.Media library. This means that an incompressible or compressible medium, a single or a multiple substance medium with one or more phases might be used with one and the same model as long as the modeling assumptions made hold. Furthermore, trace substances are supported.

Modeling assumptions can be configured globally in an outer System object. This covers in particular the initialization, uni- or bi-directional flow, and dynamic or steady-state formulation of mass, energy, and momentum balance. All assumptions can be locally refined for every component.

While Modelica.Fluid contains a reasonable set of component models, the goal of the library is not to provide a comprehensive set of models, but rather to provide interfaces and best practices for the treatment of issues such as connector design and implementation of energy, mass and momentum balances. Applications from various domains are presented.

Keywords: Modelica, thermo-fluid; one dimensional fluid flow, single substance, multi substance, trace substances

1 Introduction

Modelica.Fluid was announced together with Modelica Media at the Modelica'2003 conference, after the Modelica Association had made an attempt to standardize the most important interfaces and to provide good solutions for the basic problems of fluid modeling [1]. By now Modelica.Media is widely used. Regarding Modelica.Fluid it has not been possible to meet the ambitious goal for device-oriented modeling in realistic fluid applications so far. Still many different fluid libraries exist, each defining its own basics and each having its own downsides. Based on lessons learned, the Modelica Association has made a second attempt to standardize the basic fluid interfaces during the last year. It turned out that the regular Modelica connection approach with effort and flow variables is not sufficient for device-oriented fluid modeling. The newly introduced stream variables [3] represent properties transported by large-scale motion of a flow, such as specific enthalpy transported via convection by a mass flow. This makes it possible that the significant amount work that went into Modelica.Fluid finally yields fruits; 17 persons have contributed to the development during the last 6 years. Compared to previous beta releases, the code was reorganized and extended to cover the whole range from steady-state models to dynamic energy, mass and momentum balances. The fundamental balance equations for one-dimensional fluid flow and heat flow have been decoupled from the device models based on them. This not only simplifies the readability and understanding, but also the maintenance and further development of the library.

2 Library Structure and Interfaces

2.1 Library Structure

Figure 1 shows the package structure of Modelica.Fluid.

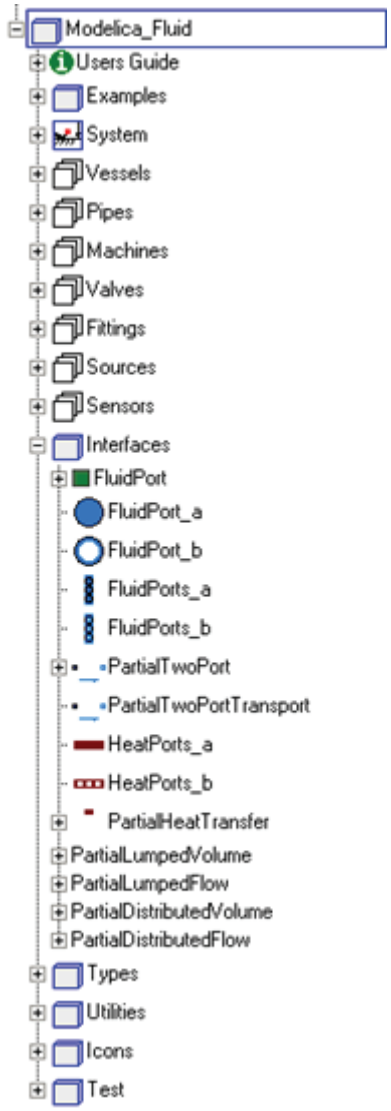

Figure 1: Library structure and interfaces of Modelica.Fluid

The overall library structure is oriented on fluid devices. The main packages are:

- **Vessels** are devices for storing fluid;
- **Pipes** are devices for transporting fluid;
- **Machines** convert between energy held in a fluid and mechanical energy;
- **Valves** regulate fluid flow;
- **Fittings** are adaptors for the connection of fluid components and for the regulation of fluid flow;
- **Sources** define boundary conditions for fluid models;
- **Sensors** are used to measure fluid properties and flow rates.

2.2 Interfaces

The Interfaces package defines both fluid connectors and partial base classes for the implementation of component models. The fluid connectors utilize the new Modelica stream variables, see [3], for the declarative modeling of convective transport of heat and substances. The fluid connector is defined as:

```
connector FluidPort
  replaceable package Medium;

  Medium.Pressure p;
  flow Medium.MassFlowRate m_flow;

  stream Medium.SpecificEnthalpy
    h_outflow;

  stream Medium.MassFraction
    Xi_outflow[Medium.nXi];

  stream Medium.ExtraProperty
    C_outflow[Medium.nC];
end FluidPort;
```

The medium characterizes the fluid passing the port. This ensures that only ports transporting the same fluid can be connected. Moreover, the medium defines numerical ranges, nominal values and equations of state appropriate for the fluid and its application domain at hand.

The purely hydraulic portion of fluid flow is described with the pressure and the mass flow rate in the port. The fluid may transport energy, modeled as specific enthalpy. The mass fractions describe the composition of multi-substance fluids. Extra properties can be used to model trace substances.

