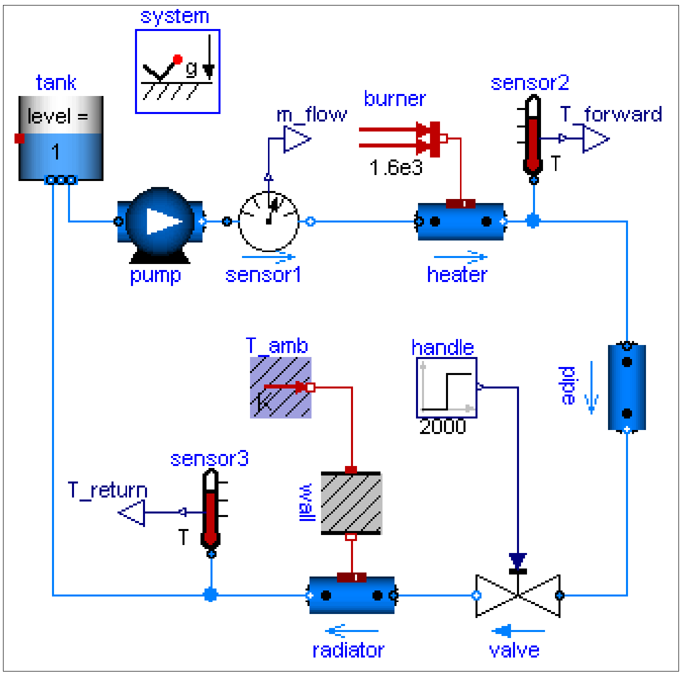
Modelica_Fluid Library

Version 1.0

January 2009

Users Guide and Reference



Modelica Association
http://www.Modelica.org

Contents

Modelica_Fluid...9

Modelica_Fluid.UsersGuide...10
UsersGuide.Overview...11
UsersGuide.GettingStarted..12
UsersGuide.ComponentDefinition..12
UsersGuide.ComponentDefinition.FluidConnectors...12
UsersGuide.ComponentDefinition.BalanceEquations...16
UsersGuide.ComponentDefinition.UpstreamDiscretization...17
UsersGuide.ComponentDefinition.RegularizingCharacteristics...18
UsersGuide.ComponentDefinition.WallFriction...19
UsersGuide.ComponentDefinition.ValveCharacteristics..24
UsersGuide.BuildingSystemModels..25
UsersGuide.BuildingSystemModels.SystemComponent..25
UsersGuide.BuildingSystemModels.MediumDefinition..26
UsersGuide.BuildingSystemModels.CustomizingModel...26
UsersGuide.ReleaseNotes..27
UsersGuide.ModelicaLicense2..33
UsersGuide.Contact...39

Modelica_Fluid.Examples...40
Examples.PumpingSystem..40
Examples.HeatingSystem...41
Examples.DrumBoiler...43
Examples.DrumBoiler.DrumBoiler...43
Examples.DrumBoiler.BaseClasses...44
Examples.DrumBoiler.BaseClasses.EquilibriumDrumBoiler...44
Examples.Tanks...45
Examples.Tanks.ThreeTanks...45
Examples.Tanks.TanksWithOverflow..46
Examples.Tanks.EmptyTanks..46
Examples.ControlledTankSystem..47
Examples.ControlledTankSystem.ControlledTanks...47
Examples.ControlledTankSystem.Utilities..49
Examples.ControlledTankSystem.Utilities.TankController..49
Examples.ControlledTankSystem.Utilities.NormalOperation...50
Examples.ControlledTankSystem.Utilities.RadioButton..50
Examples.AST_BatchPlant...51
Examples.AST_BatchPlant.BatchPlant_StandardWater...53
Examples.AST_BatchPlant.BaseClasses..54
Examples.AST_BatchPlant.BaseClasses.TriggeredTrapezoid...54
Examples.AST_BatchPlant.BaseClasses.setReal..55
Examples.AST_BatchPlant.BaseClasses.TankWith3InletOutletArraysWithEvaporatorCondensor............55
Examples.AST_BatchPlant.BaseClasses.InnerTank...56
Examples.AST_BatchPlant.BaseClasses.Controller...57
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities...57
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Adapter_Inference.....................57
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Adapter_Superposition...............57
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Block_Recipe_TBD....................57
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.BlockMain..................................58
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Buffer_Recipe_TBD....................58
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.BufferMain.................................58
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_Actuators...........................58

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_IdleTanks....59
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_Sensors....59
Examples.AST_BatchPlant.BaseClasses.Init....59
Examples.AST_BatchPlant.BaseClasses.TankWithTopPorts....59
Examples.AST_BatchPlant.Test....61
Examples.AST_BatchPlant.Test.OneTank....61
Examples.AST_BatchPlant.Test.TwoTanks....61
Examples.AST_BatchPlant.Test.TankWithEmptyingPipe1....61
Examples.AST_BatchPlant.Test.TankWithEmptyingPipe2....61
Examples.AST_BatchPlant.Test.TanksWithEmptyingPipe1....62
Examples.AST_BatchPlant.Test.TanksWithEmptyingPipe2....62
Examples.IncompressibleFluidNetwork....62
Examples.BranchingDynamicPipes....63
Examples.HeatExchanger....64
Examples.HeatExchanger.HeatExchangerSimulation....65
Examples.HeatExchanger.BaseClasses....65
Examples.HeatExchanger.BaseClasses.BasicHX....66
Examples.HeatExchanger.BaseClasses.WallConstProps....67
Examples.TraceSubstances....68
Examples.TraceSubstances.RoomCO2....68
Examples.TraceSubstances.RoomCO2WithControls....69
Examples.InverseParameterization....70
Examples.Explanatory....71
Examples.Explanatory.MomentumBalanceFittings....71

Modelica_Fluid.System....72

Modelica_Fluid.Vessels....73
Vessels.ClosedVolume....73
Vessels.OpenTank....74
Vessels.BaseClasses....75
Vessels.BaseClasses.PartialLumpedVessel....76
Vessels.BaseClasses.HeatTransfer....77
Vessels.BaseClasses.HeatTransfer.PartialVesselHeatTransfer....78
Vessels.BaseClasses.HeatTransfer.IdealHeatTransfer....78
Vessels.BaseClasses.HeatTransfer.ConstantHeatTransfer....79
Vessels.BaseClasses.VesselPortsData....79
Vessels.BaseClasses.VesselFluidPorts_a....81
Vessels.BaseClasses.VesselFluidPorts_b....81

Modelica_Fluid.Pipes....82
Pipes.StaticPipe....82
Pipes.DynamicPipe....83
Pipes.BaseClasses....85
Pipes.BaseClasses.PartialStraightPipe....85
Pipes.BaseClasses.PartialTwoPortFlow....86
Pipes.BaseClasses.FlowModels....88
Pipes.BaseClasses.FlowModels.PartialStaggeredFlowModel....88
Pipes.BaseClasses.FlowModels.NominalLaminarFlow....89
Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow....90
Pipes.BaseClasses.FlowModels.NominalTurbulentPipeFlow....91
Pipes.BaseClasses.FlowModels.TurbulentPipeFlow....93
Pipes.BaseClasses.FlowModels.DetailedPipeFlow....94
Pipes.BaseClasses.HeatTransfer....98
Pipes.BaseClasses.HeatTransfer.PartialFlowHeatTransfer....98
Pipes.BaseClasses.HeatTransfer.IdealFlowHeatTransfer....99
Pipes.BaseClasses.HeatTransfer.ConstantFlowHeatTransfer....100
Pipes.BaseClasses.HeatTransfer.PartialPipeFlowHeatTransfer....100
Pipes.BaseClasses.HeatTransfer.LocalPipeFlowHeatTransfer....101

Pipes.BaseClasses.CharacteristicNumbers....102
Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber....102
Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber_m_flow....103
Pipes.BaseClasses.CharacteristicNumbers.NusseltNumber....103
Pipes.BaseClasses.WallFriction....104
Pipes.BaseClasses.WallFriction.PartialWallFriction....105
Pipes.BaseClasses.WallFriction.PartialWallFriction.massFlowRate_dp....105
Pipes.BaseClasses.WallFriction.PartialWallFriction.massFlowRate_dp_staticHead....106
Pipes.BaseClasses.WallFriction.PartialWallFriction.pressureLoss_m_flow....106
Pipes.BaseClasses.WallFriction.PartialWallFriction.pressureLoss_m_flow_staticHead....107
Pipes.BaseClasses.WallFriction.NoFriction....108
Pipes.BaseClasses.WallFriction.NoFriction.massFlowRate_dp....108
Pipes.BaseClasses.WallFriction.NoFriction.pressureLoss_m_flow....109
Pipes.BaseClasses.WallFriction.NoFriction.massFlowRate_dp_staticHead....109
Pipes.BaseClasses.WallFriction.NoFriction.pressureLoss_m_flow_staticHead....110
Pipes.BaseClasses.WallFriction.Laminar....111
Pipes.BaseClasses.WallFriction.Laminar.massFlowRate_dp....111
Pipes.BaseClasses.WallFriction.Laminar.pressureLoss_m_flow....112
Pipes.BaseClasses.WallFriction.Laminar.massFlowRate_dp_staticHead....113
Pipes.BaseClasses.WallFriction.Laminar.pressureLoss_m_flow_staticHead....113
Pipes.BaseClasses.WallFriction.QuadraticTurbulent....114
Pipes.BaseClasses.WallFriction.QuadraticTurbulent.massFlowRate_dp....114
Pipes.BaseClasses.WallFriction.QuadraticTurbulent.pressureLoss_m_flow....115
Pipes.BaseClasses.WallFriction.QuadraticTurbulent.massFlowRate_dp_staticHead....116
Pipes.BaseClasses.WallFriction.QuadraticTurbulent.pressureLoss_m_flow_staticHead....116
Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent....117
Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.massFlowRate_dp....117
Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.pressureLoss_m_flow....118
Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.massFlowRate_dp_staticHead....119
Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.pressureLoss_m_flow_staticHead....119
Pipes.BaseClasses.WallFriction.Detailed....120
Pipes.BaseClasses.WallFriction.Detailed.massFlowRate_dp....124
Pipes.BaseClasses.WallFriction.Detailed.pressureLoss_m_flow....124
Pipes.BaseClasses.WallFriction.Detailed.massFlowRate_dp_staticHead....125
Pipes.BaseClasses.WallFriction.Detailed.pressureLoss_m_flow_staticHead....125
Pipes.BaseClasses.WallFriction.TestWallFrictionAndGravity....126

Modelica_Fluid.Machines....128
Machines.SweptVolume....128
Machines.Pump....130
Machines.ControlledPump....131
Machines.PrescribedPump....132
Machines.BaseClasses....134
Machines.BaseClasses.PartialPump....134
Machines.BaseClasses.PumpCharacteristics....136
Machines.BaseClasses.PumpCharacteristics.baseFlow....136
Machines.BaseClasses.PumpCharacteristics.basePower....136
Machines.BaseClasses.PumpCharacteristics.baseEfficiency....137
Machines.BaseClasses.PumpCharacteristics.linearFlow....137
Machines.BaseClasses.PumpCharacteristics.quadraticFlow....137
Machines.BaseClasses.PumpCharacteristics.polynomialFlow....138
Machines.BaseClasses.PumpCharacteristics.constantEfficiency....138
Machines.BaseClasses.PumpCharacteristics.linearPower....138
Machines.BaseClasses.PumpCharacteristics.quadraticPower....139
Machines.BaseClasses.assertPositiveDifference....139

Modelica_Fluid.Valves....139
Valves.ValveIncompressible....140

Valves.ValveVaporizing....141
Valves.ValveCompressible....142
Valves.ValveLinear....143
Valves.ValveDiscrete....144
Valves.BaseClasses....145
Valves.BaseClasses.PartialValve....145
Valves.BaseClasses.ValveCharacteristics....147
Valves.BaseClasses.ValveCharacteristics.baseFun....147
Valves.BaseClasses.ValveCharacteristics.linear....147
Valves.BaseClasses.ValveCharacteristics.one....147
Valves.BaseClasses.ValveCharacteristics.quadratic....148
Valves.BaseClasses.ValveCharacteristics.equalPercentage....148

Modelica_Fluid.Fittings....149
Fittings.SimpleGenericOrifice....149
Fittings.SharpEdgedOrifice....150
Fittings.AbruptAdaptor....151
Fittings.MultiPort....152
Fittings.TeeJunctionIdeal....152
Fittings.TeeJunctionVolume....153
Fittings.BaseClasses....154
Fittings.BaseClasses.lossConstant_D_zeta....154
Fittings.BaseClasses.QuadraticTurbulent....154
Fittings.BaseClasses.QuadraticTurbulent.LossFactorData....155
Fittings.BaseClasses.QuadraticTurbulent.massFlowRate_dp....157
Fittings.BaseClasses.QuadraticTurbulent.massFlowRate_dp_and_Re....157
Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow....158
Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow_and_Re....158
Fittings.BaseClasses.QuadraticTurbulent.BaseModel....159
Fittings.BaseClasses.QuadraticTurbulent.TestWallFriction....161
Fittings.BaseClasses.QuadraticTurbulent.BaseModelNonconstantCrossSectionArea....162
Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow_totalPressure....163
Fittings.BaseClasses.PartialTeeJunction....164

Modelica_Fluid.Sources....164
Sources.FixedBoundary....165
Sources.Boundary_pT....165
Sources.Boundary_ph....166
Sources.MassFlowSource_T....167
Sources.MassFlowSource_h....168
Sources.BaseClasses....169
Sources.BaseClasses.PartialSource....169

Modelica_Fluid.Sensors....169
Sensors.Pressure....170
Sensors.Density....171
Sensors.DensityTwoPort....171
Sensors.Temperature....172
Sensors.TemperatureTwoPort....172
Sensors.SpecificEnthalpy....173
Sensors.SpecificEnthalpyTwoPort....173
Sensors.SpecificEntropy....174
Sensors.SpecificEntropyTwoPort....174
Sensors.TraceSubstances....175
Sensors.TraceSubstancesTwoPort....175
Sensors.MassFlowRate....176
Sensors.VolumeFlowRate....176
Sensors.RelativePressure....177
Sensors.RelativeTemperature....177

Sensors.BaseClasses....178
Sensors.BaseClasses.PartialAbsoluteSensor....178
Sensors.BaseClasses.PartialFlowSensor....178

Modelica_Fluid.Interfaces....179
Interfaces.FluidPort....179
Interfaces.FluidPort_a....180
Interfaces.FluidPort_b....180
Interfaces.FluidPorts_a....181
Interfaces.FluidPorts_b....181
Interfaces.PartialTwoPort....181
Interfaces.PartialTwoPortTransport....182
Interfaces.HeatPorts_a....183
Interfaces.HeatPorts_b....183
Interfaces.PartialHeatTransfer....183
Interfaces.PartialLumpedVolume....184
Interfaces.PartialLumpedFlow....184
Interfaces.PartialDistributedVolume....185
Interfaces.PartialDistributedFlow....186

Modelica_Fluid.Types....187
Types.HydraulicConductance....187
Types.HydraulicResistance....187
Types.Dynamics....187
Types.CvTypes....188
Types.PortFlowDirection....189
Types.ModelStructure....189

Modelica_Fluid.Utilities....190
Utilities.checkBoundary....190
Utilities.regRoot....190
Utilities.regRoot_der....191
Utilities.regSquare....191
Utilities.regPow....192
Utilities.regRoot2....192
Utilities.regSquare2....194
Utilities.regStep....196
Utilities.evaluatePoly3_derivativeAtZero....196
Utilities.regFun3....196
Utilities.cubicHermite....199
Utilities.cubicHermite_withDerivative....199

Modelica_Fluid.Icons....200
Icons.VariantLibrary....200
Icons.BaseClassLibrary....200
Icons.ObsoleteFunction....200

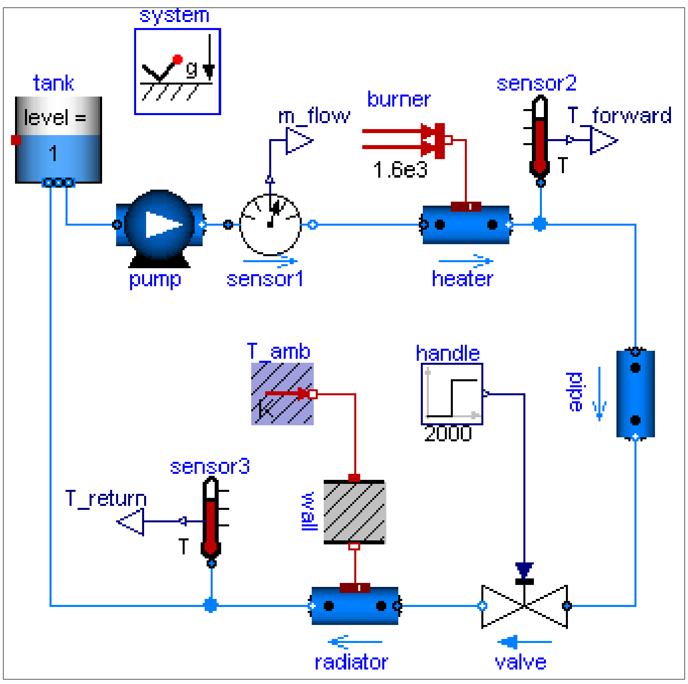
Modelica_Fluid

Modelica_Fluid, 1.0: One-dimensional thermo-fluid flow models using the Modelica.Media media description (requires package Modelica 3.0 or later, and stream connector support in the Modelica tool)

Information

The **Modelica_Fluid** library is a **free** Modelica package provided under the Modelica License 2. The library contains components describing **1-dimensional thermo-fluid flow** in networks of vessels, pipes, fluid machines, valves and fittings. A unique feature 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 especially that an incompressible or compressible medium, a single or a multiple substance medium with one or more phases might be used. The goal is to include the Modelica_Fluid library in the Modelica standard library as Modelica.Fluid.

In the next figure, several features of the library are demonstrated with a simple heating system with a closed flow cycle. By just changing one configuration parameter in the system object the equations are changed between steady-state and dynamic simulation with fixed or steady-state initial conditions.



With respect to previous versions, the design of the connectors has been changed, using the recently developed concept of streams connectors that results in much more reliable simulations (see an overview and a rationale here). This extension will be included in Modelica 3.1. As of Jan. 2009, the streams concept is supported in Dymola 7.1. Dymola 7.2 (announced for Feb. 2009) is recommended since connections to vectors of connectors are much more convenient due to a new annotation. Other tool vendors will support the streams concept as well.

The following parts are useful, when newly starting with this library:

- UsersGuide.
- UsersGuide.ReleaseNotes summarizes the changes of the library releases.
- Examples contains examples that demonstrate the usage of this library.

Politecnico di Milano.

Extends from Modelica.Icons.Library (Icon for library).

Package Content

Name	Description
UsersGuide	Users Guide
Examples	Demonstration of the usage of the library
System	System properties and default values (ambient, flow direction, initialization)
Vessels	Devices for storing fluid
Pipes	Devices for conveying fluid
Machines	Devices for converting between energy held in a fluid and mechanical energy
Valves	Components for the regulation and control of fluid flow
Fittings	Adaptors for connections of fluid components and the regulation of fluid flow
Sources	Define fixed or prescribed boundary conditions
Sensors	Ideal sensor components to extract signals from a fluid connector
Interfaces	Interfaces for steady state and unsteady, mixed-phase, multi-substance, incompressible and compressible flow
Types	Common types for fluid models
Utilities	Utility models to construct fluid components (should not be used directly)
Icons	Library of resuable icons

Modelica_Fluid.UsersGuide

The library **Modelica_Fluid** is a **free** Modelica package provided under the Modelica License 2. The library contains components describing **1-dimensional thermo-fluid flow** in networks of pipes. A unique feature 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 especially that an incompressible or compressible medium, a single or a multiple substance medium with one or more phases might be used. The goal is to include the Modelica_Fluid library in the Modelica standard library as Modelica.Fluid.

Package Content

Name	Description
Overview	Overview
GettingStarted	Getting started
ComponentDefinition	Component definition
BuildingSystemModels	Building system models

ReleaseNotes	Release notes
ModelicaLicense2	Modelica License 2
Contact	Contact

UsersGuide.Overview

The Modelica_Fluid library provides basic interfaces and components to model 1-dimensional thermo-fluid flow in networks of pipes. It is not the intention that this library covers all application cases because the fluid flow area is too large and because for special applications it is possible to implement libraries with simpler component interfaces. Instead, the goal is that the Modelica_Fluid library provides a **reasonable set of components** and that it **demonstrates** how to implement components of a fluid flow library in Modelica, in particular to cope with difficult issues such as connector design, reversing flow and initialization. It is planned to include more components in the future. User proposals are welcome.

This library has the following main features:

- The connectors Modelica_Fluid.Interfaces.FluidPort_a/_b are designed for one-dimensional flow of a **single substance** or of a **mixture of substances** with optional **multiple phases**. All media models from Modelica.Media can be utilized when connecting components. For one substance media, the additional arrays for multiple substance media have zero dimension and are therefore removed from the code during translation. The general connector definition therefore does not introduce an overhead for special cases.

- All the components of the Modelica_Fluid library are designed that they can be utilized for all media models from Modelica.Media if this is posssible. For example, all media can be utilized for the Modelica_Fluid.Sensors/Sources components. For some components only special media are possible, since additional functionality is required. For example, Modelica_Fluid.Components.Evaporator requires a two phase medium (extending from Modelica.Media.Interfaces.PartialTwoPhaseMedium).

- In order to simplify the initialization in the components, there is the restriction that only media models are supported that have T, (p,T), (p,h), (T,X), (p,T,X) or (p,h,X) as independent variables. Other media models would be possible, e.g., with (T,d) as independent variables. However, this requires to rewrite the code for the component initialization. (Note, T is temperature, p is pressure, d is density, h is specific enthalpy, and X is a mass fraction vector).

- All components work for **incompressible** and **compressible** media. This is implemented by a small change in the initialization of a component, if the medium is incompressible. Otherwise, the equations of the components are not influenced by this property.

- All components allow fluid flow in both directions, i.e., **reversing flow** is supported. However, it is possible to declare that the flow through a component only has the design direction, in order to obtain faster simulation code.

- Two or more components can be connected together. This means that the pressures of all connected ports are equal and the mass flow rates sum up to zero. Specific enthalpy, mass fractions and trace substances are mixed according to the mass flow rates.

- The **momentum balance** and the **energy balance** are only fulfilled exactly if **two ports of equal diameter** are connected. In all other cases, the balances are approximated, because kinetic and friction effect are neglected. An explicit fitting or junction should be used if these are important for the specific problem at hand. In all circuits where friction dominates, or components such as pumps determine the flow rate, kinetic pressure is typically irrelevant. You can consider the Modelica_Fluid.Examples.Explanatory.MomentumBalanceFittings model (and its documentation) to see one case where the momentum balance essentially depends on kinetic pressure, so it is necessary to use explicit fittings in order to obtain correct results.

- Given the above-mentioned limitations, there is no restriction how components can be connected together. The resulting simulation performance however often strongly depends on the model structure and modeling assumptions made. In particular the direct connection of fluid volumes generally results in high-index DAEs for the pressures. The direct connection of flow models generally results in systems of implicit nonlinear algebraic equations.

UsersGuide.GettingStarted

Please explore the Examples, which provide simple models for a broad variety of applications.

UsersGuide.ComponentDefinition

In this section it is described how the components of the Modelica_Fluid library are implemented. If you would like to introduce new components either in Modelica_Fluid or your own library, you should be aware of the issues discussed in this section.

This section is partly based on the following paper:

Elmqvist H., Tummescheit H., and Otter M.:
Object-Oriented Modeling of Thermo-Fluid Systems. Modelica 2003 Conference, Linköping, Sweden, pp. 269-286, Nov. 3-4, 2003. Download from: http://www.modelica.org/Conference2003/papers/h40_Elmqvist_fluid.pdf

Please note that the design of the connectors has been changed with respect to the design presented in that paper.

Package Content

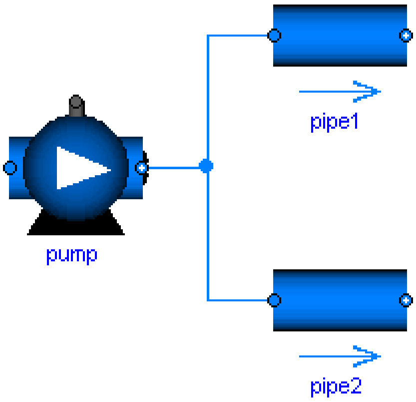
Name	Description
FluidConnectors	Fluid connectors
BalanceEquations	Balance equations
UpstreamDiscretization	Upstream discretization
RegularizingCharacteristics	Regularizing characteristics
WallFriction	Wall friction
ValveCharacteristics	Valve characteristics

UsersGuide.ComponentDefinition.FluidConnectors

In this section the design of the fluid connectors is explained.

Fluid connectors represent the points in a device (e.g. the flanges) through which a fluid can flow into or out of the component, carrying its thermodynamic properties; these flanges are assumed to be fixed in space.

A major design goal is that components can be arbitrarily connected and that the important balance equations are automatically fulfilled when 2 or more components are connected together at one point as shown in the next figure:



In such a case the balance equations define **ideal mixing**, i.e., the upstream discretization scheme of each component uses values that result from ideal mixing in an infinitely small time period. If more realistic modelling is desired that takes into account mixing losses, an explicit model has to be used in the connection point.

Single substance media

For a single substance medium, the connector definition in Modelica_Fluid.Interfaces.FluidPort reduces to

```
  connector FluidPort
     replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
              "Medium model of the fluid";
     flow Medium.MassFlowRate m_flow;
              "Mass flow rate from the connection point into the component";
     Medium.AbsolutePressure p
              "Thermodynamic pressure in the connection point";
     stream Medium.SpecificEnthalpy h_outflow
               "Specific thermodynamic enthalpy close to the connection point if
m_flow < 0";
  end FluidPort;
```

The first statement defines the Medium flowing through the connector. In a medium, medium specific types such as "Medium.AbsolutePressure" are defined that contain medium specific values for the min, max and nominal attributes. Furthermore, Medium.MassFlowRate is defined as:

```
   type MassFlowRate =
      Modelica.SIunits.MassFlowRate(quantity="MassFlowRate." + mediumName);
```

With the current library design, it is necessary to explictly select the medium model for each component in a circuit. This model is then propagated to the ports, and a Modelica translator will check that the quantity and unit attributes of connected interfaces are identical. Therefore, an error occurs, if connected FluidPorts do not have a medium with the same medium name. In the future, automatic propagation of fluid models through the ports will be introduced, but this still not possible with Modelica 3.0.

The thermodynamic pressure is an *effort* variable, which means that the connection of two or more ports states that the port pressures are the same.

The mass flow rate is a *flow* variable, which means that the connection of two or more ports states that the sum of all flow rates is zero.

The last variable is a *stream* variable, i.e., a specific quantity carried by the flow variable. The quantity on the connector always corresponds to the value close to the connection point, assuming that the fluid is flowing out of the connector, regardless of the actual direction of the flow. This helps avoiding singularities when the mass flow goes through zero. The stream properties for the other flow direction can be inquired with the built-in operator inStream(..), while the value of the stream variable corresponding to the actual flow direction can be inquired through the built-in operator actualStream(..).

The actual equations corresponding to these operators are introduced and solved automatically by the tool. In principle, they correspond to the balance equation sum(flow_variable) = 0 and sum(flow_variable*stream_variable_at_connection) = 0 applied to the set of connected ports. In this case the first equation is the mass balance sum(m_flow) = 0, and the second is the energy balance at the connection point sum(m_flow*h_connection) = 0.

In the simpler case of a one-to-one connections between port_a and port_b, inStream(port_a.h_outflow) just returns port_b.h_outflow. For multiple-way connections, mixing equations are generated, and special care is taken in order to avoid discontinuities around zero flow rates. For more details, see this presentation which illustrates the stream concept rationale and the underlying technicalities.

A connector should have only the minimal number of variables to describe the interface, otherwise there will be connection restrictions in certain cases. Therefore, in the connector no redundant variables are present, e.g., the temperature T is not present because it can be computed from the connector variables pressure p and specific enthalpy h.

Here are two simple examples to illustrate modeling with stream connectors. The first one is a rigid adiabatic volume mixing two flows, where the kinetic and gravitational terms in the energy balance are neglected for simplicity.

```
model MixingVolume "Volume that mixes two flows"
  replaceable package Medium = Modelica.Media.Interfaces.PartialPureSubstance;
  FluidPort port_a, port_b;
  parameter Modelica.SIunits.Volume V "Volume of device";
  Modelica.SIunits.Mass             m "Mass in device";
  Modelica.SIunits.Energy           U "Inner energy in device";
  Medium.BaseProperties medium(preferredMediumStates=true) "Medium in the
device";
equation
  // Definition of port variables
  port_a.p         = medium.p;
  port_b.p         = medium.p;
  port_a.h_outflow = medium.h;  // The stream variable always corresponds to the
  port_b.h_outflow = medium.h;  // properties of the fluid holdup (outgoing
flow)

  // Total quantities
  m = V*medium.d;
  U = m*medium.u;
   // Mass and energy balance (actualStream(..) is a built-in operator for
streams to
  // compute the right h, depending on the flow direction)
  der(m) = port_a.m_flow + port_b.m_flow;
  der(U) = port_a.m_flow*actualStream(port_a.h_outflow) +
           port_b.m_flow*actualStream(port_b.h_outflow);
end MixingVolume;
```

The second example is the model of a component describing a lumped pressure loss between two ports, with no energy storage and no heat transfer. An isenthalpic transformation is assumed (changes in kinetic and potential energy between inlet and outlet are neglected)

```
model PressureLoss "Pressure loss component"
  replaceable package Medium=Modelica.Media.Interfaces.PartialPureSubstance;
  FluidPort port_a, port_b:
```

```
  Medium.ThermodynamicState port_a_state_inflow "State at port_a if inflowing";
  Medium.ThermodynamicState port_b_state_inflow "State at port_b if inflowing";
  Medium density d_a, d_b "Density at ports a and b if inflowing";
  replaceable function f "Function to compute the mass flow rate";
equation
  // Medium states for inflowing fluid
  port_a_state_inflow = Medium.setState_phX(port_a.p,
inStream(port_a.h_outflow));
  port_b_state_inflow = Medium.setState_phX(port_b.p,
inStream(port_b.h_outflow));
  // Mass balance
  0 = port_a.m_flow + port_b.m_flow;
  // Instantaneous propagation of enthalpy flow between the ports with
  // isenthalpic state transformation (no storage and no loss of energy)
  port_a.h_outflow = inStream(port_b.h_outflow);
  port_b.h_outflow = inStream(port_a.h_outflow);
  // (Regularized) Momentum balance
  port_a.m_flow = f(port_a.p, port_b.p, d_a, d_b);
end PressureLoss;
```

If many such components are connected in series between two models with storage, the specific enthalpies are propagated in both directions and available to all pressure loss components, without problems when the mass flow goes through zero. The function f then uses either d_a or d_b depending on the sign of port_a.p-port_b.p, with a suitable regularization around zero to avoid discontinuities.

Please note that these models are highly idealized in order to explain the stream connector concept. Device models in the library are much more complete, handling issues such as initialization, steady vs. dynamic modelling, heat transfer from the outside, etc.

Multiple-substance media

Modelica_Fluid can handle models where the fluid contains multiple substances, so that its composition can be characterized by mass fraction vectors.

```
connector FluidPort
   replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
      "Medium model of the fluid";
   flow Medium.MassFlowRate m_flow;
      "Mass flow rate from the connection point into the component"
   Medium.AbsolutePressure p
      "Thermodynamic pressure in the connection point";
   stream Medium.SpecificEnthalpy h_outflow
       "Specific thermodynamic enthalpy close to the connection point if m_flow
< 0";
   stream Medium.MassFraction Xi_outflow[Medium.nXi]
       "Independent mixture mass fractions m_i/m close to the connection point
if m_flow < 0";
   stream Medium.ExtraProperty C_outflow[Medium.nC]
       "Properties c_i/m close to the connection point if m_flow < 0";
  end FluidPort;
```

The mass fraction vectors Xi and C are also stream quantities, as they are carried by the mass flow rate. The corresponding connection equations are sum(m_flow*Xi) and sum(m_flow*C), which correspond to mass balances for the single substances. The vector Xi contains the mass fractions of the main components of the fluid, and is used together with p and h to determine the thermodynamic state of the fluid. The vector C contains the mass fraction of the trace components, which are accounted for in mass balances, but is ignored when computing the fluid properties. This allows to easily declare and use medium models with trace components starting from existing medium models (e.g. adding CO_2 traces to Moist Air for air conditioning models).

Approximations in balance equations at connection point

Summing up, when two or more ports of the type FluidPort are connected, the following equations are generated by the tool:

```
sum(port_j.m_flow) = 0;                // Total Mass balance
port_j = port_k;                       // Momentum balance
sum(port_j.m_flow*h_connection) = 0;   // Energy balance
sum(port_j.m_flow*Xi_connection) = 0;  // Single component mass balances
sum(port_j.m_flow*C_connection) = 0;   // Trace components mass balances
```

It is **very important** to bear in mind that

- the mass balances are always exact;
- the momentum and energy balance are only exact when two port with the same diameter are connected, because there is no friction and no change in fluid velocity.

In all other cases, i.e., different port diameters and/or multple port connections:

- The momentum balance does not consider friction effects and changes of pressure due to changes in velocity.
- There might thus be errors in the momentum balance of the order of magnitude of the dynamic pressure $\rho v^2/2$.
- The energy balance does not consider the kinetic terms (gravity terms cancel out due to the infinitesimal size of the connection volume). There might thus be errors in the momentum balance of the order of magnitude of the kinetic energy v^2/2.

In many applications, where fluid speeds are low and thermal phenomena are mainly of interest, these approximations are commonly made and lead to acceptable results. In all other cases, explicit fitting and junction models should be used, that model explicitly all the kinetic phenomena with the appropriate level of detail.

UsersGuide.ComponentDefinition.BalanceEquations

For one-dimensional flow along the coordinate "x", the following partial differential equations hold

Mass balance	$\frac{\partial(\rho A)}{\partial t} + \frac{\partial(\rho A v)}{\partial x} = 0$
Momentum balance	$\frac{\partial(\rho v A)}{\partial t} + \frac{\partial(\rho v^2 A)}{\partial x} = -A\frac{\partial p}{\partial x} - F_F - A\rho g\frac{\partial z}{\partial x}$
Energy balance 1	$\frac{\partial(\rho(u+\frac{v^2}{2})A)}{\partial t} + \frac{\partial(\rho v(u+\frac{p}{\rho}+\frac{v^2}{2})A)}{\partial x} = -A\rho v g\frac{\partial z}{\partial x} + \frac{\partial}{\partial x}(kA\frac{\partial T}{\partial x}) + \dot{Q}_e$
Pipe friction	$F_F = \frac{1}{2}\rho v\lvert v\rvert fS$
	x: independent spatial coordinate (flow is along coordinate x) t: time v(x,t): mean velocity p(x,t): mean pressure T(x,t): mean temperature

	ρ(x,t): mean density u(x,t): specific internal energy z(x): height over ground A(x): area perpendicular to direction x g: gravity constant f: Fanning friction factor S: circumference

An alternative energy balance can be derived by multiplying the momentum balance with "v" and substracting it from the energy balance 1 above. This results in the "energy balance 2":

Energy balance 2	$\frac{\partial(\rho u A)}{\partial t} + \frac{\partial(\rho v (u + \frac{p}{\rho}) A)}{\partial x} = vA\frac{\partial p}{\partial x} + vF_F + \frac{\partial}{\partial x}(kA\frac{\partial T}{\partial x}) + \dot{Q}_e$

This formulation separates the internal energy of the fluid from the kinetic energy of fluid flow. The internal energy is treated by the energy balance 2, the kinetic energy is treated by the momentum balance equally well. The evaluation of medium properties, which are independent of the kinetic energy, and the formulation of many fluid models is simplified with the energy balance 2. The overall conservation of energy is achieved by considering the mutual dependencies of energy and momentum balance.

Some components in the library, like DynamicPipe, provide a rigorous implementation of mass, momentum and energy balance, using the energy balance 2 equation. Other components, like Valves and Fittings, neglect the impact of changes of the kinetic energy and potential energy on the energy balance, because they are usually irrelevant compared to changes due to heat flows. The StaticPipe component neglects the effect of kinetic energy, but includes the potential energy in the balance, which might be substantial.

All modelling assumptions and simplifications are stated in the component documentation; please note that some of the assumptions might be stated in the base classes the component inherits from.

UsersGuide.ComponentDefinition.UpstreamDiscretization

When implementing a Fluid component, the difficult arises that the value of intensive quantities (such as p, T, ρ) shall be accessed from the **upstream** volume. For example, if the fluid flows from volume A to volume B, then the intensive quantities of volume B have no influence on the fluid between the two volumes. On the other hand, if the flow direction is reversed, the intensive quantities of volume A have no influence on the fluid between the two volumes.

In the Modelica_Fluid library, such a situation is handeled with the following code fragment (from Interfaces.PartialTwoPortTransport):

```
    replaceable package Medium =
                   Modelica.Media.Interfaces.PartialMedium
                   annotation(choicesAllMatching = true);

    Interfaces.FluidPort_a port_a(redeclare package Medium = Medium);
    Interfaces.FluidPort_b port_b(redeclare package Medium = Medium);

    Medium.ThermodynamicState port_a_state_inflow
                    "Medium state close to port_a for inflowing mass flow";
    Medium.ThermodynamicState port_b_state_inflow
                    "Medium state close to port_b for inflowing mass flow";

  equation
    // Isenthalpic state transformation (no storage and no loss of energy)
    port_a.h_outflow  = inStream(port_b.h_outflow);
    port_b.h_outflow  = inStream(port_a.h_outflow);

    port_a.Xi_outflow = inStream(port_b.Xi_outflow);
```

```
    port_b.Xi_outflow = inStream(port_a.Xi_outflow);

    // Mass balance
    port_a.m_flow + port_b.m_flow = 0;

    // Medium states for inflowing medium
    port_a_state_inflow = Medium.setState_phX(port_a.p, port_b.h_outflow,
port_b.Xi_outflow);
    port_b_state_inflow = Medium.setState_phX(port_b.p, port_a.h_outflow,
port_a.Xi_outflow);

    // Densities close to the parts when mass flows in to the respective port
    port_a_rho_inflow = Medium.density(port_a_state_inflow);
    port_b_rho_inflow = Medium.density(port_b_state_inflow);

    // Pressure drop correlation (k_ab, k_ba are the loss factors for the two
flow
    // directions; e.g. for a circular device: k = 8*zeta/(pi*diameter)^2)^2)
    m_flow = Utilities.regRoot2(port_a.p - port_b.p, dp_small,
                                port_a_rho_inflow/k1, port_b_rho_inflow/k2);
```

The medium states for inflowing media can be used to compute density and dynamic viscosity which in turn can be use to formulate the pressure drop equation. The standard pressure drop equation

```
   dp = port_a - port_b;
   m_flow = sqrt(2/(zeta*diameter))*if dp >= 0 then  sqrt(dp)
                                               else -sqrt(-dp)
```

cannot be used, since the function has an infinite derivative at dp=0. Instead the region around zero mass flow rate must be regularized using one of the regularization functions of Modelica_Fluid.Utilities. This requires to have density and/or other medium properties for both flow directions at the same time. These media properties can be computed from the medium states of the inflowing fluid at the two ports.

If the above component is connected between two volumes, i.e., the independent medium variables in port_a and port_b are usually states, then port_a.h and port_b.h are either states (i.e., known quantities in the model) or are computed from states. In either case they are "known". In such a situation, all equations can be directly evaluated without any problems. Zero or reversed mass flow rate does not pose any problems because the medium properties are always computed for both flow directions and are then used in the regularization function.

If 3 or more components are connected together, it can be shown that a system of non-linear algebraic equations appear. The equations are written by purpose in such a form, that a tool can select mass flow rates and pressures as iteration variables of this system. The advantage is that these iteration variables are continuous and even often differentiable. The alternative to use the medium states as iteration variables is not good, because T,h,d are discontinuous for reversing flow direction.

UsersGuide.ComponentDefinition.RegularizingCharacteristics

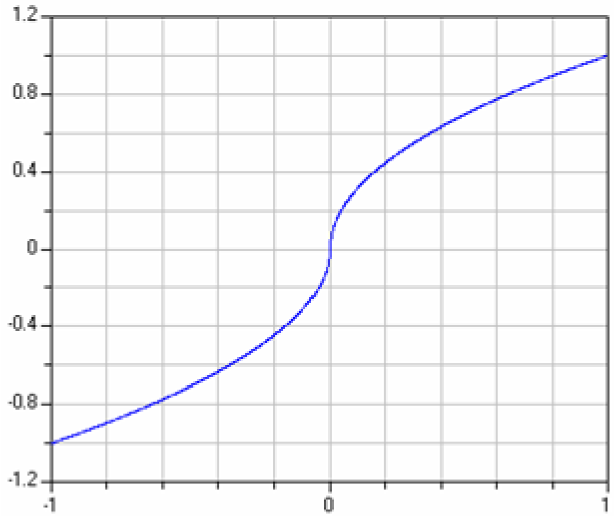
Pressure drop equations and other fluid characteristics are usually computed by **semi-empirical** equations. Unfortunately, the developers of semi-empirical equations nearly never take into account that the equation might be used in a simulation program. As a consequence, these semi-empirical equations can nearly never be used blindly but must be slightly modified or adapted in order that obvious simulation problems are avoided. Below, examples are given to demonstrate what problems occur and how to regularize the characteristics:

Square root function

In several empirical formulae, expressions of the following form are present, e.g., for turbulent flow in a pipe:

```
y = if x < 0 then -sqrt( abs(x) ) else sqrt(x)
```

A plot of this characteristic is shown in the next figure:



The difficulty with this function is that the derivative at x=0 is infinity. In reality, such a function does not exist. E.g., for pipe flow, the flow becomes laminar for small velocities and therefore around zero the sqrt() function is replaced by a linear function. Since the laminar region is usually of not much practical interest, the above approximation is used.

The direct implementation above does not work in Modelica, because an event is generated when x < 0 changes sign. In order to detect this event, an event iteration takes place. During the event iteration, the active if-branch is not changed. For example, assume that x is positive (= "else" branch) and shall become negative. During the event iteration x is slightly negative and the else branch, i.e., sqrt(x), is evaluated. Since this results in an imaginary number, an error occurs. It would be possible to fix this, by using the **noEvent**() operator to explicitly switch of an event:

```
y = if noEvent(x < 0) then -sqrt( abs(x) ) else sqrt(x)
```

Still, it is highly likely that good integrators will not work well around x=0, because they will recognize that the derivative changes very sharply and will reduce the step size drastically.

There are several solutions around this problem: Around x=0, the sqrt() function can be replaced by a polynomial of 3rd order which is determined in such a way that it smoothly touches the sqrt() function, i.e., the whole function is continuous and continuously differentiable. In the Modelica_Fluid library, implementations of such critical functions are provided in sublibrary Modelica_Fluid.Utilities. The above sqrt() type function is computed by function **Utilities.regRoot**(). This function is defined as:

```
y := x/(x*x+delta*delta)^0.25;
```

where "delta" is the size of the small region around zero where the sqrt() function is approximated by another function. The plot of the function above is practically identical to the one of the original function. However, it has a finite derivative at x=0 and is differentiable upto any order. With the default value of delta=0.01, the difference between the function above and regRoot(x) is 16% around x=0.01, 0.25% around x=0.1 and 0.0025% around x=1.

UsersGuide.ComponentDefinition.WallFriction

One important special case for a pressure loss is the friction at the wall of a pipe under the assumption of quasi steady state flow (i.e., the mass flow rate varies only slowly). In this section it is explained how this case is handeled in the Modelica_Fluid library for pipes with **nonuniform roughness**, including the smooth pipe as a special case (see

Pipes.BaseClasses.WallFriction. The treatment is non-standard in order to get a numerically well-posed description.