The library defines four different versions of the fluid port which all have the same semantics, but different icons. `FluidPort_a` and `FluidPort_b` are intended for fluid models with single flanges such as pipes. `FluidPorts_a` and `FluidPorts_b` shall be used for components with individual flanges for each connection made to them, such as vessels.

2.3 Partial Base Classes

Base classes are used to unify the implementations of similar models and equations. There finds two different kinds of base classes, which are usually combined via multiple inheritance:

- Shell models define interfaces of flow devices, such as fluid ports;
- Balance models define only balance equations, such as the momentum balance or heat transfer with the environment. No connectors are instantiated herein.

The balance models are designed such that they can either be used with inheritance or with replaceable models.

A typical concrete fluid model extends from two base classes: a shell model and a balance model; see e.g. `Fittings.SimpleGenericOrifice` extending from `PartialTwoPortTransport` and from `PartialLumpedFlow`.

Replaceable models are typically used for heat transfer with the environment; see e.g. `Vessels.BaseClasses.PartialLumpedVessel` conditionally enabling a replaceable heat transfer model with the flag `use_HeatTransfer` on the Assumptions tab.

2.3.1 Shell models

`PartialTwoPort` provides two fluid ports `port_a` and `port_b`, which are common to many components like pipes, machines and valves. `PartialTwoPortTransport` extends from `PartialTwoPort` and additionally defines steady-state mass and substance balances for component models without storage of fluid. An extending class still needs to define the energy balance (correspondingly in steady-state), which might, for example, be based on an isenthalpic state transformation for fittings or a polytropic one for machines.

2.3.2 Balance models

The balance models predefine equations for dynamic and for steady-state simulation. Moreover the initialization and the connection to a medium model are treated in these base classes.

The equations are formulated with variables that represent generic boundary and source terms of the corresponding balances. Extending classes need to define the boundary and source terms.

`PartialDistributedVolume` defines the mass and the energy balance for one-dimensional distributed flow models. The model equations are formulated for `n` flow segments, which are characterized with the variable vectors

```
SI.Volume[n] fluidVolumes;
SI.Mass[n] ms;
SI.Energy[n] Us;
```

for the volume, the mass and the internal energy of the fluid per segment. Moreover

```
SI.Mass[n,Medium.nXi] mXis;
SI.Mass[n,Medium.nC] mCs;
```

model the substance masses and the trace substance masses if a multi component medium is used.

The mass balance is defined as

```
der(ms) = mb_flows;
```

with mb_flows[n] a vector of n boundary and source terms.

The energy balance is defined as

```
der(Us) = Hb_flows + Wb_flows + Qb_flows;
```

distinguishing enthalpy flow rates Hb_flows[n], mechanical power Wb_flows[n] and heat flow rates Qb_flows[n] for boundary and source terms.

The substance mass balances and the trace substance mass balances are defined as

```
der(mXis) = mbXi_flows;
der(mCs) = mbC_flows;
```

with mbXi_flows[n,Medium.nXi] and mbC_flows[n,Medium.nC] the boundary and source terms for mass flow rates of independent substances and trace substances, respectively.

The separate base class `PartialDistributedFlow` defines the momentums

```
SI.Momentum[m] Is;
```

for `m` flow segments with the balance equation

```
der(Is) = Ib_flows - Fs_p - Fs_fg;
```

using the boundary and source terms `Ib_flows[m]` for flow of momentum across boundaries, the pressure forces `Fs_p[m]`, and the friction and gravity forces `Fs_fg[m]`.

The use of a different base class for the momentum balance allows the flexible utilization of different discretization schemes than for the mass and the energy balances. For instance, following the staggered grid approach, m=n-1 momentum balances are defined between n fluid volumes.

The additional base classes `PartialLumpedVolume` and `PartialLumpedFlow` provide lumped versions of mass, energy and momentum balances.

An interface for heat transfer with the environment is defined in the `PartialHeatTransfer` model. It predefines a vector of heat flows `Q_flows[n]` through a vector of `heatPorts[n]`. Moreover `PartialHeatTransfer` provides a simple consideration of heat losses to the ambient by using the parameter k as coefficient of heat transfer. A concrete heat transfer model extending from `PartialHeatTransfer` needs to define `Q_flows` based on the thermodynamic states, flow regime and surface areas of `n` flow segments.

2.4 System Wide Properties

Due to common interfaces and base classes, many models contain similar configuration parameters. It would be tedious to open all individual configuration dialogs in order to change the same setting in all models. This is why Modelica.Fluid defines a System object providing global defaults for all components in a fluid model.