For pipes with circular cross section the pressure drop is computed as:

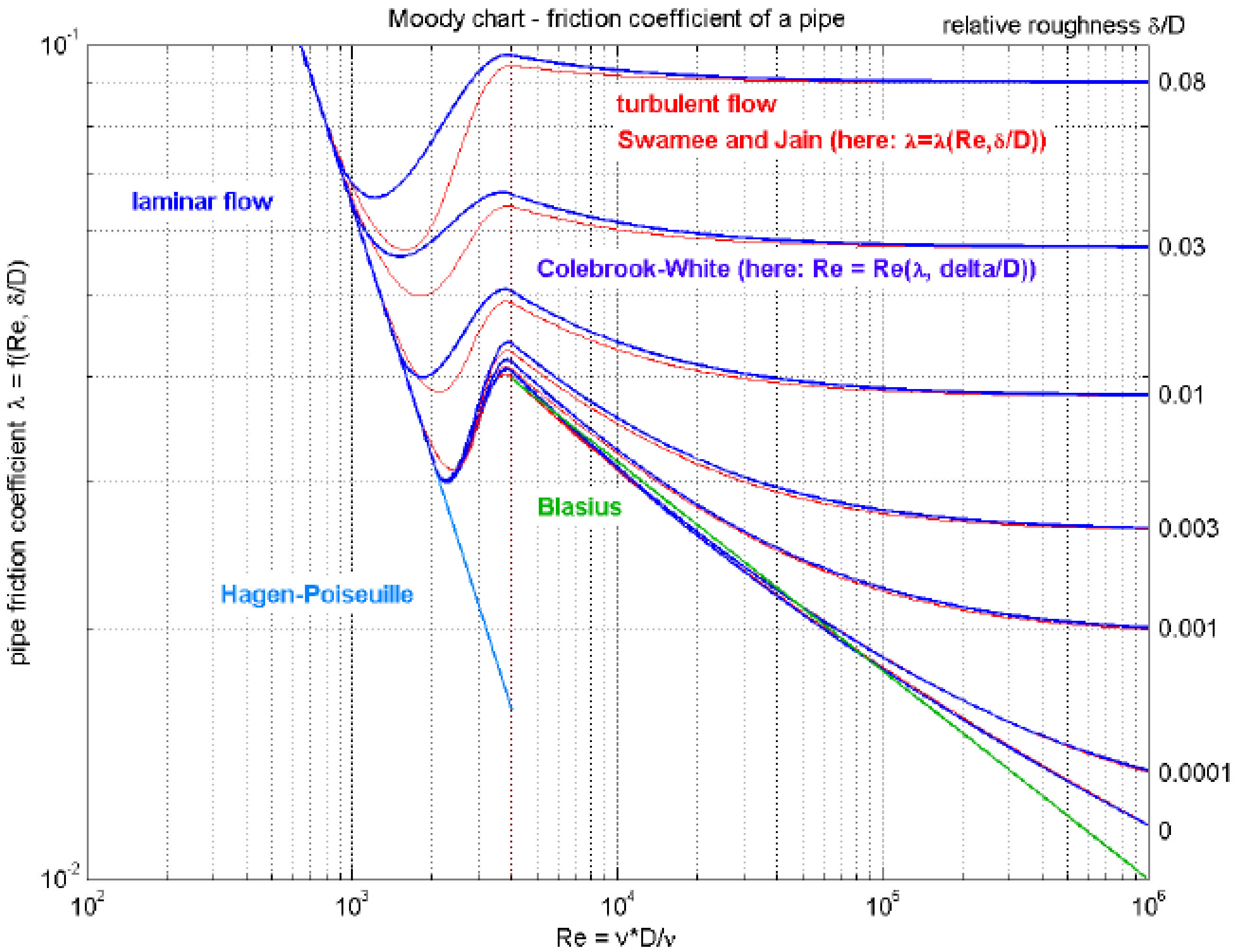
```
dp = λ(Re,Δ)*(L/D)*ρ*v*|v|/2
   = λ(Re,Δ)*8*L/(π^2*D^5*ρ)*m_flow*|m_flow|
   = λ2(Re,Δ)*k2*sign(m_flow);
```

```
with
   Re     = |v|*D*ρ/μ
          = |m_flow|*4/(π*D*μ)
   m_flow = A*v*ρ
   A      = π*(D/2)^2
   λ2     = λ*Re^2
   k2     = L*μ^2/(2*D^3*ρ)
```

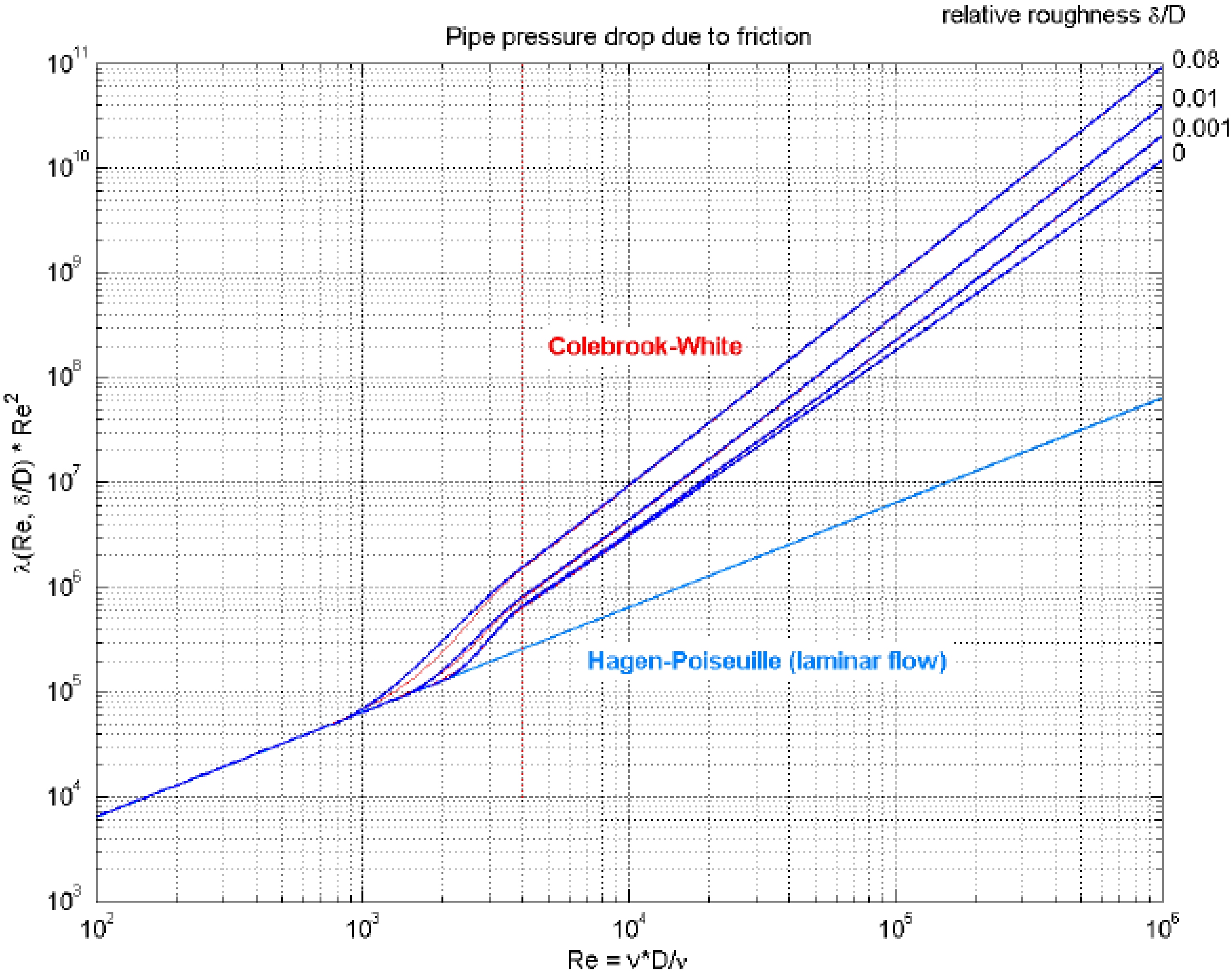
where

- L is the length of the pipe.
- D is the diameter of the pipe. If the pipe has not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.
- λ = λ(Re,Δ) is the "usual" wall friction coefficient.
- λ2 = λ*Re^2 is the used friction coefficient to get a numerically well-posed formulation.
- Re = |v|*D*ρ/μ is the Reynolds number.
- Δ = δ/D is the relative roughness where "δ" is the absolute "roughness", i.e., the averaged height of asperities in the pipe (δ may change over time due to growth of surface asperities during service, see *[Idelchick 1994, p. 85, Tables 2-1, 2-2])*.
- ρ is the upstream density.
- μ is the upstream dynamic viscosity.
- v is the mean velocity.

The first form with λ is used and presented in textbooks, see "blue" curve in the next figure:



This form is not suited for a simulation program since λ = 64/Re if Re < 2000, i.e., a division by zero occurs for zero mass flow rate because Re = 0 in this case. More useful for a simulation model is the friction coefficient λ2 = λ*Re^2, because λ2 = 64*Re if Re < 2000 and therefore no problems for zero mass flow rate occur. The characteristic of λ2 is shown in the next figure and is used in Modelica_Fluid:



The pressure loss characteristic is divided into three regions:

- **Region 1**: For **Re ≤ 2000**, the flow is **laminar** and the exact solution of the 3-dim. Navier-Stokes equations (momentum and mass balance) is used under the assumptions of steady flow, constant pressure gradient and constant density and viscosity (= Hagen-Poiseuille flow) leading to λ2 = 64*Re. Therefore:

  ```
  dp = 128*μ*L/(π*D^4*ρ)*m_flow
  ```

- **Region 3**: For **Re ≥ 4000**, the flow is **turbulent**. Depending on the calculation direction (see "inverse formulation" below) either of two explicit equations are used. If the pressure drop dp is assumed to be known, λ2 = |dp|/k2. The Colebrook-White equation *[Colebrook 1939; Idelchik 1994, p. 83, eq. (2-9)]*:

  ```
  1/sqrt(λ) = -2*lg( 2.51/(Re*sqrt(λ)) + 0.27*Δ)
  ```

  gives an implicit relationship between Re and λ. Inserting λ2 = λ*Re^2 allows to solve this equation analytically for Re:

  ```
  Re = -2*sqrt(λ2)*lg(2.51/sqrt(λ2) + 0.27*Δ)
  ```

  Finally, the mass flow rate m_flow is computed from Re via m_flow = Re*π*D*μ/4*sign(dp). These are the **red** curves in the diagrams above.

If the mass flow rate is assumed known (and therefore implicitly also the Reynolds number), then λ2 is computed by an approximation of the inverse of the Colebrook-White equation *[Swamee and Jain 1976; Miller 1990, p. 191, eq.(8.4)]* adapted to λ2:

```
λ2 = 0.25*(Re/lg(Δ/3.7 + 5.74/Re^0.9))^2
```

The pressure drop is then computed as dp = k2*λ2*sign(m_flow). These are the **blue** curves in the diagrams above.

- **Region 2**: For **2000 ≤ Re ≤ 4000** there is a transition region between laminar and turbulent flow. The value of λ2 depends on more factors as just the Reynolds number and the relative roughness, therefore only crude approximations are possible in this area.
  The deviation from the laminar region depends on the relative roughness. A laminar flow at Re=2000 is only reached for smooth pipes. The deviation Reynolds number Re1 is computed according to *[Samoilenko 1968; Idelchik 1994, p. 81, sect. 2.1.21]* as:

  ```
  Re1 = 745*e^(if Δ ≤ 0.0065 then 1 else 0.0065/Δ)
  ```

  These are the **blue** curves in the diagrams above.
  Between Re1=Re1(δ/D) and Re2=4000, λ2 is approximated by a cubic polynomial in the "lg(λ2) - lg(Re)" chart (see figures above) such that the first derivative is continuous at these two points. In order to avoid the solution of non-linear equations, two different cubic polynomials are used for the direct and the inverse formulation. This leads to some discrepancies in λ2 (= differences between the red and the blue curves). This is acceptable, because the transition region is anyway not precisely known since the actual friction coefficient depends on additional factors and since the operating points are usually not in this region.

The absolute roughness δ has usually to be estimated. In *[Idelchik 1994, pp. 105-109, Table 2-5; Miller 1990, p. 190, Table 8-1]* many examples are given. As a short summary:

Smooth pipes	Drawn brass, coper, aluminium, glass, etc.	δ = 0.0025 mm
Steel pipes	New smooth pipes	δ = 0.025 mm
	Mortar lined, average finish	δ = 0.1 mm
	Heavy rust	δ = 1 mm
Concrete pipes	Steel forms, first class workmanship	δ = 0.025 mm
	Steel forms, average workmanship	δ = 0.1 mm
	Block linings	δ = 1 mm

The equations above are valid for incompressible flow. They can also be applied for **compressible** flow up to about **Ma = 0.6** (Ma is the Mach number) with a maximum error in λ of about 3 %. The effect of gas compressibility in a wide region can be taken into account by the following formula derived by Voronin *[Voronin 1959; Idelchick 1994, p. 97, sect. 2.1.81]*:

```
λ_comp = λ*(1 + (κ-1)/2 * Ma^2)^(-0.47)
```

where κ is the isentropic coefficient (for ideal gases, κ is the ratio of specific heat capacities cp/cv). An appreciable decrease in the coefficent "λ_comp" is observed only in a narrow transonic region and also at supersonic flow velocities by about 15% *[Idelchick 1994, p. 97, sect. 2.1.81]*. This effect is not yet included in Modelica_Fluid. Another restriction is that the pressure drop model is valid only for steady state or slowly changing mass flow rate. For large fluid acceleration, the pressure drop depends additionally on the frequency of the changing mass flow rate.

Inverse formulation

In the "Advanced menu" it is possible via parameter "from_dp" to define in which form the pressure drop equation is actually evaluated (**default** is from_dp = **true**):

```
from_dp = true:   m_flow = f1(dp)
```

```
         = false:  dp     = f2(m_flow)
```

"from_dp" can be useful to avoid nonlinear systems of equations in cases where the inverse pressure loss function is needed.

Summary

A detailed pressure drop model for pipe wall friction is provided in the form m_flow = f1(dp, Δ) or dp = f2(m_flow, Δ). These functions are continuous and differentiable, are provided in an explicit form without solving non-linear equations, and do behave well also at small mass flow rates. This pressure drop model can be used stand-alone in a static momentum balance and in a dynamic momentum balance as the friction pressure drop term. It is valid for incompressible and compressible flow up to a Mach number of 0.6.

References

Colebrook F. (1939):
: **Turbulent flow in pipes with particular reference to the transition region between the smooth and rough pipe laws**. J. Inst. Civ. Eng. no. 4, 14-25.

Idelchik I.E. (1994):
: **Handbook of Hydraulic Resistance**. 3rd edition, Begell House, ISBN 0-8493-9908-4

Miller D. S. (1990):
: **Internal flow systems**. 2nd edition. Cranfield:BHRA(Information Services).

Samoilenko L.A. (1968):
: **Investigation of the Hydraulic Resistance of Pipelines in the Zone of Transition from Laminar into Turbulent Motion**. Thesis (Cand. of Technical Science), Leningrad.

Swamee P.K. and Jain A.K. (1976):
: **Explicit equations for pipe-flow problems**. Proc. ASCE, J.Hydraul. Div., 102 (HY5), pp. 657-664.

Voronin F.S. (1959):
: **Effect of contraction on the friction coefficient in a turbulent gas flow**. Inzh. Fiz. Zh., vol. 2, no. 11, pp. 81-85.

UsersGuide.ComponentDefinition.ValveCharacteristics

The control valves in Valves have the parameters **Kv** and **Cv**. They are defined as unit-less variables, but in the description text a unit is given. The reason for this definition is the following:

The basic equation for valves is:

```
q = Av*sqrt(dp/rho)
```

In SI units, [q] is m3/s, [dp] is Pascal, [rho] is [kg/m3], and Av is an area, thus [Av] = m2. Basically, the equation stems from Bernoulli's law. Av is roughly 1.4 times the area of the valve throat. Now, usually valves aren't so big that their throat area is of the order of magnitude of square meters - depending on the applications it is from a few square millimeters to a few square centimeters. Therefore, in the common engineering practice, the following equations are used:

Europe:

```
q = Kv sqrt(dp/(rho/rho0)) , with [q] = m3/h, [dp] = bar
```

US:

```
q = Cv sqrt(dp/(rho/rho0)) , with [q] = USG/min, [dp] = psi
```

In both cases rho0 is the density of cold water at 4 °C, 999 kg/m3. Note that these equations use relative, not absolute densities.

It turns out that Kv = 1e6/27.7*Av and Cv = 1e6/24*Av, so both US and EU engineers get more or less the same numbers (just by sheer luck), with a range between a few units and a few hundred units for typical industrial applications, and everybody is happy.

Now, we've got two problems here. First, depending on the unit, we change the equation: with SI units, we use the density, with non-SI units, we use the relative density. So the quantities (not only the units!) of Av and Cv/Kv are different.

Second, the units of Kv and Cv are usually labelled "m3/h" and "USG/min", but as a matter of fact they are different, as can be seen from the equations above: they are actually m3/(h*sqrt(bar)) and USG/(min*sqrt(psi)). If I have a valve with Kv = 10 m3/h, it means I get 10 m3/h "for a pressure drop of 1 bar". Unfortunately, this is not correct from the point of view of strict dimensional analysis, but nobody uses sqrt(Pa) or sqrt(bar).

You might think this is crazy (it is, expecially when you try to explain it), but as a matter of fact the valve coefficient is **never** given in square meters in any catalog or datasheet; Cv is still the most used (even in Europe), followed by Kv. So, it will be very inconvenient for users to type in Av in square meters.

The pragmatic approach used in Modelica_Fluid.ControlValves is to accept the fact that m3/h and USG/min are not the real units of Cv and Kv, so we can't use the general unit conversion mechanism, put them just as mnemonic labels in the comment, use non-dimensional coefficients in the interface, and then define properly dimensioned unit conversion within the model

---

UsersGuide.BuildingSystemModels

This section is a quick primer explaining how to build a system model using Modelica_Fluid. It covers some key issues, such as the System component, the definition of medium models in the system, and the typical customizations available in the Modelica_Fluid models.

Package Content

Name	Description
SystemComponent	System component
MediumDefinition	Definition of the medium models
CustomizingModel	Customizing a system model

UsersGuide.BuildingSystemModels.SystemComponent

The Modelica_Fluid library is designed so that each model of a system must include an instance `system` of the `System` component at the top level, in the same way as the `World` model of the MultiBody Library. The System component contains the parameters that describe the environment surrounding the components (ambient pressure and temperature, gravity acceleration), and also provides default settings for many parameters which are used consistently by the models in the library. These parameters are then propagated to the individual components using the inner/outer variable mechanism. In case the system model is structured hieararchically, it is possible to either put a single System component at the top level, or possibly to put many of them at different levels, which will only influence the system components from that level down.

All the parameters defined in the System model are used as default values for the parameters of the individual components of the system model. Note that it is always possible to ovverride these defaults locally by changing the value of the parameters in the specific component instance.

- The *General* tab of the System model allows to set the default enviroment variables (pressure, temperature and gravity) used by all the components.
- The *Assumptions* tab allows to change the default modelling assumptions used by all the components (see the section *Customizing a system model later*)

- The *Initialization* tab allows to define default start values for mass flow rates, pressures and temperatures in the model; this can be useful to help nonlinear solver converge to the solution of any nonlinear system of equations that involves such variables, by providing meaningful guess values.
- The *Advanced* tab contains default values for parameters used in the advanced settings of some components.

Remember to **always add a System component** at the top level of your system model, otherwise you will get errors when compiling the model. The tool will automatically name it `system`, so that it is recognised by all other components.

UsersGuide.BuildingSystemModels.MediumDefinition

All the models in Modelica_Fluid compute fluid properties by using medium models defined by Modelica.Media packages. Custom fluid models can also be used, provided they extend the interfaces defined in Modelica.Media.Interfaces.

All the components in Modelica_Fluid use a *replaceable* medium package, called `Medium`: the model is written for a generic fluid, and a specific fluid model can then be specified when building a system model by redeclaring the package. This can be done in different ways:

- If several components use the same medium, it is possible to select all of them within a GUI, and set them simultaneously (as they are all named Medium).
- It is also possible to declare one or more (possibly replaceable) medium packages in the model, and then use them to set up the individual components.

UsersGuide.BuildingSystemModels.CustomizingModel

Once a system model has been built, it is possible to obtain different approximations by appropriately setting the defaults in the System component (and/or the settings of specific components.

The Assumptions | allowFlowReversal parameter determines whether reversing flow conditions (i.e. flow direction opposite to design direction) are modelled or not. By default, reversing flow conditions are considered by the models, but this causes a significant increase of complexity in the equations, due to the conditional equations depending on the flow direction. If you know in advance that the flow in a certain component (or in the whole system) will always be in the design direction, then setting this parameter to false will produce a much faster and possibly more robust simulation code.

The flags in the Assumptions | Dynamics tab allow different degrees of approximation on the mass, energy, and momentum equations of the components.

- DynamicFreeInitial: dynamic equations are considered (nonzero storage), no initial equations are provided, and the start values are used as guess values.
- FixedInitial: dynamic equations are considered (nonzero storage) and initial equations are included, fixing the states to the start values provided by the component parameters.
- SteadyStateInitial: dynamic equations are considered (nonzero storage), initial equations are included, declaring that the state derivatives are zero (steady-state initialization) and the start values are used as guess values for the nonlinear solver.
- SteadyState: algebraic (or static) balance equations are considered (no storage) and the start values are used as guess values for the nonlinear solver.

It is then possible to neglect the storage of mass, momentum, and energy in the whole system (or just in parts of it) just by a few mouse clicks in a GUI, and also to change the type of initialization when considering dynamic models. Please note that some combinations of the options might be contradictory, and will therefore trigger compilation errors.

UsersGuide.ReleaseNotes

Version 1.0, 2009-01-28

Modelica_Fluid was refactored and finalized for the release:

- Refactoring of the code
  This became necessary as the previous release Modelica_Fluid Streams Beta3 still reflected the long development history, while the basic concepts had been crystalized. Please consult the subversion control (SVN) logs for individual changes.
- Device oriented package names
  The former sub-packages Junctions and PressureLosses have been combined into the new subpackage Fittings. The former Pumps and Volumes.SweptVolume have become the initial version of fluid Machines. The former Volumes package is now called Vessels.
- Complete implementation of one-dimensional fluid flow
  The balance equations as documented in UsersGuide.ComponentDefinition.BalanceEquations are now completely implemented. The implementations with generic boundary flow and source terms find in:
  - Interfaces.PartialDistributedVolume, Interfaces.PartialLumpedVolume: Energy, Mass and Substance balances
  - Interfaces.PartialDistributedFlow, Interfaces.PartialLumpedFlow: Momentum balance

  Specific models combine the balances and define the boundary flow and source terms as appropriate. For instance
  - Vessels.OpenTank extends from Interfaces.PartialLumpedVolume,
  - Fittings.SimpleGenericOrifice extends from Interfaces.PartialLumpedFlow, besides Interfaces.PartialTwoPortTransport,
  - Pipes.DynamicPipe is based on Interfaces.PartialDistributedVolume and Interfaces.PartialDistributedFlow, besides Interfaces.PartialTwoPort.

  All non-trivial mass and energy balances of Vessels, Machines and Fittings have been replaced with PartialLumpedVolume. The mass and energy balances of Pipes are based on PartialDistributedVolume.
  See Examples.BranchingDynamicPipes for an example utilizing the complete balance equations.
- New approach for the connection of distributed flow models
  The staggered grid approach offers different choices for the connection approach. So far the preferred modeling was to put full mass balances into the pipes and expose half momentum balances through the ports (ModelStructure a_v_b). This resulted in nonlinear equation systems for pressure/flow correlations in connection sets. A new default ModelStructure av_vb has been introduced putting full momentum balances into the models and exposing half mass balances through the ports (av_vb replaces the former avb). This way the nonlinear equation systems are avoided. High-index DAEs need to be treated instead in connection sets. Alternatively a Fitting like SuddenExpansion can be introduced to account for different cross flow areas of connected flow models.
- New Vessels.BaseClasses.PartialLumpedVessel treating the ports, including hydraulic resistances, for ClosedVolume, SimpleTank and SweptVolume.
- Clarification of modeling assumptions
  The documentation has been extended to better explain the modeling assumptions made. In particular the section UsersGuide.ComponentDefinition.FluidConnectors now makes clear that the ports represent the thermodynamic enthalpy, as opposed to stagnation enthalpy, and thermodynamic or static pressure, as opposed to total pressure. An new package Explanatory has been added to the examples to show the difference beteen static pressure and total pressure and possible implications. See Examples.Explanatory.MomentumBalanceFittings.
- System (former Ambient)
  The use of the global System object has been extended towards common default values for modeling assumptions, initialization, and advanced settings that are different for each application of the library but should nevertheless provide default values for reasons of convenience. In particular steady-state initialization and complete steady-state simulation can now be specified system-wide. A new Types.Init.Dynamics has been introduced, combining steady-state and initial conditions. The former Types.Init has become obsolete.

See Examples.HeatingSystem.

- Extension of pumps for better consideration of zero flow and heat transfer with environment
The simplified mass and energy balances have been replaced with a rigorous formulation. Moreover an optional heat transfer model can be configured for heat exchanged with the environment or the housing.
See Machines.BaseClasses.PartialPump
- Refinement of valves for flow reversal
All valves now use upstream discretization for reverting flow conditions.
- Finalization of trace substrances
Modelica_Fluid now provides a sound implementation for trace substances, which can easily be added to existing Media models, in order to study their evolution in a fluid system.
See Examples.TraceSubstances.RoomCO2WithControls.
- Vectorized ports for volumes
The ports of models that typically have large volumes, like Vessels and Sources, have been vectorized. Formerly the connection of multiple flow models to the same port of such volume models resulted in unintended mixing equations for stream variables in connection sets outside the volumes. The mixing takes place inside the volumes when using multiple ports. Moreover a Fittings.MultiPort has been introduced. It can be attached to components like pipes, which don't have vectorized ports on their own.
- Inverse parameterization of flow models with nominal operational conditions
Flow models have been added or extended to support the parameterization with nominal values (Machines.ControlledPump, Orifices.SimpleGenericOrifice, Pipes.BaseClasses.FlowModels.NominalTurbulentFlow). They are intended for early phases of system modeling, if geometries and flow characteristics are of secondary interest. As these models use the same interfaces, base classes and naming conventions, they can easily be replaced with more detailed models as more information shall be taken into account later on.
See Examples.InverseParameterization.
- Replaceable HeatTransfer models
The Vessels and the Machines now have replaceable HeatTransfer models, besides the Pipes. All HeatTransfer models are optional. The heat transfer models are parameterized with the Medium and the ThermodynamicState of involved flow segments.
See Interfaces.PartialHeatTransfer.
- All examples are working now (using Dymola 7.1).
The number of examples has been extended with the former critical test cases HeatingSystem and IncompressibleFluidNetwork. Moreover the HeatExchangers have been moved into Examples.

Version 1.0 Streams Beta 3, 2008-10-12

Modelica_Fluid was further improved:

- Volumes, tanks, junctions
Added asserts to require that ports are connected at most once. If a user would perform more than one connection, ideal mixing takes place for the connected components and this is nearly never what the user would like to have
- Ambient
Renamed Ambient to System, including adaptation of models.
Introduced default values system.flowDirection and as a comment system.initType.
system.flowDirection is used in two port components as default.
- GenericJunction
Corrected specification of flowDirection.
Added a HeatPort.
- PartialDistributedFlow models
Adapted determination of velocities to usage of upstream properties at ports.
Corrected and unified initialization of p_start[*] values.
- DistributedPipe models
Changed treatment of port densities and viscosities to the treatment of the lumped pipe model. This way events are avoided if the mass flow rate crosses or approaches zero.
Correct determination of Reynolds numbers.

Added test model DistributedPipeClosingValve.
- ControlValves
Changed flowCharacteristic into valveCharacteristic
Removed parameter Kv and added dp_nom, m_flow_nom from linear and discrete valve interfaces.
Added test cases.
Adapted Examples to new LinearValve and DiscreteValve, using nominal values instead of Kv.
Changed default flow coefficient selection to OpPoint
- Fixed units for Kv and Cv in control valve models.
- Updated tests for valves.
- Bug in Modelica_Fluid.Test.TestComponents.Pumps.TestWaterPump2 corrected (complicated redeclaration issue).
- Adapted AST_BatchPlant so that "Check" is sucessful. Simulation fails after 600 s.
- Introduced density_pTX(Medium.p_default, Medium.T_default, Medium.X_default) as default value for nominal densities (previously it was a literal such as 1000).
- Pumps
Updated energy balance equations for pumps (no division by zero anymore, fixed several bugs related to Np).
Added two more test cases for pumps.
Fixed pump initialization options.
- PartialPump
Explanation for the energy balanced added as comment
"h=0" replaced by "h=Medium.h_default" since otherwise an assert is triggered if "h=0" is not in the medium range.
Fluid ports positioned in the middle line and using the same size as for all other components.
- Pumps.Pump
Resized input connector, so that it has the same size as the standard input connectors.
Changed icon text to input connector to "N_in [rpm]".
Added unit 1/min to the external and internal input connector.
- PartialValve
fillcolor=white added to icon
made line Thickness = Single, since icon does not look nice sometimes
- All components
Changed %name color from black to blue (is a conversion bug, since Modelica 2 has blue as default color whereas Modelica 3 has black and Dymola is not taking care off this).
- Sources
Made icon elements unvisible, if corresponding input is disabled.
- Valves, Pipes, PressureLosses, HeatExchangers, two port senors
Added an arrow in the icon for the "design flow direction" from port_a to port_b.
- Moved default initialization in "System" in to a comment, since no effect yet
- Added the explanation from Francesco for Kv, Cv for valves in the users guide and added links in the corresponding valves to this description

"Check" for the library is successful. "Check with Simulation" (i.e., simulating all test models in the library) is successful with the exceptions:

- Examples.AST_BatchPlant.BatchPlant_StandardWater
Need to be fixed in a later release (requires quite a lot of work).
- Test.TestOverdeterminedSteadyStateInit.Test5
Test.TestOverdeterminedSteadyStateInit.Test6
These are test cases where too much initial conditions are given. The goal is to work on methods how this can be handled. So, this is a principal problem that these models do not simulate.

Version 1.0 Streams Beta 2, 2008-10-08

Modelica_Fluid was transformed to Modelica 3 and to Modelica Standard library 3.0 (by automatic conversion). Further changes:

- Emulated enumerations changed to real enumerations.
- Improved ControlValves code

- Introduced stream connectors with stream keyword (was previously an annotation)
- Introduced inStream() instead of inflow()
- Introduced m_flow*actualStream(h_outflow) instead of streamFlow() or semiLinear(m_flow, inStream(h_outflow), medium.h)
- Removed Modelica_Fluid.Media and all references to it (since now available in Modelica.Media of MSL3.0).
- Fixed PartialLumpedVolume for media with multiple substances
- New function "Utilities.RegFun3" for regularization with static head
- Fix density in static head models with the new RegFun3 functions (ticket 7)
- Minor bug in MixingVolume corrected:
  V_lumped and Wb_flow have been set as modifiers when extending from PartialLumpedVolume, although they are not declared as input. This is not allowed in Modelica 3. Fixed by replacing the modifiers by equations.
- Modelica_Fluid.Sources.FixedBoundary
  Introduced p_default, T_default, h_default as default values, since otherwise warnings will always be printed because parameter value is missing.
- Modelica_Fluid.Sources.Boundary_pT
  Modelica_Fluid.Sources.Boundary_ph
  Modelica_Fluid.Sources.MassFlowSource_T
  Changed default values of parameters reference_p, reference_T to p_default, T_default (some have been xx_default, some reference_xx, it seems best to always use the same approach)
- Modelica_Fluid.Pipes.BaseClasses.PartialDistributedFlow
  Added default value for parameter "rho_nominal" = Medium.density_pTX(Medium.p_default, Medium.T_default, Medium.X_default) in order to avoid unnecessary warning messages. Should be replaced by "Medium.rho_default", once available.
- Modelica_Fluid.Pipes.DistributedPipe
  Modelica_Fluid.Pipes.DistributedPipeSb
  Modelica_Fluid.Pipes.DistributedPipeSa
  Added default value for parameter "mu_nominal" (computed with default values of p,T,X from dynamicViscosity(..))
- Modelica_Fluid.Pipes.BaseClasses.PartialDistributedFlowLumpedPressure
  Replaced default value "rho_nominal=0.01" by Medium.density_pTX(Medium.p_default, Medium.T_default, Medium.X_default)
- Modelica_Fluid.Volumes.OpenTank
  Modelica_Fluid.Volumes.Tank
  Corrected icons of ports (wrongly sized by automatic conversion from Modelica 2 to Modelica 3).
- Examples.BranchingDistributedPipes
  Modelica_Fluid.Test.TestComponents.Junctions.TestGenericJunction
  Modelica_Fluid.Test.TestComponents.Pipes.TestDistributedPipe01
  Parameters dp_nom, m_flow_nom are not defined in junction components. Values provided.
- PressureLosses.BaseClasses.QuadraticTurbulent.BaseModel
  No default or start values for "parameter LossFactorData data" Changed the model to "partial model" to avoid warning messages

Version 1.0 Streams Beta 1, 2008-05-02

Changed connectors to stream connectors and adapted the following sublibraries:

- Volumes
- PressureLosses
- Sensors
- Sources
- ControlValves
- HeatExchangers
- Junctions
- Pipes
- Pumps
- Test and Exampleas (most of the examples and tests are simulating)

Other changes:

- Introduced HeatPorts with vectorized icon in Modelica_Fluid.Interfaces
- Deleted Modelica_Fluid.WorkInProgress since it seems to be too much work to convert it to stream connectors
- Added Modelica_Fluid.Media (contains ConstantLiquidWater medium because functions are missing in Modelica.Media),
- Added two additional test cases with LumpedPipes (to identify problems with hierarchically connected stream connectors).
- Deleted TestPortVolumes since PortVolumes can no longer be implemented with stream connectors
- Leakage flow introduced for valves
- Drumboiler Example corrected
- Regularization for sensors (T,h,...), in order that no discontinuity for bi-directional flow
- Density computation in static head corrected
- New functions Utilities.regUnitStep, regStep
- New components (TestComponents.Sensors.TestOnePortSensors1/.TestOnePortSensors2l, TestRegStep)
- PartialTwoPortTransport
    - Introduced port_a.T, port_b.T (for plotting)
    - Removed initialization menu
    - Introduced dp_start, m_flow_start
    - Removed previous start values of PartialTwoPortTransport in all models
- PartialPump: Removed p_nom, since no longer needed (only dp_nom)
- Made "%name" in the icons of all components unified (and better looking)
- Changed default value of leackage flow of valves to zero.
- Fixed Modelica_Fluid.Junctions.MassFlowRatio so that it compiles (inflow(..) currently only supported for scalars, not for vectors)
- Added script libraryinfo.mos, in order that Modelica_Fluid appears in the Dymola library window automatically (provided library is in MODELICAPATH)
- Replaced semiLinear(..) by streamFlow(..) (not yet at all places)
- Introduced check-boxes in parameter menu of Sources (is more convenient to use)
- TwoPortTransport
  Computation of V_flow and optionally port_a_T, port_b_T. Error in temperature calculation corrected
- Tank:
  Default of bottom pipe diameter changed from 0 to 0.1, since otherwise a division by zero (if not connected and not changed).
- Modelica_Fluid.ControlValves.ValveVaporizing:
  Due to changes in PartialTwoPortTransport, port_a_T_inflow does no longer exist and the usage to it is removed.
- Modelica_Fluid.Test.TestComponents.Sensors.TestTemperatureSensor:
  Due to changes in PartialTwoPortTransport, p_start does no longer exist and the usage to it is removed.
- VersionBuild introduced, as well as automatic update of VersionBuild/VersionDate

Version 1.0 Beta 4, 2008-04-26

Changes according to the Modelica Design Meetings since the last beta version. This version is used to "freeze" the current development, in order to change to a version with a new connector design using stream variables.

Version 1.0 Beta 3, 2007-06-05

Changes according to the Modelica Design Meetings since the Modelica'2006 conference, especially, improved initialization, changed Source components (input connectors must be enabled), improved tank component, moved test models from Examples to new package Test, many more test models, etc. This version is slightly non-backward compatible to version 1.0 Beta 2.

Version 1.0 Beta 2, 2006-08-28

Package considerably restructured and some new components added. New examples (ControlledTankSystem, AST_BatchPlant).

Version 0.96, 2006-01-08

- New package Modelica_Fluid.PressureLosses.
- New package Modelica_Fluid.WorkInProgress.
- New components in Modelica_Fluid.Components:
  ShortPipe, OpenTank, ValveDiscrete, StaticHead.
- New components in Modelica_Fluid.Examples.
- Improved users guide.

Version 0.910, 2005-10-25

- Changes as decided on 41th-45th Modelica Design Meetings (details, see minutes).

Version 0.900, 2004-10-18

- Changes as decided on 40th Modelica Design Meeting in Dresden (see also minutes)

Version 0.794, 2004-05-31

- Sensors.mo, Examples/DrumBoiler.mo: extend sensors with user choice for measurement unit.
- Components.mo, Types.mo: moved components and types to package Examples.
- Moved Examples from **file** Modelica_Fluid/package.mo to Modelica.Media/Examples **subdirectory** and created separate file per sub-package. This shall simplify the maintenance of examples by different authors
- Moved Interfaces from file Modelica_Fluid/package.mo to Modelica_Fluid/Interfaces.mo

Version 0.793, 2004-05-18

- Removed "semiLinear" function since available as Modelica 2.1 built-in operator in Dymola.
- Minor bug in "Components.ShortPipe" corrected.
- Bug in "Components.Orifice" corrected (dp was previously calculated in Interfaces.PartialTwoPortTransport, but this was removed and not updated in Orifice).

Version 0.792, 2003-11-07

This is the first consolidated version made up from several changes for Modelica'2003. Modelica_Fluid is still quite far away from a library that could be included in the Modelica standard library.

Previous Releases

- *Oct., 2003*
  by Martin Otter: Adapted to latest design of the Modelica.Media library.
  by Ruediger Franke: Included sensor components and Modelica_Fluid.Examples.DrumBoiler example.
- *Sept., 2003*
  by Martin Otter: Changes according to the decisions of the Modelica design meeting in Dearborn, Sept. 2-4, 2003. Fluid library splitt in to two packages: Modelica.Media that contains the media models and Modelica_Fluid that contains fluid flow components. Modelica.Media is independent of Modelica_Fluid and my be used also from other packages that may have a different design as Modelica_Fluid.
- *Aug., 2003*
  by Martin Otter: Improved documentation, PortVicinity (now called semiLinear) manually expanded, two different volume types, replaced number of massFractions from n to n-1 in order that usage of

model for single substances is easier and in order that no special cases have to be treated in the equations (previously the massFraction equations had to be removed for single substance flow; now they are removed automatically, since the dimensions are zero, and not one as previously), included asserts to check the validity of the medium models, included the dynamic viscosity in the medium models, adapted the examples and medium models to the changes in Interfaces, improved menus according to the new features in Dymola 5.1. Added "Components.ShortPipe" that contains a detailed model of the frictional losses in pipes over a very wide range.

- *Feb., 2003*
  by Martin Otter: Included several elementary components and a model for moisted air. Some elementary components, such as FixedAmbient, are adapted versions from the SimpleFlow fluid library of Anton Haumer.
- *Dec., 2002*
  by Hubertus Tummescheit: Improved version of the high precision water model (Copy from ThermoFluid library, code reorganization, enhanced documentation, additional functions).
- *Nov. 30, 2002*
  by Martin Otter: Improved the design from the design meeting: Adapted to Modelica standard library 1.5, added "choicesAllMatching=true" annotation, added short documentation to "Interfaces", added packages "Examples" and "Media" (previously called "Properties") from previous versions and adapted them to the updated "Interfaces" package.
- *Nov. 20-21, 2002*
  by Hilding Elmqvist, Mike Tiller, Allan Watson, John Batteh, Chuck Newman, Jonas Eborn: Improved at the 32nd Modelica Design Meeting.
- *Nov. 11, 2002*
  by Hilding Elmqvist, Martin Otter: improved version.
- *Nov. 6, 2002*
  by Hilding Elmqvist: first version of the basic design.

UsersGuide.ModelicaLicense2

This page contains the “Modelica License 2” which was released by the Modelica Association on Nov. 19, 2008. It is used for all material from the Modelica Association provided to the public after this date. It is recommended that other providers of free Modelica packages license their library also under “Modelica License 2”. Additionally, this document contains a description how to apply the license and has a “Frequently Asked Questions” section.

The Modelica License 2 (in other formats: standalone html, pdf, odt, doc)

How to Apply the Modelica License 2

Frequently Asked Questions

The Modelica License 2

Preamble. The goal of this license is that Modelica related model libraries, software, images, documents, data files etc. can be used freely in the original or a modified form, in open source and in commercial environments (as long as the license conditions below are fulfilled, in particular sections 2c) and 2d). The Original Work is provided free of charge and the use is completely at your own risk. Developers of free Modelica packages are encouraged to utilize this license for their work.

The Modelica License applies to any Original Work that contains the following licensing notice adjacent to the copyright notice(s) for this Original Work:

Licensed by <name of Licensor> under the Modelica License 2

1. Definitions.

1. “License” is this Modelica License.
2. “Original Work” is any work of authorship, including software, images, documents, data files, that contains the above licensing notice or that is packed together with a licensing notice referencing it.
3. “Licensor” is the provider of the Original Work who has placed this licensing notice adjacent to the copyright notice(s) for the Original Work. The Original Work is either directly provided by the owner of the Original Work, or by a licensee of the owner.
4. “Derivative Work” is any modification of the Original Work which represents, as a whole, an original work of authorship. For the matter of clarity and as examples:
    1. Derivative Work shall not include work that remains separable from the Original Work, as well as merely extracting a part of the Original Work without modifying it.
    2. Derivative Work shall not include (a) fixing of errors and/or (b) adding vendor specific Modelica annotations and/or (c) using a subset of the classes of a Modelica package, and/or (d) using a different representation, e.g., a binary representation.
    3. Derivative Work shall include classes that are copied from the Original Work where declarations, equations or the documentation are modified.
    4. Derivative Work shall include executables to simulate the models that are generated by a Modelica translator based on the Original Work (of a Modelica package).
5. “Modified Work” is any modification of the Original Work with the following exceptions: (a) fixing of errors and/or (b) adding vendor specific Modelica annotations and/or (c) using a subset of the classes of a Modelica package, and/or (d) using a different representation, e.g., a binary representation.
6. "Source Code" means the preferred form of the Original Work for making modifications to it and all available documentation describing how to modify the Original Work.
7. “You” means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License.
8. “Modelica package” means any Modelica library that is defined with the “`package <Name> ... end <Name>;`“ Modelica language element.

2. Grant of Copyright License. Licensor grants You a worldwide, royalty-free, non-exclusive, sublicensable license, for the duration of the copyright, to do the following:

1. To reproduce the Original Work in copies, either alone or as part of a collection.
2. To create Derivative Works according to Section 1d) of this License.
3. To distribute or communicate to the public copies of the Original Work or a Derivative Work under this License. No fee, neither as a copyright-license fee, nor as a selling fee for the copy as such may be charged under this License. Furthermore, a verbatim copy of this License must be included in any copy of the Original Work or a Derivative Work under this License.
   For the matter of clarity, it is permitted A) to distribute or communicate such copies as part of a (possible commercial) collection where other parts are provided under different licenses and a license fee is charged for the other parts only and B) to charge for mere printing and shipping costs.
4. To distribute or communicate to the public copies of a Derivative Work, alternatively to Section 2c), under any other license of your choice, especially also under a license for commercial/proprietary software, as long as You comply with Sections 3, 4 and 8 below.
   For the matter of clarity, no restrictions regarding fees, either as to a copyright-license fee or as to a selling fee for the copy as such apply.
5. To perform the Original Work publicly.
6. To display the Original Work publicly.

3. Acceptance. Any use of the Original Work or a Derivative Work, or any action according to either Section 2a) to 2f) above constitutes Your acceptance of this License.

4. Designation of Derivative Works and of Modified Works. The identifying designation of Derivative Work and of Modified Work must be different to the corresponding identifying designation of the Original Work. This means especially that the (root-level) name of a Modelica package under this license must be changed if the package is modified (besides fixing of errors, adding vendor specific Modelica annotations, using a subset of the classes of a Modelica package, or using another representation, e.g. a binary representation).

5. Grant of Patent License. Licensor grants You a worldwide, royalty-free, non-exclusive, sublicensable license, under patent claims owned by the Licensor or licensed to the Licensor by the owners of the Original

Work that are embodied in the Original Work as furnished by the Licensor, for the duration of the patents, to make, use, sell, offer for sale, have made, and import the Original Work and Derivative Works under the conditions as given in Section 2. For the matter of clarity, the license regarding Derivative Works covers patent claims to the extent as they are embodied in the Original Work only.

6. Provision of Source Code. Licensor agrees to provide You with a copy of the Source Code of the Original Work but reserves the right to decide freely on the manner of how the Original Work is provided.

For the matter of clarity, Licensor might provide only a binary representation of the Original Work. In that case, You may (a) either reproduce the Source Code from the binary representation if this is possible (e.g., by performing a copy of an encrypted Modelica package, if encryption allows the copy operation) or (b) request the Source Code from the Licensor who will provide it to You.

7. Exclusions from License Grant. Neither the names of Licensor, nor the names of any contributors to the Original Work, nor any of their trademarks or service marks, may be used to endorse or promote products derived from this Original Work without express prior permission of the Licensor. Except as otherwise expressly stated in this License and in particular in Sections 2 and 5, nothing in this License grants any license to Licensor’s trademarks, copyrights, patents, trade secrets or any other intellectual property, and no patent license is granted to make, use, sell, offer for sale, have made, or import embodiments of any patent claims.

No license is granted to the trademarks of Licensor even if such trademarks are included in the Original Work, except as expressly stated in this License. Nothing in this License shall be interpreted to prohibit Licensor from licensing under terms different from this License any Original Work that Licensor otherwise would have a right to license.

8. Attribution Rights. You must retain in the Source Code of the Original Work and of any Derivative Works that You create, all author, copyright, patent, or trademark notices, as well as any descriptive text identified therein as an "Attribution Notice". The same applies to the licensing notice of this License in the Original Work. For the matter of clarity, “author notice” means the notice that identifies the original author(s).

You must cause the Source Code for any Derivative Works that You create to carry a prominent Attribution Notice reasonably calculated to inform recipients that You have modified the Original Work.

In case the Original Work or Derivative Work is not provided in Source Code, the Attribution Notices shall be appropriately displayed, e.g., in the documentation of the Derivative Work.

9. Disclaimer of Warranty.
The Original Work is provided under this License on an "as is" basis and without warranty, either express or implied, including, without limitation, the warranties of non-infringement, merchantability or fitness for a particular purpose. The entire risk as to the quality of the Original Work is with You.
This disclaimer of warranty constitutes an essential part of this License. No license to the Original Work is granted by this License except under this disclaimer.

10. Limitation of Liability. Under no circumstances and under no legal theory, whether in tort (including negligence), contract, or otherwise, shall the Licensor, the owner or a licensee of the Original Work be liable to anyone for any direct, indirect, general, special, incidental, or consequential damages of any character arising as a result of this License or the use of the Original Work including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses. This limitation of liability shall not apply to the extent applicable law prohibits such limitation.

11. Termination. This License conditions your rights to undertake the activities listed in Section 2 and 5, including your right to create Derivative Works based upon the Original Work, and doing so without observing these terms and conditions is prohibited by copyright law and international treaty. Nothing in this License is intended to affect copyright exceptions and limitations. This License shall terminate immediately and You may no longer exercise any of the rights granted to You by this License upon your failure to observe the conditions of this license.

12. Termination for Patent Action. This License shall terminate automatically and You may no longer exercise any of the rights granted to You by this License as of the date You commence an action, including a cross-claim or counterclaim, against Licensor, any owners of the Original Work or any licensee alleging that the Original Work infringes a patent. This termination provision shall not apply for an action alleging patent infringement through combinations of the Original Work under combination with other software or hardware.

13. Jurisdiction. Any action or suit relating to this License may be brought only in the courts of a jurisdiction wherein the Licensor resides and under the laws of that jurisdiction excluding its conflict-of-law provisions.

The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any use of the Original Work outside the scope of this License or after its termination shall be subject to the requirements and penalties of copyright or patent law in the appropriate jurisdiction. This section shall survive the termination of this License.

14. Attorneys' Fees. In any action to enforce the terms of this License or seeking damages relating thereto, the prevailing party shall be entitled to recover its costs and expenses, including, without limitation, reasonable attorneys' fees and costs incurred in connection with such action, including any appeal of such action. This section shall survive the termination of this License.

15. Miscellaneous.

1. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.
2. No verbal ancillary agreements have been made. Changes and additions to this License must appear in writing to be valid. This also applies to changing the clause pertaining to written form.
3. You may use the Original Work in all ways not otherwise restricted or conditioned by this License or by law, and Licensor promises not to interfere with or be responsible for such uses by You.

How to Apply the Modelica License 2

At the top level of your Modelica package and at every important subpackage, add the following notices in the info layer of the package:

> Licensed by <Licensor> under the Modelica License 2
> Copyright © <year1>-<year2>, <name of copyright holder(s)>.
>
> *This Modelica package is free software and the use is completely at your own risk; it can be redistributed and/or modified under the terms of the Modelica license 2, see the license conditions (including the disclaimer of warranty) here or at http://www.modelica.org/licenses/ModelicaLicense2.*

Include a copy of the Modelica License 2 under **<library>.UsersGuide.ModelicaLicense2** (use http://www.modelica.org/licenses/ModelicaLicense2.mo). Furthermore, add the list of authors and contributors under **<library>.UsersGuide.Contributors** or **<library>.UsersGuide.Contact**.

For example, sublibrary Modelica.Blocks of the Modelica Standard Library may have the following notices:

> Licensed by Modelica Association under the Modelica License 2
> Copyright © 1998-2008, Modelica Association.
>
> *This Modelica package is free software and the use is completely at your own risk; it can be redistributed and/or modified under the terms of the Modelica license 2, see the license conditions (including the disclaimer of warranty) here or at http://www.modelica.org/licenses/ModelicaLicense2.*

For C-source code and documents, add similar notices in the corresponding file.

For images, add a "readme.txt" file to the directories where the images are stored and include a similar notice in this file.

In these cases, save a copy of the Modelica License 2 in one directory of the distribution, e.g., http://www.modelica.org/licenses/ModelicaLicense2.html in directory **<library>/help/documentation/ModelicaLicense2.html**.

Frequently Asked Questions

This section contains questions/answer to users and/or distributors of Modelica packages and/or documents under Modelica License 2. Note, the answers to the questions below are not a legal interpretation of the Modelica License 2. In case of a conflict, the language of the license shall prevail.

Using or Distributing a Modelica Package under the Modelica License 2

What are the main differences to the previous version of the Modelica License?

1. Modelica License 1 is unclear whether the licensed Modelica package can be distributed under a different license. Version 2 explicitly allows that “Derivative Work” can be distributed under any license of Your choice, see examples in Section 1d) as to what qualifies as Derivative Work (so, version 2 is clearer).
2. If You modify a Modelica package under Modelica License 2 (besides fixing of errors, adding vendor specific Modelica annotations, using a subset of the classes of a Modelica package, or using another representation, e.g., a binary representation), you must rename the root-level name of the package for your distribution. In version 1 you could keep the name (so, version 2 is more restrictive). The reason of this restriction is to reduce the risk that Modelica packages are available that have identical names, but different functionality.
3. Modelica License 1 states that “It is not allowed to charge a fee for the original version or a modified version of the software, besides a reasonable fee for distribution and support“. Version 2 has a similar intention for all Original Work under Modelica License 2 (to remain free of charge and open source) but states this more clearly as “No fee, neither as a copyright-license fee, nor as a selling fee for the copy as such may be charged”. Contrary to version 1, Modelica License 2 has no restrictions on fees for Derivative Work that is provided under a different license (so, version 2 is clearer and has fewer restrictions).
4. Modelica License 2 introduces several useful provisions for the licensee (articles 5, 6, 12), and for the licensor (articles 7, 12, 13, 14) that have no counter part in version 1.
5. Modelica License 2 can be applied to all type of work, including documents, images and data files, contrary to version 1 that was dedicated for software only (so, version 2 is more general).