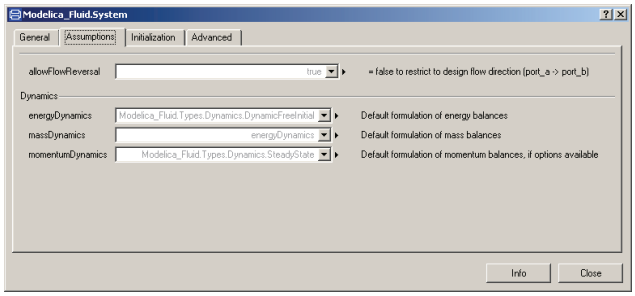

Figure 2: Parameter dialog of the System object

Figure 2 shows the configuration of modeling assumptions. The system wide default values cover:

- General parameters that define ambient pressure, temperature and the force of gravity.
- Assumptions that declare dynamics options, ranging from steady-state to dynamic with fixed or free initial values. These assumptions are made per balance type. A typical fluid model would have dynamic energy and mass balances together with steady-state momentum balances. Moreover the analysis of a fluid model can be restricted to only cover the design flow direction from `port_a` to `port_b`.
- Initialization with common start values.
- Advanced settings to provide default values for mass flow rates and pressure drops that shall be considered small for the numerical analysis of reverting flow conditions.

These system-wide settings can, however, be overridden at the component level to allow, for example, the use of steady-state and dynamic components within the same system model.

3 Rigorous Implementation of One-Dimensional Fluid Flow

The Pipes sub-package provides a rigorous implementation for one-dimensional thermo-fluid flow. The governing equations of pipe flow are the mass balance

$$\underbrace{\frac{\partial(\rho A)}{\partial t}}_{der(ms)} = \underbrace{-\frac{\partial(\rho A v)}{\partial x}}_{mb\_flows},$$

the energy balance

$$\underbrace{\frac{\partial(\rho u A)}{\partial t}}_{der(Us)} = \underbrace{-\frac{\partial\left(\rho v\left(u+\frac{p}{\rho}\right)A\right)}{\partial x}}_{Hb\_flows} + \underbrace{vA\frac{\partial p}{\partial x} + vF_F}_{Wb\_flows} + \underbrace{\frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right) + \dot{Q}_e}_{Qb\_flows}$$

and the momentum balance

$$\underbrace{\frac{\partial(\rho v A)}{\partial t}}_{der(Is)} = \underbrace{-\frac{\partial(\rho v^2 A)}{\partial x}}_{Ib\_flows} \underbrace{- A\frac{\partial p}{\partial x}}_{Fs\_p} \underbrace{- \left(F_F + A\rho g\frac{\partial z}{\partial x}\right)}_{Fs\_fg}.$$

Below the curly braces the names of corresponding variables that are predefined in the base classes are given; see Section 2.3.2.

Here t is the time and x is the independent spatial coordinate along the direction of fluid flow. The fluid is characterized by the density ρ, the specific internal energy u, the temperature T, the velocity v of the flow, and the thermal conductivity k. The flow device is characterized by the area A perpendicular to the direction x, the Fanning friction factor F_F, and the heat flow $\dot{Q}_e$ exchanged with the environment. Moreover g is the constant of gravity and z is the spatial coordinate along the gravity.

Note that the energy balance does not contain contributions of the momentum. The kinetic energy of fluid flow is treated by the momentum balance; see [1].

`Pipes.BaseClasses.PartialTwoPortFlow` implements the balances. It extends from `Interfaces.PartialDistributedVolume` and applies the finite volume approach to the discretization along the spatial coordinate x with `n` flow segments. Figure 3 gives a graphical overview.

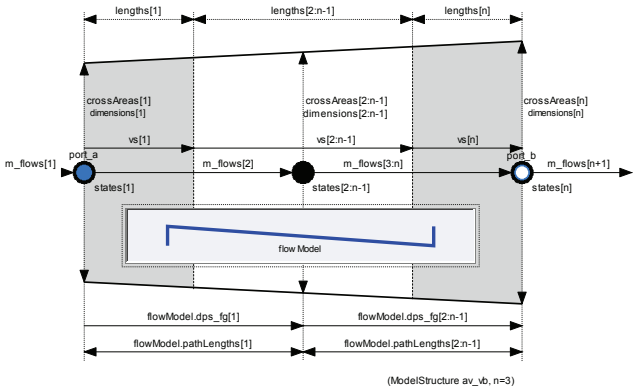

Figure 3: Overview of PartialTwoPortFlow

The replaceable `flowModel` defines m=n-1 momentum balances. Concrete implementations of flow models are provided in the subpackage `Pipes.BaseClasses.FlowModels`. They include laminar, turbulent and detailed one-phase flow. The flow models are either parameterized with pipe geometries or, alternatively, with nominal flow conditions.