Can I distribute a Modelica package (under Modelica License 2) as part of my commercial Modelica modeling and simulation environment?

Yes, according to Section 2c). However, you are not allowed to charge a fee for this part of your environment. Of course, you can charge for your part of the environment.

Can I distribute a Modelica package (under Modelica License 2) under a different license?

No. The license of an unmodified Modelica package cannot be changed according to Sections 2c) and 2d). This means that you cannot sell copies of it, any distribution has to be free of charge.

Can I distribute a Modelica package (under Modelica License 2) under a different license when I first encrypt the package?

No. Merely encrypting a package does not qualify for Derivative Work and therefore the encrypted package has to stay under Modelica License 2.

Can I distribute a Modelica package (under Modelica License 2) under a different license when I first add classes to the package?

No. The package itself remains unmodified, i.e., it is Original Work, and therefore the license for this part must remain under Modelica License 2. The newly added classes can be, however, under a different license.

Can I copy a class out of a Modelica package (under Modelica License 2) and include it unmodified in a Modelica package under a commercial/proprietary license?

No, according to article 2c). However, you can include model, block, function, package, record and connector classes in your Modelica package under Modelica License 2. This means that your Modelica package could be under a commercial/proprietary license, but one or more classes of it are under Modelica License 2.
Note, a “type” class (e.g., type Angle = Real(unit=”rad”)) can be copied and included unmodified under a commercial/proprietary license (for details, see the next question).

Can I copy a type class or part of a model, block, function, record, connector class, out of a Modelica package (under Modelica License 2) and include it modified or unmodified in a Modelica package under a commercial/proprietary license

Yes, according to article 2d), since this will in the end usually qualify as Derivative Work. The reasoning is the following: A type class or part of another class (e.g., an equation, a declaration, part of a class description) cannot be utilized “by its own”. In order to make this “usable”, you have to add additional code in order that the class can be utilized. This is therefore usually Derivative Work and Derivative Work can be provided under a different license. Note, this only holds, if the additional code introduced is sufficient to qualify for Derivative Work. Merely, just copying a class and changing, say, one character in the documentation of this class would be no Derivative Work and therefore the copied code would have to stay under Modelica License 2.

Can I copy a class out of a Modelica package (under Modelica License 2) and include it in modified form in a commercial/proprietary Modelica package?

Yes. If the modification can be seen as a “Derivative Work”, you can place it under your commercial/proprietary license. If the modification does not qualify as “Derivative Work” (e.g., bug fixes, vendor specific annotations), it must remain under Modelica License 2. This means that your Modelica package could be under a commercial/proprietary license, but one or more parts of it are under Modelica License 2.

Can I distribute a “save total model” under my commercial/proprietary license, even if classes under Modelica License 2 are included?

Your classes of the “save total model” can be distributed under your commercial/proprietary license, but the classes under Modelica License 2 must remain under Modelica License 2. This means you can distribute a “save total model”, but some parts might be under Modelica License 2.

Can I distribute a Modelica package (under Modelica License 2) in encrypted form?

Yes. Note, if the encryption does not allow “copying” of classes (in to unencrypted Modelica source code), you have to send the Modelica source code of this package to your customer, if he/she wishes it, according to article 6.

Can I distribute an executable under my commercial/proprietary license, if the model from which the executable is generated uses models from a Modelica package under Modelica License 2?

Yes, according to article 2d), since this is seen as Derivative Work. The reasoning is the following: An executable allows the simulation of a concrete model, whereas models from a Modelica package (without pre-processing, translation, tool run-time library) are not able to be simulated without tool support. By the processing of the tool and by its run-time libraries, significant new functionality is added (a model can be simulated whereas previously it could not be simulated) and functionality available in the package is removed (e.g., to build up a new model by dragging components of the package is no longer possible with the executable).

Is my modification to a Modelica package (under Modelica License 2) a Derivative Work?

It is not possible to give a general answer to it. To be regarded as "an original work of authorship", a derivative work must be different enough from the original or must contain a substantial amount of new material. Making minor changes or additions of little substance to a preexisting work will not qualify the work as a new version for such purposes.

Using or Distributing a Modelica Document under the Modelica License 2

This section is devoted especially for the following applications:

1. A Modelica tool extracts information out of a Modelica package and presents the result in form of a “manual” for this package in, e.g., html, doc, or pdf format.
2. The Modelica language specification is a document defining the Modelica language. It will be licensed under Modelica License 2.
3. Someone writes a book about the Modelica language and/or Modelica packages and uses information which is available in the Modelica language specification and/or the corresponding Modelica package.

Can I sell a manual that was basically derived by extracting information automatically from a Modelica package under Modelica License 2 (e.g., a “reference guide” of the Modelica Standard Library):

Yes. Extracting information from a Modelica package, and providing it in a human readable, suitable format, like html, doc or pdf format, where the content is significantly modified (e.g. tables with interface information are constructed from the declarations of the public variables) qualifies as Derivative Work and there are no restrictions to charge a fee for Derivative Work under alternative 2d).

Can I copy a text passage out of a Modelica document (under Modelica License 2) and use it unmodified in my document (e.g. the Modelica syntax description in the Modelica Specification)?

Yes. In case you distribute your document, the copied parts are still under Modelica License 2 and you are not allowed to charge a license fee for this part. You can, of course, charge a fee for the rest of your document.

Can I copy a text passage out of a Modelica document (under Modelica License 2) and use it in modified form in my document?

Yes, the creation of Derivative Works is allowed. In case the content is significantly modified this qualifies as Derivative Work and there are no restrictions to charge a fee for Derivative Work under alternative 2d).

Can I sell a printed version of a Modelica document (under Modelica License 2), e.g., the Modelica Language Specification?

No, if you are not the copyright-holder, since article 2c) does not allow a selling fee for a (in this case physical) copy. However, mere printing and shipping costs may be recovered.

UsersGuide.Contact

The Modelica_Fluid library (this Modelica package) is developed by many people from different organizations (see list below). It is licensed under the Modelica License 2 by:

Modelica Association
(Ideella Föreningar 822003-8858 in Linköping)
c/o PELAB, IDA, Linköpings Universitet
S-58183 Linköping
Sweden
email: Board@Modelica.org
web: http://www.Modelica.org

The development of the Modelica_Fluid package is organized by

Francesco Casella
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5
I-20133 Milano, Italy
email: casella@elet.polimi.it

and

Rüdiger Franke
ABB AG
PTSP-E22
Kallstadter Str. 1
D-68163, Germany
email: ruediger.franke@de.abb.com

Acknowledgements:

The development of this library has been a collaborative effort and many have contributed.

- The previous design of this library (until beginning of 2008) was based on the paper Elmqvist H., Tummescheit H., and Otter M.: Object-Oriented Modeling of Thermo-Fluid Systems. Modelica 2003 Conference, Linköping, Sweden, pp. 269-286, Nov. 3-4, 2003.
  This design has been partly changed, especially by the introduction of the streams concept.
- The Fluid library development was organized in 2002-2004 by Martin Otter, since 2004 it is organized by Francesco Casella, and since 2008 it is organized jointly by Francesco Casella and Rüdiger Franke.
- Francesco Casella included several components of his ThermoPower library with some rewriting. The stream connector concept used in Modelica_Fluid is based on a similar concept developed by him for the ThermoPower library.
- Rüdiger Franke initiated the stream connector concept as an extension and improved version of the ThermoPower concept. In Nov. 2008 - Jan. 2009 he greatly restructured and improved the library.

- Michael Wetter introduced trace constituents in Modelica_Fluid consistently and provided corresponding examples under Examples.TraceSubstances.
- The following people contributed to the fluid component models, examples, and the further design of the library (alphabetical list):
  John Batteh, Francesco Casella, Jonas Eborn, Hilding Elmqvist, Rüdiger Franke, Manuel Gräber, Henning Knigge, Sven Erik Mattsson, Chuck Newman, Hans Olsson, Martin Otter, Katrin Prölß, Christoph Richter, Michael Sielemann, Mike Tiller, Hubertus Tummescheit, Allan Watson, Michael Wetter.

Partial financial support of ABB and DLR by BMBF (BMBF Förderkennzeichen: 01IS07022F) for the further development of this library within the ITEA project EUROSYSLIB is highly appreciated.

Modelica_Fluid.Examples

Demonstration of the usage of the library

Information

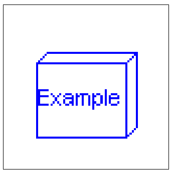
Extends from Modelica.Icons.Library (Icon for library).

Package Content

Name	Description
PumpingSystem	Model of a pumping system for drinking water
HeatingSystem	Simple model of a heating system
DrumBoiler	Drum boiler example, see Franke, Rode, Krueger: On-line Optimization of Drum Boiler Startup, 3rd International Modelica Conference, Linkoping, 2003
Tanks	Library demonstrating the usage of the tank model
ControlledTankSystem	Tank system with controller, start/stop/shut operation and diagram animation
AST_BatchPlant	Model of the experimental batch plant at Process Control Laboratory at University of Dortmund (Prof. Engell)
IncompressibleFluidNetwork	Multi-way connections of pipes and incompressible medium model
BranchingDynamicPipes	Multi-way connections of pipes with dynamic momentum balance, pressure wave and flow reversal
HeatExchanger	Demo of a heat exchanger model
TraceSubstances	Library demonstrating the usage of trace substances
InverseParameterization	Demonstrates the parameterization of a pump and a pipe for given nominal values
Explanatory	A set of examples illustrating when special attention has to be paid

---

Examples.PumpingSystem



Model of a pumping system for drinking water

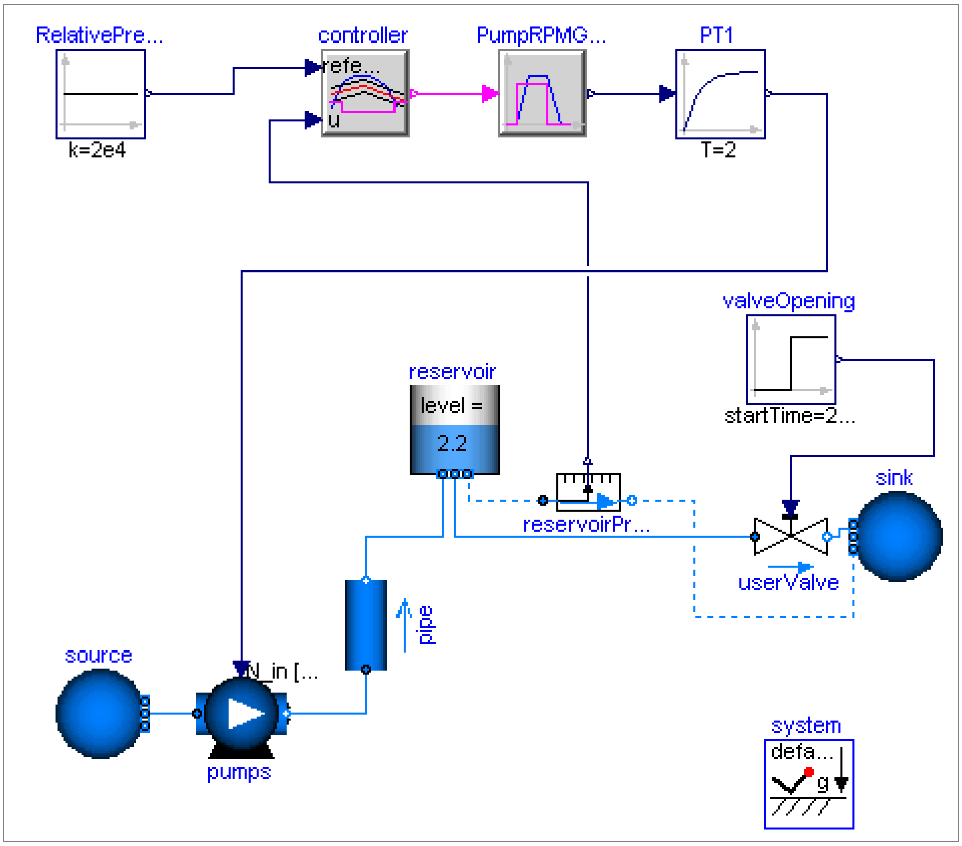
Information

Water is pumped from a source by a pump (fitted with check valves), through a pipe whose outlet is 50 m 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. A small but nonzero rotational speed is used to represent the standby state of the pumps, in order to avoid singularities in the flow characteristic.

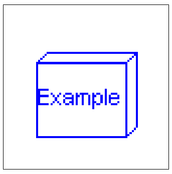
Simulate for 2000 s. When the valve is opened at time t=200, the pump starts turning on and off to keep the reservoir level around 2 meters, which roughly corresponds to a gauge pressure of 200 mbar

If using Dymola, turn off "Equidistant time grid" to avoid numerical errors.



Extends from Modelica.Icons.Example (Icon for an example model).

Examples.HeatingSystem



Simple model of a heating system

Information

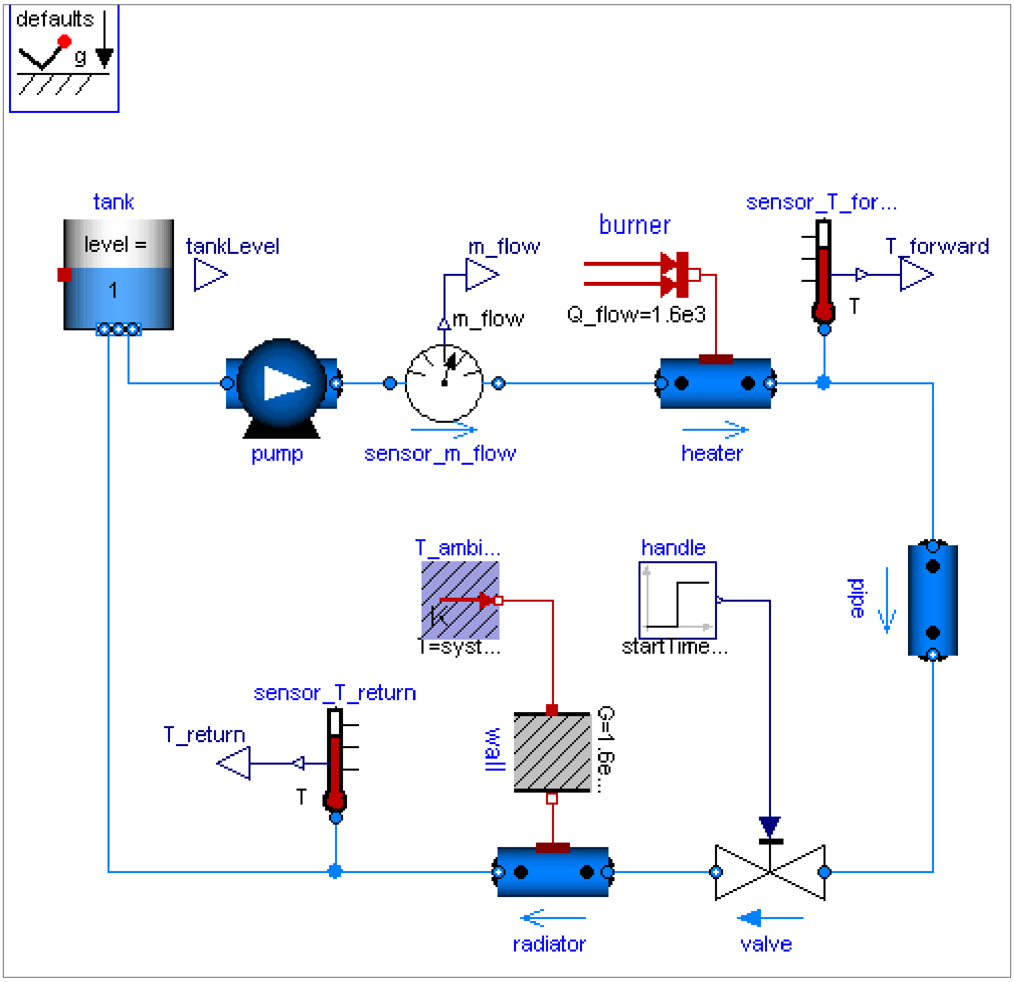
Simple heating system with a closed flow cycle. It is set up for steady-state initial values. After 2000s of simulation time the valve fully opens. A simple idealized control is embedded into the respective components, so that the heating system can be regulated with the valve: the pump controls the pressure, the burner controls the temperature.

One can investigate the temperatures and flows for different settings of `system.energyDynamics` (see Assumptions tab of the system object). With `system.energyDynamics==Types.Dynamics.SteadyState` all but one dynamic states are eliminated. The left state `tank.m` is to account for the closed flow cycle. It is constant as outflow and inflow are equal in a steady-state simulation.

Note that a closed flow cycle generally causes circular equalities for the mass flow rates and leaves the pressure undefined. This is why the tank.massDynamics, i.e. the tank level determining the port pressure, is modified locally to Types.Dynamics.FixedInitial.

Also note that the tank is thermally isolated againts its ambient. This way the temperature of the tank is also well defined for zero flow rate in the heating system, e.g. for valveOpening.offset=0 at the beginning of a simulation. The pipe however is assumed to be perfectly isolated. If steady-state values shall be obtained with the valve fully closed, then a thermal coupling between the pipe and its ambient should be defined as well.

Moreover it is worth noting that the idealized direct connection between the heater and the pipe, resulting in equal port pressures, is treated as high-index DAE, as opposed to a nonlinear equation system for connected pressure loss correlations. A pressure loss correlation could be additionally introduced to model the fitting between the heater and the pipe, e.g. to adapt different diameters.



Extends from Modelica.Icons.Example (Icon for an example model).

Parameters

Type	Name	Description
replaceable package Medium		

Connectors

Type	Name	Description
replaceable package Medium		

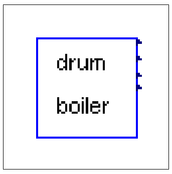
Examples.DrumBoiler

Drum boiler example, see Franke, Rode, Krueger: On-line Optimization of Drum Boiler Startup, 3rd International Modelica Conference, Linkoping, 2003

Package Content

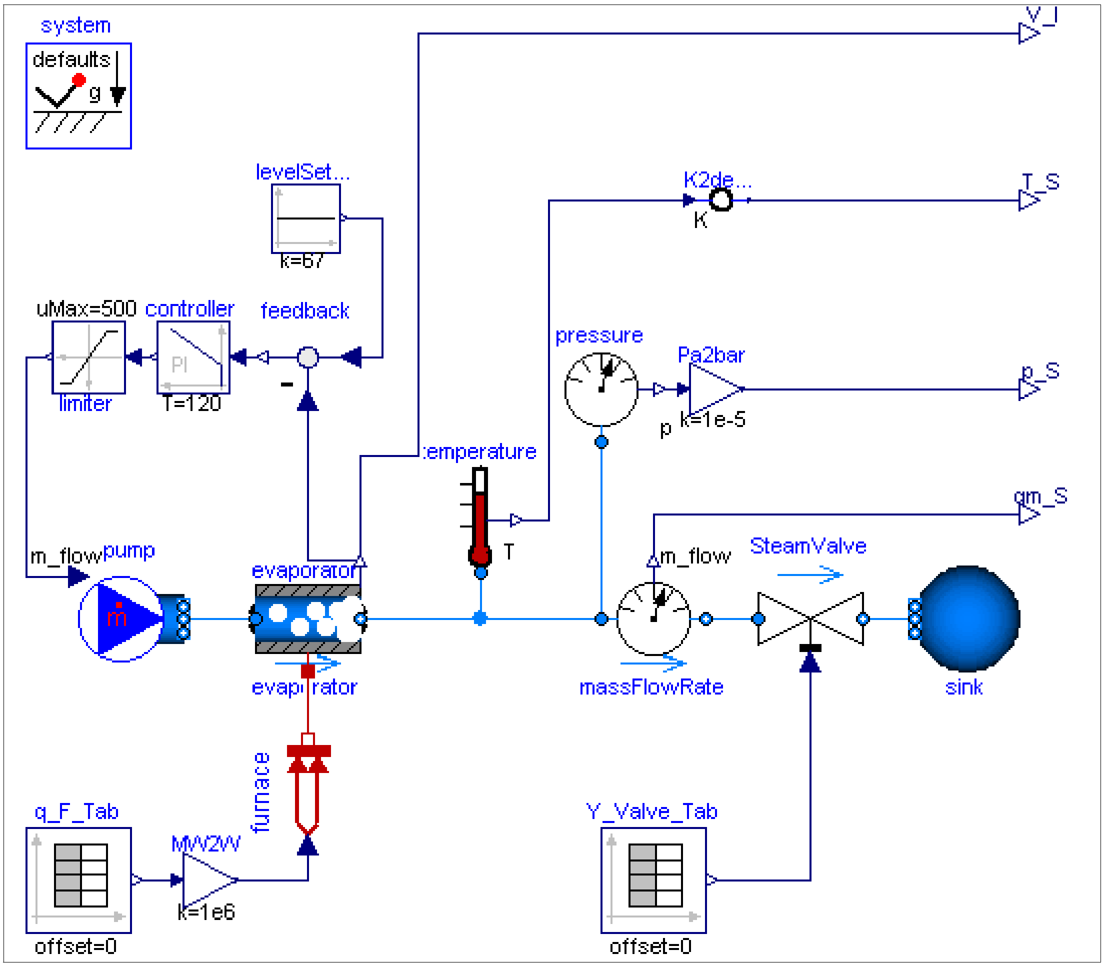
Name	Description
DrumBoiler	Complete drum boiler model, including evaporator and supplementary components
BaseClasses	Additional components for drum boiler example

Examples.DrumBoiler.DrumBoiler



Complete drum boiler model, including evaporator and supplementary components

Information



Extends from Modelica.Icons.Example (Icon for an example model).

Connectors

Type	Name	Description

output RealOutput	T_S	
output RealOutput	p_S	
output RealOutput	qm_S	
output RealOutput	V_l	

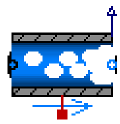
Examples.DrumBoiler.BaseClasses

Additional components for drum boiler example

Package Content

Name	Description
 EquilibriumDrumBoiler	Simple Evaporator with two states, see Astroem, Bell: Drum-boiler dynamics, Automatica 36, 2000, pp.363-378

Examples.DrumBoiler.BaseClasses.EquilibriumDrumBoiler



Simple Evaporator with two states, see Astroem, Bell: Drum-boiler dynamics, Automatica 36, 2000, pp.363-378

Information

Model of a simple evaporator with two states. The model assumes two-phase equilibrium inside the component; saturated steam goes out of the steam outlet.

References: Astroem, Bell: Drum-boiler dynamics, Automatica 36, 2000, pp.363-378

Extends from Interfaces.PartialTwoPort (Partial component with two ports).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Mass	m_D	mass of surrounding drum metal [kg]
SpecificHeatCapacity	cp_D	specific heat capacity of drum metal [J/(kg.K)]
Volume	V_t	total volume inside drum [m3]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Volume	V_l_start	Start value of liquid volumeStart value of volume [m3]

Connectors

Type	Name	Description
replaceable package Medium		Medium in the component
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)

FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
HeatPort_a	heatPort	
output RealOutput	V	liquid volume

Examples.Tanks

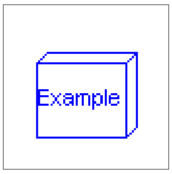
Library demonstrating the usage of the tank model

Package Content

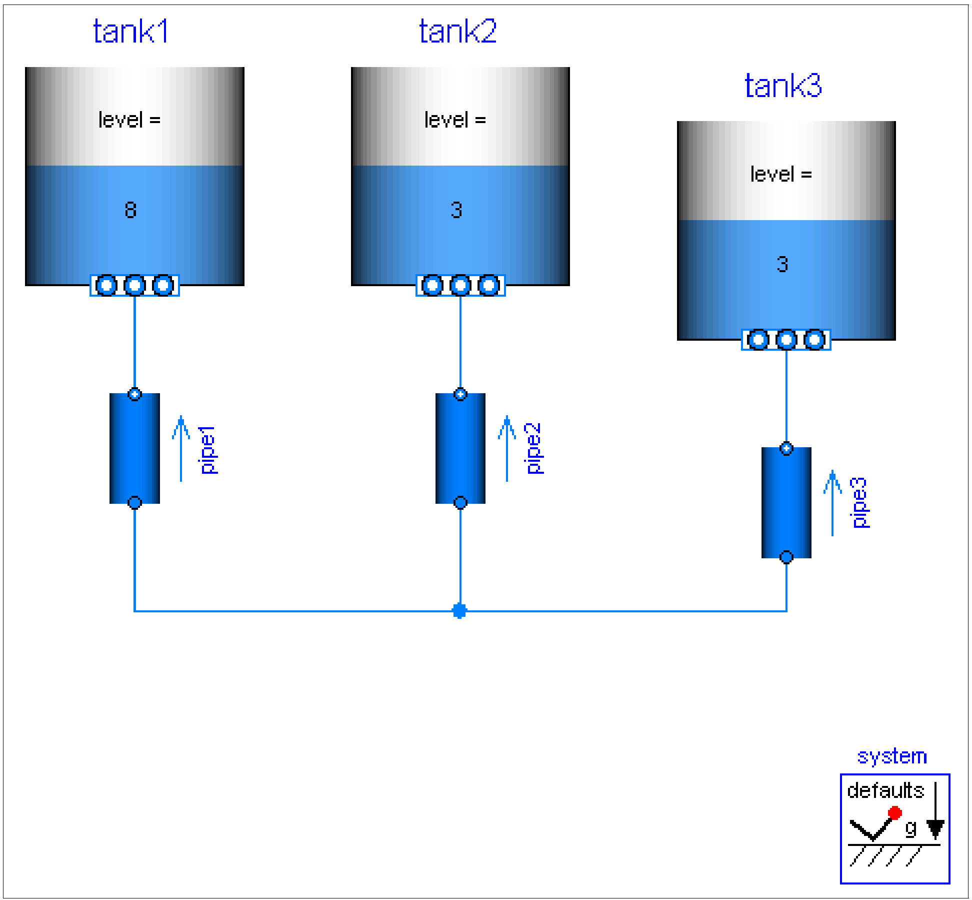
Name	Description
ThreeTanks	Demonstrating the usage of SimpleTank
TanksWithOverflow	Two tanks connected with pipes at different heights
EmptyTanks	Show the treatment of empty tanks

Examples.Tanks.ThreeTanks

Demonstrating the usage of SimpleTank

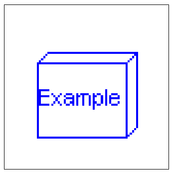


Information



Extends from Modelica.Icons.Example (Icon for an example model).

Examples.Tanks.TanksWithOverflow



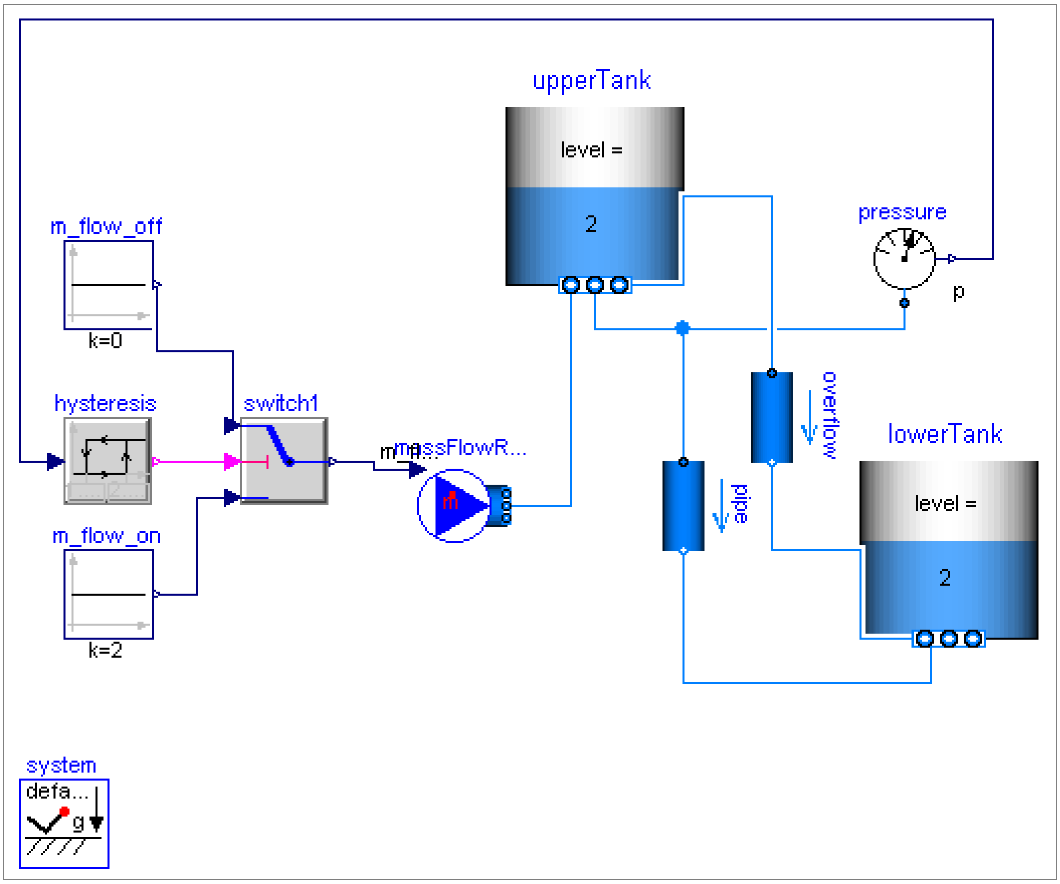
Two tanks connected with pipes at different heights

Information

The mass flow rate to the upper tank is controlled by the static pressure at its bottom. The fluid flows through a pipe and forced by different heights from the upper tank to the lower tank.

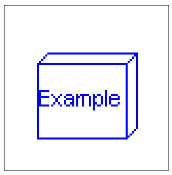
Additional fluid flows through an overflow pipe if the level of the upper tank exceeds 10m. Initially the overflow enters the lower tank above its fluid level; later on the fluid level exceeds the overflow port.

Note that the number of solver intervals has been increased, accounting for the long simulation time horizon. Otherwise the simulation may fail due to too large steps subject to events. Alternatively the simulation accuracy could be increased in order to avoid errors.



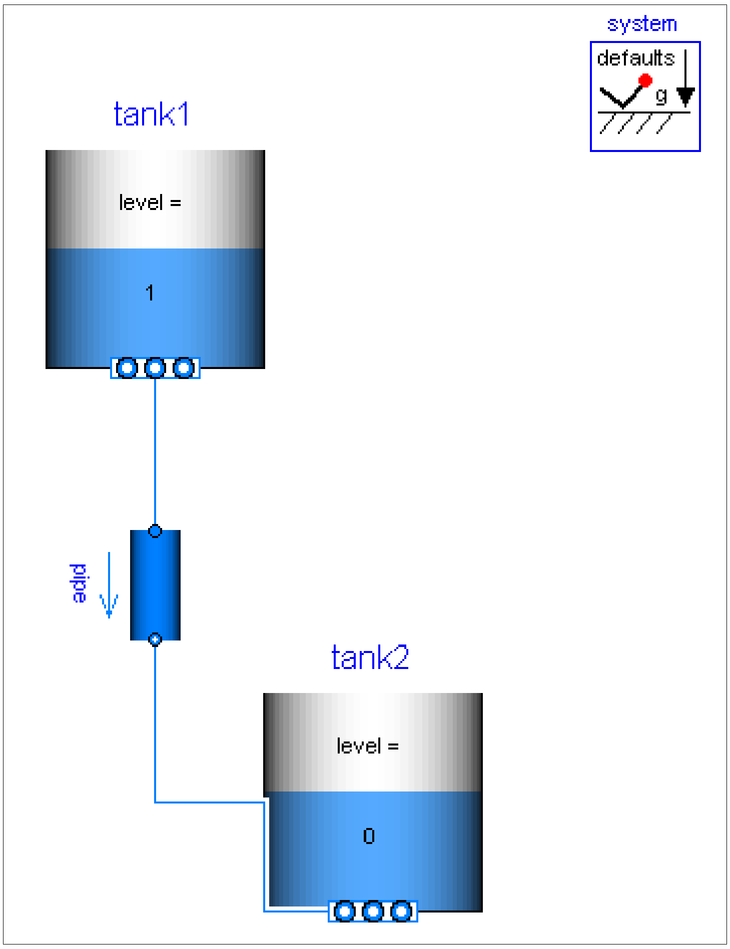
Extends from Modelica.Icons.Example (Icon for an example model).

Examples.Tanks.EmptyTanks



Show the treatment of empty tanks

Information



Extends from Modelica.Icons.Example (Icon for an example model).

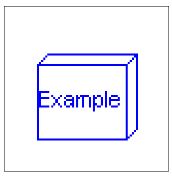
Examples.ControlledTankSystem

Tank system with controller, start/stop/shut operation and diagram animation

Package Content

Name	Description
ControlledTanks	Demonstrating the controller of a tank filling/emptying system
Utilities	

Examples.ControlledTankSystem.ControlledTanks



Demonstrating the controller of a tank filling/emptying system

Information

With this example, the controller of a tank filling/emptying system is demonstrated.

The basic operation is to fill and empty the two tanks:

1. Valve 1 is opened and tank 1 is filled.

2. When tank 1 reaches its fill level limit, valve 1 is closed.
3. After a waiting time, valve 2 is opened and the fluid flows from tank 1 into tank 2.
4. When tank 1 reaches its minimum level, valve 2 is closed.
5. After a waiting time, valve 3 is opened and the fluid flows out of tank 2
6. When tank 2 reaches its minimum level, valve 3 is closed

The above "normal" process can be influenced by three buttons:

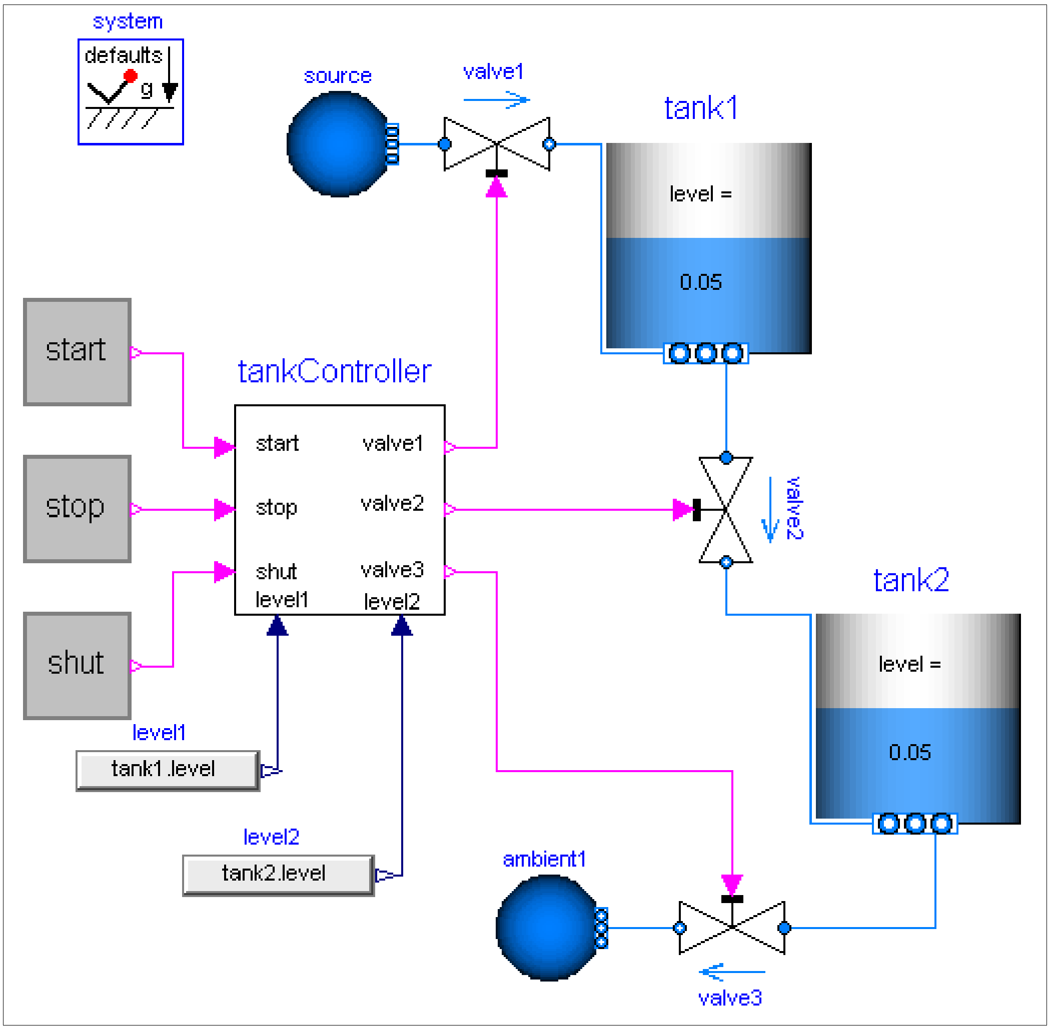
- Button **start** starts the above process. When this button is pressed after a "stop" or "shut" operation, the process operation continues. .
- Button **stop** stops the above process by closing all valves. Then, the controller waits for further input (either "start" or "shut" operation).
- Button **shut** is used to shutdown the process, by emptying at once both tanks by opening valve 2 and valve 3. When this is achieved, the process goes back to its start configuration where all 3 valves are closed. Clicking on "start", restarts the process.

The demo-run uses the following button presses:

- Button **start** pressed at 20 s.
- Button **stop** pressed at 220 s
- Button **start** pressed at 280 s
- Button **stop** pressed at 650 s
- Button **shut** pressed at 700 s
- Simulate for 900 s

This example is based on

Dressler I. (2004):
Code Generation From JGrafchart to Modelica. Master thesis, supervisor: Karl-Erik Arzen, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, March 30, 2004



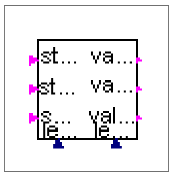
Extends from Modelica.Icons.Example (Icon for an example model).

Examples.ControlledTankSystem.Utilities

Package Content

Name	Description
TankController	Controller for tank system
NormalOperation	Normal operation of tank system (button start pressed)
RadioButton	Button that sets its output to true when pressed and is reset when an element of 'reset' becomes true

Examples.ControlledTankSystem.Utilities.TankController



Controller for tank system

Parameters

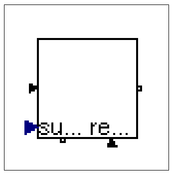
Type	Name	Description

Height	maxLevel	Fill level of tank 1 [m]
Height	minLevel	Lowest level of tank 1 and 2 [m]
Time	waitTime	Wait time, between operations [s]

Connectors

Type	Name	Description
input BooleanInput	start	
input BooleanInput	stop	
input BooleanInput	shut	
input RealInput	level1	
input RealInput	level2	
output BooleanOutput	valve1	
output BooleanOutput	valve2	
output BooleanOutput	valve3	

Examples.ControlledTankSystem.Utilities.NormalOperation



Normal operation of tank system (button start pressed)

Information

Extends from Modelica.StateGraph.PartialCompositeStep (Superclass of a subgraph, i.e., a composite step that has internally a StateGraph).

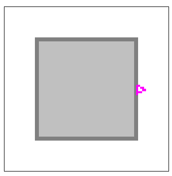
Parameters

Type	Name	Description
Height	maxLevel	Fill level of tank 1 [m]
Height	minLevel	Lowest level of tank 1 and 2 [m]
Time	waitTime	Wait time between operations [s]
Exception connections		
Integer	nSuspend	Number of suspend ports
Integer	nResume	Number of resume ports

Connectors

Type	Name	Description
Step_in	inPort	
Step_out	outPort	
CompositeStep_suspend	suspend[nSuspend]	
CompositeStep_resume	resume[nResume]	
input RealInput	level1	

Examples.ControlledTankSystem.Utilities.RadioButton



Button that sets its output to true when pressed and is reset when an element of 'reset' becomes true

Information

Parameters

Type	Name	Description
Time	buttonTimeTable[:]	Time instants where button is pressend [s]
Time varying expressions		
Boolean	reset[:]	Reset button to false, if an element of reset becomes true

Connectors

Type	Name	Description
output BooleanOutput	on	

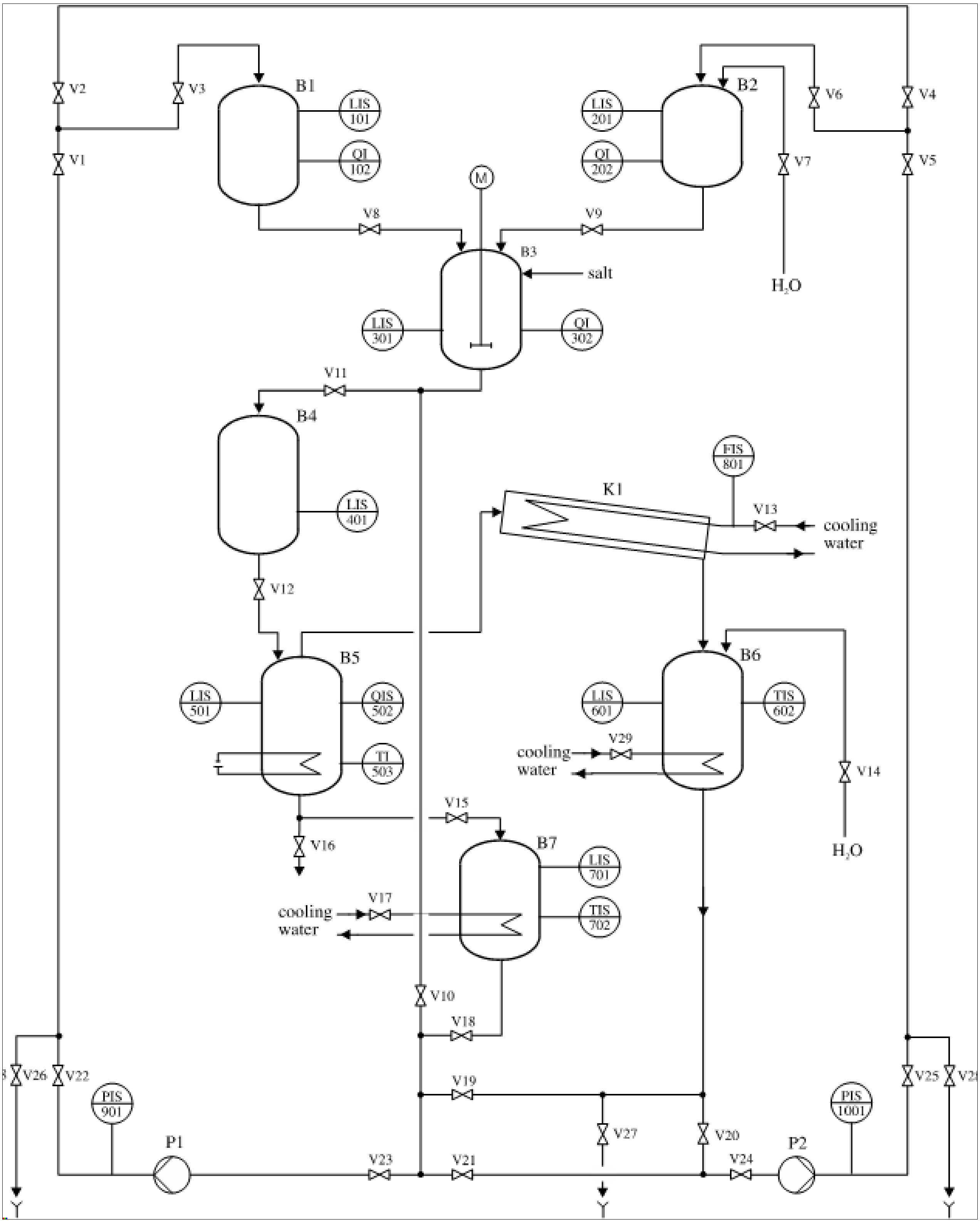
Examples.AST_BatchPlant

Model of the experimental batch plant at Process Control Laboratory at University of Dortmund (Prof. Engell)

Information

The process under consideration is an evaporation plant for a student lab at the Process Control Laboratory (AST) of the University of Dortmund that evaporates a water sodium chloride mixture so that a higher concentrated solution is produced. The task of the students is to learn how to program the process control system. A picture of the batch plant is shown in the figure below.

The flow sheet diagram is shown in the next figure.

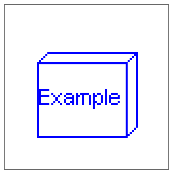


Pure water from tank B1 and concentrated sodium chloride solution from tank B2 are mixed in a mixing tank B3. After buffering in tank B4 the mixture flows to the evaporator B5. Here the water sodium chloride mixture is evaporated until the desired con-centration is reached. The steam is condensed in the condenser K1 and cooled afterwards in the cooling tank B6. The concentrated solution is also led to a cooling tank B7. The cooled fluids are pumped back to the charging vessels by the pumps P1 and P2. Be-tween the tanks several valves are present that are regulated by a central control system.

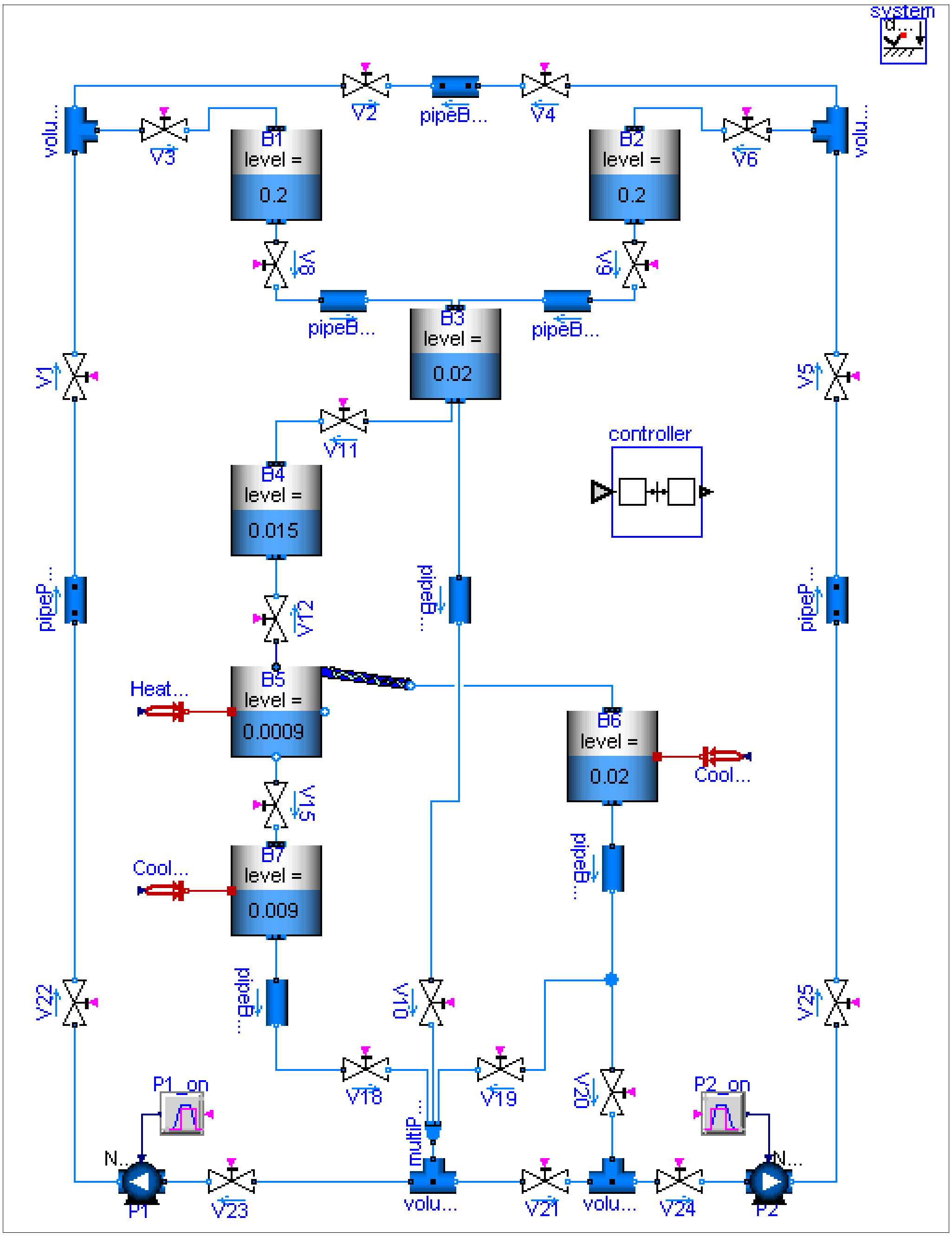
Package Content

Name	Description
BatchPlant_StandardWater	
BaseClasses	
Test	Test of used tank models

Examples.AST_BatchPlant.BatchPlant_StandardWater



Information



Extends from Modelica.Icons.Example (Icon for an example model).

Parameters

Type	Name	Description
replaceable package BatchMedium		Component media
Length	pipeDiameter	[m]

Connectors

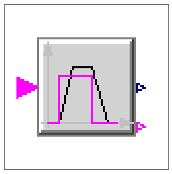
Type	Name	Description
replaceable package BatchMedium		Component media

Examples.AST_BatchPlant.BaseClasses

Package Content

Name	Description
TriggeredTrapezoid	Triggered trapezoid generator
setReal	Set output signal to a time varying Real expression
TankWith3InletOutletArraysWithEvaporatorCondensor	Tank with Heating and Evaporation
InnerTank	
Controller	
ControllerUtilities	
Init	Enumeration to define initialization options
TankWithTopPorts	Tank with inlet/outlet ports and with inlet ports at the top

Examples.AST_BatchPlant.BaseClasses.TriggeredTrapezoid



Triggered trapezoid generator

Information

The block TriggeredTrapezoid has a boolean input and a real output signal and requires the parameters *amplitude*, *rising*, *falling* and *offset*. The output signal **y** represents a trapezoidal signal dependent on the input signal **u**.

The behaviour is as follows: Assume the initial input to be false. In this case, the output will be *offset*. After a rising edge (i.e. the input changes from false to true), the output is rising during *rising* to the sum of *offset* and *amplitude*. In contrast, after a falling edge (i.e. the input changes from true to false), the output is falling during *falling* to a value of *offset*.

Note, that the case of edges before expiration of rising or falling is handled properly.

Extends from Modelica.Blocks.Interfaces.partialBooleanBlockIcon (Basic graphical layout of logical block).

Parameters

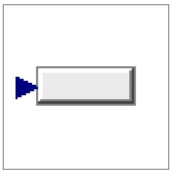
Type	Name	Description

Real	amplitude	Amplitude of trapezoid
Time	rising	Rising duration of trapezoid [s]
Time	falling	Falling duration of trapezoid [s]
Real	offset	Offset of output signal

Connectors

Type	Name	Description
input BooleanInput	u	Connector of Boolean input signal
output RealOutput	y	Connector of Real output signal
output BooleanOutput	y_high	

Examples.AST_BatchPlant.BaseClasses.setReal



Set output signal to a time varying Real expression

Information

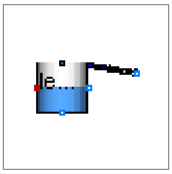
Parameters

Type	Name	Description
Time varying input signal		
RealInput	u	Set value of Real input

Connectors

Type	Name	Description
Time varying input signal		
input RealInput	u	Set value of Real input

Examples.AST_BatchPlant.BaseClasses.TankWith3InletOutletArraysWithEvaporatorCondensor



Tank with Heating and Evaporation

Information

This tank has the same geometric variables as TankWith3InletOutletArrays plus the feature of a HeatPort and the possibility of evaporation. (Assumption: The gas is condensed emidiatly afterwards so that a liquid boiling fluid is created.)

The tank can be initialized with the following options:

- GuessValues: no explicit initial conditions
- InitialValues: initial values of temperature (or specific enthalpy), composition and level are specified
- SteadyStateHydraulic: initial values of temperature (or specific enthalpy) and composition are specified; the initial level is determined so that levels and pressure are at steady state.

Full steady state initialization is not supported, because the corresponding intial equations for temperature/enthalpy are undetermined (the flow rate through the port at steady state is zero).

Parameters

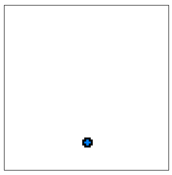
Type	Name	Description

Area	crossArea	Tank area [m2]
Area	top_pipeArea[n_TopPorts]	Area of outlet pipe [m2]
Area	side_pipeArea[n_SidePorts]	Area of outlet pipe [m2]
Area	bottom_pipeArea[n_BottomPorts]	Area of outlet pipe [m2]
Height	height	Height of Tank [m]
Volume	V0	Volume of the liquid when the level is zero [m3]
Real	side_heights[n_SidePorts]	
Real	bottom_heights[n_BottomPorts]	
Real	top_heights[n_TopPorts]	
AbsolutePressure	p_ambient	Tank surface pressure [Pa]
Temperature	T_ambient	Tank surface Temperature [K]
Integer	n_TopPorts	number of Top connectors
Integer	n_SidePorts	number of side connectors
Integer	n_BottomPorts	number of bootom connectors
Real	min_level_for_heating	
Initialization		
Height	level_start	Initial tank level [m]
Init	initType	Initialization option
Boolean	use_T_start	Use T_start if true, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]

Connectors

Type	Name	Description
FluidPort_b	BottomFluidPort[n_BottomPorts]	
FluidPort_a	TopFluidPort[n_TopPorts]	
FluidPort_b	SideFluidPort[n_SidePorts]	
FluidPort_b	Condensed	
HeatPort_a	heatPort	

Examples.AST_BatchPlant.BaseClasses.InnerTank



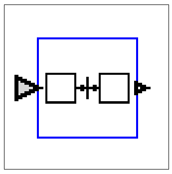
Parameters

Type	Name	Description
MassFraction	Xi[Medium.nXi]	Actual mass fractions of fluid in tank [kg/kg]

Connectors

Type	Name	Description
FluidPort_a	port	

Examples.AST_BatchPlant.BaseClasses.Controller



Parameters

Type	Name	Description
Real	w_dilution	
Real	w_concentrate	
Real	startTime	
Real	T5_batch_level	

Connectors

Type	Name	Description
Port_Sensors	sensors	
Port_Actuators	actuators	

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities

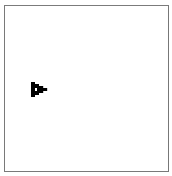
Package Content

Name	Description
Adapter_Inference	
Adapter_Superposition	
Block_Recipe_TBD	
BlockMain	
Buffer_Recipe_TBD	
BufferMain	
Port_Actuators	
Port_IdleTanks	
Port_Sensors	

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Adapter_Inference

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Adapter_Superposition

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Block_Recipe_TBD

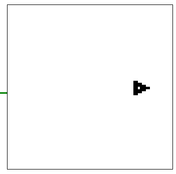


Parameters

Type	Name	Description
Real	startTime	
Real	w_dilution	
Real	w_concentrat	
Real	T3_batch_level	

Real	T5_batch_level	

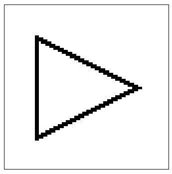
Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.BlockMain



Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Buffer_Recipe_TBD

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.BufferMain

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_Actuators



Contents

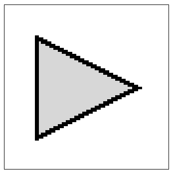
Type	Name	Description
Boolean	P1	
Boolean	P2	
Boolean	T5_Heater	
Boolean	T7_Cooling	
Boolean	T6_Cooling	
Boolean	V1	
Boolean	V2	
Boolean	V3	
Boolean	V4	
Boolean	V5	
Boolean	V6	
Boolean	V8	
Boolean	V9	
Boolean	V10	
Boolean	V11	
Boolean	V12	
Boolean	V15	
Boolean	V18	
Boolean	V19	
Boolean	V20	
Boolean	V21	
Boolean	V22	
Boolean	V23	
Boolean	V24	
Boolean	V25	

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_IdleTanks

Contents

Type	Name	Description
Boolean	T5_idle	
Boolean	T7_idle	

Examples.AST_BatchPlant.BaseClasses.ControllerUtilities.Port_Sensors



Contents

Type	Name	Description
Real	LIS_301	
Real	QI_302	
Real	LIS_501	
Real	QIS_502	
Real	TI_503	
Real	LIS_601	
Real	TIS_602	
Real	LIS_701	
Real	TIS_702	

Examples.AST_BatchPlant.BaseClasses.Init

Enumeration to define initialization options

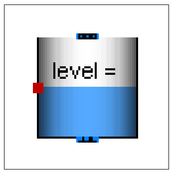
Information

Integer type that can have the following values (to be selected via choices menu):

Types.Init.	Meaning
GuessValues	GuessValues -- Guess values (not fixed) for p, T or h, X, C
InitialValues	Initial values for p, T or h, X, C
SteadyStateMomentum	Steady state momentum
SteadyStateHydraulic	Hydraulic steady state (der(p)=0), guess value for p, initial values for T or h, X, C
SteadyState	Steady state (guess values for p, T or h, X, C)

Examples.AST_BatchPlant.BaseClasses.TankWithTopPorts

Tank with inlet/outlet ports and with inlet ports at the top



Information

Model of a tank that is open to the environment at the fixed pressure `p_ambient`. The tank is filled with a single or multiple-substance liquid, assumed to have uniform temperature and mass fractions.

At the top of the tank over the maximal fill level **height** a vector of FluidPorts, called **topPorts**, is present. The assumption is made that fluid flows always in to the tank via these ports (and never back in to the connector).

The vector of connectors **ports** are fluid ports at the bottom and side of the tank at a defineable height. Fluid can flow either out of or in to this port. The fluid level of the tank may be below one of these ports. This case is approximated by introducing a large pressure flow coefficient so that the mass flow rate through this port is very small in this case.

If the tank starts to over flow (i.e., level > height), an assertion is triggered.

When the diagram layer is open in the plot environment, the level of the tank is dynamically visualized. Note, the speed of the diagram animation in Dymola can be set via command **animationSpeed**(), e.g., animationSpeed(speed = 10)

Extends from Interfaces.PartialLumpedVolume (Lumped volume with mass and energy balance).

Parameters

Type	Name	Description
Height	height	Maximum level of tank before it overflows [m]
Area	crossArea	Area of tank [m2]
Volume	V0	Volume of the liquid when level = 0 [m3]
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
VesselPortsData	portsData[nPorts]	Data of inlet/outlet ports at side and bottom of tank
Assumptions		
Ambient		
AbsolutePressure	p_ambient	Tank surface pressure [Pa]
Temperature	T_ambient	Tank surface Temperature [K]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
Initialization		
Height	level_start	Start value of tank level [m]
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Port properties		
Real	hysteresisFactor	Hysteresis for empty pipe = diameter*hysteresisFactor
Boolean	stiffCharacteristicForEmptyPort	=true, if steep pressure loss characteristic for empty pipe port
Real	zetaLarge	Large pressure loss factor if mass flows out of empty pipe port
MassFlowRate	m_flow_small	Regularization range at zero mass flow rate [kg/s]

Connectors

Type	Name	Description
VesselFluidPorts_a	topPorts[nTopPorts]	Inlet ports over height at top of tank (fluid flows only from the port in to the tank)
VesselFluidPorts_b	ports[nPorts]	inlet/outlet ports at bottom or side of tank (fluid flows in to or out of port; a port might be above the fluid level)
HeatPort_a	heatPort	

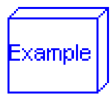
Examples.AST_BatchPlant.Test

Test of used tank models

Package Content

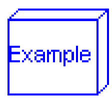
Name	Description
OneTank	Tank with one time-varying top inlet mass flow rate and a bottom outlet into the ambient
TwoTanks	
TankWithEmptyingPipe1	Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient
TankWithEmptyingPipe2	Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient
TanksWithEmptyingPipe1	Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient
TanksWithEmptyingPipe2	Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient

Examples.AST_BatchPlant.Test.OneTank



Tank with one time-varying top inlet mass flow rate and a bottom outlet into the ambient

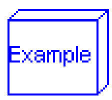
Examples.AST_BatchPlant.Test.TwoTanks



Parameters

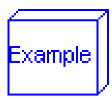
Type	Name	Description
Boolean	stiffCharacteristicForEmptyPort	

Examples.AST_BatchPlant.Test.TankWithEmptyingPipe1



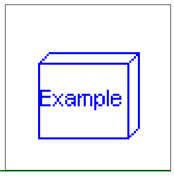
Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient

Examples.AST_BatchPlant.Test.TankWithEmptyingPipe2



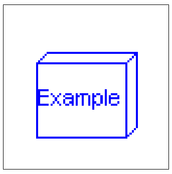
Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient

Examples.AST_BatchPlant.Test.TanksWithEmptyingPipe1



Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient

Examples.AST_BatchPlant.Test.TanksWithEmptyingPipe2

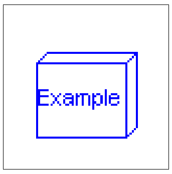


Demonstrates a tank with one constant top inlet mass flow rate and a bottom outlet into the ambient

Parameters

Type	Name	Description
Boolean	stiffCharacteristicForEmptyPort	

Examples.IncompressibleFluidNetwork



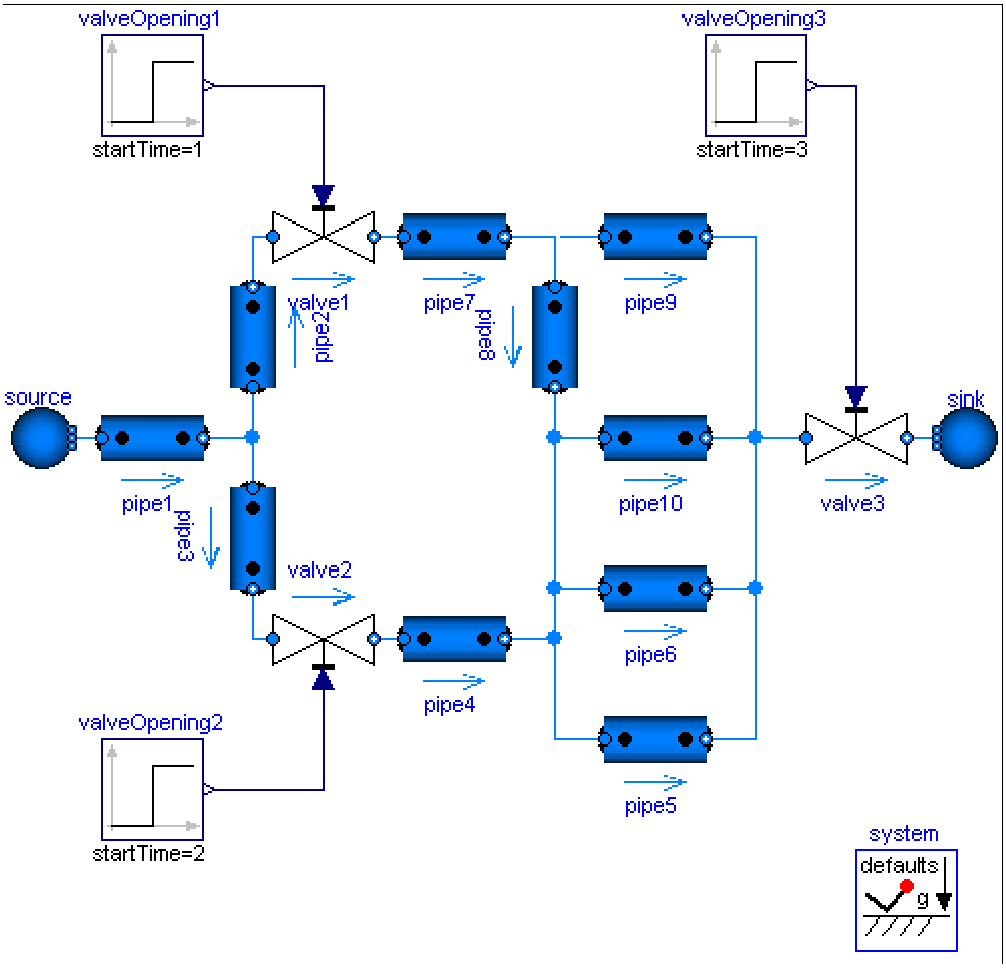
Multi-way connections of pipes and incompressible medium model

Information

This example demonstrates two aspects: the efficient treatment of multi-way connections and the usage of an incompressible medium model.

Normally one would expect bad equation systems in multi-way connections and possibly introduce mixing volumes to work around this. Here the problem is treated with the the modelStructure=av_vb in the DynamicPipe model. Each pipe exposes the states of the outer fluid segments to the respective fluid ports. Consequently the pressures of all connected pipe segments get lumped together into one mass balance spanning the whole connection set. With the stream concept in the fluid ports, the energy and substance balances remain independent in the connected pipe segments.

The model does not contain pressure dynamics as an incompressible medium is used (Essotherm650). Pressure dynamics becomes present with a compressible medium model (e.g. StandardWater).



Extends from Modelica.Icons.Example (Icon for an example model).

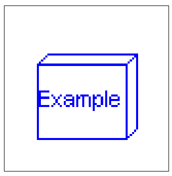
Parameters

Type	Name	Description
replaceable package Medium		

Connectors

Type	Name	Description
replaceable package Medium		

Examples.BranchingDynamicPipes



Multi-way connections of pipes with dynamic momentum balance, pressure wave and flow reversal

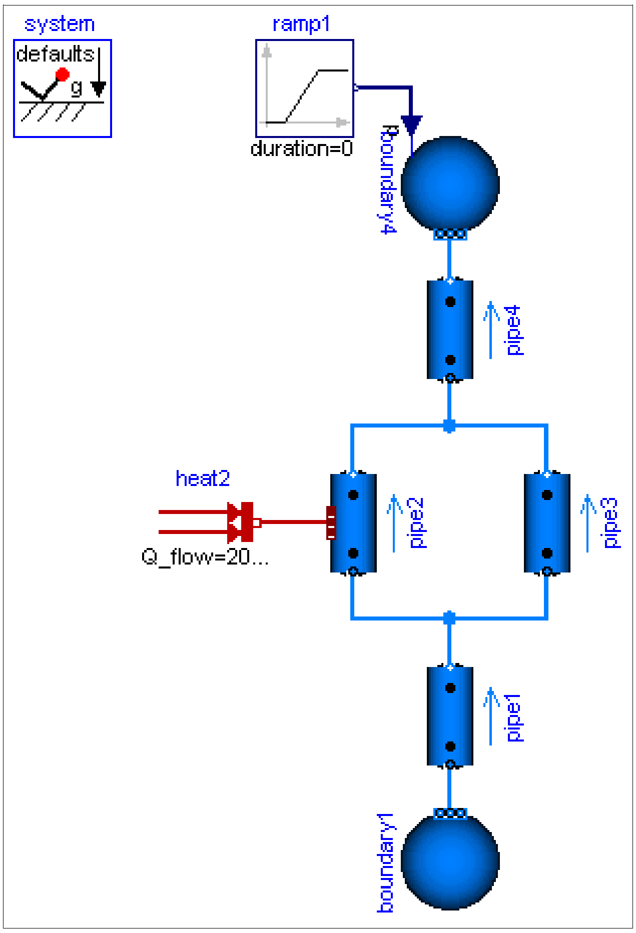
Information

This model demonstrates the use of distributed pipe models with dynamic energy, mass and momentum balances. At time=2s the pressure of boundary4 jumps, which causes a pressure wave and flow reversal.

Change system.momentumDynamics on the Assumptions tab of the system object from DynamicFreeInitial to SteadyState, in order to assume a steady-state momentum balance. This is the default for all models of the library.

Change the Medium from MoistAir to StandardWater, in order to investigate a medium with significantly different density. Note the static head caused by the elevation of the pipes.

Note, pipe4.modelStructure = av_b, i.e., the pipe has no volume at port_b. It is not possible to have a volume at port_b, since otherwise the pressure of the volume is defined by the connected boundary source. This in turn means that the derivative of the pressure of the boundary source is needed, since the volume requires this derivative. It is, however, not possible to compute this derivative because the input pressure is changing disontinuously and its derivative would be a dirac impulse.



Extends from Modelica.Icons.Example (Icon for an example model).

Parameters

Type	Name	Description
replaceable package Medium		

Connectors

Type	Name	Description
replaceable package Medium		

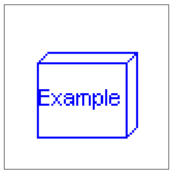
Examples.HeatExchanger

Demo of a heat exchanger model

Package Content

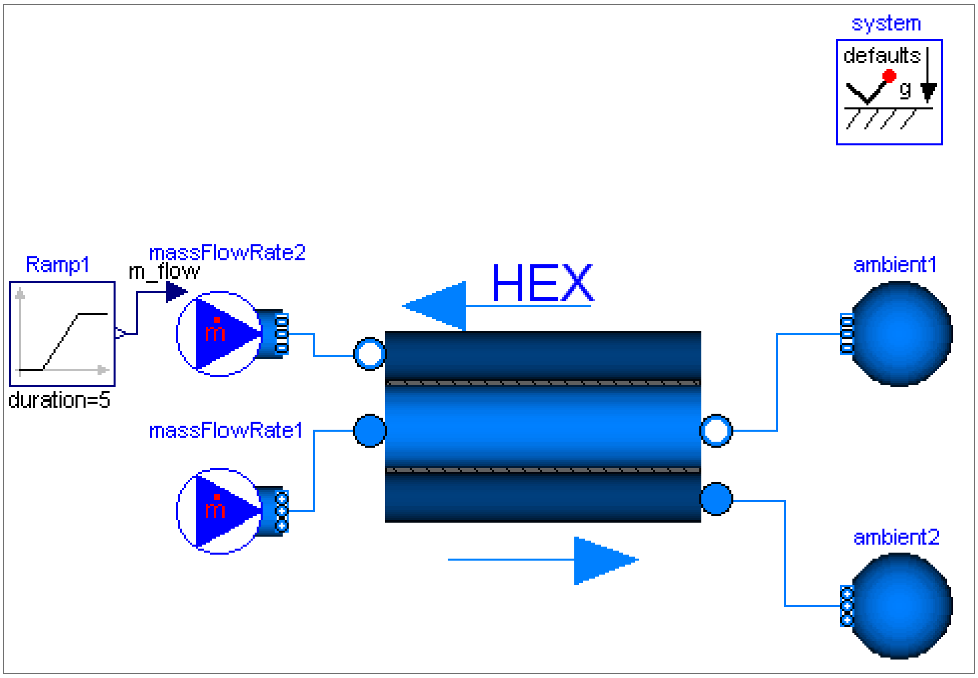
Name	Description
HeatExchangerSimulation	simulation for the heat exchanger model
BaseClasses	Additional models for heat exchangers

Examples.HeatExchanger.HeatExchangerSimulation



simulation for the heat exchanger model

Information



Extends from Modelica.Icons.Example (Icon for an example model).

Parameters

Type	Name	Description
replaceable package Medium		

Connectors

Type	Name	Description
replaceable package Medium		

Examples.HeatExchanger.BaseClasses

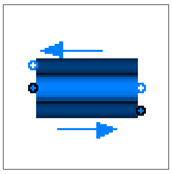
Additional models for heat exchangers

Package Content

Name	Description

BasicHX	Simple heat exchanger model
WallConstProps	Pipe wall with capacitance, assuming 1D heat conduction and constant material properties

Examples.HeatExchanger.BaseClasses.BasicHX



Simple heat exchanger model

Information

Simple model of a heat exchanger consisting of two pipes and one wall in between. For both fluids geometry parameters, such as heat transfer area and cross section as well as heat transfer and pressure drop correlations may be chosen. The flow scheme may be concurrent or counterflow, defined by the respective flow directions of the fluids entering the component. The design flow direction with positive m_flow variables is counterflow.

Parameters

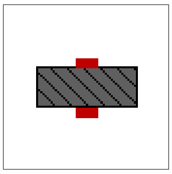
Type	Name	Description
Integer	nNodes	Spatial segmentation
Length	length	Length of flow path for both fluids [m]
Length	s_wall	Wall thickness [m]
Fluid 1		
Area	crossArea_1	Cross sectional area [m2]
Length	perimeter_1	Flow channel perimeter [m]
Area	area_h_1	Heat transfer area [m2]
Length	roughness_1	Absolute roughness of pipe (default = smooth steel pipe) [m]
Fluid 2		
Area	crossArea_2	Cross sectional area [m2]
Length	perimeter_2	Flow channel perimeter [m]
Area	area_h_2	Heat transfer area [m2]
Length	roughness_2	Absolute roughness of pipe (default = smooth steel pipe) [m]
Solid material properties		
Density	rho_wall	Density of wall material [kg/m3]
SpecificHeatCapacity	c_wall	Specific heat capacity of wall material [J/(kg.K)]
ThermalConductivity	k_wall	Thermal conductivity of wall material [W/(m.K)]
Assumptions		
Boolean	allowFlowReversal	allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Dynamics	momentumDynamics	Formulation of momentum balance, if pressureLoss options available
Initialization		
Wall		
Temperature	Twall_start	Start value of wall temperature [K]

Temperature	dT	Start value for pipe_1.T - pipe_2.T [K]
Boolean	use_T_start	Use T_start if true, otherwise h_start
Fluid 1		
AbsolutePressure	p_a_start1	Start value of pressure [Pa]
AbsolutePressure	p_b_start1	Start value of pressure [Pa]
Temperature	T_start_1	Start value of temperature [K]
SpecificEnthalpy	h_start_1	Start value of specific enthalpy [J/kg]
MassFraction	X_start_1[Medium_1.nX]	Start value of mass fractions m_i/m [kg/kg]
MassFlowRate	m_flow_start_1	Start value of mass flow rate [kg/s]
Fluid 2		
AbsolutePressure	p_a_start2	Start value of pressure [Pa]
AbsolutePressure	p_b_start2	Start value of pressure [Pa]
Temperature	T_start_2	Start value of temperature [K]
SpecificEnthalpy	h_start_2	Start value of specific enthalpy [J/kg]
MassFraction	X_start_2[Medium_2.nX]	Start value of mass fractions m_i/m [kg/kg]
MassFlowRate	m_flow_start_2	Start value of mass flow rate [kg/s]

Connectors

Type	Name	Description
FluidPort_b	port_b1	
FluidPort_a	port_a1	
FluidPort_b	port_b2	
FluidPort_a	port_a2	

Examples.HeatExchanger.BaseClasses.WallConstProps



Pipe wall with capacitance, assuming 1D heat conduction and constant material properties

Information

Simple model of circular (or any other closed shape) wall to be used for pipe (or duct) models. Heat conduction is regarded one dimensional, capacitance is lumped at the arithmetic mean temperature. The spatial discretization (parameter `n`) is meant to correspond to a connected fluid model discretization.

Parameters

Type	Name	Description
Integer	n	Segmentation perpendicular to heat conduction
Length	s	Wall thickness [m]
Area	area_h	Heat transfer area [m2]
Density	rho_wall	Density of wall material [kg/m3]
SpecificHeatCapacity	c_wall	Specific heat capacity of wall material [J/(kg.K)]
ThermalConductivity	k_wall	Thermal conductivity of wall material [W/(m.K)]
Mass	m[n]	Distribution of wall mass [kg]
Temperature	T_start	Wall temperature start value [K]

Temperature	dT	Start value for port_b.T - port_a.T [K]
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance

Connectors

Type	Name	Description
HeatPort_a	heatPort_a[n]	Thermal port
HeatPort_a	heatPort_b[n]	Thermal port

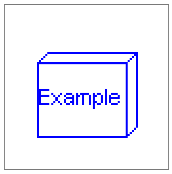
Examples.TraceSubstances

Library demonstrating the usage of trace substances

Package Content

Name	Description
RoomCO2	Demonstrates a room volume with CO2 accumulation
RoomCO2WithControls	Demonstrates a room volume with CO2 controls

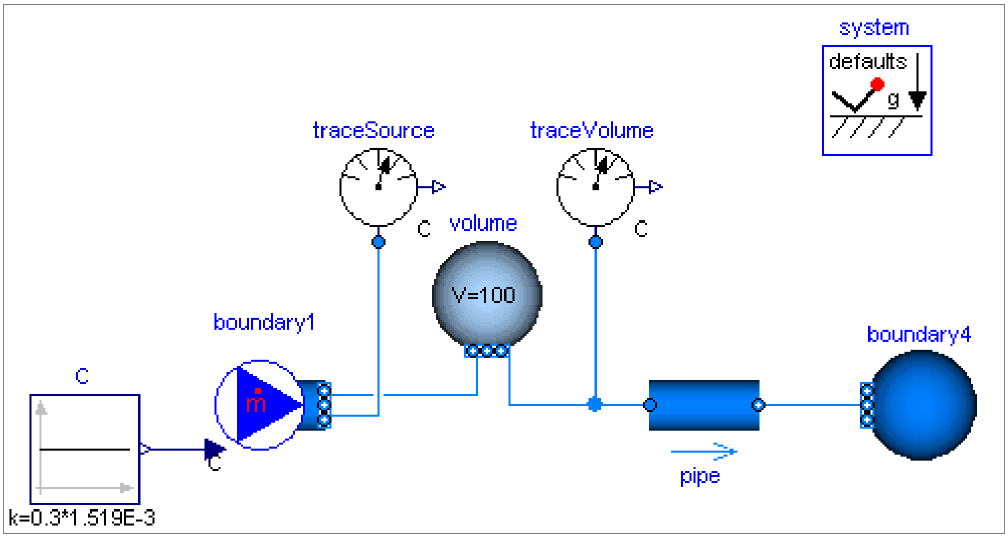
Examples.TraceSubstances.RoomCO2



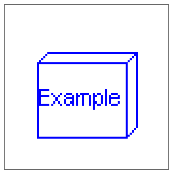
Demonstrates a room volume with CO2 accumulation

Information

This example consists of a volume with a carbon dioxide concentration that corresponds to about 1000 PPM. There is a fresh air stream with a carbon dioxide concentration of about 300 PPM. The fresh air stream is such that the air exchange rate is about 5 air changes per hour. After 1 hour of ventilation, the volume's carbon dioxide concentration is close to the concentration of the fresh air.



Extends from Modelica.Icons.Example (Icon for an example model).



Examples.TraceSubstances.RoomCO2WithControls

Demonstrates a room volume with CO2 controls

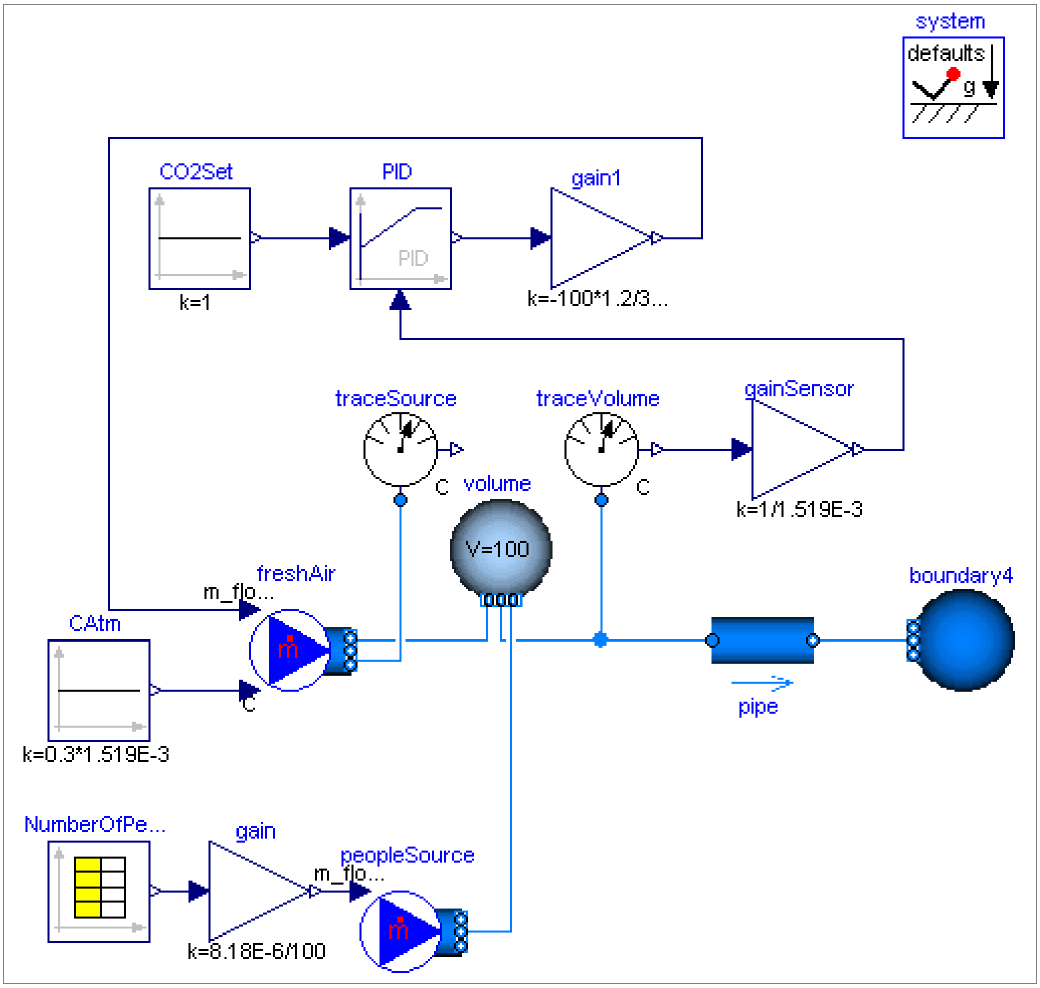
Information

This example illustrates a room volume with a CO2 source and a fresh air supply with feedback control. The CO2 emission rate is proportional to the room occupancy, which is defined by a schedule. The fresh air flow rate is controlled such that the room CO2 concentration does not exceed `1000 PPM (=1.519E-3 kg/kg)`. The fresh air has a CO2 concentration of `300 PPM` which corresponds to a typical CO2 concentration in the outside air.

The CO2 emission from the occupants is implemented as a mass flow source. Depending on the activity and size, a person emits about `8.18E-6 kg/s` CO2. In the model, this value is multiplied by the number of occupants. Since the mass flow rate associate with the CO2 source model contributes to the volume's energy balance, this mass flow rate should be kept small. Thus, in the source model, we set the CO2 concentration to `C={100} kg/kg`, and scaled the mass flow rate using

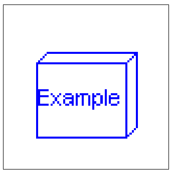
```
m_flow = 1/100 * nPeo * 8.18E-6 kg/(s*person)
```

where `nPeo` is the number of people in the room. This results in a mass flow rate that is about 5 orders of magnitudes smaller than the supply air flow rate, and hence its contribution to the volume's energy balance is negligible.



Extends from Modelica.Icons.Example (Icon for an example model).

Examples.InverseParameterization



Demonstrates the parameterization of a pump and a pipe for given nominal values

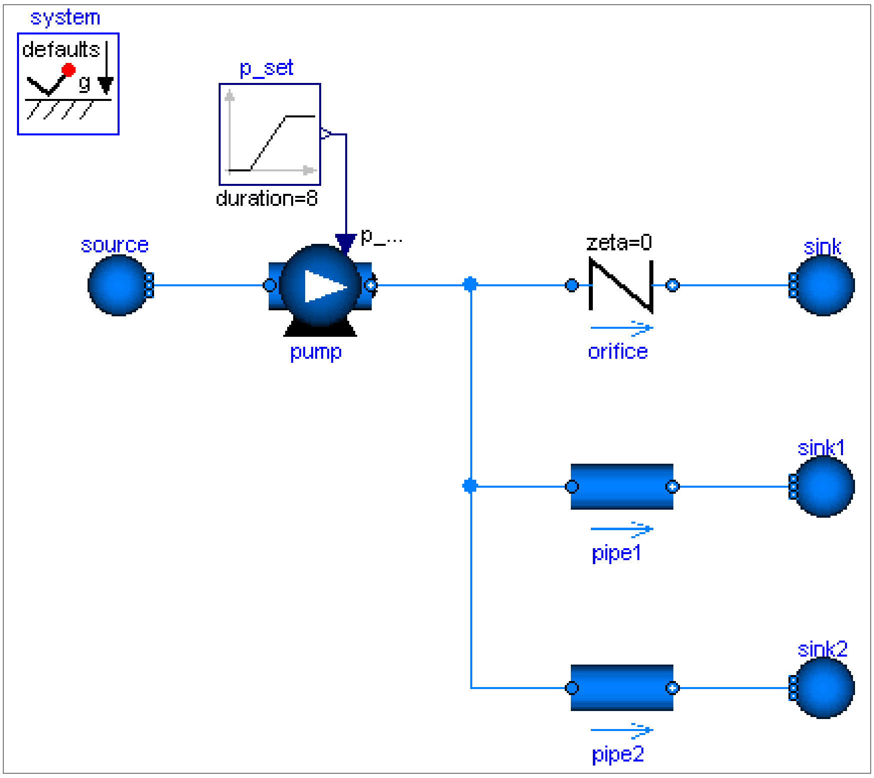
Information

A pump, an orifice and two pipes are parameterized with simple nominal values. Note that pipe1 and pipe2 use the flowModel NominalTurbulentFlow and NominalLaminarFlow, respectively, which do not require the specification of geometry data. Instead pathLengths_nominal are obtained internally for given nominal pressure loss and nominal mass flow rate.

The pump controls a pressure ramp from 1.9 bar to 2.1 bar. This causes an appropriate ramp on the mass flow rate of the orifice, which has a boundary pressure of 1 bar. Flow reversal occurs in the pipes, which have a boundary pressure of 2 bar. The Command plotResults can be used to see the pump speed N, which is controlled ideally to obtain the pressure ramp. Moreover the internally obtained nominal design values that fulfill the nominal operating conditions as well as the Reynolds number, m_flows_turbulent, and dps_fg_turbulent are plotted.

Note that the large value for pipe2.flowModel.pathLengths_nominal[1] is only meaningful under the made assumption of laminar flow, which is hardly possible for a real pipe.

Once the geometries have been designed, the NominalTurbulentPipeFlow correlations can easily be replaced with TurbulentPipeFlow or DetailedPipeFlow correlations. Similarily the ControlledPump can be replaced with a PrescribedPump to investigate a real controller or with a Pump with rotational shaft to investigate inertia effects.



Extends from Modelica.Icons.Example (Icon for an example model).

Parameters

Type	Name	Description
replaceable package Medium		

Connectors

Type	Name	Description
replaceable package Medium		

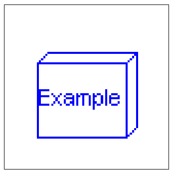
Examples.Explanatory

A set of examples illustrating when special attention has to be paid

Package Content

Name	Description
MomentumBalanceFittings	Illustrating a case in which kinetic terms play a major role in the momentum balance

Examples.Explanatory.MomentumBalanceFittings



Illustrating a case in which kinetic terms play a major role in the momentum balance

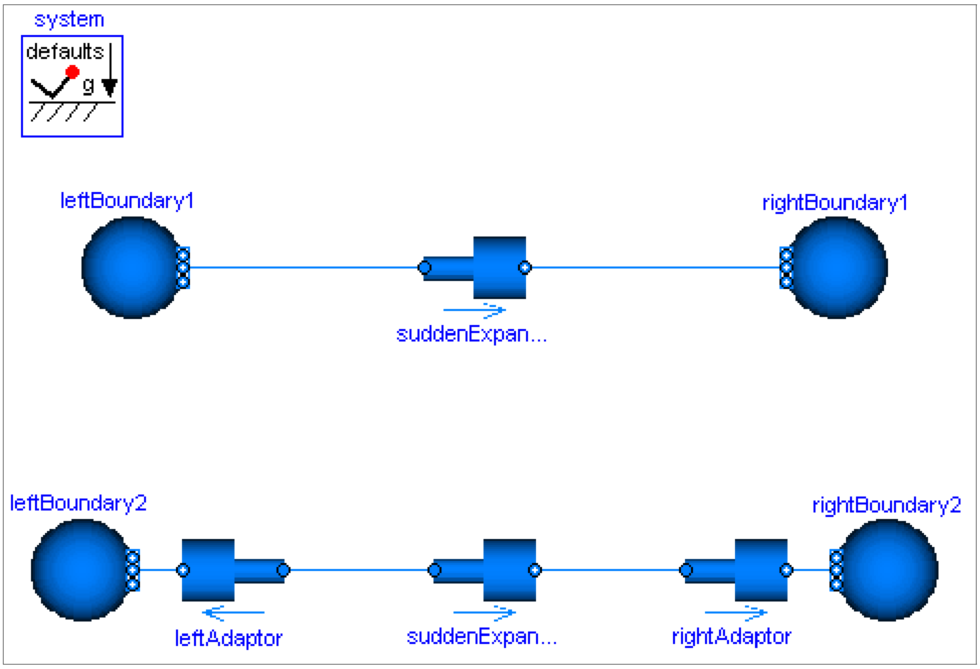
Information

This example shows the use of a sudden expansion / contraction model, which is connected to two boundary conditions prescribing static pressure. Notice that the prescribed static pressure on the right boundary is higher than on the left one. Still, the fluid flows from left to right.

The reason for this is that the boundary conditions model infinite reservoirs with an infinite diameter and thus zero flow velocity. The sudden expansion model does however have two ends with finite diameters, and, as explained in the Overview of the Users' Guide, the momentum balance is not fulfilled exactly for this type of connections. Using a simple `connect()`-statement, the difference of the kinetic terms is neglected, which is not reasonable in the present model: At the left boundary condition it is zero, and on the left side of the sudden expansion it has a non-zero value. It is not reasonable to neglect it in the shown model, because there is little friction and therefore these kinetic effects dominate. Consequently, only modelling these effects explicitly leads to the correct results.

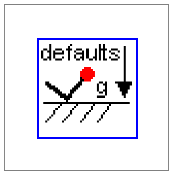
To do so, two additional sudden expansions / contractions are included in the model. The diameter is set to `inf` close to the boundaries and the proper values close to the original model. These additional components now introduce *exact* momentum balances and the results are as expected.

The total pressures offer an additional perspective on the model. After setting the parameter `show_totalPressures` on the Advanced tab of the `AbruptAdaptor`s to `true`, the total pressures are included in said models and may be plotted. This allows to confirm that the **total** pressure *always* reduces along the flow direction, even in the upper model.



Extends from Modelica.Icons.Example (Icon for an example model).

Modelica_Fluid.System



System properties and default values (ambient, flow direction, initialization)

Information

A system component is needed in each fluid model to provide system-wide settings, such as ambient conditions and overall modeling assumptions. The system settings are propagated to the fluid models using the inner/outer mechanism.

A model should never directly use system parameters. Instead a local parameter should be declared, which uses the global setting as default. The only exception currently made is the gravity system.g.

Parameters

Type	Name	Description
Environment		
AbsolutePressure	p_ambient	Default ambient pressure [Pa]
Temperature	T_ambient	Default ambient temperature [K]
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= false to restrict to design flow direction (port_a -> port_b)
Dynamics		
Dynamics	energyDynamics	Default formulation of energy balances
Dynamics	massDynamics	Default formulation of mass balances
Dynamics	momentumDynamics	Default formulation of momentum balances, if options available
Initialization		

MassFlowRate	m_flow_start	Default start value for mass flow rates [kg/s]
AbsolutePressure	p_start	Default start value for pressures [Pa]
Temperature	T_start	Default start value for temperatures [K]
Advanced		
MassFlowRate	m_flow_small	Default small laminar mass flow rate for regularization of zero flow [kg/s]
AbsolutePressure	dp_small	Default small pressure drop for regularization of laminar and zero flow [Pa]

Modelica_Fluid.Vessels

Devices for storing fluid

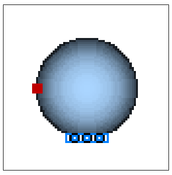
Information

Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
 ClosedVolume	Volume of fixed size, closed to the ambient, with inlet/outlet ports
 OpenTank	Simple tank with inlet/outlet ports
 BaseClasses	Base classes used in the Vessels package (only of interest to build new component models)

Vessels.ClosedVolume



Volume of fixed size, closed to the ambient, with inlet/outlet ports

Information

Ideally mixed volume of constant size with two fluid ports and one medium model. The flow properties are computed from the upstream quantities, pressures are equal in both nodes and the medium model if `use_portsData=false`. Heat transfer through a thermal port is possible, it equals zero if the port remains unconnected. A spherical shape is assumed for the heat transfer area, with V=4/3*pi*r^3, A=4*pi*r^2. Ideal heat transfer is assumed per default; the thermal port temperature is equal to the medium temperature.

If `use_portsData=true`, the port pressures represent the pressures just after the outlet (or just before the inlet) in the attached pipe. The hydraulic resistances `portsData.zeta_in` and `portsData.zeta_out` determine the dissipative pressure drop between volume and port depending on the direction of mass flow. See VesselPortsData and *[Idelchik, Handbook of Hydraulic Resistance, 2004]*.

Extends from Vessels.BaseClasses.PartialLumpedVessel (Lumped volume with a vector of fluid ports and replaceable heat transfer model).

Parameters

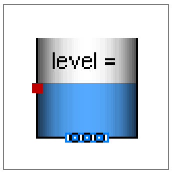
Type	Name	Description
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
Volume	V	Volume [m3]

Ports		
Boolean	use_portsData	= false to neglect pressure loss and kinetic energy
VesselPortsData	portsData[nPorts]	Data of inlet/outlet ports
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
replaceable model	HeatTransfer	Wall heat transfer
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Port properties		
MassFlowRate	m_flow_small	Regularization range at zero mass flow rate [kg/s]

Connectors

Type	Name	Description
VesselFluidPorts_b	ports[nPorts]	Fluid inlets and outlets
HeatPort_a	heatPort	

Vessels.OpenTank



Simple tank with inlet/outlet ports

Information

Model of a tank that is open to the ambient at the fixed pressure `p_ambient`.

The vector of connectors **ports** represents fluid ports at configurable heights, relative to the bottom of tank. Fluid can flow either out of or in to each port.

The following assumptions are made:

- The tank is filled with a single or multiple-substance medium having a density higher than the density of the ambient medium.
- The fluid has uniform density, temperature and mass fractions
- No liquid is leaving the tank through the open top; the simulation breaks with an assertion if the liquid level growths over the height.

The port pressures represent the pressures just after the outlet (or just before the inlet) in the attached pipe. The hydraulic resistances `portsData.zeta_in` and `portsData.zeta_out` determine the dissipative pressure drop between tank and port depending on the direction of mass flow. See VesselPortsData and *[Idelchik, Handbook of Hydraulic Resistance, 2004]*.

With the setting `use_portsData=false`, the port pressure represents the static head at the height of the respective port. The relationship between pressure drop and mass flow rate at the port must then be provided by connected components; Heights of ports as well as kinetic and potential energy of fluid enering or leaving are not taken into account anymore.

Extends from Vessels.BaseClasses.PartialLumpedVessel (Lumped volume with a vector of fluid ports and replaceable heat transfer model).