The overall model structure follows the staggered grid approach with nNodes=n flow segments. Momentum balance equations are formulated for neighboring flow segments. There are different choices for the formulation of the boundary conditions at the fluid ports, which can be configured with the parameter `ModelStructure` on the Advanced tab:

- `av_vb`: Symmetric setting with nNodes-1 momentum balances between nNodes flow segments. The ports `port_a` and `port_b` expose the first and the last thermodynamic state, respectively. Connecting two or more flow devices therefore may result in high-index DAEs for the pressures of connected flow segments;
- `a_v_b`: Alternative symmetric setting with nNodes+1 momentum balances across nNodes flow segments. Half momentum balances are placed between port_a and the first flow segment as well as between the last flow segment and port_b. Connecting two or more flow devices therefore results in algebraic pressures at the ports. The specification of good start values for the port pressures is essential for the solution of large nonlinear equation systems;
- `av_b`: Unsymmetric setting with nNodes full momentum balances, one between the n-th volume and `port_b`, potential pressure state at `port_a`;
- a_vb: Unsymmetric setting with nNodes full momentum balances, one between the first volume and `port_a`, potential pressure state at `port_b`.

The model structure influences the equations that result from the interconnection of multiple flow models. The connection of models that expose steady-state momentum balances through their ports (model structure `a_v_b`) generally results in a nonlinear equation system for the connection point. The connection of models that expose states of fluid volumes with storage through their ports (model structure `av_vb`) results in high-index DAEs for pressure states. Note that the states representing enthalpy or temperature still stay separated due to the used stream connectors.

A specific pipe flow model still needs to add the source terms `Qb_flow` and `Wb_flow` for heat and work exchanged with the environment. The component model `Pipes.DynamicPipe` extends from `PartialTwoPortFlow` and defines these terms.

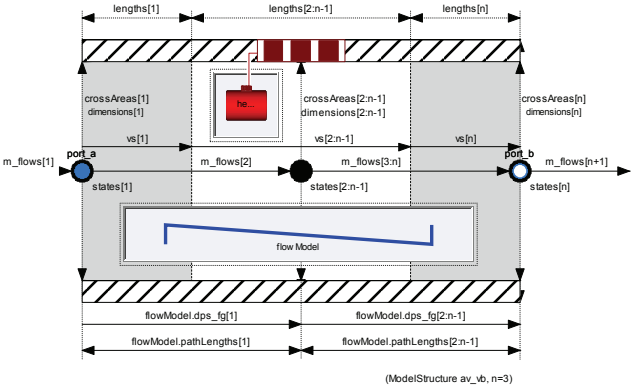

Figure 4: Overview of Pipes.DynamicPipe that extends from Pipes.BaseClasses.PartialTwoPortFlow and defines the terms Wb_flows and Qb_flows

Figure 4 gives an overview of `Pipes.DynamicPipe`. The term `Wb_flow` has been implemented according to the energy balance given above. The term `Qb_flow` is defined by a replaceable wall heat transfer model. Some predefined choices can be found in `Pipes.BaseClasses.HeatTransfer`.

4 Treatment of Wall Friction and Flow Reversal

In a one-dimensional model of fluid dynamics, the wall shear stress cannot be established directly as product of dynamic viscosity and gradient of fluid velocity perpendicular to the fluid surface, as all changes perpendicular to the bulk flow velocity are not resolved in such a model. Consequently, the transfer of momentum between the fluid and an adjacent surface is modeled using a lumped approach. The pressure drop due to wall friction is a computed as product of dynamic pressure and a loss factor ζ ,

$$\Delta p_t = \zeta \frac{\rho v |v|}{2}.$$

If a component models a domain of zero “length” (no extension with respect to the flow direction) then the loss factor is established directly via an appropriate correlation (e.g. a valve). For pipes and other components of finite length, the loss factor is defined

using a wall friction coefficient λ, pipe length l, and diameter d,

$$\zeta = \lambda \frac{l}{d}.$$

All correlations have been regularized to be continuous and have a finite, non-zero, and smooth first derivative. The functions are all guaranteed to be strictly monotonic, which guarantees that a unique inverse exists. Wherever possible, the correlations are provided in two versions, one to calculate mass flow as a function of pressure drop and one to compute pressure drop from mass flow rate. Furthermore, most functions contain a single statement only such that Modelica tools may inline them to avoid function call overhead.

4.1 Pipe Wall Friction

In case of pipe models, the wall friction correlations are provided in form of replaceable packages defined in `Pipes.BaseClasses.WallFriction`. A replaceable instance of the corresponding base class is added to the staggered grid momentum balance in `Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow`. Implementation details are given in [1], regularization is discussed in [2]. Figure 5 provides a Moody chart of an exemplary pipe wall friction correlation.

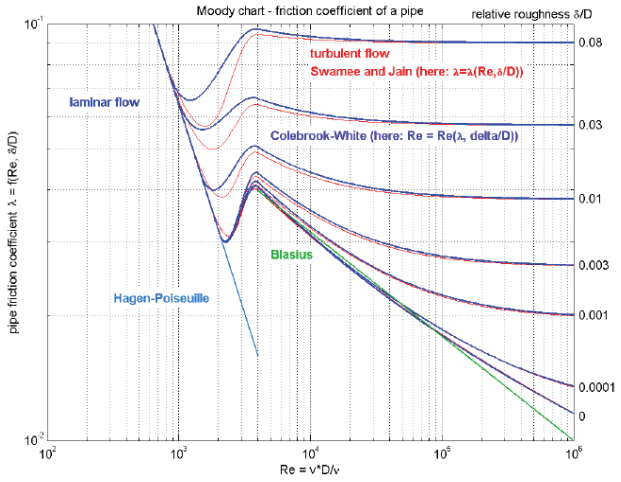

Figure 5: Moody chart of correlations provided in Pipes.BaseClasses.WallFriction

4.2 Pipe Wall Friction and Static Head

When static head is of relevant order of magnitude in a pipe (e.g. natural circulation), properly modeling pipe capacity is important to have a well-defined density and therefore pressure difference due to static head $\Delta p_{sh} = \rho g z$. This is supported by the pipe models using an average of the control volume densities to the left and the right of each momentum balance. In some cases it is reasonable to neglect the pipe capacity however, e.g. when the cost of the associated state variables is high. In this simplification, each pipe segment is filled instantaneously with fluid of the upstream density. The resulting exact solution is not monotonic and thus not bijective if the rate of change of elevation z along the domain has the same sign as the rate of change of density. This is shown in blue in figure 6 below.

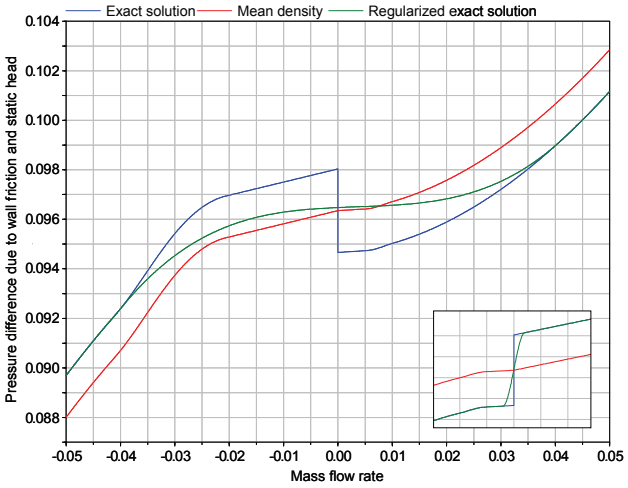

Figure 6: Pressure difference due to wall friction and static head over mass flow rate, non-bijective and bijective case (inset)

This problem cannot be regularized using the Fritsch-Carlson splines employed in [2]. Instead, a modified algorithm was developed based on [5]. It is available under `Utilities.regFun3()` and used to regularize this problem.

Figure 6 illustrates the different options for this particular problem. The exact solution is shown in blue, a simplified approach using a mean density independent of the actual flow direction in red, and the regularized exact version in green. Note in particular the large difference in the necessary width of the regularization interval required to yield a bijective approximation.

4.3 Fittings with Non-Constant Cross Section

According to their definition, loss factors ζ establish a pressure difference in total pressure. For several components, not only the density but also the cross section area is identical on both ends. In this case the dynamic pressure is constant over the component and therefore the difference in total pressure equals the difference in static pressure, i.e. the difference between the port pressures. Additionally to components falling in the latter category, Mode-

lica.Fluid provides component models for fittings with cross section area changing over the device.

Herein, we discuss this type of components at the example of an adaptor model in which the cross section changes abruptly, `Fittings.AbruptAdaptor`. In [6], two different correlations are given, one for a sudden expansion, and a second one for a sudden contraction, which were combined into the given model. Similar to other wall friction correlations for devices with changing cross section area, this type of correlation does not refer to *upstream* density and velocity. Instead, the pressure drop is defined as loss factor times dynamic pressure at the *smallest* cross section area.

Based on the definition of total pressure $p_t = p + 1/2\rho v^2$ and the definition of the loss factor ζ we arrive at the following equation relating the *static* pressure drop and the mass flow rate.

$$\Delta p = \frac{1}{2}\dot{m}^2\left(\frac{\pm\zeta}{\rho_{sca}A_{sca}^2} - \frac{1}{\rho_a A_a^2} + \frac{1}{\rho_b A_b^2}\right)$$

Herein, indices *a* and *b* each refer to one of the ends of the adaptor. Index *sca* refers to the end with smaller cross section area. Furthermore, the positive sign refers to the case with positive mass flow rate.

When substituting the particular correlations mentioned above, the parenthesis turns out to be negative independently of the flow direction. Therefore, the function is not bijective and therefore cannot be inverted.

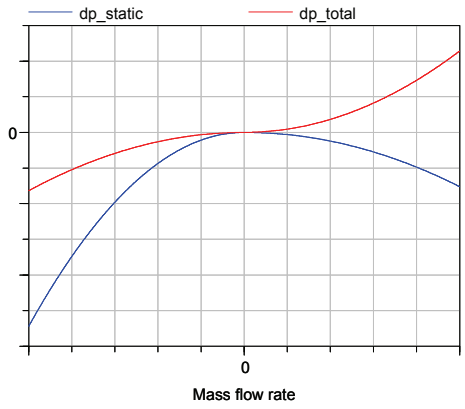

Figure 7: Pressure difference of static pressure and total pressure for flow reversal in an adaptor

Figure 7 provides exemplary results for a particular parametrization of the adaptor (d_a=0.1m, d_b=0.2m). As can be seen in the illustration, the pressure drop in static pressure is not monotone, only the pressure drop in total pressure is.