Parameters

Type	Name	Description
Height	height	Height of tank [m]
Area	crossArea	Area of tank [m2]
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
Ports		
Boolean	use_portsData	= false to neglect pressure loss and kinetic energy
VesselPortsData	portsData[nPorts]	Data of inlet/outlet ports
Assumptions		
Ambient		
AbsolutePressure	p_ambient	Tank surface pressure [Pa]
Temperature	T_ambient	Tank surface Temperature [K]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
replaceable model HeatTransfer		Wall heat transfer
Initialization		
Height	level_start	Start value of tank level [m]
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Port properties		
MassFlowRate	m_flow_small	Regularization range at zero mass flow rate [kg/s]

Connectors

Type	Name	Description
VesselFluidPorts_b	ports[nPorts]	Fluid inlets and outlets
HeatPort_a	heatPort	

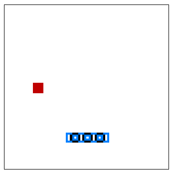
Vessels.BaseClasses

Base classes used in the Vessels package (only of interest to build new component models)

Package Content

Name	Description
PartialLumpedVessel	Lumped volume with a vector of fluid ports and replaceable heat transfer model
HeatTransfer	HeatTransfer models for vessels
VesselPortsData	Data to describe inlet/outlet ports at vessels: diameter -- Inner (hydraulic) diameter of inlet/outlet port height -- Height over the bottom of the vessel zeta_out -- Hydraulic resistance out of vessel, default 0.5 for small diameter mounted flush with the wall zeta_in -- Hydraulic resistance into vessel, default 1.04 for small diameter mounted flush with the wall
VesselFluidPorts_a	Fluid connector with filled, large icon to be used for horizontally aligned vectors of FluidPorts (vector dimensions must be added after dragging)
VesselFluidPorts_b	Fluid connector with outlined, large icon to be used for horizontally aligned vectors of FluidPorts (vector dimensions must be added after dragging)

Vessels.BaseClasses.PartialLumpedVessel



Lumped volume with a vector of fluid ports and replaceable heat transfer model

Information

This base class extends PartialLumpedVolume with a vector of fluid ports and a replaceable wall HeatTransfer model.

The following modeling assumption are made:

- homogenous medium, i.e. phase seperation is not taken into account,
- no kinetic energy in the fluid, i.e. kinetic energy dissipates into the internal energy,
- pressure loss definitions at vessel ports assume incompressible fluid,
- outflow of ambient media is prevented at each port assuming check valve behavior. If `fluidlevel < portsData_height[i]` and `ports[i].p < vessel_ps_static[i]` massflow at the port is set to 0.

Each port has a (hydraulic) diameter and a height above the bottom of the vessel, which can be configured using the **`portsData`** record. Alternatively the impact of port geometries can be neglected with `use_portsData=false`. This might be useful for early design studies. Note that this means to assume an infinite port diameter at the bottom of the vessel. Pressure drops and heights of the ports as well as kinetic and potential energy fluid entering or leaving the vessel are neglected then.

The following variables need to be defined by an extending model:

- `input fluidVolume`, the volume of the fluid in the vessel,
- `vessel_ps_static[nPorts]`, the static pressures inside the vessel at the height of the corresponding ports, at zero flow velocity, and
- `Wb_flow`, work term of the energy balance, e.g. p*der(V) if the volume is not constant or stirrer power.

An extending model should define:

- `parameter vesselArea` (default: Modelica.Constants.inf m2), the area of the vessel, to be related to cross flow areas of the ports for the consideration of dynamic pressure effects.

Optionally the fluid level may vary in the vessel, which effects the flow through the ports at configurable `portsData_height[nPorts]`. This is why an extending model with varying fluid level needs to define:

- `input fluidLevel (default: 0m)`, the level the fluid in the vessel, and
- `parameter fluidLevel_max (default: 1m)`, the maximum level that must not be exceeded. Ports at or above fluidLevel_max can only receive inflow.

An extending model should not access the `portsData` record defined in the configuration dialog, as an access to `portsData` may fail for `use_portsData=false` or `nPorts=0`. Instead the predefined variables

- `portsData_diameter[nPorts]`,
- `portsData_height[nPorts]`,
- `portsData_zeta_in[nPorts]`, and
- `portsData_zeta_out[nPorts]`

should be used if these values are needed.

Extends from Interfaces.PartialLumpedVolume (Lumped volume with mass and energy balance).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Ports		
Boolean	use_portsData	= false to neglect pressure loss and kinetic energy
VesselPortsData	portsData[nPorts]	Data of inlet/outlet ports
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Port properties		
MassFlowRate	m_flow_small	Regularization range at zero mass flow rate [kg/s]

Connectors

Type	Name	Description
VesselFluidPorts_b	ports[nPorts]	Fluid inlets and outlets
HeatPort_a	heatPort	

Vessels.BaseClasses.HeatTransfer

HeatTransfer models for vessels

Information

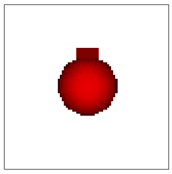
Heat transfer correlations for pipe models

Package Content

Name	Description
PartialVesselHeatTransfer	Base class for vessel heat transfer models
IdealHeatTransfer	IdealHeatTransfer: Ideal heat transfer without thermal resistance
ConstantHeatTransfer	ConstantHeatTransfer: Constant heat transfer coefficient

---

Vessels.BaseClasses.HeatTransfer.PartialVesselHeatTransfer



Base class for vessel heat transfer models

Information

Base class for vessel heat transfer models.

Extends from Interfaces.PartialHeatTransfer (Common interface for heat transfer models).

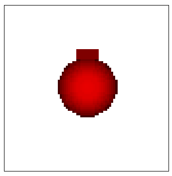
Parameters

Type	Name	Description
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Vessels.BaseClasses.HeatTransfer.IdealHeatTransfer



IdealHeatTransfer: Ideal heat transfer without thermal resistance

Information

Ideal heat transfer without thermal resistance.

Extends from PartialVesselHeatTransfer (Base class for vessel heat transfer models).

Parameters

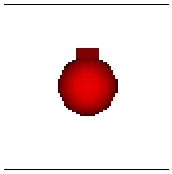
Type	Name	Description
Ambient		

CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Vessels.BaseClasses.HeatTransfer.ConstantHeatTransfer



ConstantHeatTransfer: Constant heat transfer coefficient

Information

Simple heat transfer correlation with constant heat transfer coefficient.

Extends from PartialVesselHeatTransfer (Base class for vessel heat transfer models).

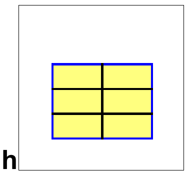
Parameters

Type	Name	Description
CoefficientOfHeatTransfer	alpha0	constant heat transfer coefficient [W/(m2.K)]
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Vessels.BaseClasses.VesselPortsData



Data to describe inlet/outlet ports at vessels: diameter -- Inner (hydraulic) diameter of inlet/outlet port height -- Height over the bottom of the vessel zeta_out -- Hydraulic resistance out of vessel, default 0.5 for small diameter mounted flush with the wall zeta_in -- Hydraulic resistance into vessel, default 1.04 for small diameter mounted flush with the wall

Information

Vessel Port Data

This record describes the **ports** of a **vessel**. The variables in it are mostly self-explanatory (see list below);

only the ζ loss factors are discussed further. All data is quoted from Idelchik (1994).

Outlet Coefficients

If a **straight pipe with constant cross section is mounted flush with the wall**, its outlet pressure loss coefficient will be `ζ = 0.5` (Idelchik, p. 160, Diagram 3-1, paragraph 2).

If a **straight pipe with constant cross section is mounted into a vessel such that the entrance into it is at a distance** `b` from the wall (inside) the following table can be used. Herein, δ is the tube wall thickness (Idelchik, p. 160, Diagram 3-1, paragraph 1).

		b / D_hyd				
		0.000	**0.005**	**0.020**	**0.100**	**0.500-∞**
	0.000	0.50	0.63	0.73	0.86	1.00
	0.008	0.50	0.55	0.62	0.74	0.88
δ / D_hyd	**0.016**	0.50	0.51	0.55	0.64	0.77
	0.024	0.50	0.50	0.52	0.58	0.68
	0.040	0.50	0.50	0.51	0.51	0.54

Pressure loss coefficients for outlets, entrance at a distance from wall

If a **straight pipe with a circular bellmouth inlet (collector) without baffle is mounted flush with the wall** then its pressure loss coefficient can be established from the following table. Herein, r is the radius of the bellmouth inlet surface (Idelchik, p. 164 f., Diagram 3-4, paragraph b)

	r / D_hyd					
	0.01	**0.03**	**0.05**	**0.08**	**0.16**	**≥0.20**
ζ	0.44	0.31	0.22	0.15	0.06	0.03

Pressure loss coefficients for outlets, bellmouth flush with wall

If a **straight pipe with a circular bellmouth inlet (collector) without baffle is mounted at a distance from a wall** then its pressure loss coefficient can be established from the following table. Herein, r is the radius of the bellmouth inlet surface (Idelchik, p. 164 f., Diagram 3-4, paragraph a)

	r / D_hyd					
	0.01	**0.03**	**0.05**	**0.08**	**0.16**	**≥0.20**
ζ	0.87	0.61	0.40	0.20	0.06	0.03

Pressure loss coefficients for outlets, bellmouth at a distance of wall

Inlet Coefficients

If a **straight pipe with constant circular cross section is mounted flush with the wall**, its vessel inlet pressure loss coefficient will be according to the following table (Idelchik, p. 209 f., Diagram 4-2 with `A_port/A_vessel = 0` and Idelchik, p. 640, Diagram 11-1, graph a). According to the text, `m = 9` is appropriate for fully developed turbulent flow.

	m					
	1.0	**2.0**	**3.0**	**4.0**	**7.0**	**9.0**
ζ	2.70	1.50	1.25	1.15	1.06	1.04

Pressure loss coefficients for inlets, circular tube flush with wall

For larger port diameters, relative to the area of the vessel, the inlet pressure loss coefficient will be according to the following table (Idelchik, p. 209 f., Diagram 4-2 with `m = 7`).

	A_port / A_vessel					
	0.0	**0.1**	**0.2**	**0.4**	**0.6**	**0.8**
ζ	1.04	0.84	0.67	0.39	0.18	0.06

Pressure loss coefficients for inlets, circular tube flush with wall

References

Idelchik I.E. (1994):
    Handbook of Hydraulic Resistance. 3rd edition, Begell House, ISBN 0-8493-9908-4

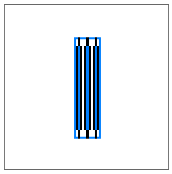
Extends from Modelica.Icons.Record (Icon for a record).

Parameters

Type	Name	Description
Diameter	diameter	Inner (hydraulic) diameter of inlet/outlet port [m]
Height	height	Height over the bottom of the vessel [m]
Real	zeta_out	Hydraulic resistance out of vessel, default 0.5 for small diameter mounted flush with the wall
Real	zeta_in	Hydraulic resistance into vessel, default 1.04 for small diameter mounted flush with the wall

---

Vessels.BaseClasses.VesselFluidPorts_a



Fluid connector with filled, large icon to be used for horizontally aligned vectors of FluidPorts (vector dimensions must be added after dragging)

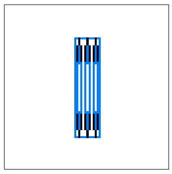
Parameters

Type	Name	Description
replaceable package Medium		Medium model

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Vessels.BaseClasses.VesselFluidPorts_b



Fluid connector with outlined, large icon to be used for horizontally aligned vectors of FluidPorts (vector dimensions must be added after dragging)

Parameters

Type	Name	Description
replaceable package Medium		Medium model

Contents

Type	Name	Description

flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Modelica_Fluid.Pipes

Devices for conveying fluid

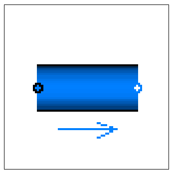
Information

Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
StaticPipe	Basic pipe flow model without storage of mass or energy
DynamicPipe	Dynamic pipe model with storage of mass and energy
BaseClasses	Base classes used in the Pipes package (only of interest to build new component models)

Pipes.StaticPipe



Basic pipe flow model without storage of mass or energy

Information

Model of a straight pipe with constant cross section and with steady-state mass, momentum and energy balances, i.e. the model does not store mass or energy. There exist two thermodynamic states, one at each fluid port. The momentum balance is formulated for the two states, taking into account momentum flows, friction and gravity. The same result can be obtained by using DynamicPipe with steady-state dynamic settings. The intended use is to provide simple connections of vessels or other devices with storage, as it is done in:

- Examples.Tanks.EmptyTanks
- Examples.InverseParameterization.

Numerical Issues

With the stream connectors the thermodynamic states on the ports are generally defined by models with storage or by sources placed upstream and downstream of the static pipe. Other non storage components in the flow path may yield to state transformation. Note that this generally leads to nonlinear equation systems if multiple static pipes, or other flow models without storage, are directly connected.

Extends from Pipes.BaseClasses.PartialStraightPipe (Base class for straight pipe models).

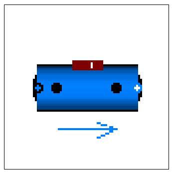
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Geometry		
Real	nParallel	Number of identical parallel pipes
Length	length	Length [m]
Boolean	isCircular	= true if cross sectional area is circular
Diameter	diameter	Diameter of circular pipe [m]
Area	crossArea	Inner cross section area [m2]
Length	perimeter	Inner perimeter [m]
Height	roughness	Average height of surface asperities (default: smooth steel pipe) [m]
Static head		
Length	height_ab	Height(port_b) - Height(port_a) [m]
Pressure loss		
replaceable model FlowModel		Wall friction, gravity, momentum flow
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Initialization		
AbsolutePressure	p_a_start	Start value of pressure at port a [Pa]
AbsolutePressure	p_b_start	Start value of pressure at port b [Pa]
MassFlowRate	m_flow_start	Start value for mass flow rate [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Pipes.DynamicPipe



Dynamic pipe model with storage of mass and energy

Information

Model of a straight pipe with distributed mass, energy and momentum balances. It provides the complete balance equations for one-dimensional fluid flow as formulated in UsersGuide.ComponentDefinition.BalanceEquations.

The partial differential equations are treated with the finite volume method and a staggered grid scheme for momentum balances. The pipe is split into nNodes equally spaced segments along the flow path. The default value is nNodes=2. This results in two lumped mass and energy balances and one lumped momentum balance across the dynamic pipe.

Note that this generally leads to high-index DAEs for pressure states if dynamic pipes are directly connected to each other, or generally to models with storage exposing a thermodynamic state through the port. This may not be valid if the dynamic pipe is connected to a model with non-differentiable pressure, like a Sources.Boundary_pT with prescribed jumping pressure. The **`modelStructure`** can be configured as appropriate in such situations, in order to place a momentum balance between a pressure state of the pipe and a non-differentiable boundary condition.

The default **`modelStructure`** is `av_vb` (see Advanced tab). The simplest possible alternative symetric configuration, avoiding potential high-index DAEs at the cost of the potential introduction of nonlinear equation systems, is obtained with the setting `nNodes=1, modelStructure=a_v_b`. Depending on the configured model structure, the first and the last pipe segment, or the flow path length of the first and the last momentum balance, are of half size. See the documentation of the base class Pipes.BaseClasses.PartialTwoPortFlow, also covering asymmetric configurations.

The **`HeatTransfer`** component specifies the source term `Qb_flows` of the energy balance. The default component uses a constant coefficient for the heat transfer between the bulk flow and the segment boundaries exposed through the `heatPorts`. The `HeatTransfer` model is replaceable and can be exchanged with any model extended from BaseClasses.HeatTransfer.PartialFlowHeatTransfer.

The intended use is for complex networks of pipes and other flow devices, like valves. See e.g.

- Examples.BranchingDynamicPipes, or
- Examples.IncompressibleFluidNetwork.

Extends from Pipes.BaseClasses.PartialStraightPipe (Base class for straight pipe models), BaseClasses.PartialTwoPortFlow (Base class for distributed flow models).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Geometry		
Real	nParallel	Number of identical parallel pipes
Length	length	Length [m]
Boolean	isCircular	= true if cross sectional area is circular
Diameter	diameter	Diameter of circular pipe [m]
Area	crossArea	Inner cross section area [m2]
Length	perimeter	Inner perimeter [m]
Height	roughness	Average height of surface asperities (default: smooth steel pipe) [m]
Length	lengths[n]	lengths of flow segments [m]
Area	crossAreas[n]	cross flow areas of flow segments [m2]
Length	dimensions[n]	hydraulic diameters of flow segments [m]
Height	roughnesses[n]	Average heights of surface asperities [m]
Static head		
Length	height_ab	Height(port_b) - Height(port_a) [m]
Length	dheights[n]	Differences in heigths of flow segments [m]
Pressure loss		
replaceable model FlowModel		Wall friction, gravity, momentum flow
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	energyDynamics	Formulation of energy balances
Dynamics	massDynamics	Formulation of mass balances
Dynamics	momentumDynamics	Formulation of momentum balances
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
Initialization		
AbsolutePressure	p_a_start	Start value of pressure at port a [Pa]

AbsolutePressure	p_b_start	Start value of pressure at port b [Pa]
Boolean	use_T_start	Use T_start if true, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
MassFlowRate	m_flow_start	Start value for mass flow rate [kg/s]
Advanced		
Integer	nNodes	Number of discrete flow volumes
ModelStructure	modelStructure	Determines whether flow or volume models are present at the ports
Boolean	useLumpedPressure	=true to lump pressure states together
Boolean	useInnerPortProperties	=true to take port properties for flow models from internal control volumes

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
HeatPorts_a	heatPorts[nNodes]	

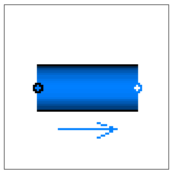
Pipes.BaseClasses

Base classes used in the Pipes package (only of interest to build new component models)

Package Content

Name	Description
PartialStraightPipe	Base class for straight pipe models
PartialTwoPortFlow	Base class for distributed flow models
FlowModels	Flow models for pipes, including wall friction, static head and momentum flow
HeatTransfer	Heat transfer for flow models
CharacteristicNumbers	Functions to compute characteristic numbers
WallFriction	Different variants for pressure drops due to pipe wall friction

Pipes.BaseClasses.PartialStraightPipe



Base class for straight pipe models

Information

Base class for one dimensional flow models. It specializes a PartialTwoPort with a parameter interface and icon graphics.

Extends from Interfaces.PartialTwoPort (Partial component with two ports).

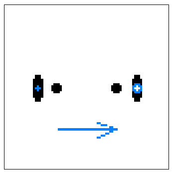
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Geometry		
Real	nParallel	Number of identical parallel pipes
Length	length	Length [m]
Boolean	isCircular	= true if cross sectional area is circular
Diameter	diameter	Diameter of circular pipe [m]
Area	crossArea	Inner cross section area [m2]
Length	perimeter	Inner perimeter [m]
Height	roughness	Average height of surface asperities (default: smooth steel pipe) [m]
Static head		
Length	height_ab	Height(port_b) - Height(port_a) [m]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Pipes.BaseClasses.PartialTwoPortFlow



Base class for distributed flow models

Information

Base class for distributed flow models. The total volume is split into nNodes segments along the flow path. The default value is nNodes=2.

Mass and Energy balances

The mass and energy balances are inherited from Interfaces.PartialDistributedVolume. One total mass and one energy balance is formed across each segment according to the finite volume approach. Substance mass balances are added if the medium contains more than one component.

An extending model needs to define the geometry and the difference in heights between the flow segments (static head). Moreover it needs to define two vectors of source terms for the distributed energy balance:

- `Qb_flows[nNodes]`, the heat flow source terms, e.g. conductive heat flows across segment boundaries, and
- `Wb_flows[nNodes]`, the work source terms.

Momentum balance

The momentum balance is determined by the `FlowModel` component, which can be replaced with any model extended from BaseClasses.FlowModels.PartialStaggeredFlowModel. The default setting is DetailedPipeFlow. This considers

- pressure drop due to friction and other dissipative losses, and
- gravity effects for non-horizontal devices.
- variation of flow velocity along the flow path, which occur due to changes in the cross sectional area or the fluid density, provided that `flowModel.use_Ib_flows` is true.

Model Structure

The momentum balances are formulated across the segment boundaries along the flow path according to the staggered grid approach. The configurable **`modelStructure`** determines the formulation of the boundary conditions at `port_a` and `port_b`. The options include (default: av_vb):

- `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 momentum balances, one between nth volume and `port_b`, potential pressure state at `port_a`
- `a_vb`: Unsymmetric setting with nNodes momentum balance, one between first volume and `port_a`, potential pressure state at `port_b`

When connecting two components, e.g. two pipes, the momentum balance across the connection point reduces to

```
pipe1.port_b.p = pipe2.port_a.p
```

This is only true if the flow velocity remains the same on each side of the connection. Consider using a fitting for any significant change in diameter or fluid density, if the resulting effects, such as change in kinetic energy, cannot be neglected. This also allows for taking into account friction losses with respect to the actual geometry of the connection point.

Extends from Interfaces.PartialTwoPort (Partial component with two ports), Interfaces.PartialDistributedVolume (Base class for distributed volume models).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Integer	n	Number of discrete volumes
Volume	fluidVolumes[n]	Discretized volume, determine in inheriting class [m3]
Geometry		
Real	nParallel	Number of identical parallel flow devices
Length	lengths[n]	lengths of flow segments [m]
Area	crossAreas[n]	cross flow areas of flow segments [m2]
Length	dimensions[n]	hydraulic diameters of flow segments [m]
Height	roughnesses[n]	Average heights of surface asperities [m]
Static head		
Length	dheights[n]	Differences in heigths of flow segments [m]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	energyDynamics	Formulation of energy balances
Dynamics	massDynamics	Formulation of mass balances
Dynamics	momentumDynamics	Formulation of momentum balances

Initialization		
AbsolutePressure	p_a_start	Start value of pressure at port a [Pa]
AbsolutePressure	p_b_start	Start value of pressure at port b [Pa]
Boolean	use_T_start	Use T_start if true, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
MassFlowRate	m_flow_start	Start value for mass flow rate [kg/s]
Advanced		
Integer	nNodes	Number of discrete flow volumes
ModelStructure	modelStructure	Determines whether flow or volume models are present at the ports
Boolean	useLumpedPressure	=true to lump pressure states together
Boolean	useInnerPortProperties	=true to take port properties for flow models from internal control volumes

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

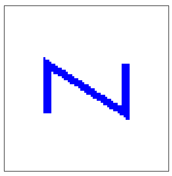
Pipes.BaseClasses.FlowModels

Flow models for pipes, including wall friction, static head and momentum flow

Package Content

Name	Description
PartialStaggeredFlowModel	Base class for momentum balances in flow models
NominalLaminarFlow	NominalLaminarFlow: Linear pressure loss for nominal values
PartialGenericPipeFlow	GenericPipeFlow: Pipe flow pressure loss and gravity with replaceable WallFriction package
NominalTurbulentPipeFlow	NominalTurbulentPipeFlow: Quadratic turbulent pressure loss for nominal values
TurbulentPipeFlow	TurbulentPipeFlow: Pipe wall friction in the quadratic turbulent regime (simple characteristic, mu not used)
DetailedPipeFlow	DetailedPipeFlow: Pipe wall friction in the laminar and turbulent regime (detailed characteristic)

Pipes.BaseClasses.FlowModels.PartialStaggeredFlowModel



Base class for momentum balances in flow models

Information

This paratial model defines a common interface for `m=n-1` flow models between `n` device segments. The flow models provide a steady-state or dynamic momentum balance using an upwind discretization scheme per default. Extending models must add pressure loss terms for friction and gravity.

The fluid is specified in the interface with the thermodynamic `states[n]` for a given `Medium` model. The geometry is specified with the `pathLengths[n-1]` between the device segments as well as with the `crossAreas[n]` and the `roughnesses[n]` of the device segments. Moreover the fluid flow is characterized for different types of devices by the characteristic `dimensions[n]` and the average velocities `vs[n]` of fluid flow in the device segments. See Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber for examplary definitions.

The parameter `Re_turbulent` can be specified for the least mass flow rate of the turbulent regime. It defaults to 4000, which is appropriate for pipe flow. The `m_flows_turbulent[n-1]` resulting from `Re_turbulent` can optionally be calculated together with the Reynolds numbers `Res[n]` of the device segments (`show_Res=true`).

Using the thermodynamic states[n] of the device segments, the densities rhos[n] and the dynamic viscosities mus[n] of the segments as well as the actual densities rhos_act[n-1] and the actual viscosities mus_act[n-1] of the flows are predefined in this base model. Note that no events are raised on flow reversal. This needs to be treated by an extending model, e.g. with numerical smoothing or by raising events as appropriate.

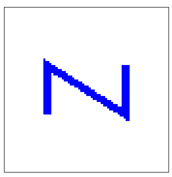
Extends from Interfaces.PartialDistributedFlow (Base class for a distributed momentum balance).

Parameters

Type	Name	Description
Integer	m	Number of flow segments
ReynoldsNumber	Re_turbulent	Start of turbulent regime, depending on type of flow device [1]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		
Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Internal Interface		
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Pipes.BaseClasses.FlowModels.NominalLaminarFlow

NominalLaminarFlow: Linear pressure loss for nominal values



Information

This model defines a simple lineaer pressure loss assuming laminar flow for specified `dp_nominal` and `m_flow_nominal`.

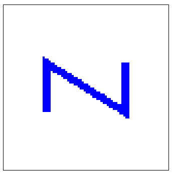
Select `show_Res = true` to analyze the actual flow and the lengths of a pipe that would fulfill the specified nominal values for given geometry parameters `crossAreas`, `dimensions` and `roughnesses`.

Extends from Pipes.BaseClasses.FlowModels.PartialStaggeredFlowModel (Base class for momentum balances in flow models).

Parameters

Type	Name	Description
ReynoldsNumber	Re_turbulent	Start of turbulent regime, depending on type of flow device [1]
AbsolutePressure	dp_nominal	Nominal pressure loss [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal [kg/s]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		
Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow



GenericPipeFlow: Pipe flow pressure loss and gravity with replaceable WallFriction package

Information

This model describes pressure losses due to **wall friction** in a pipe and due to **gravity**. Correlations of different complexity and validity can be seleted via the replaceable package **WallFriction** (see parameter menu below). The details of the pipe wall friction model are described in the UsersGuide. Basically, different variants of the equation

```
dp = λ(Re,Δ)*(L/D)*ρ*v*|v|/2.
```

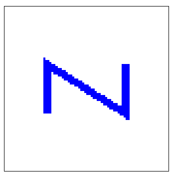
By default, the correlations are computed with media data at the actual time instant. In order to reduce non-linear equation systems, the parameters **use_mu_nominal** and **use_rho_nominal** provide the option to compute the correlations with constant media values at the desired operating point. This might speed-up the simulation and/or might give a more robust simulation.

Extends from Pipes.BaseClasses.FlowModels.PartialStaggeredFlowModel (Base class for momentum balances in flow models).

Parameters

Type	Name	Description
ReynoldsNumber	Re_turbulent	Start of turbulent regime, depending on type of flow device [1]
AbsolutePressure	dp_nominal	Nominal pressure loss (for nominal models) [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal (for nominal models) [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp), otherwise dp = f(m_flow)
AbsolutePressure	dp_small	Within regularization if |dp| < dp_small (may be wider for large discontinuities in static head) [Pa]
MassFlowRate	m_flow_small	Within regularization if |m_flows| < m_flow_small (may be wider for large discontinuities in static head) [kg/s]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		
Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Pipes.BaseClasses.FlowModels.NominalTurbulentPipeFlow



NominalTurbulentPipeFlow: Quadratic turbulent pressure loss for nominal values

Information

This model defines the pressure loss assuming turbulent flow for specified `dp_nominal` and `m_flow_nominal`. It takes into account the fluid density of each flow segment and obtaines appropriate `pathLengths_nominal` values for an inverse parameterization of the TurbulentPipeFlow model. Per default the upstream and downstream densities are averaged with the setting `useUpstreamScheme = false`, in order to avoid discontinuous `pathLengths_nominal` values in the case of flow reversal.

The geometry parameters `crossAreas`, `diameters` and `roughnesses` do not effect simulation results of this nominal pressure loss model. As the geometry is specified however, the optionally calculated Reynolds number as well as `m_flows_turbulent` and `dps_fg_turbulent` become meaningful and can be related to `m_flow_small` and `dp_small`.

Optional Variables if show_Res

Type	Name	Description
ReynoldsNumber	Res[n]	Reynolds numbers of pipe flow per flow segment
MassFlowRate	m_flows_turbulent[n-1]	mass flow rates at start of turbulent region for Re_turbulent=4000
AbsolutePressure	dps_fg_turbulent[n-1]	pressure losses due to friction and gravity corresponding to m_flows_turbulent

Extends from Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow (GenericPipeFlow: Pipe flow pressure loss and gravity with replaceable WallFriction package).

Parameters

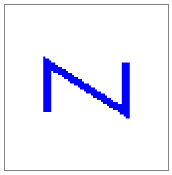
Type	Name	Description
Length	pathLengths_internal[n - 1]	pathLengths used internally; to be defined by extending class [m]
AbsolutePressure	dp_nominal	Nominal pressure loss (for nominal models) [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal (for nominal models) [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp), otherwise dp = f(m_flow)
AbsolutePressure	dp_small	Within regularization if |dp| < dp_small (may be wider for large discontinuities in static head) [Pa]
MassFlowRate	m_flow_small	Within regularization if |m_flows| < m_flow_small (may be wider for large discontinuities in static head) [kg/s]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		
Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Wall friction		
replaceable package WallFriction		Wall friction model
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]

Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Connectors

Type	Name	Description
Wall friction		
replaceable package WallFriction		Wall friction model

Pipes.BaseClasses.FlowModels.TurbulentPipeFlow



TurbulentPipeFlow: Pipe wall friction in the quadratic turbulent regime (simple characteristic, mu not used)

Information

This model defines only the quadratic turbulent regime of wall friction: dp = k*m_flow*|m_flow|, where "k" depends on density and the roughness of the pipe and is not a function of the Reynolds number. This relationship is only valid for large Reynolds numbers. The turbulent pressure loss correlation might be useful to optimize models that are only facing turbular flow.

Extends from Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow (GenericPipeFlow: Pipe flow pressure loss and gravity with replaceable WallFriction package).

Parameters

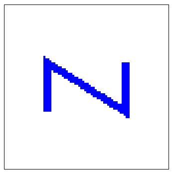
Type	Name	Description
Length	pathLengths_internal[n - 1]	pathLengths used internally; to be defined by extending class [m]
AbsolutePressure	dp_nominal	Nominal pressure loss (for nominal models) [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal (for nominal models) [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp), otherwise dp = f(m_flow)
AbsolutePressure	dp_small	Within regularization if |dp| < dp_small (may be wider for large discontinuities in static head) [Pa]
MassFlowRate	m_flow_small	Within regularization if |m_flows| < m_flow_small (may be wider for large discontinuities in static head) [kg/s]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		
Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Wall friction		
replaceable package WallFriction		Wall friction model

Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Connectors

Type	Name	Description
Wall friction		
replaceable package WallFriction		Wall friction model

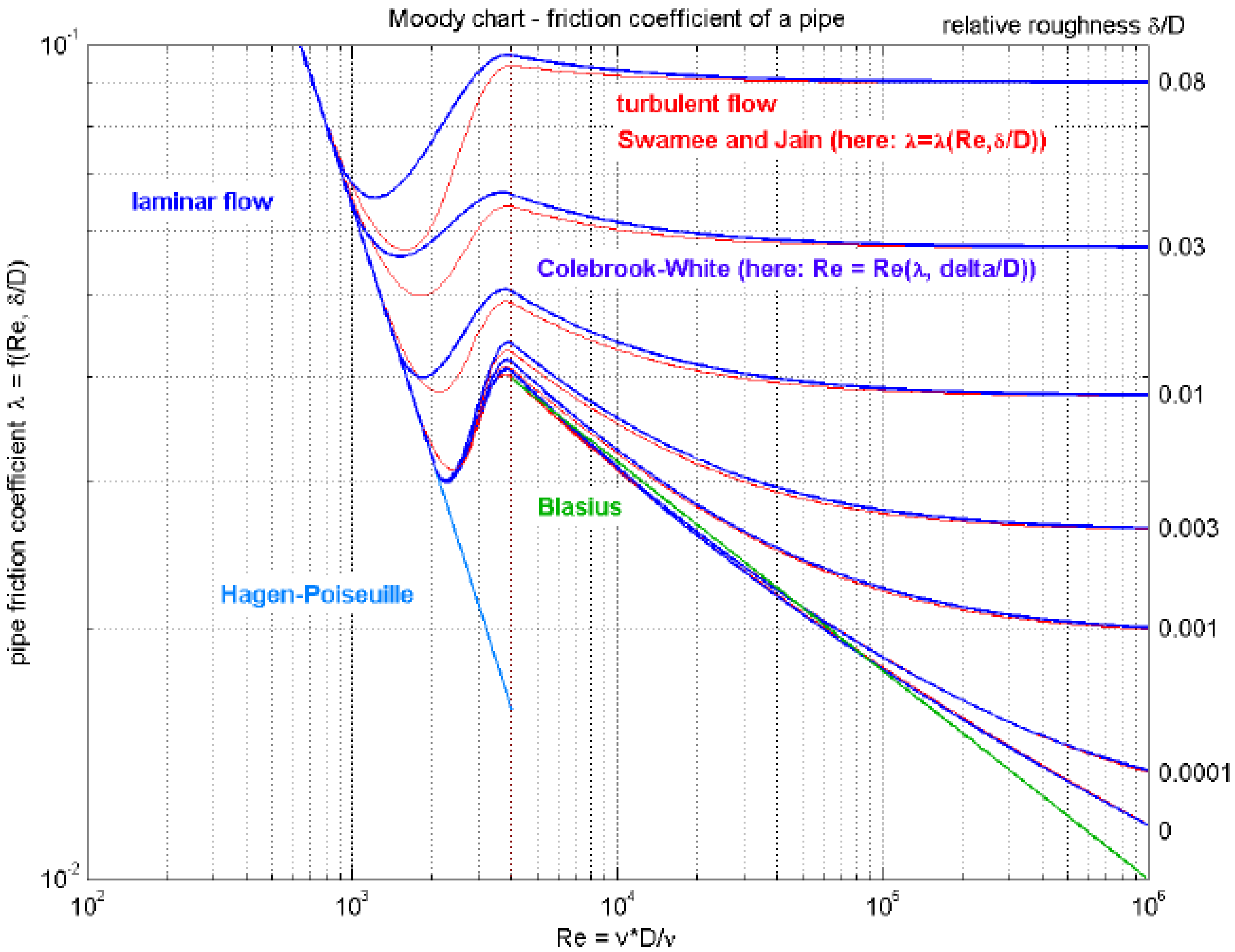
Pipes.BaseClasses.FlowModels.DetailedPipeFlow



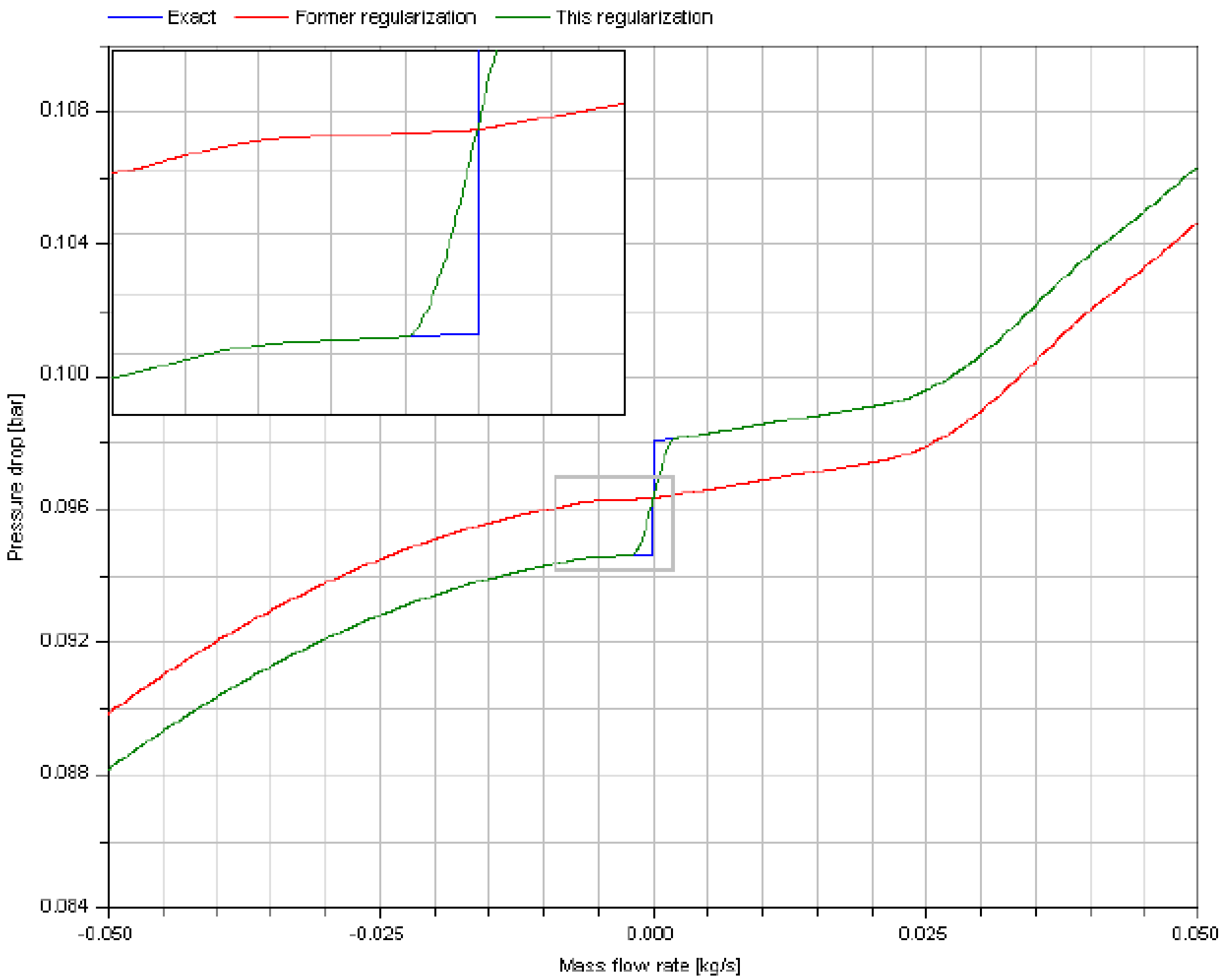
DetailedPipeFlow: Pipe wall friction in the laminar and turbulent regime (detailed characteristic)

Information

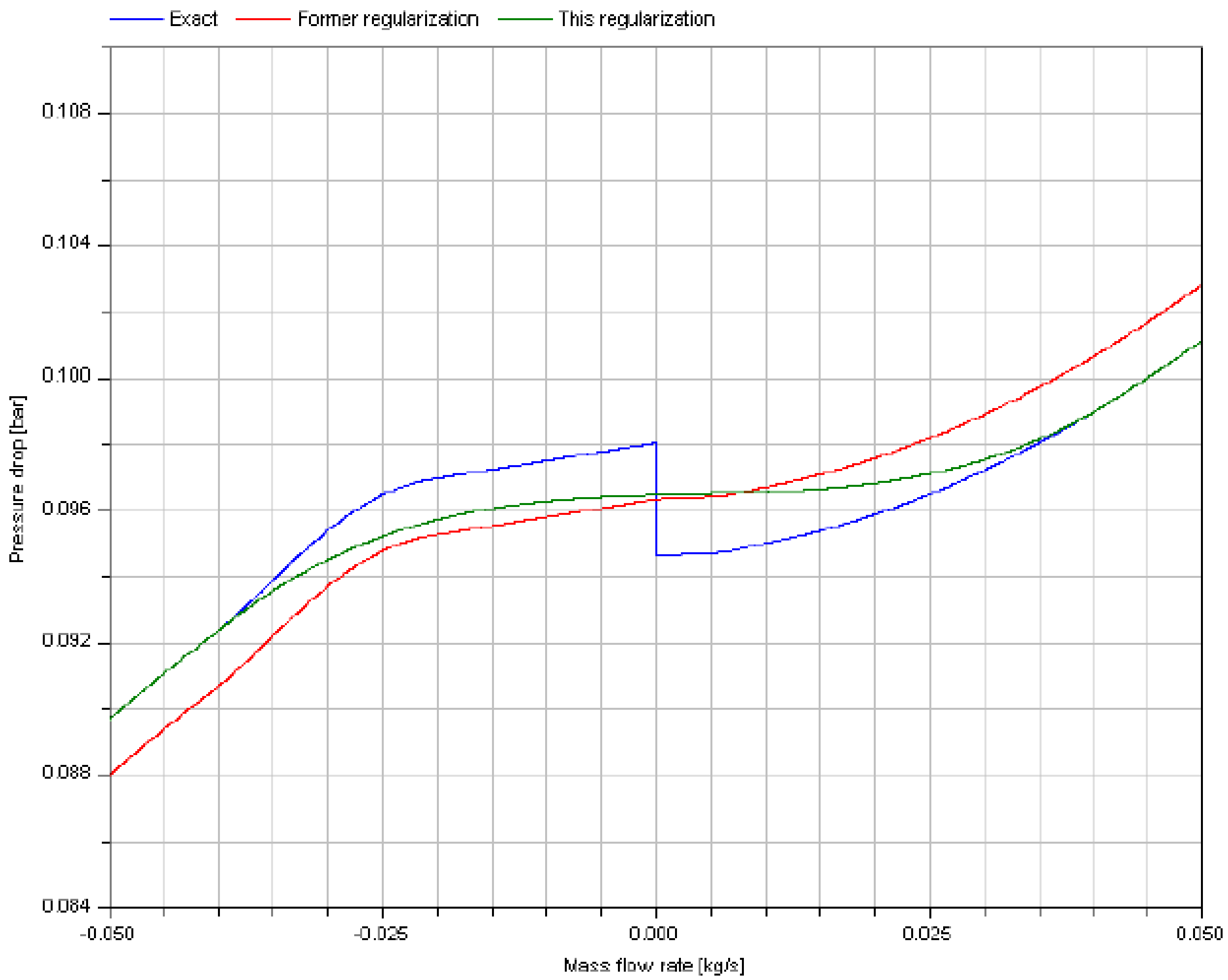
This component defines the complete regime of wall friction. The details are described in the UsersGuide. The functional relationship of the friction loss factor λ is displayed in the next figure. Function massFlowRate_dp() defines the "red curve" ("Swamee and Jain"), where as function pressureLoss_m_flow() defines the "blue curve" ("Colebrook-White"). The two functions are inverses from each other and give slightly different results in the transition region between Re = 1500 .. 4000, in order to get explicit equations without solving a non-linear equation.



Additionally to wall friction, this component properly implements static head. With respect to the latter, two cases can be distinguished. In the case shown next, the change of elevation with the path from a to b has the opposite sign of the change of density.



In the case illustrated second, the change of elevation with the path from a to b has the same sign of the change of density.



Extends from Pipes.BaseClasses.FlowModels.PartialGenericPipeFlow (GenericPipeFlow: Pipe flow pressure loss and gravity with replaceable WallFriction package).

Parameters

Type	Name	Description
Length	pathLengths_internal[n - 1]	pathLengths used internally; to be defined by extending class [m]
AbsolutePressure	dp_nominal	Nominal pressure loss (for nominal models) [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal (for nominal models) [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp), otherwise dp = f(m_flow)
AbsolutePressure	dp_small	Within regularization if |dp| < dp_small (may be wider for large discontinuities in static head) [Pa]
MassFlowRate	m_flow_small	Within regularization if |m_flows| < m_flow_small (may be wider for large discontinuities in static head) [kg/s]
Advanced		
Boolean	useUpstreamScheme	= false to average upstream and downstream properties across flow segments
Boolean	use_Ib_flows	= true to consider differences in flow of momentum through boundaries
Diagnostics		

Boolean	show_Res	= true, if Reynolds numbers are included for plotting
Wall friction		
replaceable package WallFriction		Wall friction model
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of discrete flow volumes
Geometry		
Real	nParallel	number of identical parallel flow devices
Static head		
Acceleration	g	Constant gravity acceleration [m/s2]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (states[1] -> states[n+1])
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]
AbsolutePressure	p_a_start	Start value for p[1] at design inflow [Pa]
AbsolutePressure	p_b_start	Start value for p[n+1] at design outflow [Pa]

Connectors

Type	Name	Description
Wall friction		
replaceable package WallFriction		Wall friction model

Pipes.BaseClasses.HeatTransfer

Heat transfer for flow models

Information

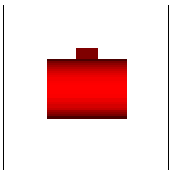
Heat transfer correlations for pipe models

Package Content

Name	Description
PartialFlowHeatTransfer	base class for any pipe heat transfer correlation
IdealFlowHeatTransfer	IdealHeatTransfer: Ideal heat transfer without thermal resistance
ConstantFlowHeatTransfer	ConstantHeatTransfer: Constant heat transfer coefficient
PartialPipeFlowHeatTransfer	Base class for pipe heat transfer correlation in terms of Nusselt number heat transfer in a circular pipe for laminar and turbulent one-phase flow
LocalPipeFlowHeatTransfer	LocalPipeFlowHeatTransfer: Laminar and turbulent forced convection in pipes, local coefficients

---

Pipes.BaseClasses.HeatTransfer.PartialFlowHeatTransfer



base class for any pipe heat transfer correlation

Information

Base class for heat transfer models of flow devices.

The geometry is specified in the interface with the `surfaceAreas[n]`, the `roughnesses[n]` and the lengths[n] along the flow path. Moreover the fluid flow is characterized for different types of devices by the characteristic `dimensions[n+1]` and the average velocities `vs[n+1]` of fluid flow. See Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber for examplary definitions.

Extends from Interfaces.PartialHeatTransfer (Common interface for heat transfer models).

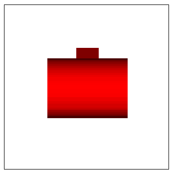
Parameters

Type	Name	Description
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation
Geometry		
Real	nParallel	number of identical parallel flow devices

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Pipes.BaseClasses.HeatTransfer.IdealFlowHeatTransfer



IdealHeatTransfer: Ideal heat transfer without thermal resistance

Information

Ideal heat transfer without thermal resistance.

Extends from PartialFlowHeatTransfer (base class for any pipe heat transfer correlation).

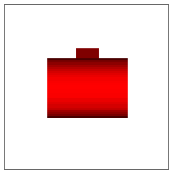
Parameters

Type	Name	Description
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation
Geometry		
Real	nParallel	number of identical parallel flow devices

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Pipes.BaseClasses.HeatTransfer.ConstantFlowHeatTransfer



ConstantHeatTransfer: Constant heat transfer coefficient

Information

Simple heat transfer correlation with constant heat transfer coefficient, used as default component in

Extends from PartialFlowHeatTransfer (base class for any pipe heat transfer correlation).

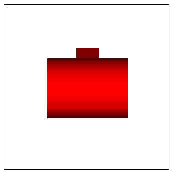
Parameters

Type	Name	Description
CoefficientOfHeatTransfer	alpha0	heat transfer coefficient [W/(m2.K)]
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation
Geometry		
Real	nParallel	number of identical parallel flow devices

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Pipes.BaseClasses.HeatTransfer.PartialPipeFlowHeatTransfer



Base class for pipe heat transfer correlation in terms of Nusselt number heat transfer in a circular pipe for laminar and turbulent one-phase flow

Information

Base class for heat transfer models that are expressed in terms of the Nusselt number and which can be used in distributed pipe models.

Extends from PartialFlowHeatTransfer (base class for any pipe heat transfer correlation).

Parameters

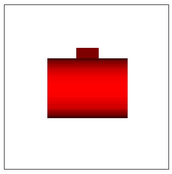
Type	Name	Description
CoefficientOfHeatTransfer	alpha0	guess value for heat transfer coefficients [W/(m2.K)]
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]

Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation
Geometry		
Real	nParallel	number of identical parallel flow devices

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Pipes.BaseClasses.HeatTransfer.LocalPipeFlowHeatTransfer



LocalPipeFlowHeatTransfer: Laminar and turbulent forced convection in pipes, local coefficients

Information

Heat transfer model for laminar and turbulent flow in pipes. Range of validity:

- fully developed pipe flow
- forced convection
- one phase Newtonian fluid
- (spatial) constant wall temperature in the laminar region
- 0 ≤ Re ≤ 1e6, 0.6 ≤ Pr ≤ 100, d/L ≤ 1
- The correlation holds for non-circular pipes only in the turbulent region. Use diameter=4*crossArea/perimeter as characteristic length.