5 Heat transfer

Due to the underlying one-dimensional approach are temperature gradients perpendicular to the main flow not resolved. Therefore, heat transport between the fluid bulk flow and its environment, as for example a pipe wall, is described by a lumped approach based on a heat transfer coefficient *k*, the heat transfer area A_h and the driving temperature difference between bulk flow and the wall inside.

$$\dot{Q} = k \cdot A_h \cdot (T_{port} - T_{fluid})$$

The computation of the transport coefficient may range from simple constant parameters to complex correlations depending on fluid properties, geometry information and flow position.

The resulting heat flow is used in the energy balance, which looks the same for a wide range of specific component models, while the heat transfer correlation itself needs to be very flexible even at the very top level of a component model which is ready to be used in a larger system. For this reason the heat transfer correlation is implemented as a replaceable model with a defined public interface and can be propagated across several hierarchical levels (see e.g. Fig. 4).

Geometry parameters that are specific to a certain correlation can be entered as a class modification at the top level.

`DistributedPipe` and `LumpedVessel` use each their own constraining type, based on a common model, which are the starting points for example models in the library as well as possible user extensions. Additional heat transfer resistances as walls and insulation materials can be added in each correlation as an option. Thermal capacitance must be covered by an external component connected to the heat ports.

The following correlations are already implemented in the library:

- Zero resistance for lumped and distributed flow. The fluid temperature is then directly exposed to the heat port, which may cause higher index systems if a fixed temperature boundary condition is directly connected.
- Constant heat transfer coefficient for lumped and distributed flow: The heat transfer coefficient is entered as a constant parameter in the user interface
- Nusselt-number (Nu) based, forced convection driven heat transfer for distributed pipe flows: The heat transfer coefficient is deter-

mined from fluid flow properties according to a correlation for laminar and turbulent pipe flow in [7].

The last model inherits from a template class which provides an easy to use interface for other Nu-based correlations, which are e.g. used in heat exchangers.

All models and their corresponding interface classes are found in their respective component category folder, `Vessels.BaseClasses.HeatTransfer` and `Pipes.BaseClasses.HeatTransfer`.

6 Examples

Modelica.Fluid contains a number of simple examples from various application domains. They are intended for the exploration of the library and may serve as a starting point for own developments.

6.1 PumpingSystem

This example models a supply system for drinking water; see Figure 8. It features lumped mass and momentum balances as well as an embedded control.

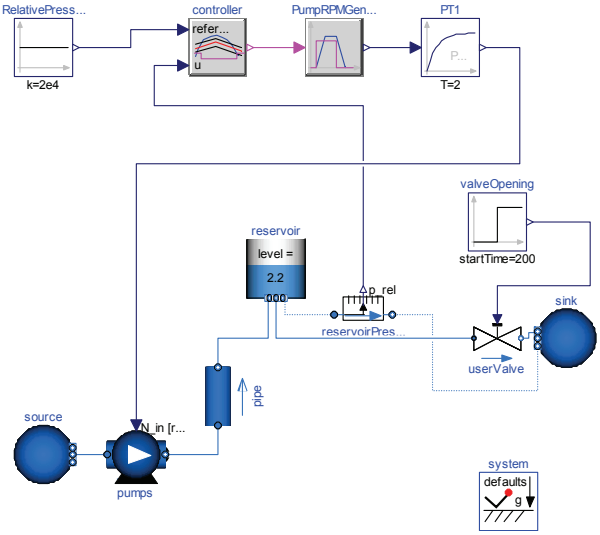

Figure 8: PumpingSystem example

Water is pumped from a source by a pump (fitted with check valves) through a pipe whose outlet is 50m higher than the source, into a reservoir. The users are represented by an equivalent valve, connected to the reservoir.

The water controller is a simple on-off controller, regulating on the gauge pressure measured at the base of the tower; the output of the controller is the rotational speed of the pump, which is represented by the output of a first-order system.

6.2 HeatingSystem

This example models a home heating system; see Figure 9. It features a closed flow cycle and idealized embedded controls in the pump and in the heater.

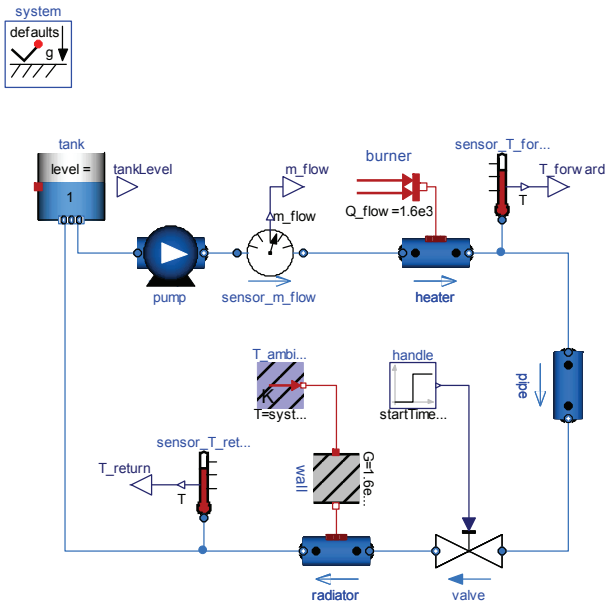

Figure 9: HeatingSystem example

After 2000s of simulation time the handle valve fully opens. A simple idealized control is embedded into the respective components, so that the heating system can be regulated with the valve: the pump controls the pressure, the burner controls the temperature.