The correlation takes into account the spatial position along the pipe flow, which changes discontinuously at flow reversal. However, the heat transfer coefficient itself is continuous around zero flow rate, but not its derivative.

References

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

Extends from PartialPipeFlowHeatTransfer (Base class for pipe heat transfer correlation in terms of Nusselt number heat transfer in a circular pipe for laminar and turbulent one-phase flow).

Parameters

Type	Name	Description
CoefficientOfHeatTransfer	alpha0	guess value for heat transfer coefficients [W/(m2.K)]
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]
Internal Interface		
replaceable package Medium		Medium in the component
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation

Geometry		
Real	nParallel	number of identical parallel flow devices

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Pipes.BaseClasses.CharacteristicNumbers

Functions to compute characteristic numbers

Package Content

Name	Description
ReynoldsNumber	Return Reynolds number from v, rho, mu, D
ReynoldsNumber_m_flow	Return Reynolds number from m_flow, mu, D, A
NusseltNumber	Return Nusselt number

Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber

Return Reynolds number from v, rho, mu, D

Information

Calculation of Reynolds Number

$$Re = |v|\rho D/\mu$$

a measure of the relationship between inertial forces (vρ) and viscous forces (D/μ).

The following table gives examples for the characteristic dimension D and the velocity v for different fluid flow devices:

Device Type	Characteristic Dimension D	Velocity v
Circular Pipe	diameter	m_flow/ρ/crossArea
Rectangular Duct	4*crossArea/perimeter	m_flow/ρ/crossArea
Wide Duct	distance between narrow, parallel walls	m_flow/ρ/crossArea
Packed Bed	diameterOfSpericalParticles/(1-fluidFractionOfTotalVolume)	m_flow/ρ/crossArea (without particles)
Device with rotating agitator	diameterOfRotor	RotationalSpeed*diameterOfRotor

Inputs

Type	Name	Description
Velocity	v	Mean velocity of fluid flow [m/s]
Density	rho	Fluid density [kg/m3]
DynamicViscosity	mu	Dynamic (absolute) viscosity [Pa.s]
Length	D	Characteristic dimension (hydraulic diameter of pipes) [m]

Outputs

Type	Name	Description
ReynoldsNumber	Re	Reynolds number [1]

Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber_m_flow

Return Reynolds number from m_flow, mu, D, A

Information

Simplified calculation of Reynolds Number for flow through pipes or orifices; using the mass flow rate `m_flow` instead of the velocity `v` to express inertial forces.

```
  Re = |m_flow|*diameter/A/μ
with
  m_flow = v*ρ*A
```

See also Pipes.BaseClasses.CharacteristicNumbers.ReynoldsNumber.

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate [kg/s]
DynamicViscosity	mu	Dynamic viscosity [Pa.s]
Length	D	Characteristic dimension (hydraulic diameter of pipes or orifices) [m]
Area	A	Cross sectional area of fluid flow [m2]

Outputs

Type	Name	Description
ReynoldsNumber	Re	Reynolds number [1]

Pipes.BaseClasses.CharacteristicNumbers.NusseltNumber

Return Nusselt number

Information

```
Nusselt number Nu = alpha*D/lambda
```

Inputs

Type	Name	Description
CoefficientOfHeatTransfer	alpha	Coefficient of heat transfer [W/(m2.K)]
Length	D	Characteristic dimension [m]
ThermalConductivity	lambda	Thermal conductivity [W/(m.K)]

Outputs

Type	Name	Description
NusseltNumber	Nu	Nusselt number [1]

Pipes.BaseClasses.WallFriction

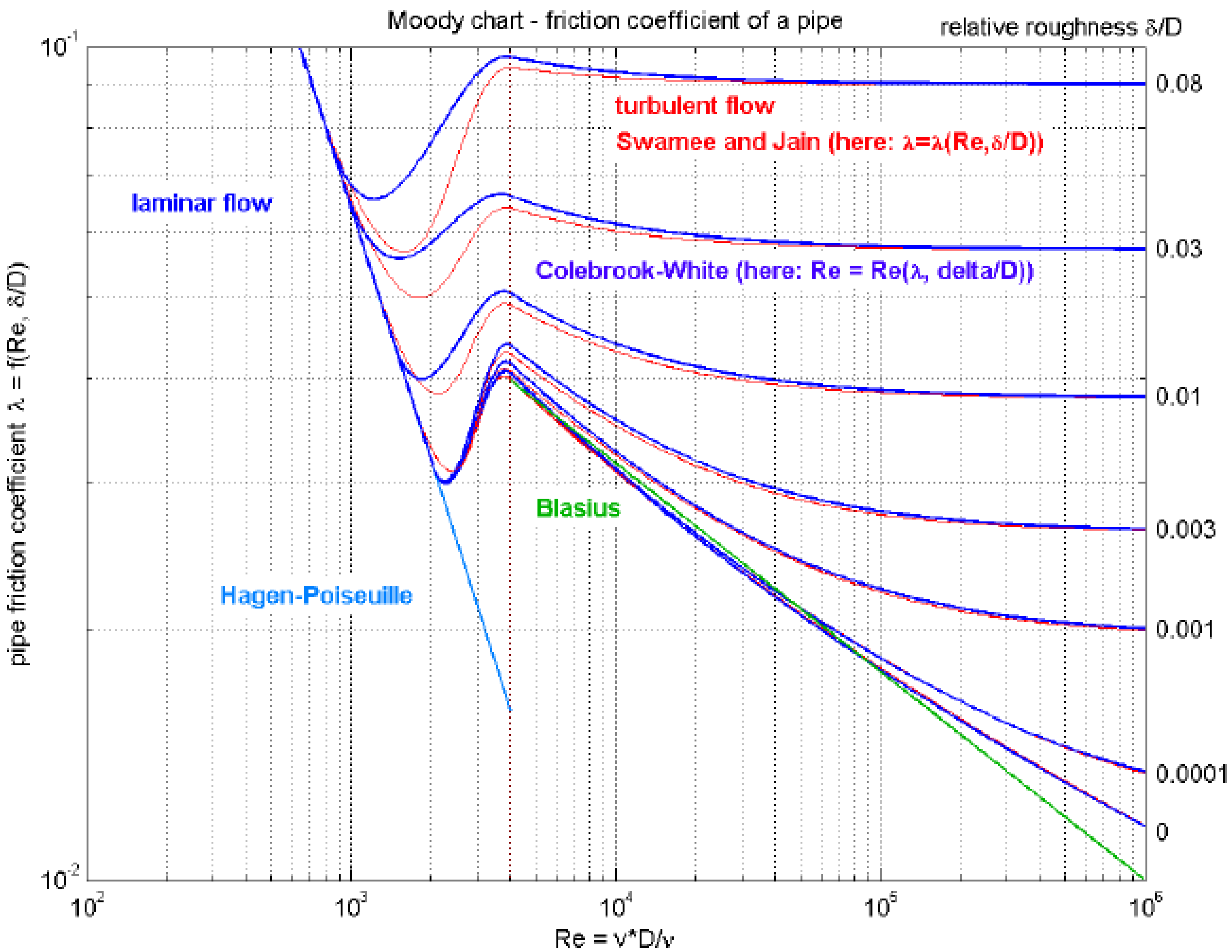
Different variants for pressure drops due to pipe wall friction

Information

This package provides functions to compute pressure losses due to **wall friction** in a pipe. Every correlation is defined by a package that is derived by inheritance from the package WallFriction.PartialWallFriction. The details of the underlying pipe wall friction model are described in the UsersGuide. Basically, different variants of the equation

```
dp = λ(Re,Δ)*(L/D)*ρ*v*|v|/2
```

are used, where the friction loss factor λ is shown in the next figure:



Package Content

Name	Description
PartialWallFriction	Partial wall friction characteristic (base package of all wall friction characteristics)
NoFriction	No pipe wall friction, no static head

Laminar	Pipe wall friction in the laminar regime (linear correlation)
QuadraticTurbulent	Pipe wall friction in the quadratic turbulent regime (simple characteristic, mu not used)
LaminarAndQuadraticTurbulent	Pipe wall friction in the laminar and quadratic turbulent regime (simple characteristic)
Detailed	Pipe wall friction in the whole regime (detailed characteristic)
TestWallFrictionAndGravity	Pressure loss in pipe due to wall friction and gravity (only for test purposes; if needed use Pipes.StaticPipe instead)

Pipes.BaseClasses.WallFriction.PartialWallFriction

Partial wall friction characteristic (base package of all wall friction characteristics)

Information

Package Content

Name	Description
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Pipes.BaseClasses.WallFriction.PartialWallFriction.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]

Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.PartialWallFriction.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.PartialWallFriction.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.PartialWallFriction.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.NoFriction

No pipe wall friction, no static head

Information

This component sets the pressure loss due to wall friction to zero, i.e., it allows to switch off pipe wall friction.

Extends from PartialWallFriction (Partial wall friction characteristic (base package of all wall friction characteristics)).

Package Content

Name	Description
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head
Inherited	
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)

Pipes.BaseClasses.WallFriction.NoFriction.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]

DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.NoFriction.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.NoFriction.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.NoFriction.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.Laminar

Pipe wall friction in the laminar regime (linear correlation)

Information

This component defines only the laminar region of wall friction: dp = k*m_flow, where "k" depends on density and dynamic viscosity. The roughness of the wall does not have an influence on the laminar flow and therefore argument roughness is ignored. Since this is a linear relationship, the occuring systems of equations are usually much simpler (e.g. either linear instead of non-linear). By using nominal values for density and dynamic viscosity, the systems of equations can still further be reduced.

In UsersGuide the complete friction regime is illustrated. This component describes only the **Hagen-Poiseuille** equation.

Extends from PartialWallFriction (Partial wall friction characteristic (base package of all wall friction characteristics)).

Package Content

Name	Description
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head
Inherited	
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)

Pipes.BaseClasses.WallFriction.Laminar.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.Laminar.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.Laminar.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.Laminar.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height	Gravity times (Height(port_b) - Height(port_a))

	_ab	
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.QuadraticTurbulent

Pipe wall friction in the quadratic turbulent regime (simple characteristic, mu not used)

Information

This component defines only the quadratic turbulent regime of wall friction: dp = k*m_flow*|m_flow|, where "k" depends on density and the roughness of the pipe and is no longer a function of the Reynolds number. This relationship is only valid for large Reynolds numbers.

In UsersGuide the complete friction regime is illustrated. This component describes only the asymptotic behaviour for large Reynolds numbers, i.e., the values at the right ordinate where λ is constant.

Extends from PartialWallFriction (Partial wall friction characteristic (base package of all wall friction characteristics)).

Package Content

Name	Description
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head
Inherited	
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)

Pipes.BaseClasses.WallFriction.QuadraticTurbulent.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.QuadraticTurbulent.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description

Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.QuadraticTurbulent.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.QuadraticTurbulent.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]

Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent

Pipe wall friction in the laminar and quadratic turbulent regime (simple characteristic)

Information

This component defines the quadratic turbulent regime of wall friction: dp = k*m_flow*|m_flow|, where "k" depends on density and the roughness of the pipe and is no longer a function of the Reynolds number. This relationship is only valid for large Reynolds numbers. At Re=4000, a polynomial is constructed that approaches the constant λ (for large Reynolds-numbers) at Re=4000 smoothly and has a derivative at zero mass flow rate that is identical to laminar wall friction.

Extends from PartialWallFriction (Partial wall friction characteristic (base package of all wall friction characteristics)).

Package Content

Name	Description
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head
Inherited	
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)

Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description

Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.LaminarAndQuadraticTurbulent.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]

DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

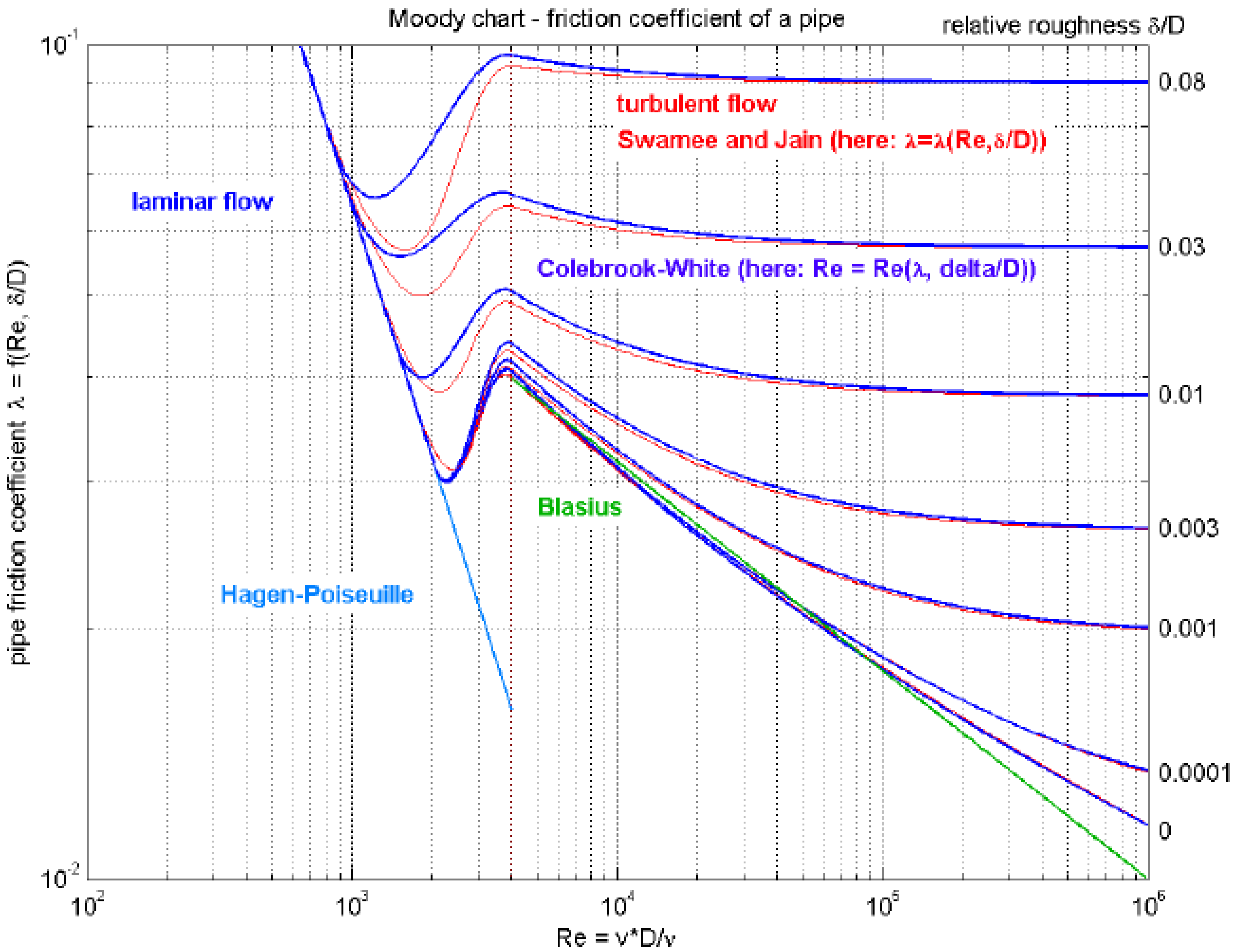
Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.Detailed

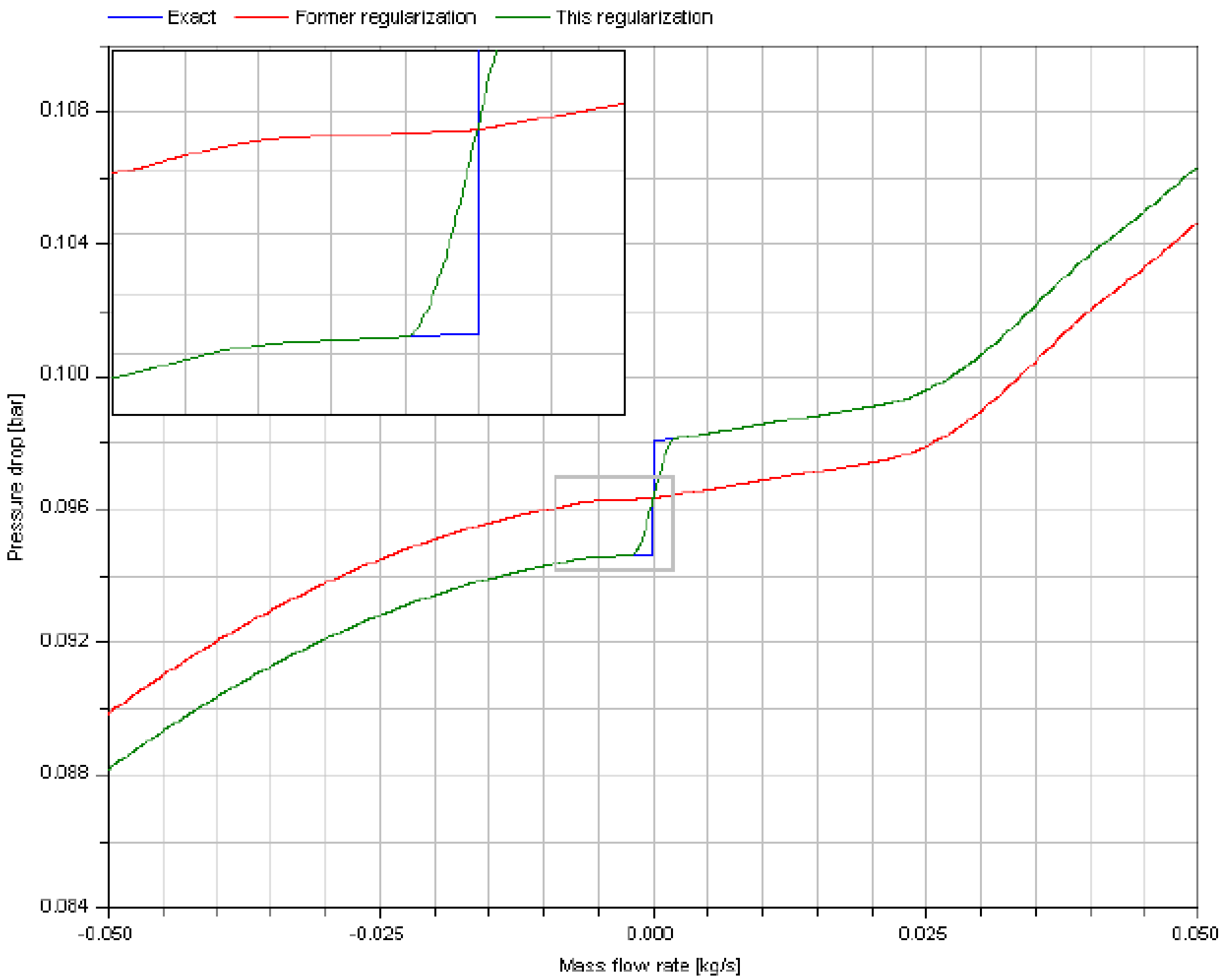
Pipe wall friction in the whole regime (detailed characteristic)

Information

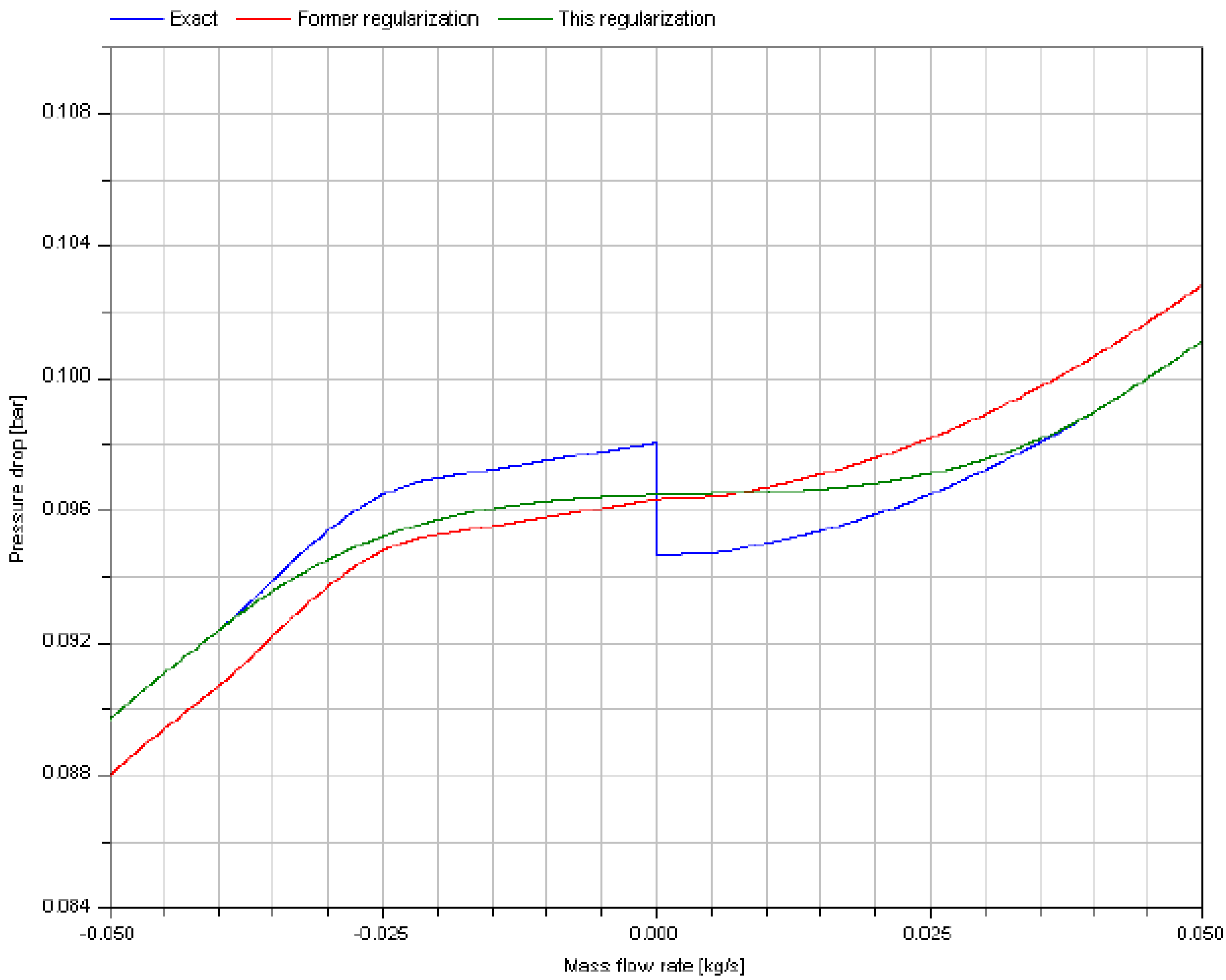
This component defines the complete regime of wall friction. The details are described in the UsersGuide. The functional relationship of the friction loss factor λ is displayed in the next figure. Function massFlowRate_dp() defines the "red curve" ("Swamee and Jain"), where as function pressureLoss_m_flow() defines the "blue curve" ("Colebrook-White"). The two functions are inverses from each other and give slightly different results in the transition region between Re = 1500 .. 4000, in order to get explicit equations without solving a non-linear equation.



Additionally to wall friction, this component properly implements static head. With respect to the latter, two cases can be distinguished. In the case shown next, the change of elevation with the path from a to b has the opposite sign of the change of density.



In the case illustrated second, the change of elevation with the path from a to b has the same sign of the change of density.



Extends from PartialWallFriction (Partial wall friction characteristic (base package of all wall friction characteristics)).

Package Content

Name	Description
massFlowRate_dp	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction
pressureLoss_m_flow	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction
massFlowRate_dp_staticHead	Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head
pressureLoss_m_flow_staticHead	Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head
Inherited	
use_mu=true	= true, if mu_a/mu_b are used in function, otherwise value is not used
use_roughness=true	= true, if roughness is used in function, otherwise value is not used
use_dp_small=true	= true, if dp_small is used in function, otherwise value is not used
use_m_flow_small=true	= true, if m_flow_small is used in function, otherwise value is not used
dp_is_zero=false	= true, if no wall friction is present, i.e., dp = 0 (function massFlowRate_dp() cannot be used)

Pipes.BaseClasses.WallFriction.Detailed.massFlowRate_dp

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction

Information

Extends from (Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction).

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.Detailed.pressureLoss_m_flow

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction

Information

Extends from (Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy

		if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.Detailed.massFlowRate_dp_staticHead

Return mass flow rate m_flow as function of pressure loss dp, i.e., m_flow = f(dp), due to wall friction and static head

Inputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small (dummy if use_dp_small = false) [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Pipes.BaseClasses.WallFriction.Detailed.pressureLoss_m_flow_staticHead

Return pressure loss dp as function of mass flow rate m_flow, i.e., dp = f(m_flow), due to wall friction and static head

Inputs

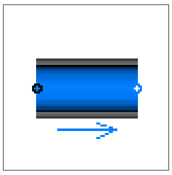
Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a (dummy if use_mu = false) [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b (dummy if use_mu = false) [Pa.s]
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]

Real	g_times_height_ab	Gravity times (Height(port_b) - Height(port_a))
Length	roughness	Absolute roughness of pipe, with a default for a smooth steel pipe (dummy if use_roughness = false) [m]
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small (dummy if use_m_flow_small = false) [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure loss (dp = port_a.p - port_b.p) [Pa]

Pipes.BaseClasses.WallFriction.TestWallFrictionAndGravity



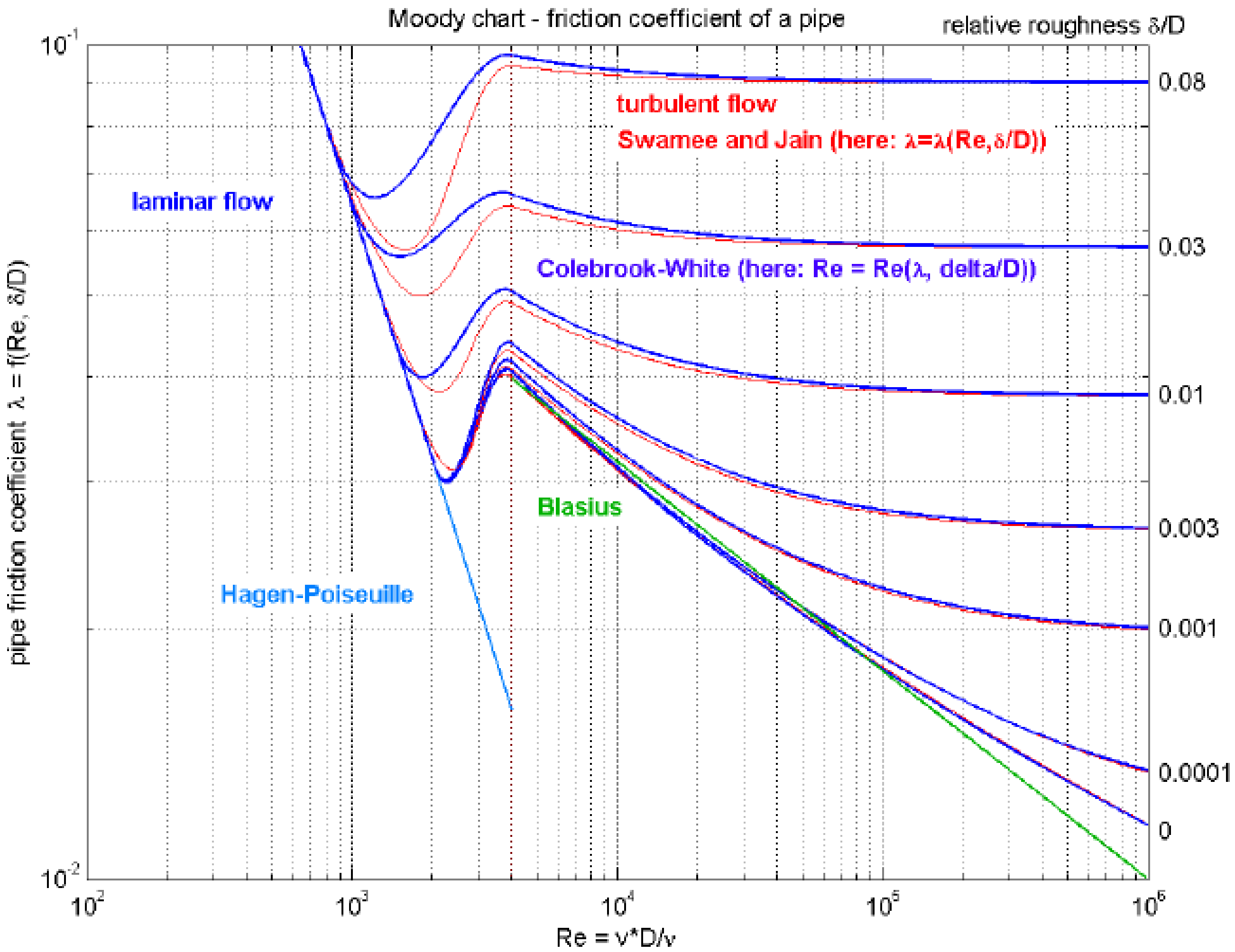
Pressure loss in pipe due to wall friction and gravity (only for test purposes; if needed use Pipes.StaticPipe instead)

Information

This model describes pressure losses due to **wall friction** in a pipe and due to gravity. It is assumed that no mass or energy is stored in the pipe. Correlations of different complexity and validity can be seleted via the replaceable package **WallFriction** (see parameter menu below). The details of the pipe wall friction model are described in the UsersGuide. Basically, different variants of the equation

```
dp = λ(Re,Δ)*(L/D)*ρ*v*|v|/2
```

are used, where the friction loss factor λ is shown in the next figure:



By default, the correlations are computed with media data at the actual time instant. In order to reduce non-linear equation systems, parameter **use_nominal** provides the option to compute the correlations with constant media values at the desired operating point. This might speed-up the simulation and/or might give a more robust simulation.

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Length	length	Length of pipe [m]
Diameter	diameter	Inner (hydraulic) diameter of pipe [m]
Length	height_ab	Height(port_b) - Height(port_a) [m]
Length	roughness	Absolute roughness of pipe (default = smooth steel pipe) [m]
Boolean	use_nominal	= true, if mu_nominal and rho_nominal are used, otherwise computed from medium
DynamicViscosity	mu_nominal	Nominal dynamic viscosity (e.g. mu_liquidWater = 1e-3, mu_air = 1.8e-5) [Pa.s]
Density	rho_nominal	Nominal density (e.g. rho_liquidWater = 995, rho_air = 1.2) [kg/m3]
Assumptions		

Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Boolean	show_Re	= true, if Reynolds number is included for plotting
Boolean	from_dp	= true, use m_flow = f(dp), otherwise dp = f(m_flow)
AbsolutePressure	dp_small	Within regularization if |dp| < dp_small (may be wider for large discontinuities in static head) [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Modelica_Fluid.Machines

Devices for converting between energy held in a fluid and mechanical energy

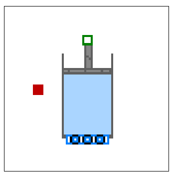
Information

Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
SweptVolume	varying cylindric volume depending on the postition of the piston
Pump	Centrifugal pump with mechanical connector for the shaft
ControlledPump	Centrifugal pump with ideally controlled mass flow rate
PrescribedPump	Centrifugal pump with ideally controlled speed
BaseClasses	Base classes used in the Machines package (only of interest to build new component models)

Machines.SweptVolume



varying cylindric volume depending on the postition of the piston

Information

Mixing volume with varying size. The size of the volume is given by:

- cross sectional piston area
- piston stroke given by the flange position s
- clearance (volume at flang position = 0)

Losses are neglected. The shaft power is completely converted into mechanical work on the fluid.

The flange position has to be equal or greater than zero. Otherwise the simulation stops. The force of the flange results from the pressure difference between medium and ambient pressure and the cross sectional piston area. For using the component, a top level instance of the ambient model with the inner attribute is needed.

The pressure at both fluid ports equals the medium pressure in the volume. No suction nor discharge valve is included in the model.

The thermal port is directly connected to the medium. The temperature of the thermal port equals the medium temperature. The heat capacity of the cylinder and the piston are not includes in the model.

Extends from Vessels.BaseClasses.PartialLumpedVessel (Lumped volume with a vector of fluid ports and replaceable heat transfer model).

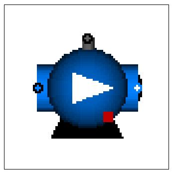
Parameters

Type	Name	Description
Area	pistonCrossArea	cross sectional area of pistion [m2]
Volume	clearance	remaining volume at zero piston stroke [m3]
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
Ports		
Boolean	use_portsData	= false to neglect pressure loss and kinetic energy
VesselPortsData	portsData[nPorts]	Data of inlet/outlet ports
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use the HeatTransfer model
replaceable model HeatTransfer		Wall heat transfer
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Port properties		
MassFlowRate	m_flow_small	Regularization range at zero mass flow rate [kg/s]

Connectors

Type	Name	Description
VesselFluidPorts_b	ports[nPorts]	Fluid inlets and outlets
HeatPort_a	heatPort	
Flange_b	flange	translation flange for piston

Machines.Pump



Centrifugal pump with mechanical connector for the shaft

Information

This model describes a centrifugal pump (or a group of `nParallel` pumps) with a mechanical rotational connector for the shaft, to be used when the pump drive has to be modelled explicitly. In the case of `nParallel` pumps, the mechanical connector is relative to a single pump.

The model extends `PartialPump`

Extends from Machines.BaseClasses.PartialPump (Base model for centrifugal pumps).

Parameters

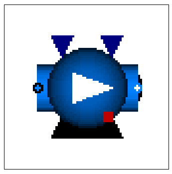
Type	Name	Description
replaceable package Medium		Medium in the component
Characteristics		
Integer	nParallel	Number of pumps in parallel
replaceable function flowCharacteristic		Head vs. V_flow characteristic at nominal speed and density
AngularVelocity_rpm	N_nominal	Nominal rotational speed for flow characteristic [1/min]
Density	rho_nominal	Nominal fluid density for characteristic [kg/m3]
Boolean	use_powerCharacteristic	Use powerCharacteristic (vs. efficiencyCharacteristic)
replaceable function powerCharacteristic		Power consumption vs. V_flow at nominal speed and density
replaceable function efficiencyCharacteristic		Efficiency vs. V_flow at nominal speed and density
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	= true to prevent reverse flow
Volume	V	Volume inside the pump [m3]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use a HeatTransfer model, e.g. for a housing
replaceable model HeatTransfer		Wall heat transfer
Initialization		
AbsolutePressure	p_a_start	Guess value for inlet pressure [Pa]
AbsolutePressure	p_b_start	Guess value for outlet pressure [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Diagnostics		

Boolean	show_NPSHa	= true to compute Net Positive Suction Head available

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
HeatPort_a	heatPort	
Flange_a	shaft	

Machines.ControlledPump



Centrifugal pump with ideally controlled mass flow rate

Information

This model describes a centrifugal pump (or a group of `nParallel` pumps) with ideally controlled mass flow rate or pressure.

Nominal values are used to predefine an exemplary pump characteristics and to define the operation of the pump. The input connectors `m_flow_set` or `p_set` can optionally be enabled to provide time varying set points.

Use this model if the pump characteristics is of secondary interest. The actual characteristics can be configured later on for the appropriate rotational speed N. Then the model can be replaced with a Pump with rotational shaft or with a PrescribedPump.

Extends from Machines.BaseClasses.PartialPump (Base model for centrifugal pumps).

Parameters

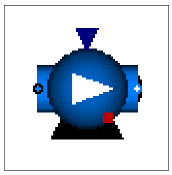
Type	Name	Description
replaceable package Medium		Medium in the component
AbsolutePressure	p_a_nominal	Nominal inlet pressure for predefined pump characteristics [Pa]
AbsolutePressure	p_b_nominal	Nominal outlet pressure, fixed if not control_m_flow and not use_p_set [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flow rate, fixed if control_m_flow and not use_m_flow_set [kg/s]
Boolean	control_m_flow	= false to control outlet pressure port_b.p instead of m_flow
Boolean	use_m_flow_set	= true to use input signal m_flow_set instead of m_flow_nominal
Boolean	use_p_set	= true to use input signal p_set instead of p_b_nominal
Characteristics		
Integer	nParallel	Number of pumps in parallel
replaceable function flowCharacteristic		Head vs. V_flow characteristic at nominal speed and density
AngularVelocity_rpm	N_nominal	Nominal rotational speed for flow characteristic [1/min]
Density	rho_nominal	Nominal fluid density for characteristic [kg/m3]
Boolean	use_powerCharacteristic	Use powerCharacteristic (vs. efficiencyCharacteristic)
replaceable function powerCharacteristic		Power consumption vs. V_flow at nominal speed and density
replaceable function efficiencyCharacteristic		Efficiency vs. V_flow at nominal speed and density

Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	= true to prevent reverse flow
Volume	V	Volume inside the pump [m3]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use a HeatTransfer model, e.g. for a housing
replaceable model HeatTransfer		Wall heat transfer
Initialization		
AbsolutePressure	p_a_start	Guess value for inlet pressure [Pa]
AbsolutePressure	p_b_start	Guess value for outlet pressure [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Diagnostics		
Boolean	show_NPSHa	= true to compute Net Positive Suction Head available

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
HeatPort_a	heatPort	
input RealInput	m_flow_set	Prescribed mass flow rate
input RealInput	p_set	Prescribed outlet pressure
Characteristics		
replaceable function flowCharacteristic		Head vs. V_flow characteristic at nominal speed and density

Machines.PrescribedPump



Centrifugal pump with ideally controlled speed

Information

This model describes a centrifugal pump (or a group of `nParallel` pumps) with prescribed speed, either fixed or provided by an external signal.

The model extends `PartialPump`

If the `N_in` input connector is wired, it provides rotational speed of the pumps (rpm); otherwise, a constant rotational speed equal to `n_const` (which can be different from `N_nominal`) is assumed.

Extends from Machines.BaseClasses.PartialPump (Base model for centrifugal pumps).

Parameters

Type	Name	Description
replaceable package	Medium	Medium in the component
Boolean	use_N_in	Get the rotational speed from the input connector
AngularVelocity_rpm	N_const	Constant rotational speed [1/min]
Characteristics		
Integer	nParallel	Number of pumps in parallel
replaceable function	flowCharacteristic	Head vs. V_flow characteristic at nominal speed and density
AngularVelocity_rpm	N_nominal	Nominal rotational speed for flow characteristic [1/min]
Density	rho_nominal	Nominal fluid density for characteristic [kg/m3]
Boolean	use_powerCharacteristic	Use powerCharacteristic (vs. efficiencyCharacteristic)
replaceable function	powerCharacteristic	Power consumption vs. V_flow at nominal speed and density
replaceable function	efficiencyCharacteristic	Efficiency vs. V_flow at nominal speed and density
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	= true to prevent reverse flow
Volume	V	Volume inside the pump [m3]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use a HeatTransfer model, e.g. for a housing
replaceable model	HeatTransfer	Wall heat transfer
Initialization		
AbsolutePressure	p_a_start	Guess value for inlet pressure [Pa]
AbsolutePressure	p_b_start	Guess value for outlet pressure [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		
Diagnostics		
Boolean	show_NPSHa	= true to compute Net Positive Suction Head available

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)

FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
HeatPort_a	heatPort	
input RealInput	N_in	Prescribed rotational speed [1/min]

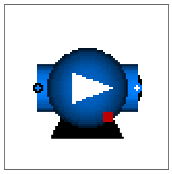
Machines.BaseClasses

Base classes used in the Machines package (only of interest to build new component models)

Package Content

Name	Description
PartialPump	Base model for centrifugal pumps
PumpCharacteristics	Functions for pump characteristics
assertPositiveDifference	

Machines.BaseClasses.PartialPump



Base model for centrifugal pumps

Information

This is the base model for pumps.

The model describes a centrifugal pump, or a group of `nParallel` identical pumps. The pump model is based on the theory of kinematic similarity: the pump characteristics are given for nominal operating conditions (rotational speed and fluid density), and then adapted to actual operating condition, according to the similarity equations.

Pump characteristics

The nominal hydraulic characteristic (head vs. volume flow rate) is given by the the replaceable function `flowCharacteristic`.

The pump energy balance can be specified in two alternative ways:

- `use_powerCharacteristic = false` (default option): the replaceable function `efficiencyCharacteristic` (efficiency vs. volume flow rate in nominal conditions) is used to determine the efficiency, and then the power consumption. The default is a constant efficiency of 0.8.
- `use_powerCharacteristic = true`: the replaceable function `powerCharacteristic` (power consumption vs. volume flow rate in nominal conditions) is used to determine the power consumption, and then the efficiency. Use `powerCharacteristic` to specify a non-zero power consumption for zero flow rate.

Several functions are provided in the package `PumpCharacteristics` to specify the characteristics as a function of some operating points at nominal conditions.

Depending on the value of the `checkValve` parameter, the model either supports reverse flow conditions, or includes a built-in check valve to avoid flow reversal.

It is possible to take into account the heat capacity of the fluid inside the pump by specifying its volume `V`; this is necessary to avoid singularities in the computation of the outlet enthalpy in case of zero flow rate. If zero flow rate conditions are always avoided, this dynamic effect can be neglected by leaving the default value `V = 0`, thus avoiding a fast state variable in the model.

Dynamics options

Steady-state mass and energy balances are assumed per default, neglecting the holdup of fluid in the pump.

Dynamic mass and energy balance can be used by setting the corresponding dynamic parameters. This might be desirable if the pump is assembled together with valves before port_a and behind port_b. If both valves are closed, then the fluid is useful to define the thermodynamic state and in particular the absolute pressure in the pump. Note that the `flowCharacteristic` only specifies a pressure difference.

Heat transfer

The boolean paramter `use_HeatTransfer` can be set to true if heat exchanged with the environment should be taken into account or to model a housing. This might be desirable if a pump with realistic `powerCharacteristic` for zero flow operates while a valve prevents fluid flow.

Diagnostics of Cavitation

The boolean parameter show_NPSHa can set true to compute the Net Positive Suction Head available and check for cavitation, provided a two-phase medium model is used.

Extends from Interfaces.PartialTwoPort (Partial component with two ports), Interfaces.PartialLumpedVolume (Lumped volume with mass and energy balance).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
Characteristics		
Integer	nParallel	Number of pumps in parallel
AngularVelocity_rpm	N_nominal	Nominal rotational speed for flow characteristic [1/min]
Density	rho_nominal	Nominal fluid density for characteristic [kg/m3]
Boolean	use_powerCharacteristic	Use powerCharacteristic (vs. efficiencyCharacteristic)
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	= true to prevent reverse flow
Volume	V	Volume inside the pump [m3]
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Heat transfer		
Boolean	use_HeatTransfer	= true to use a HeatTransfer model, e.g. for a housing
Initialization		
AbsolutePressure	p_a_start	Guess value for inlet pressure [Pa]
AbsolutePressure	p_b_start	Guess value for outlet pressure [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances
Advanced		

Diagnostics		
Boolean	show_NPSHa	= true to compute Net Positive Suction Head available

Connectors

Type	Name	Description
HeatPort_a	heatPort	

Machines.BaseClasses.PumpCharacteristics

Functions for pump characteristics

Package Content

Name	Description
baseFlow	Base class for pump flow characteristics
basePower	Base class for pump power consumption characteristics
baseEfficiency	Base class for efficiency characteristics
linearFlow	Linear flow characteristic
quadraticFlow	Quadratic flow characteristic
polynomialFlow	Polynomial flow characteristic
constantEfficiency	Constant efficiency characteristic
linearPower	Linear power consumption characteristic
quadraticPower	Quadratic power consumption characteristic

Machines.BaseClasses.PumpCharacteristics.baseFlow

Base class for pump flow characteristics

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]

Outputs

Type	Name	Description
Height	head	Pump head [m]

Machines.BaseClasses.PumpCharacteristics.basePower

Base class for pump power consumption characteristics

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]

Outputs

Type	Name	Description
Power	consumption	Power consumption [W]

Machines.BaseClasses.PumpCharacteristics.baseEfficiency

Base class for efficiency characteristics

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]

Outputs

Type	Name	Description
Real	eta	Efficiency

Machines.BaseClasses.PumpCharacteristics.linearFlow

Linear flow characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
VolumeFlowRate	V_flow_nominal[2]	Volume flow rate for two operating points (single pump) [m3/s]
Height	head_nominal[2]	Pump head for two operating points [m]

Outputs

Type	Name	Description
Height	head	Pump head [m]

Machines.BaseClasses.PumpCharacteristics.quadraticFlow

Quadratic flow characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
VolumeFlowRate	V_flow_nominal[3]	Volume flow rate for three operating points (single pump) [m3/s]
Height	head_nominal[3]	Pump head for three operating points [m]

Outputs

Type	Name	Description

Height	head	Pump head [m]

Machines.BaseClasses.PumpCharacteristics.polynomialFlow

Polynomial flow characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
VolumeFlowRate	V_flow_nominal[:]	Volume flow rate for N operating points (single pump) [m3/s]
Height	head_nominal[:]	Pump head for N operating points [m]

Outputs

Type	Name	Description
Height	head	Pump head [m]

Machines.BaseClasses.PumpCharacteristics.constantEfficiency

Constant efficiency characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
Real	eta_nominal	Nominal efficiency

Outputs

Type	Name	Description
Real	eta	Efficiency

Machines.BaseClasses.PumpCharacteristics.linearPower

Linear power consumption characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
VolumeFlowRate	V_flow_nominal[2]	Volume flow rate for two operating points (single pump) [m3/s]
Power	W_nominal[2]	Power consumption for two operating points [W]

Outputs

Type	Name	Description
Power	consumption	Power consumption [W]

Machines.BaseClasses.PumpCharacteristics.quadraticPower

Quadratic power consumption characteristic

Inputs

Type	Name	Description
VolumeFlowRate	V_flow	Volumetric flow rate [m3/s]
VolumeFlowRate	V_flow_nominal[3]	Volume flow rate for three operating points (single pump) [m3/s]
Power	W_nominal[3]	Power consumption for three operating points [W]

Outputs

Type	Name	Description
Power	consumption	Power consumption [W]

Machines.BaseClasses.assertPositiveDifference

Inputs

Type	Name	Description
Pressure	p	[Pa]
Pressure	p_sat	[Pa]
String	message	

Outputs

Type	Name	Description
Pressure	dp	[Pa]

Modelica_Fluid.Valves

Components for the regulation and control of fluid flow

Information

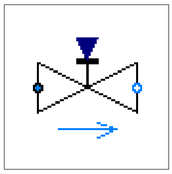
Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
ValveIncompressible	Valve for (almost) incompressible fluids
ValveVaporizing	Valve for possibly vaporizing (almost) incompressible fluids, accounts for choked flow conditions
ValveCompressible	Valve for compressible fluids, accounts for choked flow conditions
ValveLinear	Valve for water/steam flows with linear pressure drop
ValveDiscrete	Valve for water/steam flows with linear pressure drop
BaseClasses	Base classes used in the Valves package (only of interest to build new

	component models)

Valves.ValveIncompressible



Valve for (almost) incompressible fluids

Information

Valve model according to the IEC 534/ISA S.75 standards for valve sizing, incompressible fluids.

This model assumes that the fluid has a low compressibility, which is always the case for liquids. It can also be used with gases, provided that the pressure drop is lower than 0.2 times the absolute pressure at the inlet, so that the fluid density does not change much inside the valve.

If `checkValve` is false, the valve supports reverse flow, with a symmetric flow characteric curve. Otherwise, reverse flow is stopped (check valve behaviour).

The treatment of parameters **Kv** and **Cv** is explained in detail in the Users Guide.

Extends from BaseClasses.PartialValve (Base model for valves).