The simulation can be turned from dynamic to steady-state or, steady-state initial conditions by selecting the `energy-`, `mass-`, and `momentumDynamics` under the Assumptions tab of the global system object. The selection gets propagated to all component models.

It is important to note that the `massDynamics` of the tank is set to `FixedInitial`, in order to obtain sensible initial conditions for the closed flow cycle.

Also note that the heat transfer model of the tank is enabled, in order to define a thermal insulation against the environment. This way the steady-state initial temperature of the tank is also defined in the case of zero flow.

6.3 DrumBoiler

This example models the drum boiler of a power plant; see Figure 10. It features two phase flow. The locally defined evaporator component explicitly models the phase change from water to steam.

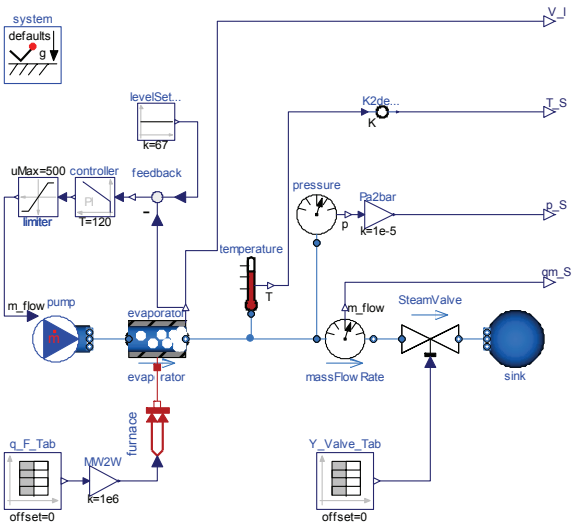

Figure 10: DrumBoiler example

During simulation, the boiler is ramped up from standstill to full load. The prescribed control of fuel flow rate and steam valve position is specified in two lookup tables.

More details about this example, including also the calculation of an optimal start-up control, can be found in [4].

6.4 BranchingDynamicPipes

This example models long pipes; see Figure 11. It features dynamic momentum balances subject to flow reversal.

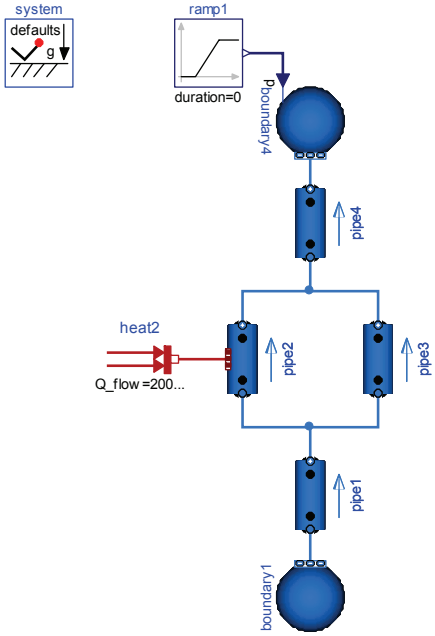

Figure 11: BranchingDynamicPipes example

Applying the default model structure `av_vb` of the `DynamicPipe` models, the idealized junctions are treated as high-index DAEs. The pressure states of connected pipes get lumped together – there is no need to explicitly introduce junction models into the connection points.

At simulation time 2s, the pressure of boundary4 jumps, which causes a pressure wave and flow reversal.

6.5 TraceSubstances

This example models an air conditioning system controlling the CO2 content in a room. It features trace substances.

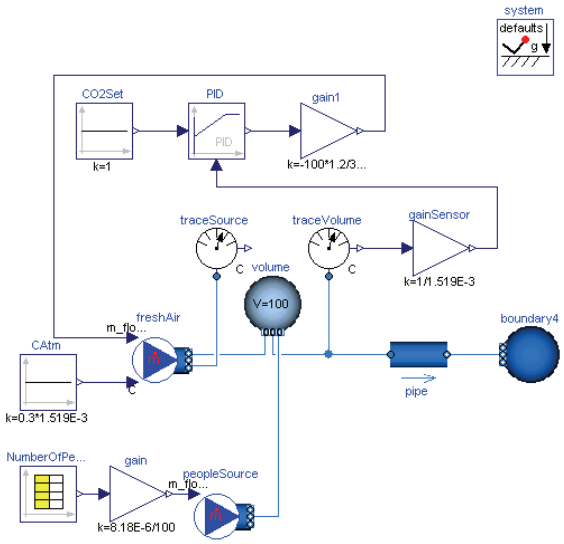

Figure 12: Example model for trace substances.