Parameters

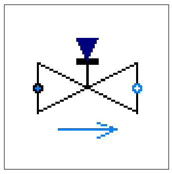
Type	Name	Description
replaceable package Medium		Medium in the component
replaceable function valveCharacteristic		Inherent flow characteristic
Flow Coefficient		
CvTypes	CvData	Selection of flow coefficient
Area	Av	Av (metric) flow coefficient [m2]
Real	Kv	Kv (metric) flow coefficient [m3/h]
Real	Cv	Cv (US) flow coefficient [USG/min]
Nominal operating point		
Pressure	dp_nominal	Nominal pressure drop [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate [kg/s]
Density	rho_nominal	Nominal inlet density [kg/m3]
Real	opening_nominal	Nominal opening
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	Reverse flow stopped
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Pressure	dp_small	Regularisation of zero flow [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description

FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input RealInput	opening	Valve position in the range 0-1

Valves.ValveVaporizing



Valve for possibly vaporizing (almost) incompressible fluids, accounts for choked flow conditions

Information

Valve model according to the IEC 534/ISA S.75 standards for valve sizing, incompressible fluid at the inlet, and possibly two-phase fluid at the outlet, including choked flow conditions.

The model operating range includes choked flow operation, which takes place for low outlet pressures due to flashing in the vena contracta; otherwise, non-choking conditions are assumed.

This model requires a two-phase medium model, to describe the liquid and (possible) two-phase conditions.

The default liquid pressure recovery coefficient `Fl` is constant and given by the parameter `Fl_nominal`. The relative change (per unit) of the recovery coefficient can be specified as a given function of the valve opening by replacing the `FlCharacteristic` function.

If `checkValve` is false, the valve supports reverse flow, with a symmetric flow characteric curve. Otherwise, reverse flow is stopped (check valve behaviour).

The treatment of parameters **Kv** and **Cv** is explained in detail in the Users Guide.

Extends from BaseClasses.PartialValve (Base model for valves).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
replaceable function valveCharacteristic		Inherent flow characteristic
Real	Fl_nominal	Liquid pressure recovery factor
replaceable function FlCharacteristic		Pressure recovery characteristic
Flow Coefficient		
CvTypes	CvData	Selection of flow coefficient
Area	Av	Av (metric) flow coefficient [m2]
Real	Kv	Kv (metric) flow coefficient [m3/h]
Real	Cv	Cv (US) flow coefficient [USG/min]
Nominal operating point		
Pressure	dp_nominal	Nominal pressure drop [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate [kg/s]
Density	rho_nominal	Nominal inlet density [kg/m3]
Real	opening_nominal	Nominal opening
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	Reverse flow stopped
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]

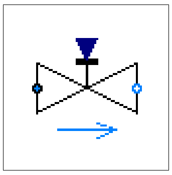
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Pressure	dp_small	Regularisation of zero flow [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
replaceable package Medium		Medium in the component
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input RealInput	opening	Valve position in the range 0-1
replaceable function FlCharacteristic		Pressure recovery characteristic

---

Valves.ValveCompressible



Valve for compressible fluids, accounts for choked flow conditions

Information

Valve model according to the IEC 534/ISA S.75 standards for valve sizing, compressible fluid, no phase change, also covering choked-flow conditions.

This model can be used with gases and vapours, with arbitrary pressure ratio between inlet and outlet.

The product Fk*xt is given by the parameter `Fxt_full`, and is assumed constant by default. The relative change (per unit) of the xt coefficient with the valve opening can be specified by replacing the `xtCharacteristic` function.

If `checkValve` is false, the valve supports reverse flow, with a symmetric flow characteric curve. Otherwise, reverse flow is stopped (check valve behaviour).

The treatment of parameters **Kv** and **Cv** is explained in detail in the Users Guide.

Extends from BaseClasses.PartialValve (Base model for valves).

Parameters

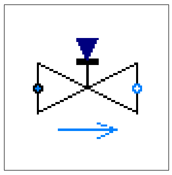
Type	Name	Description
replaceable package Medium		Medium in the component
replaceable function valveCharacteristic		Inherent flow characteristic
Real	Fxt_full	Fk*xt critical ratio at full opening
replaceable function xtCharacteristic		Critical ratio characteristic
Flow Coefficient		
CvTypes	CvData	Selection of flow coefficient
Area	Av	Av (metric) flow coefficient [m2]
Real	Kv	Kv (metric) flow coefficient [m3/h]
Real	Cv	Cv (US) flow coefficient [USG/min]

Nominal operating point		
Pressure	dp_nominal	Nominal pressure drop [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate [kg/s]
Density	rho_nominal	Nominal inlet density [kg/m3]
Real	opening_nominal	Nominal opening
AbsolutePressure	p_nominal	Nominal inlet pressure [Pa]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	Reverse flow stopped
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Pressure	dp_small	Regularisation of zero flow [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input RealInput	opening	Valve position in the range 0-1
replaceable function xtCharacteristic		Critical ratio characteristic

Valves.ValveLinear



Valve for water/steam flows with linear pressure drop

Information

This very simple model provides a pressure drop which is proportional to the flowrate and to the `opening` input, without computing any fluid property. It can be used for testing purposes, when a simple model of a variable pressure loss is needed.

A medium model must be nevertheless be specified, so that the fluid ports can be connected to other components using the same medium model.

The model is adiabatic (no heat losses to the ambient) and neglects changes in kinetic energy from the inlet to the outlet.

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component

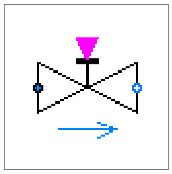
AbsolutePressure	dp_nominal	Nominal pressure drop at full opening [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate at full opening [kg/s]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input RealInput	opening	=1: completely open, =0: completely closed

---

Valves.ValveDiscrete



Valve for water/steam flows with linear pressure drop

Information

This very simple model provides a (small) pressure drop which is proportional to the flowrate if the Boolean open signal is **true**. Otherwise, the mass flow rate is zero. If opening_min > 0, a small leakage mass flow rate occurs when open = **false**.

This model can be used for simplified modelling of on-off valves, when it is not important to accurately describe the pressure loss when the valve is open. Although the medium model is not used to determine the pressure loss, it must be nevertheless be specified, so that the fluid ports can be connected to other components using the same medium model.

The model is adiabatic (no heat losses to the ambient) and neglects changes in kinetic energy from the inlet to the outlet.

In a diagram animation, the valve is shown in "green", when it is open.

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Pressure	dp_nominal	Nominal pressure drop at full opening=1 [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate at full opening=1 [kg/s]
Real	opening_min	Remaining opening if closed, causing small leakage flow
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)

Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input BooleanInput	open	

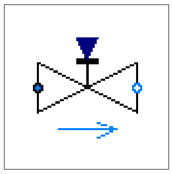
Valves.BaseClasses

Base classes used in the Valves package (only of interest to build new component models)

Package Content

Name	Description
PartialValve	Base model for valves
ValveCharacteristics	Functions for valve characteristics

Valves.BaseClasses.PartialValve



Base model for valves

Information

This is the base model for the `ValveIncompressible`, `ValveVaporizing`, and `ValveCompressible` valve models. The model is based on the IEC 534 / ISA S.75 standards for valve sizing.

The model optionally supports reverse flow conditions (assuming symmetrical behaviour) or check valve operation, and has been suitably regularized, compared to the equations in the standard, in order to avoid numerical singularities around zero pressure drop operating conditions.

The model assumes adiabatic operation (no heat losses to the ambient); changes in kinetic energy from inlet to outlet are neglected in the energy balance.

Modelling options

The following options are available to specify the valve flow coefficient in fully open conditions:

- `CvData = Modelica_Fluid.Types.CvTypes.Av`: the flow coefficient is given by the metric `Av` coefficient (m^2).
- `CvData = Modelica_Fluid.Types.CvTypes.Kv`: the flow coefficient is given by the metric `Kv` coefficient (m^3/h).
- `CvData = Modelica_Fluid.Types.CvTypes.Cv`: the flow coefficient is given by the US `Cv` coefficient (USG/min).
- `CvData = Modelica_Fluid.Types.CvTypes.OpPoint`: the flow is computed from the nominal operating point specified by `p_nominal`, `dp_nominal`, `m_flow_nominal`, `rho_nominal`, `opening_nominal`.

The nominal pressure drop `dp_nominal` must always be specified; to avoid numerical singularities, the flow characteristic is modified for pressure drops less than `b*dp_nominal` (the default value is 1% of the nominal pressure drop). Increase this parameter if numerical problems occur in valves with very low pressure drops.

If `checkValve` is true, then the flow is stopped when the outlet pressure is higher than the inlet pressure; otherwise, reverse flow takes place. Use this option only when neede, as it increases the numerical complexity of the problem.

The valve opening characteristic `valveCharacteristic`, linear by default, can be replaced by any user-defined function. Quadratic and equal percentage with customizable rangeability are already provided by the library.

The treatment of parameters **Kv** and **Cv** is explained in detail in the Users Guide.

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Flow Coefficient		
CvTypes	CvData	Selection of flow coefficient
Area	Av	Av (metric) flow coefficient [m2]
Real	Kv	Kv (metric) flow coefficient [m3/h]
Real	Cv	Cv (US) flow coefficient [USG/min]
Nominal operating point		
Pressure	dp_nominal	Nominal pressure drop [Pa]
MassFlowRate	m_flow_nominal	Nominal mass flowrate [kg/s]
Density	rho_nominal	Nominal inlet density [kg/m3]
Real	opening_nominal	Nominal opening
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Boolean	checkValve	Reverse flow stopped
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Pressure	dp_small	Regularisation of zero flow [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
input RealInput	opening	Valve position in the range 0-1

Valves.BaseClasses.ValveCharacteristics

Functions for valve characteristics

Package Content

Name	Description
baseFun	Base class for valve characteristics
linear	Linear characteristic
one	Constant characteristic
quadratic	Quadratic characteristic
equalPercentage	Equal percentage characteristic

Valves.BaseClasses.ValveCharacteristics.baseFun

Base class for valve characteristics

Inputs

Type	Name	Description
Real	pos	Opening position (per unit)

Outputs

Type	Name	Description
Real	rc	Relative flow coefficient (per unit)

Valves.BaseClasses.ValveCharacteristics.linear

Linear characteristic

Inputs

Type	Name	Description
Real	pos	Opening position (per unit)

Outputs

Type	Name	Description
Real	rc	Relative flow coefficient (per unit)

Valves.BaseClasses.ValveCharacteristics.one

Constant characteristic

Inputs

Type	Name	Description
Real	pos	Opening position (per unit)

Outputs

Type	Name	Description
Real	rc	Relative flow coefficient (per unit)

Valves.BaseClasses.ValveCharacteristics.quadratic

Quadratic characteristic

Inputs

Type	Name	Description
Real	pos	Opening position (per unit)

Outputs

Type	Name	Description
Real	rc	Relative flow coefficient (per unit)

Valves.BaseClasses.ValveCharacteristics.equalPercentage

Equal percentage characteristic

Information

This characteristic is such that the relative change of the flow coefficient is proportional to the change in the opening position:

d(rc)/d(pos) = k d(pos).

The constant k is expressed in terms of the rangeability, i.e. the ratio between the maximum and the minimum useful flow coefficient:

rangeability = exp(k) = rc(1.0)/rc(0.0).

The theoretical characteristic has a non-zero opening when pos = 0; the implemented characteristic is modified so that the valve closes linearly when pos < delta.

Extends from baseFun (Base class for valve characteristics).

Inputs

Type	Name	Description
Real	pos	Opening position (per unit)
Real	rangeability	Rangeability
Real	delta	

Outputs

Type	Name	Description
Real	rc	Relative flow coefficient (per unit)

Modelica_Fluid.Fittings

Adaptors for connections of fluid components and the regulation of fluid flow

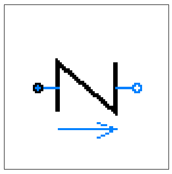
Information

Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
SimpleGenericOrifice	Simple generic orifice defined by pressure loss coefficient and diameter (only for flow from port_a to port_b)
SharpEdgedOrifice	Pressure drop due to sharp edged orifice (for both flow directions)
AbruptAdaptor	Pressure drop in pipe due to suddenly expanding or reducing area (for both flow directions)
MultiPort	Multiply a port; useful if multiple connections shall be made to a port exposing a state
TeeJunctionIdeal	Splitting/joining component with static balances for an infinitesimal control volume
TeeJunctionVolume	Splitting/joining component with static balances for a dynamic control volume
BaseClasses	Base classes used in the Fittings package (only of interest to build new component models)

Fittings.SimpleGenericOrifice



Simple generic orifice defined by pressure loss coefficient and diameter (only for flow from port_a to port_b)

Information

This pressure drop component defines a simple, generic orifice, where the loss factor ζ is provided for one flow direction (e.g., from loss table of a book):

```
Δp = 0.5*ζ*ρ*v*|v|
   = 8*ζ/(π^2*D^4*ρ) * m_flow*|m_flow|
```

where

- Δp is the pressure drop: Δp = port_a.p - port_b.p
- D is the diameter of the orifice at the position where ζ is defined (either at port_a or port_b). If the orifice has not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.
- ζ is the loss factor with respect to D that depends on the geometry of the orifice. In the turbulent flow regime, it is assumed that ζ is constant.
  For small mass flow rates, the flow is laminar and is approximated by a polynomial that has a finite derivative for m_flow=0.
- v is the mean velocity.
- ρ is the upstream density.

Since the pressure loss factor zeta is provided only for a mass flow from port_a to port_b, the pressure loss is not correct when the flow is reversing. If reversing flow only occurs in a short time interval, this is most likely uncritical. If significant reversing flow can appear, this component should not be used.

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy), Interfaces.PartialLumpedFlow (Base class for a lumped momentum balance).

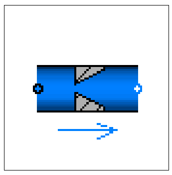
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Length	pathLength	Length flow path [m]
Diameter	diameter	Diameter of orifice [m]
Real	zeta	Loss factor for flow of port_a -> port_b
Boolean	use_zeta	= false to obtain zeta from dp_nominal and m_flow_nominal
AbsolutePressure	dp_nominal	Nominal pressure drop [Pa]
MassFlowRate	m_flow_nominal	Mass flow rate for dp_nominal [kg/s]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	momentumDynamics	Formulation of momentum balance
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp) else dp = f(m_flow)
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.SharpEdgedOrifice



Pressure drop due to sharp edged orifice (for both flow directions)

Information

Extends from BaseClasses.QuadraticTurbulent.BaseModel (Generic pressure drop component with constant turbulent loss factor data and without an icon).

Parameters

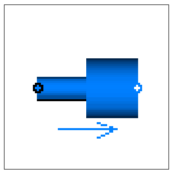
Type	Name	Description
replaceable package Medium		Medium in the component
LossFactorData	data	Loss factor data
Length	length	Length of orifice [m]
Diameter	diameter	Inner diameter of pipe (= same at port_a and port_b) [m]

Diameter	leastDiameter	Smallest diameter of orifice [m]
Angle_deg	alpha	Angle of orifice [deg]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp) else dp = f(m_flow)
Boolean	use_Re	= true, if turbulent region is defined by Re, otherwise by dp_small or m_flow_small
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed
Boolean	show_Re	= true, if Reynolds number is included for plotting

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.AbruptAdaptor



Pressure drop in pipe due to suddenly expanding or reducing area (for both flow directions)

Information

Extends from BaseClasses.QuadraticTurbulent.BaseModelNonconstantCrossSectionArea (Generic pressure drop component with constant turbulent loss factor data and without an icon, for non-constant cross section area).

Parameters

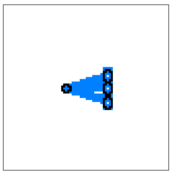
Type	Name	Description
replaceable package Medium		Medium in the component
LossFactorData	data	Loss factor data
Diameter	diameter_a	Inner diameter of pipe at port_a [m]
Diameter	diameter_b	Inner diameter of pipe at port_b [m]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]

AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed
Boolean	show_Re	= true, if Reynolds number is included for plotting
Boolean	show_totalPressures	= true, if total pressures are included for plotting
Boolean	show_portVelocities	= true, if port velocities are included for plotting

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.MultiPort



Multiply a port; useful if multiple connections shall be made to a port exposing a state

Information

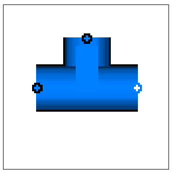
This model is useful if multiple connections shall be made to a port of a volume model exposing a state, like a pipe with ModelStructure av_vb. The mixing is shifted into the volume connected to port_a and the result is propagated back to each ports_b.

If multiple connections were directly made to the volume, then ideal mixing would take place in the connection set, outside the volume. This is normally not intended.

Connectors

Type	Name	Description
FluidPort_a	port_a	
FluidPorts_b	ports_b[nPorts_b]	

Fittings.TeeJunctionIdeal



Splitting/joining component with static balances for an infinitesimal control volume

Information

This model is the simplest implementation for a splitting/joining component for three flows. Its use is not required. It just formulates the balance equations in the same way that the connect symmantics would formulate them anyways. The main advantage of using this component is, that the user does not get confused when looking at the specific enthalpy at each port which might be confusing when not using a splitting/joining component. The reason for the confusion is that one exmanins the mixing enthalpy of the infinitesimal control volume introduced with the connect statement when looking at the specific enthalpy in the connector which might not be equal to the specific enthalpy at the port in the "real world".

Extends from Fittings.BaseClasses.PartialTeeJunction (Base class for a splitting/joining component with three ports).

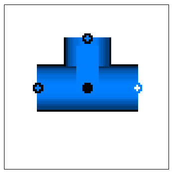
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component

Connectors

Type	Name	Description
FluidPort_a	port_1	
FluidPort_b	port_2	
FluidPort_a	port_3	

Fittings.TeeJunctionVolume



Splitting/joining component with static balances for a dynamic control volume

Information

This model introduces a mixing volume into a junction. This might be useful to examine the non-ideal mixing taking place in a real junction.

Extends from Fittings.BaseClasses.PartialTeeJunction (Base class for a splitting/joining component with three ports), Interfaces.PartialLumpedVolume (Lumped volume with mass and energy balance).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Volume	fluidVolume	Volume [m3]
Volume	V	Mixing volume inside junction [m3]
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances

Connectors

Type	Name	Description
FluidPort_a	port_1	
FluidPort_b	port_2	
FluidPort_a	port_3	

Fittings.BaseClasses

Base classes used in the Fittings package (only of interest to build new component models)

Package Content

Name	Description
lossConstant_D_zeta	Return the loss constant 8*zeta/(pi^2*D^4)
QuadraticTurbulent	Pressure loss components that are mainly defined by a quadratic turbulent regime with constant loss factor data
PartialTeeJunction	Base class for a splitting/joining component with three ports

---

Fittings.BaseClasses.lossConstant_D_zeta

Return the loss constant 8*zeta/(pi^2*D^4)

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Diameter	D	Diameter at port_a or port_b [m]
Real	zeta	Constant pressure loss factor with respect to D (i.e., either port_a or port_b)

Outputs

Type	Name	Description
Real	k	Loss constant (= 8*zeta/(pi^2*D^4))

Fittings.BaseClasses.QuadraticTurbulent

Pressure loss components that are mainly defined by a quadratic turbulent regime with constant loss factor data

Information

This library provides pressure loss factors of a pipe segment (orifice, bending etc.) with a minimum amount of data. If available, data can be provided for **both flow directions**, i.e., flow from port_a to port_b and from port_b to port_a, as well as for the **laminar** and the **turbulent** region. It is also an option to provide the loss factor **only** for the **turbulent** region for a flow from port_a to port_b. Basically, the pressure drop is defined by the following equation:

```
Δp = 0.5*ζ*ρ*v*|v|
   = 0.5*ζ/A^2 * (1/ρ) * m_flow*|m_flow|
   = 8*ζ/(π^2*D^4*ρ) * m_flow*|m_flow|
```

where

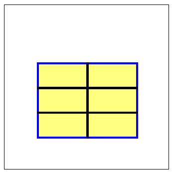
- Δp is the pressure drop: Δp = port_a.p - port_b.p
- v is the mean velocity.
- ρ is the density.

- ζ is the loss factor that depends on the geometry of the pipe. In the turbulent flow regime, it is assumed that ζ is constant and is given by "zeta1" and "zeta2" depending on the flow direction.
- D is the diameter of the pipe segment. If this is not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.

Package Content

Name	Description
LossFactorData	Data structure defining constant loss factor data for dp = zeta*rho*v*|v|/2 and functions providing the data for some loss types
massFlowRate_dp	Return mass flow rate from constant loss factor data and pressure drop (m_flow = f(dp))
massFlowRate_dp_and_Re	Return mass flow rate from constant loss factor data, pressure drop and Re (m_flow = f(dp))
pressureLoss_m_flow	Return pressure drop from constant loss factor and mass flow rate (dp = f(m_flow))
pressureLoss_m_flow_and_Re	Return pressure drop from constant loss factor, mass flow rate and Re (dp = f(m_flow))
BaseModel	Generic pressure drop component with constant turbulent loss factor data and without an icon
TestWallFriction	Pressure drop in pipe due to wall friction (only for test purposes; if needed use Pipes.StaticPipe instead)
BaseModelNonconstantCrossSectionArea	Generic pressure drop component with constant turbulent loss factor data and without an icon, for non-constant cross section area
pressureLoss_m_flow_totalPressure	Return pressure drop from constant loss factor and mass flow rate (dp = f(m_flow))

Fittings.BaseClasses.QuadraticTurbulent.LossFactorData



Data structure defining constant loss factor data for dp = zeta*rho*v*|v|/2 and functions providing the data for some loss types

Information

This record defines the pressure loss factors of a pipe segment (orifice, bending etc.) with a minimum amount of data. If available, data should be provided for **both flow directions**, i.e., flow from port_a to port_b and from port_b to port_a, as well as for the **laminar** and the **turbulent** region. It is also an option to provide the loss factor **only** for the **turbulent** region for a flow from port_a to port_b.

The following equations are used:

```
   Δp = 0.5*ζ*ρ*v*|v|
      = 0.5*ζ/A^2 * (1/ρ) * m_flow*|m_flow|
      = 8*ζ/(π^2*D^4*ρ) * m_flow*|m_flow|
        Re = |v|*D*ρ/μ
```

flow type	ζ =	flow region
turbulent	**zeta1** = const.	Re ≥ Re_turbulent, v ≥ 0
	zeta2 = const.	Re ≥ Re_turbulent, v < 0
laminar	**c0**/Re	both flow directions, Re small; c0 = const.

where

- Δp is the pressure drop: Δp = port_a.p - port_b.p
- v is the mean velocity.
- ρ is the density.
- ζ is the loss factor that depends on the geometry of the pipe. In the turbulent flow regime, it is assumed that ζ is constant and is given by "zeta1" and "zeta2" depending on the flow direction. When the Reynolds number Re is below "Re_turbulent", the flow is laminar for small flow velocities. For higher velocities there is a transition region from laminar to turbulent flow. The loss factor for laminar flow at small velocities is defined by the often occuring approximation c0/Re. If c0 is different for the two flow directions, the mean value has to be used (c0 = (c0_ab + c0_ba)/2).
- The equation "Δp = 0.5*ζ*ρ*v*|v|" is either with respect to port_a or to port_b, depending on the definition of the particular loss factor ζ (in some references loss factors are defined with respect to port_a, in other references with respect to port_b).
- Re = |v|*D_Re*ρ/μ = |m_flow|*D_Re/(A_Re*μ) is the Reynolds number at the smallest cross section area. This is often at port_a or at port_b, but can also be between the two ports. In the record, the diameter D_Re of this smallest cross section area has to be provided, as well, as Re_turbulent, the absolute value of the Reynolds number at which the turbulent flow starts. If Re_turbulent is different for the two flow directions, use the smaller value as Re_turbulent.
- D is the diameter of the pipe. If the pipe has not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.
- A is the cross section area with A = π(D/2)^2.
- μ is the dynamic viscosity.

The laminar and the transition region is usually of not much technical interest because the operating point is mostly in the turbulent regime. For simplification and for numercial reasons, this whole region is described by two polynomials of third order, one polynomial for m_flow ≥ 0 and one for m_flow < 0. The polynomials start at Re = |m_flow|*4/(π*D_Re*μ), where D_Re is the smallest diameter between port_a and port_b. The common derivative of the two polynomials at Re = 0 is computed from the equation "c0/Re". Note, the pressure drop equation above in the laminar region is always defined with respect to the smallest diameter D_Re.

If no data for c0 is available, the derivative at Re = 0 is computed in such a way, that the second derivatives of the two polynomials are identical at Re = 0. The polynomials are constructed, such that they smoothly touch the characteristic curves in the turbulent regions. The whole characteristic is therefore **continuous** and has a **finite**, **continuous first derivative everywhere**. In some cases, the constructed polynomials would "vibrate". This is avoided by reducing the derivative at Re=0 in such a way that the polynomials are guaranteed to be monotonically increasing. The used sufficient criteria for monotonicity follows from:

Fritsch F.N. and Carlson R.E. (1980):
: **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Modelica.Icons.Record (Icon for a record).

Parameters

Type	Name	Description
Diameter	diameter_a	Diameter at port_a [m]
Diameter	diameter_b	Diameter at port_b [m]
Real	zeta1	Loss factor for flow port_a -> port_b
Real	zeta2	Loss factor for flow port_b -> port_a
ReynoldsNumber	Re_turbulent	Loss factors suited for Re >= Re_turbulent [1]
Diameter	D_Re	Diameter used to compute Re [m]
Boolean	zeta1_at_a	dp = zeta1*(if zeta1_at_a then rho_a*v_a^2/2 else rho_b*v_b^2/2)
Boolean	zeta2_at_a	dp = -zeta2*(if zeta2_at_a then rho_a*v_a^2/2 else rho_b*v_b^2/2)
Boolean	zetaLaminarKnown	= true, if zeta = c0/Re in laminar region
Real	c0	zeta = c0/Re; dp = zeta*rho_Re*v_Re^2/2, Re=v_Re*D_Re*rho_Re/

		mu_Re)

Fittings.BaseClasses.QuadraticTurbulent.massFlowRate_dp

Return mass flow rate from constant loss factor data and pressure drop (m_flow = f(dp))

Information

Compute mass flow rate from constant loss factor and pressure drop (m_flow = f(dp)). For small pressure drops (dp < dp_small), the characteristic is approximated by a polynomial in order to have a finite derivative at zero mass flow rate.

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Pressure	dp	Pressure drop (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
LossFactorData	data	Constant loss factors for both flow directions
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Fittings.BaseClasses.QuadraticTurbulent.massFlowRate_dp_and_Re

Return mass flow rate from constant loss factor data, pressure drop and Re (m_flow = f(dp))

Information

Compute mass flow rate from constant loss factor and pressure drop (m_flow = f(dp)). If the Reynolds-number Re ≥ data.Re_turbulent, the flow is treated as a turbulent flow with constant loss factor zeta. If the Reynolds-number Re < data.Re_turbulent, the flow is laminar and/or in a transition region between laminar and turbulent. This region is approximated by two polynomials of third order, one polynomial for m_flow ≥ 0 and one for m_flow < 0. The common derivative of the two polynomials at Re = 0 is computed from the equation "data.c0/Re".

If no data for c0 is available, the derivative at Re = 0 is computed in such a way, that the second derivatives of the two polynomials are identical at Re = 0. The polynomials are constructed, such that they smoothly touch the characteristic curves in the turbulent regions. The whole characteristic is therefore **continuous** and has a **finite**, **continuous first derivative everywhere**. In some cases, the constructed polynomials would "vibrate". This is avoided by reducing the derivative at Re=0 in such a way that the polynomials are guaranteed to be monotonically increasing. The used sufficient criteria for monotonicity follows from:

Fritsch F.N. and Carlson R.E. (1980):
> **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Pressure	dp	Pressure drop (dp = port_a.p - port_b.p) [Pa]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b [Pa.s]
LossFactorData	data	Constant loss factors for both flow directions

Outputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]

Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow

Return pressure drop from constant loss factor and mass flow rate (dp = f(m_flow))

Information

Compute pressure drop from constant loss factor and mass flow rate (dp = f(m_flow)). For small mass flow rates(|m_flow| < m_flow_small), the characteristic is approximated by a polynomial in order to have a finite derivative at zero mass flow rate.

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
LossFactorData	data	Constant loss factors for both flow directions
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure drop (dp = port_a.p - port_b.p) [Pa]

Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow_and_Re

Return pressure drop from constant loss factor, mass flow rate and Re (dp = f(m_flow))

Information

Compute pressure drop from constant loss factor and mass flow rate (dp = f(m_flow)). If the Reynolds-number Re ≥ data.Re_turbulent, the flow is treated as a turbulent flow with constant loss factor zeta. If the Reynolds-number Re < data.Re_turbulent, the flow is laminar and/or in a transition region between laminar and turbulent. This region is approximated by two polynomials of third order, one polynomial for m_flow ≥ 0 and one for m_flow < 0. The common derivative of the two polynomials at Re = 0 is computed from the equation "data.c0/Re".

If no data for c0 is available, the derivative at Re = 0 is computed in such a way, that the second derivatives of the two polynomials are identical at Re = 0. The polynomials are constructed, such that they smoothly touch the characteristic curves in the turbulent regions. The whole characteristic is therefore **continuous** and has a **finite**, **continuous first derivative everywhere**. In some cases, the constructed polynomials would "vibrate". This is avoided by reducing the derivative at Re=0 in such a way that the polynomials are guaranteed to be monotonically increasing. The used sufficient criteria for monotonicity follows from:

Fritsch F.N. and Carlson R.E. (1980):
> **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Modelica.Icons.Function (Icon for a function).

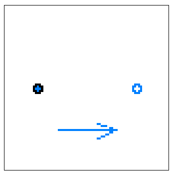
Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a	Density at port_a [kg/m3]
Density	rho_b	Density at port_b [kg/m3]
DynamicViscosity	mu_a	Dynamic viscosity at port_a [Pa.s]
DynamicViscosity	mu_b	Dynamic viscosity at port_b [Pa.s]
LossFactorData	data	Constant loss factors for both flow directions

Outputs

Type	Name	Description
Pressure	dp	Pressure drop (dp = port_a.p - port_b.p) [Pa]

Fittings.BaseClasses.QuadraticTurbulent.BaseModel



Generic pressure drop component with constant turbulent loss factor data and without an icon

Information

This model computes the pressure loss of a pipe segment (orifice, bending etc.) with a minimum amount of data provided via parameter **data**. If available, data should be provided for **both flow directions**, i.e., flow from port_a to port_b and from port_b to port_a, as well as for the **laminar** and the **turbulent** region. It is also an option to provide the loss factor **only** for the **turbulent** region for a flow from port_a to port_b.

The following equations are used:

```
Δp = 0.5*ζ*ρ*v*|v|
   = 0.5*ζ/A^2 * (1/ρ) * m_flow*|m_flow|
     Re = |v|*D*ρ/μ
```

flow type	ζ =	flow region
turbulent	**zeta1** = const.	Re ≥ Re_turbulent, v ≥ 0
	zeta2 = const.	Re ≥ Re_turbulent, v < 0
laminar	**c0**/Re	both flow directions, Re small; c0 = const.

where

- Δp is the pressure drop: Δp = port_a.p - port_b.p
- v is the mean velocity.
- ρ is the density.

- ζ is the loss factor that depends on the geometry of the pipe. In the turbulent flow regime, it is assumed that ζ is constant and is given by "zeta1" and "zeta2" depending on the flow direction. When the Reynolds number Re is below "Re_turbulent", the flow is laminar for small flow velocities. For higher velocities there is a transition region from laminar to turbulent flow. The loss factor for laminar flow at small velocities is defined by the often occuring approximation c0/Re. If c0 is different for the two flow directions, the mean value has to be used (c0 = (c0_ab + c0_ba)/2).
- The equation "Δp = 0.5*ζ*ρ*v*|v|" is either with respect to port_a or to port_b, depending on the definition of the particular loss factor ζ (in some references loss factors are defined with respect to port_a, in other references with respect to port_b).
- Re = |v|*D_Re*ρ/μ = |m_flow|*D_Re/(A_Re*μ) is the Reynolds number at the smallest cross section area. This is often at port_a or at port_b, but can also be between the two ports. In the record, the diameter D_Re of this smallest cross section area has to be provided, as well, as Re_turbulent, the absolute value of the Reynolds number at which the turbulent flow starts. If Re_turbulent is different for the two flow directions, use the smaller value as Re_turbulent.
- D is the diameter of the pipe. If the pipe has not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.
- A is the cross section area with A = π(D/2)^2.
- μ is the dynamic viscosity.

The laminar and the transition region is usually of not much technical interest because the operating point is mostly in the turbulent regime. For simplification and for numercial reasons, this whole region is described by two polynomials of third order, one polynomial for m_flow ≥ 0 and one for m_flow < 0. The polynomials start at Re = |m_flow|*4/(π*D_Re*μ), where D_Re is the smallest diameter between port_a and port_b. The common derivative of the two polynomials at Re = 0 is computed from the equation "c0/Re". Note, the pressure drop equation above in the laminar region is always defined with respect to the smallest diameter D_Re.

If no data for c0 is available, the derivative at Re = 0 is computed in such a way, that the second derivatives of the two polynomials are identical at Re = 0. The polynomials are constructed, such that they smoothly touch the characteristic curves in the turbulent regions. The whole characteristic is therefore **continuous** and has a **finite**, **continuous first derivative everywhere**. In some cases, the constructed polynomials would "vibrate". This is avoided by reducing the derivative at Re=0 in such a way that the polynomials are guaranteed to be monotonically increasing. The used sufficient criteria for monotonicity follows from:

Fritsch F.N. and Carlson R.E. (1980):
> **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy), Interfaces.PartialLumpedFlow (Base class for a lumped momentum balance).

Parameters

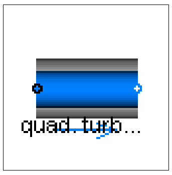
Type	Name	Description
replaceable package Medium		Medium in the component
Length	pathLength	Length flow path [m]
LossFactorData	data	Loss factor data
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	momentumDynamics	Formulation of momentum balance
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]

MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp) else dp = f(m_flow)
Boolean	use_Re	= true, if turbulent region is defined by Re, otherwise by dp_small or m_flow_small
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed
Boolean	show_Re	= true, if Reynolds number is included for plotting

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.BaseClasses.QuadraticTurbulent.TestWallFriction



Pressure drop in pipe due to wall friction (only for test purposes; if needed use Pipes.StaticPipe instead)

Information

Extends from BaseModel (Generic pressure drop component with constant turbulent loss factor data and without an icon).

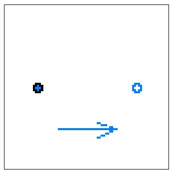
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
LossFactorData	data	Loss factor data
Length	length	Length of pipe [m]
Diameter	diameter	Inner diameter of pipe [m]
Length	roughness	Absolute roughness of pipe (> 0 required, details see info layer) [m]
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Boolean	from_dp	= true, use m_flow = f(dp) else dp = f(m_flow)
Boolean	use_Re	= true, if turbulent region is defined by Re, otherwise by dp_small or m_flow_small
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed
Boolean	show_Re	= true, if Reynolds number is included for plotting

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.BaseClasses.QuadraticTurbulent.BaseModelNonconstantCrossSectionArea



Generic pressure drop component with constant turbulent loss factor data and without an icon, for non-constant cross section area

Information

This model computes the pressure loss of a pipe segment (orifice, bending etc.) with a minimum amount of data provided via parameter **data**. If available, data should be provided for **both flow directions**, i.e., flow from port_a to port_b and from port_b to port_a, as well as for the **laminar** and the **turbulent** region. It is also an option to provide the loss factor **only** for the **turbulent** region for a flow from port_a to port_b.

The following equations are used:

```
Δp = 0.5*ζ*ρ*v*|v|
   = 0.5*ζ/A^2 * (1/ρ) * m_flow*|m_flow|
    Re = |v|*D*ρ/μ
```

flow type	ζ =	flow region
turbulent	**zeta1** = const.	Re ≥ Re_turbulent, v ≥ 0
	zeta2 = const.	Re ≥ Re_turbulent, v < 0
laminar	**c0**/Re	both flow directions, Re small; c0 = const.

where

- Δp is the pressure drop: Δp = port_a.p - port_b.p
- v is the mean velocity.
- ρ is the density.
- ζ is the loss factor that depends on the geometry of the pipe. In the turbulent flow regime, it is assumed that ζ is constant and is given by "zeta1" and "zeta2" depending on the flow direction. When the Reynolds number Re is below "Re_turbulent", the flow is laminar for small flow velocities. For higher velocities there is a transition region from laminar to turbulent flow. The loss factor for laminar flow at small velocities is defined by the often occuring approximation c0/Re. If c0 is different for the two flow directions, the mean value has to be used (c0 = (c0_ab + c0_ba)/2).
- The equation "Δp = 0.5*ζ*ρ*v*|v|" is either with respect to port_a or to port_b, depending on the definition of the particular loss factor ζ (in some references loss factors are defined with respect to port_a, in other references with respect to port_b).
- Re = |v|*D_Re*ρ/μ = |m_flow|*D_Re/(A_Re*μ) is the Reynolds number at the smallest cross section area. This is often at port_a or at port_b, but can also be between the two ports. In the record, the diameter D_Re of this smallest cross section area has to be provided, as well, as Re_turbulent, the absolute value of the Reynolds number at which the turbulent flow starts. If Re_turbulent is different for the two flow directions, use the smaller value as Re_turbulent.
- D is the diameter of the pipe. If the pipe has not a circular cross section, D = 4*A/P, where A is the cross section area and P is the wetted perimeter.
- A is the cross section area with A = π(D/2)^2.
- μ is the dynamic viscosity.

The laminar and the transition region is usually of not much technical interest because the operating point is mostly in the turbulent regime. For simplification and for numercial reasons, this whole region is described by two polynomials of third order, one polynomial for m_flow ≥ 0 and one for m_flow < 0. The polynomials start at Re = |m_flow|*4/(π*D_Re*μ), where D_Re is the smallest diameter between port_a and port_b. The common derivative of the two polynomials at Re = 0 is computed from the equation "c0/Re". Note, the pressure drop equation above in the laminar region is always defined with respect to the smallest diameter D_Re.

If no data for c0 is available, the derivative at Re = 0 is computed in such a way, that the second derivatives of the two polynomials are identical at Re = 0. The polynomials are constructed, such that they smoothly touch the characteristic curves in the turbulent regions. The whole characteristic is therefore **continuous** and has a **finite**, **continuous first derivative everywhere**. In some cases, the constructed polynomials would "vibrate". This is avoided by reducing the derivative at Re=0 in such a way that the polynomials are guaranteed to be monotonically increasing. The used sufficient criteria for monotonicity follows from:

Fritsch F.N. and Carlson R.E. (1980):
> **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Interfaces.PartialTwoPortTransport (Partial element transporting fluid between two ports without storage of mass or energy), Interfaces.PartialLumpedFlow (Base class for a lumped momentum balance).

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Length	pathLength	Length flow path [m]
LossFactorData	data	Loss factor data
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Dynamics		
Dynamics	momentumDynamics	Formulation of momentum balance
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
AbsolutePressure	dp_small	Turbulent flow if |dp| >= dp_small [Pa]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed
Boolean	show_Re	= true, if Reynolds number is included for plotting
Boolean	show_totalPressures	= true, if total pressures are included for plotting
Boolean	show_portVelocities	= true, if port velocities are included for plotting

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Fittings.BaseClasses.QuadraticTurbulent.pressureLoss_m_flow_totalPressure

Return pressure drop from constant loss factor and mass flow rate (dp = f(m_flow))

Information

Compute pressure drop from constant loss factor and mass flow rate (dp = f(m_flow)). For small mass flow rates(|m_flow| < m_flow_small), the characteristic is approximated by a polynomial in order to have a finite derivative at zero mass flow rate.

Extends from Modelica.Icons.Function (Icon for a function).

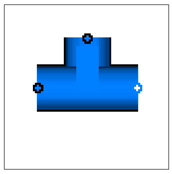
Inputs

Type	Name	Description
MassFlowRate	m_flow	Mass flow rate from port_a to port_b [kg/s]
Density	rho_a_des	Density at port_a, mass flow in design direction a -> b [kg/m3]
Density	rho_b_des	Density at port_b, mass flow in design direction a -> b [kg/m3]
Density	rho_b_nondes	Density at port_b, mass flow against design direction a <- b [kg/m3]
Density	rho_a_nondes	Density at port_a, mass flow against design direction a <- b [kg/m3]
LossFactorData	data	Constant loss factors for both flow directions
MassFlowRate	m_flow_small	Turbulent flow if |m_flow| >= m_flow_small [kg/s]

Outputs

Type	Name	Description
Pressure	dp	Pressure drop (dp = port_a.p - port_b.p) [Pa]

Fittings.BaseClasses.PartialTeeJunction



Base class for a splitting/joining component with three ports

Connectors

Type	Name	Description
FluidPort_a	port_1	
FluidPort_b	port_2	
FluidPort_a	port_3	

Modelica_Fluid.Sources

Define fixed or prescribed boundary conditions

Information

Package **Sources** contains generic sources for fluid connectors to define fixed or prescribed ambient conditions.

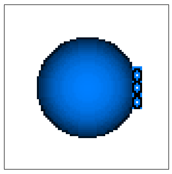
Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
FixedBoundary	Boundary source component
Boundary_pT	Boundary with prescribed pressure, temperature, composition and trace substances

Boundary_ph	Boundary with prescribed pressure, specific enthalpy, composition and trace substances
MassFlowSource_T	Ideal flow source that produces a prescribed mass flow with prescribed temperature, mass fraction and trace substances
MassFlowSource_h	Ideal flow source that produces a prescribed mass flow with prescribed specific enthalpy, mass fraction and trace substances
BaseClasses	Base classes used in the Sources package (only of interest to build new component models)

Sources.FixedBoundary



Boundary source component

Information

Model **FixedBoundary** defines constant values for boundary conditions:

- Boundary pressure or boundary density.
- Boundary temperature or boundary specific enthalpy.
- Boundary composition (only for multi-substance or trace-substance flow).

Note, that boundary temperature, density, specific enthalpy, mass fractions and trace substances have only an effect if the mass flow is from the Boundary into the port. If mass is flowing from the port into the boundary, the boundary definitions, with exception of boundary pressure, do not have an effect.

Extends from Sources.BaseClasses.PartialSource (Partial component source with one fluid connector).

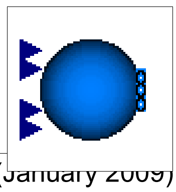
Parameters

Type	Name	Description
replaceable package Medium		Medium model within the source
Boundary pressure or Boundary density		
Boolean	use_p	select p or d
AbsolutePressure	p	Boundary pressure [Pa]
Density	d	Boundary density [kg/m3]
Boundary temperature or Boundary specific enthalpy		
Boolean	use_T	select T or h
Temperature	T	Boundary temperature [K]
SpecificEnthalpy	h	Boundary specific enthalpy [J/kg]
Only for multi-substance flow		
MassFraction	X[Medium.nX]	Boundary mass fractions m_i/m [kg/kg]
Only for trace-substance flow		
ExtraProperty	C[Medium.nC]	Boundary trace substances

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	

Sources.Boundary_pT



Boundary with prescribed pressure, temperature, composition and trace substances

Information

Defines prescribed values for boundary conditions:

- Prescribed boundary pressure.
- Prescribed boundary temperature.
- Boundary composition (only for multi-substance or trace-substance flow).

If `use_p_in` is false (default option), the `p` parameter is used as boundary pressure, and the `p_in` input connector is disabled; if `use_p_in` is true, then the `p` parameter is ignored, and the value provided by the input connector is used instead.

The same thing goes for the temperature, composition and trace substances.

Note, that boundary temperature, mass fractions and trace substances have only an effect if the mass flow is from the boundary into the port. If mass is flowing from the port into the boundary, the boundary definitions, with exception of boundary pressure, do not have an effect.

Extends from Sources.BaseClasses.PartialSource (Partial component source with one fluid connector).

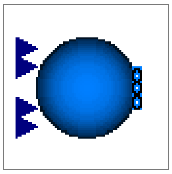
Parameters

Type	Name	Description
replaceable package Medium		Medium model within the source
Boolean	use_p_in	Get the pressure from the input connector
Boolean	use_T_in	Get the temperature from the input connector
Boolean	use_X_in	Get the composition from the input connector
Boolean	use_C_in	Get the trace substances from the input connector
AbsolutePressure	p	Fixed value of pressure [Pa]
Temperature	T	Fixed value of temperature [K]
MassFraction	X[Medium.nX]	Fixed value of composition [kg/kg]
ExtraProperty	C[Medium.nC]	Fixed values of trace substances

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	
input RealInput	p_in	Prescribed boundary pressure
input RealInput	T_in	Prescribed boundary temperature
input RealInput	X_in[Medium.nX]	Prescribed boundary composition
input RealInput	C_in[Medium.nC]	Prescribed boundary trace substances

Sources.Boundary_ph



Boundary with prescribed pressure, specific enthalpy, composition and trace substances

Information

Defines prescribed values for boundary conditions:

- Prescribed boundary pressure.
- Prescribed boundary temperature.
- Boundary composition (only for multi-substance or trace-substance flow).

If `use_p_in` is false (default option), the `p` parameter is used as boundary pressure, and the `p_in` input connector is disabled; if `use_p_in` is true, then the `p` parameter is ignored, and the value provided by the input connector is used instead.

The same thing goes for the specific enthalpy and composition

Note, that boundary temperature, mass fractions and trace substances have only an effect if the mass flow is from the boundary into the port. If mass is flowing from the port into the boundary, the boundary definitions, with exception of boundary pressure, do not have an effect.

Extends from Sources.BaseClasses.PartialSource (Partial component source with one fluid connector).

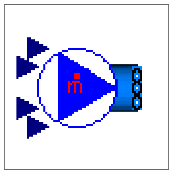
Parameters

Type	Name	Description
replaceable package Medium		Medium model within the source
Boolean	use_p_in	Get the pressure from the input connector
Boolean	use_h_in	Get the specific enthalpy from the input connector
Boolean	use_X_in	Get the composition from the input connector
Boolean	use_C_in	Get the trace substances from the input connector
AbsolutePressure	p	Fixed value of pressure [Pa]
SpecificEnthalpy	h	Fixed value of specific enthalpy [J/kg]
MassFraction	X[Medium.nX]	Fixed value of composition [kg/kg]
ExtraProperty	C[Medium.nC]	Fixed values of trace substances

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	
input RealInput	p_in	Prescribed boundary pressure
input RealInput	h_in	Prescribed boundary specific enthalpy
input RealInput	X_in[Medium.nX]	Prescribed boundary composition
input RealInput	C_in[Medium.nC]	Prescribed boundary trace substances

Sources.MassFlowSource_T



Ideal flow source that produces a prescribed mass flow with prescribed temperature, mass fraction and trace substances

Information

Models an ideal flow source, with prescribed values of flow rate, temperature, composition and trace substances:

- Prescribed mass flow rate.
- Prescribed temperature.
- Boundary composition (only for multi-substance or trace-substance flow).

If `use_m_flow_in` is false (default option), the `m_flow` parameter is used as boundary pressure, and the `m_flow_in` input connector is disabled; if `use_m_flow_in` is true, then the `m_flow` parameter is ignored, and the value provided by the input connector is used instead.

The same thing goes for the temperature and composition

Note, that boundary temperature, mass fractions and trace substances have only an effect if the mass flow is from the boundary into the port. If mass is flowing from the port into the boundary, the boundary definitions, with exception of boundary flow rate, do not have an effect.

Extends from Sources.BaseClasses.PartialSource (Partial component source with one fluid connector).