Figure 12 shows the example model `Fluid.Examples.TraceSubstances.RoomCO2WithControls`. It models a room volume with a CO_2 source and a fresh air supply with feedback control. The CO_2 emission rate is proportional to the room occupancy, which is defined by a schedule. The fresh air mass flow rate is controlled such that the room CO_2 concentration does not exceed 1000 PPM (=1.519E-3 kg/kg). In the model, the implementation of the feedback control normalizes the measured CO_2 concentration so that the controller tracks a set point of one.

The fresh air supply has a CO_2 concentration of 300 PPM, which corresponds to a typical CO_2 concentration in the outside air. The CO_2 emission from the occupants is implemented using a mass flow source. Depending on activity and size, the CO_2 emission rate per person is about 8.18E-6 kg/s. In the model, this value is multiplied by the number of occupants that is read from a time table. Notice that when modeling CO_2 emitted by people, we want to add CO_2 to the room, but no bulk mass flow rate. However, the CO_2 source model at the bottom of Figure 12 outputs

a non-zero air mass flow rate with some prescribed CO_2 concentration. Since the air mass flow rate associate with the CO_2 source model contributes to the volume's energy balance, this mass flow rate needs to be kept small. Thus, in the source model, the CO_2 concentration is set to *C=100 kg/kg*, whereas the output of the gain is scaled such that the mass flow rate is *m_flow = 1/100 * nPeo * 8.18E-6 kg/(s*person)*, where *nPeo* is the number of people in the room. This results in an air mass flow rate that is about 5 orders of magnitudes smaller than the mass flow rate of the fresh air supply, and hence its contribution to the volume's energy balance is negligible.

7 Conclusions

Modelica.Fluid attempts to standardize the most important interfaces and to provide good solutions for the basic problems of fluid modeling. Its design is a collaborative effort; 17 persons have contributed to the development during the last 6 years.

Modelica.Fluid uses stream connectors to model the convective transport of energy and substances. It provides rigorous implementations of mass, energy and momentum balances for one-dimensional thermo-fluid flow. Moreover the library provides a detailed implementation of pipe friction.

Modelica.Fluid contains a reasonable set of simple models of flow device, such as vessels, pipes and pumps. The medium models are treated separately in Modelica.Media. They are freely configurable for each device model, covering compressible and incompressible single- and multi-substance media as well as trace substances.

Several examples from various application domains demonstrate the application of Modelica.Fluid. The library does not attempt to provide a complete set of device models for all of these application domains. Application specific libraries shall base on Modelica.Fluid. This simplifies model exchange and the sharing of knowledge.

Modelica.Fluid supports dynamic and steady-state simulations for one and the same model by specifying global assumptions in a system object.

Modelica.Fluid does not yet cover multi-phase flow. Further restrictions are seen in the connection approach that does not allow the propagation of medium models and geometrical information, such as heights and diameters, through fluid ports.

The future development of Modelica.Fluid will be driven by its applications and by the people who contribute.

References

[1] H. Elmqvist, H. Tummescheit, M. Otter: Object-Oriented Modeling of Thermo-Fluid Systems, Modelica 2003 Conference, Linköping, November 2003. www.modelica.org/events/Conference2003/papers/h40_Elmqvist_fluid.pdf

[2] F. Casella, M. Otter, K. Proelss, C. Richter, H. Tummescheit: The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-Fluid Pipe Networks, Modelica 2006 Conference, Vienna, September 2006. www.modelica.org/events/modelica2006/Proceedings/sessions/Session6b1.pdf

[3] R. Franke, F. Casella, M. Otter, M.Sielemann, H.Elmqvist, S.E. Mattson, H. Olsson: Stream Connectors, Modelica 2009 Conference.

[4] R. Franke, K. Krüger, M. Rode: On-line Optimization of Drum Boiler Startup, Modelica 2003 Conference, Linköping, November 2003. www.modelica.org/events/Conference2003/papers/h29_Franke.pdf

[5] M. G. Gasparo and R. Morandi: Piecewise cubic monotone interpolation with assigned slopes. Computing 46, pages 355-365, 1991.

[6] I.E. Idelchik: Handbook of Hydraulic Resistance. Jaico Publishing House, 2005.

[7] Verein Deutscher Ingenieure (1997): VDI Wärmeatlas. Springer Verlag, Ed. 8, 1997.

Acknowledgements

The design of the Modelica.Fluid library is a collaborative effort and many have contributed. The authors like to particularily thank Jonas Eborn, Hilding Elmqvist, Manuel Gräber, Carsten Heinrich, Kilian Link, Christoph Richter, and Hubertus Tummescheit for their valuable contributions.

Partial financial support by ABB and by DLR for this work within the ITEA project EUROSYSLIB is highly appreciated (BMBF Föderkennzeichen: 01IS07022F). This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.