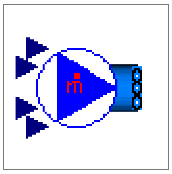
Parameters

Type	Name	Description
replaceable package Medium		Medium model within the source
Boolean	use_m_flow_in	Get the mass flow rate from the input connector
Boolean	use_T_in	Get the temperature from the input connector
Boolean	use_X_in	Get the composition from the input connector
Boolean	use_C_in	Get the trace substances from the input connector
MassFlowRate	m_flow	Fixed mass flow rate going out of the fluid port [kg/s]
Temperature	T	Fixed value of temperature [K]
MassFraction	X[Medium.nX]	Fixed value of composition [kg/kg]
ExtraProperty	C[Medium.nC]	Fixed values of trace substances

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	
input RealInput	m_flow_in	Prescribed mass flow rate
input RealInput	T_in	Prescribed fluid temperature
input RealInput	X_in[Medium.nX]	Prescribed fluid composition
input RealInput	C_in[Medium.nC]	Prescribed boundary trace substances

Sources.MassFlowSource_h



Ideal flow source that produces a prescribed mass flow with prescribed specific enthalpy, mass fraction and trace substances

Information

Models an ideal flow source, with prescribed values of flow rate, temperature and composition:

- Prescribed mass flow rate.
- Prescribed specific enthalpy.
- Boundary composition (only for multi-substance or trace-substance flow).

If `use_m_flow_in` is false (default option), the `m_flow` parameter is used as boundary pressure, and the `m_flow_in` input connector is disabled; if `use_m_flow_in` is true, then the `m_flow` parameter is ignored, and the value provided by the input connector is used instead.

The same thing goes for the temperature and composition

Note, that boundary temperature, mass fractions and trace substances have only an effect if the mass flow is from the boundary into the port. If mass is flowing from the port into the boundary, the boundary definitions, with exception of boundary flow rate, do not have an effect.

Extends from Sources.BaseClasses.PartialSource (Partial component source with one fluid connector).

Parameters

Type	Name	Description
replaceable package Medium		Medium model within the source
Boolean	use_m_flow_in	Get the mass flow rate from the input connector
Boolean	use_h_in	Get the specific enthalpy from the input connector
Boolean	use_X_in	Get the composition from the input connector

Boolean	use_C_in	Get the trace substances from the input connector
MassFlowRate	m_flow	Fixed mass flow rate going out of the fluid port [kg/s]
SpecificEnthalpy	h	Fixed value of specific enthalpy [J/kg]
MassFraction	X[Medium.nX]	Fixed value of composition [kg/kg]
ExtraProperty	C[Medium.nC]	Fixed values of trace substances

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	
input RealInput	m_flow_in	Prescribed mass flow rate
input RealInput	h_in	Prescribed fluid specific enthalpy
input RealInput	X_in[Medium.nX]	Prescribed fluid composition
input RealInput	C_in[Medium.nC]	Prescribed boundary trace substances

---

Sources.BaseClasses

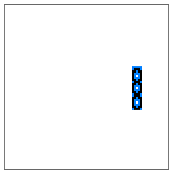
Base classes used in the Sources package (only of interest to build new component models)

Package Content

Name	Description
PartialSource	Partial component source with one fluid connector

---

Sources.BaseClasses.PartialSource



Partial component source with one fluid connector

Information

Partial component to model the **volume interface** of a **source** component, such as a mass flow source. The essential features are:

- The pressure in the connection port (= ports.p) is identical to the pressure in the volume.
- The outflow enthalpy rate (= port.h_outflow) and the composition of the substances (= port.Xi_outflow) are identical to the respective values in the volume.

Connectors

Type	Name	Description
FluidPorts_b	ports[nPorts]	

Modelica_Fluid.Sensors

Ideal sensor components to extract signals from a fluid connector

Information

Package **Sensors** consists of idealized sensor components that provide variables of a medium model and/or fluid ports as output signals. These signals can be, e.g., further processed with components of the

Modelica.Blocks library. Also more realistic sensor models can be built, by further processing (e.g., by attaching block Modelica.Blocks.FirstOrder to model the time constant of the sensor).

For the thermodynamic state variables temperature, specific entalpy, specific entropy and density the fluid library provides two different types of sensors: **regular one port** and **two port** sensors.

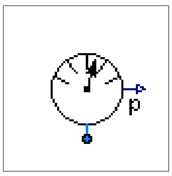
- The **regular one port** sensors have the advantage of easy introduction and removal from a model, as no connections have to be broken. A potential drawback is that the obtained value jumps as flow reverts. Test.TestComponents.Sensors.TestTemperatureSensor provides a test case, which demonstrates this.
- The **two port** sensors offer the advantages of an adjustable regularized step function around zero flow. Moreover the obtained result is restricted to the value flowing into port_a if allowFlowReversal is false.

Extends from Icons.VariantLibrary (Icon for a library that contains several variants of one component).

Package Content

Name	Description
Pressure	Ideal pressure sensor
Density	Ideal one port density sensor
DensityTwoPort	Ideal two port density sensor
Temperature	Ideal one port temperature sensor
TemperatureTwoPort	Ideal two port temperature sensor
SpecificEnthalpy	Ideal one port specific enthalpy sensor
SpecificEnthalpyTwoPort	Ideal two port sensor for the specific enthalpy
SpecificEntropy	Ideal one port specific entropy sensor
SpecificEntropyTwoPort	Ideal two port sensor for the specific entropy
TraceSubstances	Ideal one port trace substances sensor
TraceSubstancesTwoPort	Ideal two port sensor for trace substance
MassFlowRate	Ideal sensor for mass flow rate
VolumeFlowRate	Ideal sensor for volume flow rate
RelativePressure	Ideal relative pressure sensor
RelativeTemperature	Ideal relative temperature sensor
BaseClasses	Base classes used in the Sensors package (only of interest to build new component models)

Sensors.Pressure



Ideal pressure sensor

Information

This component monitors the absolute pressure at its fluid port. The sensor is ideal, i.e., it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

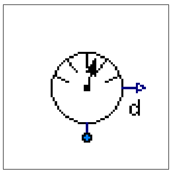
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	p	Pressure at port [Pa]

Sensors.Density



Ideal one port density sensor

Information

This component monitors the density of the fluid passing its port. The sensor is ideal, i.e. it does not influence the fluid.

If using the one port sensor please read the Information first.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

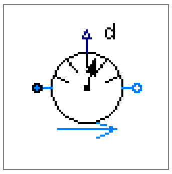
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	d	Density in port medium [kg/m3]

Sensors.DensityTwoPort



Ideal two port density sensor

Information

This component monitors the density of the fluid flowing from port_a to port_b. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

Parameters

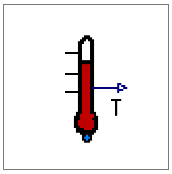
Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a ->

		port_b)
Advanced		
MassFlowRate	m_flow_small	For bi-directional flow, density is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	d	Density of the passing fluid [kg/m3]

Sensors.Temperature



Ideal one port temperature sensor

Information

This component monitors the temperature of the fluid passing its port. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable).

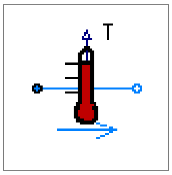
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	T	Temperature in port medium [K]

Sensors.TemperatureTwoPort



Ideal two port temperature sensor

Information

This component monitors the temperature of the passing fluid. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties).

Parameters

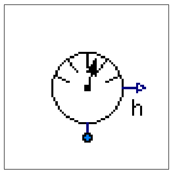
Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a ->

		port_b)
Advanced		
MassFlowRate	m_flow_small	For bi-directional flow, temperature is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	T	Temperature of the passing fluid [K]

Sensors.SpecificEnthalpy



Ideal one port specific enthalpy sensor

Information

This component monitors the specific enthalpy of the fluid passing its port. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

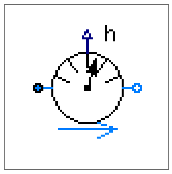
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	h_out	Specific enthalpy in port medium [J/kg]

Sensors.SpecificEnthalpyTwoPort



Ideal two port sensor for the specific enthalpy

Information

This component monitors the specific enthalpy of a passing fluid. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

Parameters

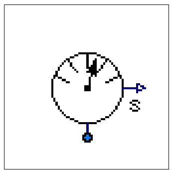
Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		

<table>
<tr><td>Boolean</td><td>allowFlowReversal</td><td>= true to allow flow reversal, false restricts to design direction (port_a -> port_b)</td></tr>
<tr><td colspan="3">Advanced</td></tr>
<tr><td>MassFlowRate</td><td>m_flow_small</td><td>For bi-directional flow, specific enthalpy is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]</td></tr>
</table>

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	h_out	Specific enthalpy of the passing fluid [J/kg]

Sensors.SpecificEntropy



Ideal one port specific entropy sensor

Information

This component monitors the specific entropy of the fluid passing its port. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

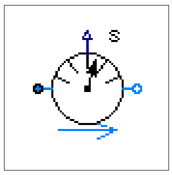
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	s	Specific entropy in port medium [J/(kg.K)]

Sensors.SpecificEntropyTwoPort



Ideal two port sensor for the specific entropy

Information

This component monitors the specific entropy of the passing fluid. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

Parameters

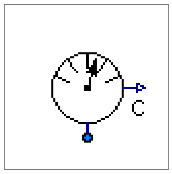
Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		

Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
MassFlowRate	m_flow_small	For bi-directional flow, specific entropy is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	s	Specific entropy of the passing fluid [J/(kg.K)]

Sensors.TraceSubstances



Ideal one port trace substances sensor

Information

This component monitors the trace substances contained in the fluid passing its port. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialAbsoluteSensor (Partial component to model a sensor that measures a potential variable), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

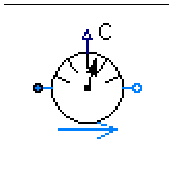
Parameters

Type	Name	Description
replaceable package Medium		Medium in the sensor
String	substanceName	Name of trace substance

Connectors

Type	Name	Description
FluidPort_a	port	
output RealOutput	C	Trace substance in port medium

Sensors.TraceSubstancesTwoPort



Ideal two port sensor for trace substance

Information

This component monitors the trace substance of the passing fluid. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

Parameters

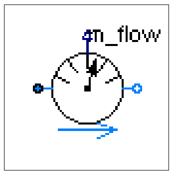
Type	Name	Description
replaceable package Medium		Medium in the component

String	substanceName	Name of trace substance
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
MassFlowRate	m_flow_small	For bi-directional flow, trace substance is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	C	Trace substance of the passing fluid

Sensors.MassFlowRate



Ideal sensor for mass flow rate

Information

This component monitors the mass flow rate flowing from port_a to port_b. The sensor is ideal, i.e., it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

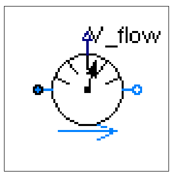
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	m_flow	Mass flow rate from port_a to port_b [kg/s]

Sensors.VolumeFlowRate



Ideal sensor for volume flow rate

Information

This component monitors the volume flow rate flowing from port_a to port_b. The sensor is ideal, i.e. it does not influence the fluid.

Extends from Sensors.BaseClasses.PartialFlowSensor (Partial component to model sensors that measure flow properties), Modelica.Icons.RotationalSensor (Icon representing rotational measurement device).

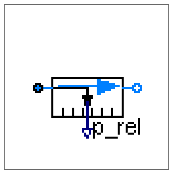
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
MassFlowRate	m_flow_small	For bi-directional flow, density is regularized in the region |m_flow| < m_flow_small (m_flow_small > 0 required) [kg/s]

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)
output RealOutput	V_flow	Volume flow rate from port_a to port_b [m3/s]

Sensors.RelativePressure



Ideal relative pressure sensor

Information

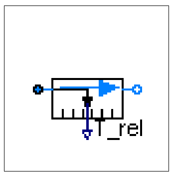
The relative pressure "port_a.p - port_b.p" is determined between the two ports of this component and is provided as output signal. The sensor should be connected in parallel with other equipment, no flow through the sensor is allowed.

Extends from Modelica.Icons.TranslationalSensor (Icon representing translational measurement device).

Connectors

Type	Name	Description
FluidPort_a	port_a	
FluidPort_b	port_b	
output RealOutput	p_rel	Relative pressure signal [Pa]

Sensors.RelativeTemperature



Ideal relative temperature sensor

Information

The relative temperature "T(port_a) - T(port_b)" is determined between the two ports of this component and is provided as output signal. The sensor should be connected in parallel with other equipment, no flow through the sensor is allowed.

Extends from Modelica.Icons.TranslationalSensor (Icon representing translational measurement device).

Connectors

Type	Name	Description
FluidPort_a	port_a	

FluidPort_b	port_b	
output RealOutput	T_rel	Relative temperature signal [K]

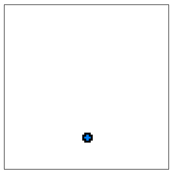
Sensors.BaseClasses

Base classes used in the Sensors package (only of interest to build new component models)

Package Content

Name	Description
PartialAbsoluteSensor	Partial component to model a sensor that measures a potential variable
PartialFlowSensor	Partial component to model sensors that measure flow properties

Sensors.BaseClasses.PartialAbsoluteSensor



Partial component to model a sensor that measures a potential variable

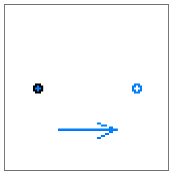
Information

Partial component to model an **absolute sensor**. Can be used for pressure sensor models. Use for other properties such as temperature or density is discouraged, because the enthalpy at the connector can have different meanings, depending on the connection topology. Use `PartialFlowSensor` instead. as signal.

Connectors

Type	Name	Description
FluidPort_a	port	

Sensors.BaseClasses.PartialFlowSensor



Partial component to model sensors that measure flow properties

Information

Partial component to model a **sensor** that measures any intensive properties of a flow, e.g., to get temperature or density in the flow between fluid connectors.
The model includes zero-volume balance equations. Sensor models inheriting from this partial class should add a medium instance to calculate the measured property.

Extends from Interfaces.PartialTwoPort (Partial component with two ports).

Parameters

Type	Name	Description
replaceable package	Medium	Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)

Connectors

Type	Name	Description

FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Modelica_Fluid.Interfaces

Interfaces for steady state and unsteady, mixed-phase, multi-substance, incompressible and compressible flow

Information

Extends from Modelica.Icons.Library (Icon for library).

Package Content

Name	Description
FluidPort	Interface for quasi one-dimensional fluid flow in a piping network (incompressible or compressible, one or more phases, one or more substances)
FluidPort_a	Generic fluid connector at design inlet
FluidPort_b	Generic fluid connector at design outlet
FluidPorts_a	Fluid connector with filled, large icon to be used for vectors of FluidPorts (vector dimensions must be added after dragging)
FluidPorts_b	Fluid connector with outlined, large icon to be used for vectors of FluidPorts (vector dimensions must be added after dragging)
PartialTwoPort	Partial component with two ports
PartialTwoPortTransport	Partial element transporting fluid between two ports without storage of mass or energy
HeatPorts_a	HeatPort connector with filled, large icon to be used for vectors of HeatPorts (vector dimensions must be added after dragging)
HeatPorts_b	HeatPort connector with filled, large icon to be used for vectors of HeatPorts (vector dimensions must be added after dragging)
PartialHeatTransfer	Common interface for heat transfer models
PartialLumpedVolume	Lumped volume with mass and energy balance
PartialLumpedFlow	Base class for a lumped momentum balance
PartialDistributedVolume	Base class for distributed volume models
PartialDistributedFlow	Base class for a distributed momentum balance

Interfaces.FluidPort

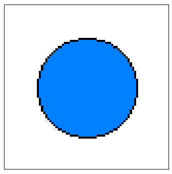
Interface for quasi one-dimensional fluid flow in a piping network (incompressible or compressible, one or more phases, one or more substances)

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]

stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Interfaces.FluidPort_a



Generic fluid connector at design inlet

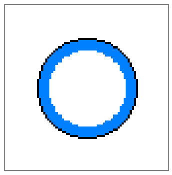
Parameters

Type	Name	Description
replaceable package	Medium	Medium model

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Interfaces.FluidPort_b



Generic fluid connector at design outlet

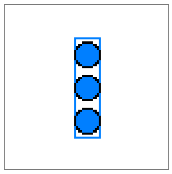
Parameters

Type	Name	Description
replaceable package	Medium	Medium model

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Interfaces.FluidPorts_a



Fluid connector with filled, large icon to be used for vectors of FluidPorts (vector dimensions must be added after dragging)

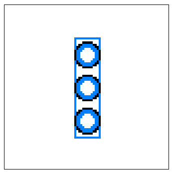
Parameters

Type	Name	Description
replaceable package	Medium	Medium model

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Interfaces.FluidPorts_b



Fluid connector with outlined, large icon to be used for vectors of FluidPorts (vector dimensions must be added after dragging)

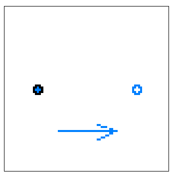
Parameters

Type	Name	Description
replaceable package	Medium	Medium model

Contents

Type	Name	Description
flow MassFlowRate	m_flow	Mass flow rate from the connection point into the component [kg/s]
AbsolutePressure	p	Thermodynamic pressure in the connection point [Pa]
stream SpecificEnthalpy	h_outflow	Specific thermodynamic enthalpy close to the connection point if m_flow < 0 [J/kg]
stream MassFraction	Xi_outflow[Medium.nXi]	Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0 [kg/kg]
stream ExtraProperty	C_outflow[Medium.nC]	Properties c_i/m close to the connection point if m_flow < 0

Interfaces.PartialTwoPort



Partial component with two ports

Information

This partial model defines an interface for components with two ports. The treatment of the design flow direction and of flow reversal are predefined based on the parameter **allowFlowReversal**. The component may transport fluid and may have internal storage for a given fluid **Medium**.

An extending model providing direct access to internal storage of mass or energy through port_a or port_b should redefine the protected parameters **port_a_exposesState** and **port_b_exposesState** appropriately. This will be visualized at the port icons, in order to improve the understanding of fluid model diagrams.

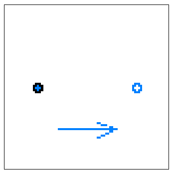
Parameters

Type	Name	Description
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Interfaces.PartialTwoPortTransport



Partial element transporting fluid between two ports without storage of mass or energy

Information

This component transports fluid between its two ports, without storing mass or energy. Energy may be exchanged with the environment though, e.g. in the form of work. `PartialTwoPortTransport` is intended as base class for devices like orifices, valves and simple fluid machines.

Three equations need to be added by an extending class using this component:

- the momentum balance specifying the relationship between the pressure drop `dp` and the mass flow rate `m_flow`,
- `port_b.h_outflow` for flow in design direction, and
- `port_a.h_outflow` for flow in reverse direction.

Extends from PartialTwoPort (Partial component with two ports).

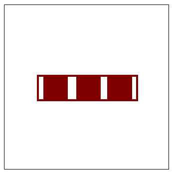
Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (port_a -> port_b)
Advanced		
AbsolutePressure	dp_start	Guess value of dp = port_a.p - port_b.p [Pa]
MassFlowRate	m_flow_start	Guess value of m_flow = port_a.m_flow [kg/s]
MassFlowRate	m_flow_small	Small mass flow rate for regularization of zero flow [kg/s]
Diagnostics		
Boolean	show_T	= true, if temperatures at port_a and port_b are computed
Boolean	show_V_flow	= true, if volume flow rate at inflowing port is computed

Connectors

Type	Name	Description
FluidPort_a	port_a	Fluid connector a (positive design flow direction is from port_a to port_b)
FluidPort_b	port_b	Fluid connector b (positive design flow direction is from port_a to port_b)

Interfaces.HeatPorts_a

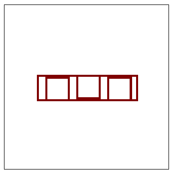


HeatPort connector with filled, large icon to be used for vectors of HeatPorts (vector dimensions must be added after dragging)

Contents

Type	Name	Description
Temperature	T	Port temperature [K]
flow HeatFlowRate	Q_flow	Heat flow rate (positive if flowing from outside into the component) [W]

Interfaces.HeatPorts_b

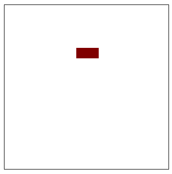


HeatPort connector with filled, large icon to be used for vectors of HeatPorts (vector dimensions must be added after dragging)

Contents

Type	Name	Description
Temperature	T	Port temperature [K]
flow HeatFlowRate	Q_flow	Heat flow rate (positive if flowing from outside into the component) [W]

Interfaces.PartialHeatTransfer



Common interface for heat transfer models

Information

This component is a common interface for heat transfer models. The heat flow rates `Q_flows[n]` through the boundaries of n flow segments are obtained as function of the thermodynamic `states` of the flow segments for a given fluid `Medium`, the `surfaceAreas[n]` and the boundary temperatures `heatPorts[n].T`.

The heat loss coefficient `k` can be used to model a thermal isolation between `heatPorts.T` and `T_ambient`.

An extending model implementing this interface needs to define one equation: the relation between the predefined fluid temperatures `Ts[n]`, the boundary temperatures `heatPorts[n].T`, and the heat flow rates `Q_flows[n]`.

Parameters

Type	Name	Description
Ambient		
CoefficientOfHeatTransfer	k	Heat transfer coefficient to ambient [W/(m2.K)]
Temperature	T_ambient	Ambient temperature [K]

Internal Interface		
Integer	n	Number of heat transfer segments
Boolean	use_k	= true to use k value for thermal isolation

Connectors

Type	Name	Description
HeatPorts_a	heatPorts[n]	Heat port to component boundary

Interfaces.PartialLumpedVolume

Lumped volume with mass and energy balance

Information

Interface and base class for an ideally mixed fluid volume with the ability to store mass and energy. The following boundary flow and source terms are part of the energy balance and must be specified in an extending class:

- **`Qb_flow`**, e.g. convective or latent heat flow rate across segment boundary, and
- **`Wb_flow`**, work term, e.g. p*der(fluidVolume) if the volume is not constant.

The component volume **`fluidVolume`** is an input that needs to be set in the extending class to complete the model.

Further source terms must be defined by an extending class for fluid flow across the segment boundary:

- **`Hb_flow`**, enthalpy flow,
- **`mb_flow`**, mass flow,
- **`mbXi_flow`**, substance mass flow, and
- **`mbC_flow`**, trace substance mass flow.

Parameters

Type	Name	Description
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balance
Dynamics	massDynamics	Formulation of mass balance
Initialization		
AbsolutePressure	p_start	Start value of pressure [Pa]
Boolean	use_T_start	= true, use T_start, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances

---

Interfaces.PartialLumpedFlow

Base class for a lumped momentum balance

Information

Interface and base class for a momentum balance, defining the mass flow rate **m_flow** of a given Medium in a flow model.

The following boundary flow and force terms are part of the momentum balance and must be specified in an extending model (to zero if not considered):

- **Ib_flow**, the flow of momentum across model boundaries,
- **F_p[m]**, pressure force, and
- **F_fg[m]**, friction and gravity forces.

The length of the flow path **pathLength** is an input that needs to be set in an extending class to complete the model.

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (m_flow >= 0)
Dynamics		
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]

Connectors

Type	Name	Description
replaceable package Medium		Medium in the component

Interfaces.PartialDistributedVolume

Base class for distributed volume models

Information

Interface and base class for **n** ideally mixed fluid volumes with the ability to store mass and energy. It is inteded to model a one-dimensional spatial discretization of fluid flow according to the finite volume method. The following boundary flow and source terms are part of the energy balance and must be specified in an extending class:

- **Qb_flows[n]**, heat flow term, e.g. conductive heat flows across segment boundaries, and
- **Wb_flows[n]**, work term.

The component volumes **fluidVolumes[n]** are an input that needs to be set in an extending class to complete the model.

Further source terms must be defined by an extending class for fluid flow across the segment boundary:

- **Hb_flows[n]**, enthalpy flow,
- **mb_flows[n]**, mass flow,
- **mbXi_flows[n]**, substance mass flow, and
- **mbC_flows[n]**, trace substance mass flow.

Parameters

Type	Name	Description
Integer	n	Number of discrete volumes
Assumptions		
Dynamics		
Dynamics	energyDynamics	Formulation of energy balances
Dynamics	massDynamics	Formulation of mass balances
Initialization		
AbsolutePressure	p_a_start	Start value of pressure at port a [Pa]
AbsolutePressure	p_b_start	Start value of pressure at port b [Pa]
Boolean	use_T_start	Use T_start if true, otherwise h_start
Temperature	T_start	Start value of temperature [K]
SpecificEnthalpy	h_start	Start value of specific enthalpy [J/kg]
MassFraction	X_start[Medium.nX]	Start value of mass fractions m_i/m [kg/kg]
ExtraProperty	C_start[Medium.nC]	Start value of trace substances

Interfaces.PartialDistributedFlow

Base class for a distributed momentum balance

Information

Interface and base class for `m` momentum balances, defining the mass flow rates `m_flows[m]` of a given `Medium` in `m` flow segments.

The following boundary flow and force terms are part of the momentum balances and must be specified in an extending model (to zero if not considered):

- `Ib_flows[m]`, the flows of momentum across segment boundaries,
- `Fs_p[m]`, pressure forces, and
- `Fs_fg[m]`, friction and gravity forces.

The lengths along the flow path `pathLengths[m]` are an input that needs to be set in an extending class to complete the model.

Parameters

Type	Name	Description
replaceable package Medium		Medium in the component
Integer	m	Number of flow segments
Assumptions		
Boolean	allowFlowReversal	= true to allow flow reversal, false restricts to design direction (m_flows >= zeros(m))
Dynamics		
Dynamics	momentumDynamics	Formulation of momentum balance
Initialization		
MassFlowRate	m_flow_start	Start value of mass flow rates [kg/s]

Connectors

Type	Name	Description

replaceable package Medium	Medium in the component

Modelica_Fluid.Types

Common types for fluid models

Information

Package Content

Name	Description
HydraulicConductance	Real type for hydraulic conductance
HydraulicResistance	Real type for hydraulic resistance
Dynamics	Enumeration to define definition of balance equations
CvTypes	Enumeration to define the choice of valve flow coefficient
PortFlowDirection	Enumeration to define whether flow reversal is allowed
ModelStructure	Enumeration with choices for model structure in distributed pipe model

Types.HydraulicConductance

Real type for hydraulic conductance

Parameters

Type	Name	Description
	quantity	
	unit	

Types.HydraulicResistance

Real type for hydraulic resistance

Parameters

Type	Name	Description
	quantity	
	unit	

Types.Dynamics

Enumeration to define definition of balance equations

Information

Enumeration to define the formulation of balance equations (to be selected via choices menu):

Dynamics.	Meaning
DynamicFreeInitial	Dynamic balance, Initial guess value

FixedInitial	Dynamic balance, Initial value fixed
SteadyStateInitial	Dynamic balance, Steady state initial with guess value
SteadyState	Steady state balance, Initial guess value

The enumeration "Dynamics" is used for the mass, energy and momentum balance equations respectively. The exact meaning for the three balance equations is stated in the following tables:

Mass balance		
Dynamics.	**Balance equation**	**Initial condition**
DynamicFreeInitial	no restrictions	no initial conditions
FixedInitial	no restrictions	**if** Medium.singleState **then** no initial condition **else** p=p_start
SteadyStateInitial	no restrictions	**if** Medium.singleState **then** no initial condition **else der**(p)=0
SteadyState	**der**(m)=0	
no initial conditions		

Energy balance		
Dynamics.	**Balance equation**	**Initial condition**
DynamicFreeInitial	no restrictions	no initial conditions
FixedInitial	no restrictions	T=T_start or h=h_start
SteadyStateInitial	no restrictions	**der**(T)=0 or **der**(h)=0
SteadyState	**der**(U)=0	
no initial conditions		

Momentum balance		
Dynamics.	**Balance equation**	**Initial condition**
DynamicFreeInitial	no restrictions	no initial conditions
FixedInitial	no restrictions	m_flow = m_flow_start
SteadyStateInitial	no restrictions	**der**(m_flow)=0
SteadyState	**der**(m_flow)=0	
no initial conditions		

In the tables above, the equations are given for one-substance fluids. For multiple-substance fluids and for trace substances, equivalent equations hold.

Medium.singleState is a medium property and defines whether the medium is only described by one state (+ the mass fractions in case of a multi-substance fluid). In such a case one initial condition less must be provided. For example, incompressible media have Medium.singleState = **true**.

Types.CvTypes

Enumeration to define the choice of valve flow coefficient

Information

Enumeration to define the choice of valve flow coefficient (to be selected via choices menu):

CvTypes.	**Meaning**
Av	Av (metric) flow coefficient
Kv	Kv (metric) flow coefficient

Cv	Cv (US) flow coefficient
OpPoint	Av defined by operating point

The details of the coefficients are explained in the Users Guide .

Types.PortFlowDirection

Enumeration to define whether flow reversal is allowed

Information

Enumeration to define the assumptions on the model for the direction of fluid flow at a port (to be selected via choices menu):

PortFlowDirection.	Meaning
Entering	Fluid flow is only entering the port from the outside
Leaving	Fluid flow is only leaving the port to the outside
Bidirectional	No restrictions on fluid flow (flow reversal possible)

The default is "PortFlowDirection.Bidirectional". If you are completely sure that the flow is only in one direction, then the other settings may make the simulation of your model faster.

Types.ModelStructure

Enumeration with choices for model structure in distributed pipe model

Information

Enumeration to define the discretization structure of distributed pipe models according to the staggered grid scheme:

ModelStructure.	Meaning
av_vb	port_a - volume - flow model - volume - port_b
a_v_b	port_a - flow model - volume - flow model - port_b
av_b	port_a - volume - flow model - port_b
a_vb	port_a - flow model - volume - port_b

The default is "ModelStructure.av_vb", i.e., the distributed pipe has "volumes" at its both ends. The advantage is that connections of the pipe to flow models (like fittings) lead to the desirable structure of alternating volume and flow models, which means that no non-linear algebraic equations occur.

Direct connections of distributed pipes with this option means that two volumes are directly connected together. Due to the stream concept this means that the pressures of the two connected volumes are identical, but the temperatures are not set equal (this corresponds to volumes that are connected together with a very short distance and it needs some time until different volume temperatures are equilibrated). Since the pressures of the volumes are identical, the number of states is reduced and index reduction takes place (which means that medium equations depending on pressure are differentiated and the number of required initial conditions is reduced by one).

The default option "av_vb" cannot be used, if the dynamic pipe is connected to a model with non-differentiable pressure, like a Sources.Boundary_pT with prescribed jumping pressure. The modelStructure can be configured as appropriate in such situations, in order to place a momentum balance between a pressure state of the pipe and a non-differentiable boundary condition (e.g. if the jumping pressure component is connected to port_a, use model structure ModelStructure.a_vb).

Modelica_Fluid.Utilities

Utility models to construct fluid components (should not be used directly)

Information

Extends from Modelica.Icons.Library (Icon for library).

Package Content

Name	Description
checkBoundary	Check whether boundary definition is correct
regRoot	Anti-symmetric square root approximation with finite derivative in the origin
regRoot_der	Derivative of regRoot
regSquare	Anti-symmetric square approximation with non-zero derivative in the origin
regPow	Anti-symmetric power approximation with non-zero derivative in the origin
regRoot2	Anti-symmetric approximation of square root with discontinuous factor so that the first derivative is finite and continuous
regSquare2	Anti-symmetric approximation of square with discontinuous factor so that the first derivative is non-zero and is continuous
regStep	Approximation of a general step, such that the characteristic is continuous and differentiable
evaluatePoly3_derivativeAtZero	Evaluate polynomial of order 3 that passes the origin with a predefined derivative
regFun3	Co-monotonic and C1 smooth regularization function
cubicHermite	Evaluate a cubic Hermite spline
cubicHermite_withDerivative	Evaluate a cubic Hermite spline, return value and derivative

Utilities.checkBoundary

Check whether boundary definition is correct

Inputs

Type	Name	Description
String	mediumName	
String	substanceNames[:]	Names of substances
Boolean	singleState	
Boolean	define_p	
Real	X_boundary[:]	
String	modelName	

Utilities.regRoot

Anti-symmetric square root approximation with finite derivative in the origin

Information

This function approximates sqrt(abs(x))*sgn(x), such that the derivative is finite and smooth in x=0.

Function	Approximation	Range
y = regRoot(x)	y ~= sqrt(abs(x))*sgn(x)	abs(x) >> delta
y = regRoot(x)	y ~= x/sqrt(delta)	abs(x) << delta

With the default value of delta=0.01, the difference between sqrt(x) and regRoot(x) is 16% around x=0.01, 0.25% around x=0.1 and 0.0025% around x=1.

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	
Real	delta	Range of significant deviation from sqrt(abs(x))*sgn(x)

Outputs

Type	Name	Description
Real	y	

Utilities.regRoot_der

Derivative of regRoot

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	
Real	delta	Range of significant deviation from sqrt(x)
Real	dx	Derivative of x

Outputs

Type	Name	Description
Real	dy	

Utilities.regSquare

Anti-symmetric square approximation with non-zero derivative in the origin

Information

This function approximates x^2*sgn(x), such that the derivative is non-zero in x=0.

Function	Approximation	Range
y = regSquare(x)	y ~= x^2*sgn(x)	abs(x) >> delta
y = regSquare(x)	y ~= x*delta	abs(x) << delta

With the default value of delta=0.01, the difference between x^2 and regSquare(x) is 41% around x=0.01, 0.4% around x=0.1 and 0.005% around x=1.

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	
Real	delta	Range of significant deviation from x^2*sgn(x)

Outputs

Type	Name	Description
Real	y	

Utilities.regPow

Anti-symmetric power approximation with non-zero derivative in the origin

Information

This function approximates abs(x)^a*sign(x), such that the derivative is positive, finite and smooth in x=0.

Function	Approximation	Range
y = regPow(x)	y ~= abs(x)^a*sgn(x)	abs(x) >> delta
y = regPow(x)	y ~= x*delta^(a-1)	abs(x) << delta

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	
Real	a	
Real	delta	Range of significant deviation from x^a*sgn(x)

Outputs

Type	Name	Description
Real	y	

Utilities.regRoot2

Anti-symmetric approximation of square root with discontinuous factor so that the first derivative is finite and continuous

Information

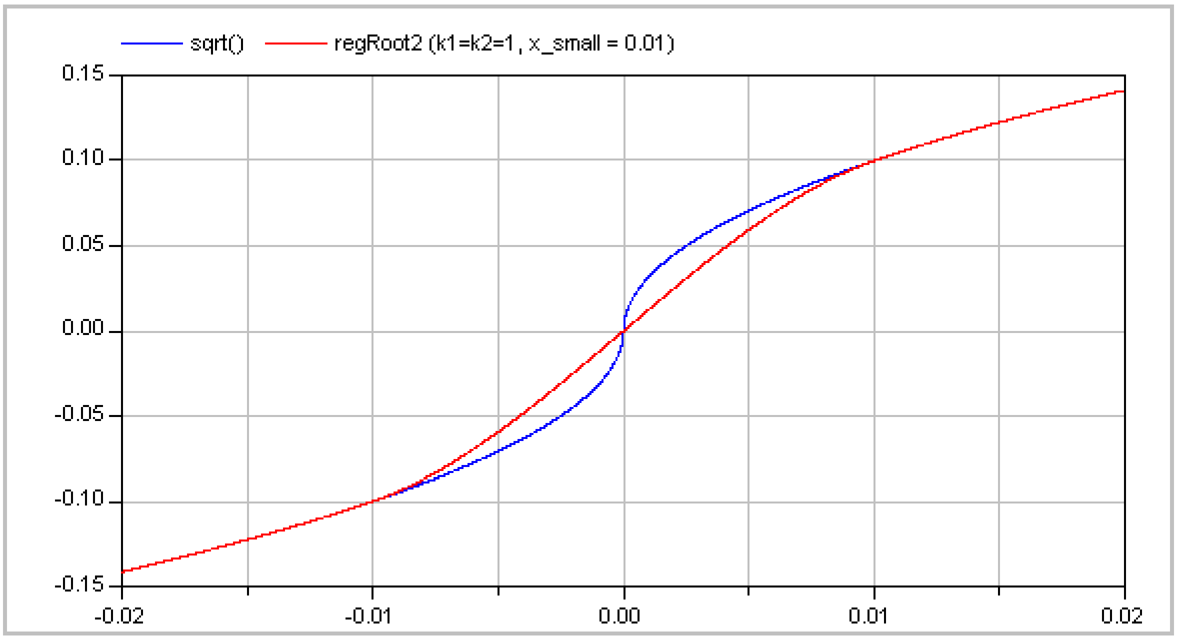
Approximates the function

```
y = if x ≥ 0 then sqrt(k1*x) else -sqrt(k2*abs(x)), with k1, k2 > 0
```

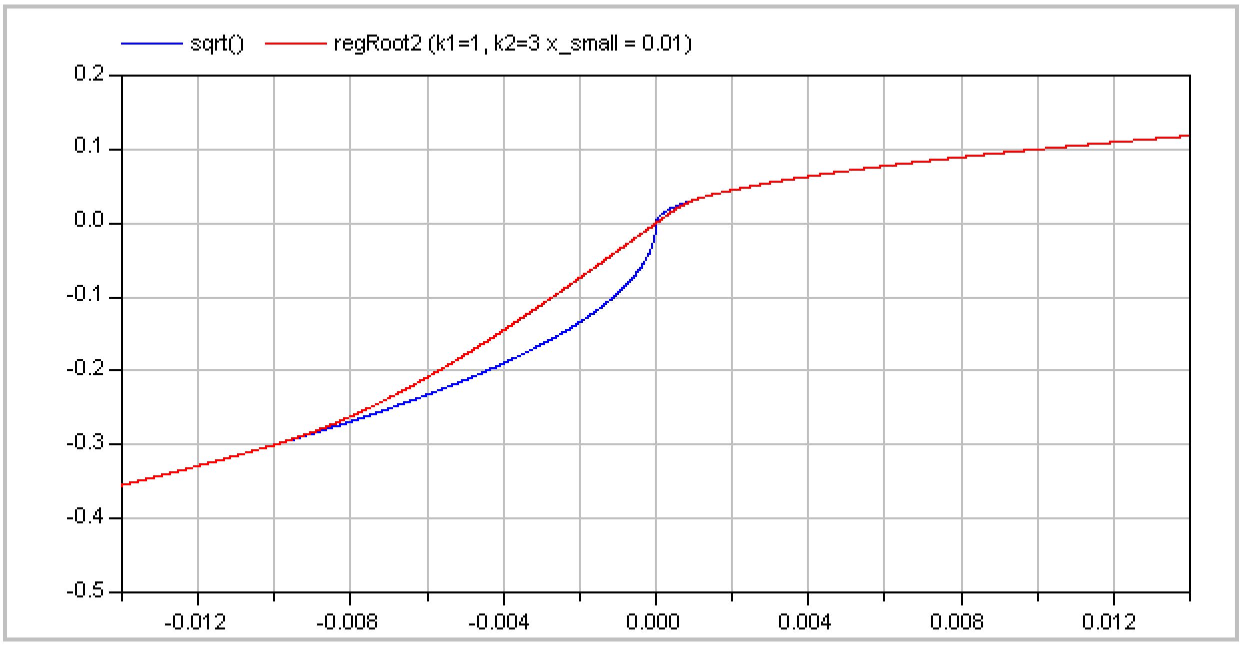
in such a way that within the region -x_small ≤ x ≤ x_small, the function is described by two polynomials of third order (one in the region -x_small .. 0 and one within the region 0 .. x_small) such that

- The derivative at x=0 is finite.
- The overall function is continuous with a continuous first derivative everywhere.
- If parameter use_yd0 = **false**, the two polynomials are constructed such that the second derivatives at x=0 are identical. If use_yd0 = **true**, the derivative at x=0 is explicitly provided via the additional argument yd0. If necessary, the derivative yd0 is automatically reduced in order that the polynomials are strict monotonically increasing *[Fritsch and Carlson, 1980]*.

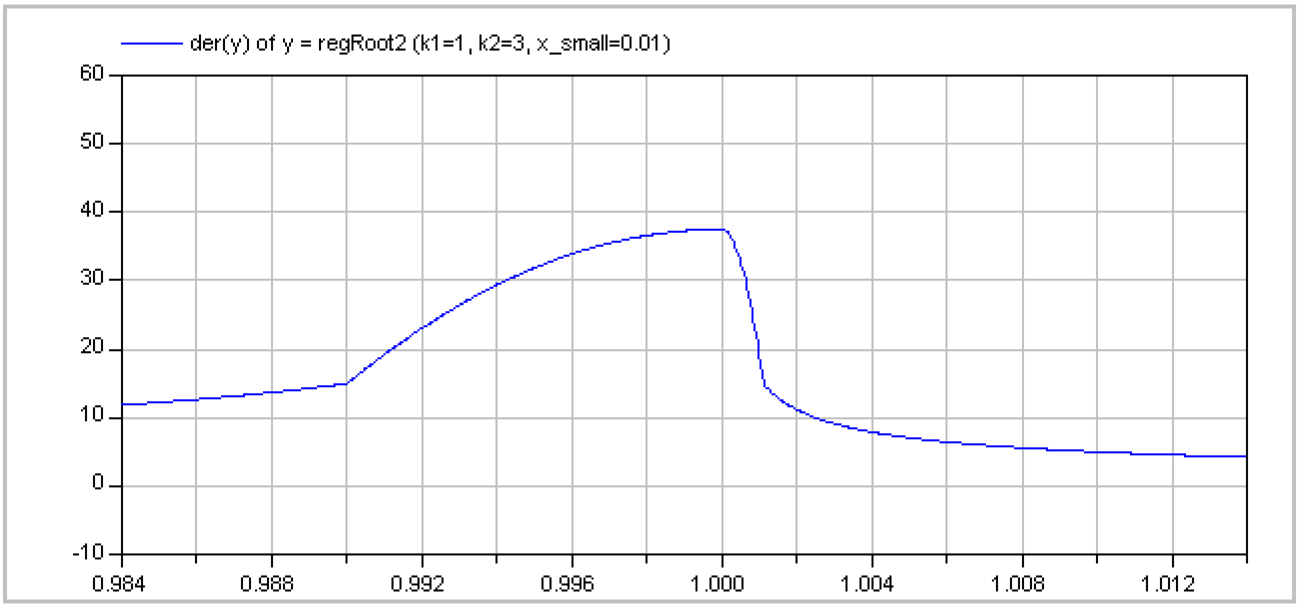
Typical screenshots for two different configurations are shown below. The first one with k1=k2=1:



and the second one with k1=1 and k2=3:



The (smooth) derivative of the function with k1=1, k2=3 is shown in the next figure:



Literature

Fritsch F.N. and Carlson R.E. (1980):
> **Monotone piecewise cubic interpolation**. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	abscissa value
Real	x_small	approximation of function for |x| <= x_small
Real	k1	y = if x>=0 then sqrt(k1*x) else -sqrt(k2*|x|)
Real	k2	y = if x>=0 then sqrt(k1*x) else -sqrt(k2*|x|)
Boolean	use_yd0	= true, if yd0 shall be used
Real	yd0	Desired derivative at x=0: dy/dx = yd0

Outputs

Type	Name	Description
Real	y	ordinate value

Utilities.regSquare2

Anti-symmetric approximation of square with discontinuous factor so that the first derivative is non-zero and is continuous

Information

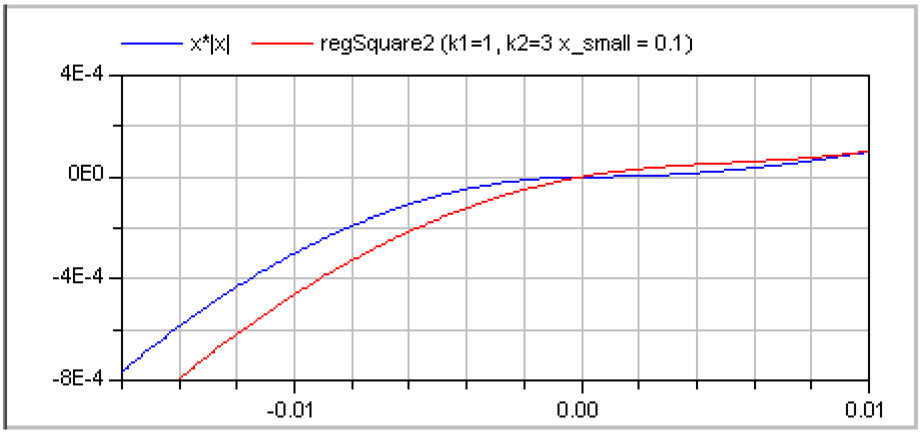
Approximates the function

```
y = if x ≥ 0 then k1*x*x else -k2*x*x, with k1, k2 > 0
```

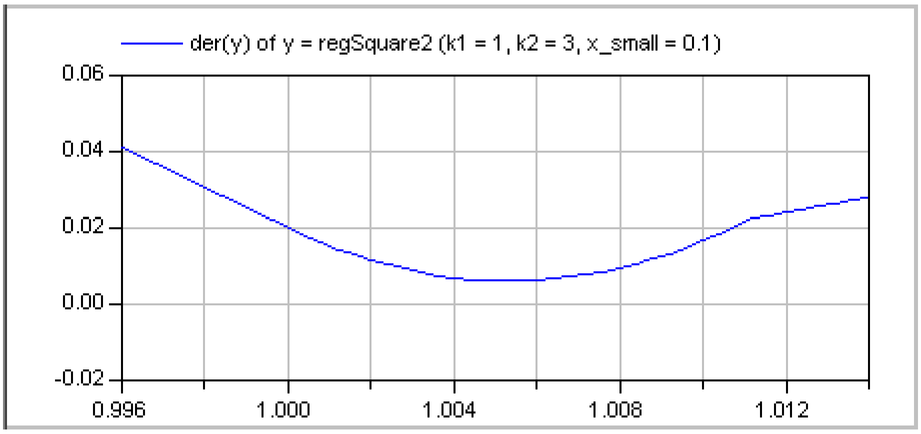
in such a way that within the region -x_small ≤ x ≤ x_small, the function is described by two polynomials of third order (one in the region -x_small .. 0 and one within the region 0 .. x_small) such that

- The derivative at x=0 is non-zero (in order that the inverse of the function does not have an infinite derivative).
- The overall function is continuous with a continuous first derivative everywhere.
- If parameter use_yd0 = **false**, the two polynomials are constructed such that the second derivatives at x=0 are identical. If use_yd0 = **true**, the derivative at x=0 is explicitly provided via the additional argument yd0. If necessary, the derivative yd0 is automatically reduced in order that the polynomials are strict monotonically increasing *[Fritsch and Carlson, 1980]*.

A typical screenshot for k1=1, k2=3 is shown in the next figure:



The (smooth, non-zero) derivative of the function with k1=1, k2=3 is shown in the next figure:



Literature

Fritsch F.N. and Carlson R.E. (1980):

Monotone piecewise cubic interpolation. SIAM J. Numerc. Anal., Vol. 17, No. 2, April 1980, pp. 238-246

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	abscissa value
Real	x_small	approximation of function for |x| <= x_small
Real	k1	y = (if x>=0 then k1 else k2)*x*|x|
Real	k2	y = (if x>=0 then k1 else k2)*x*|x|

Boolean	use_yd0	= true, if yd0 shall be used
Real	yd0	Desired derivative at x=0: dy/dx = yd0

Outputs

Type	Name	Description
Real	y	ordinate value

Utilities.regStep

Approximation of a general step, such that the characteristic is continuous and differentiable

Inputs

Type	Name	Description
Real	x	Abscissa value
Real	y1	Ordinate value for x > 0
Real	y2	Ordinate value for x < 0
Real	x_small	Approximation of step for -x_small <= x <= x_small; x_small > 0 required

Outputs

Type	Name	Description
Real	y	Ordinate value to approximate y = if x > 0 then y1 else y2

Utilities.evaluatePoly3_derivativeAtZero

Evaluate polynomial of order 3 that passes the origin with a predefined derivative

Information

Extends from Modelica.Icons.Function (Icon for a function).

Inputs

Type	Name	Description
Real	x	Value for which polynomial shall be evaluated
Real	x1	Abscissa value
Real	y1	y1=f(x1)
Real	y1d	First derivative at y1
Real	y0d	First derivative at f(x=0)

Outputs

Type	Name	Description
Real	y	

Utilities.regFun3

Co-monotonic and C1 smooth regularization function

Information

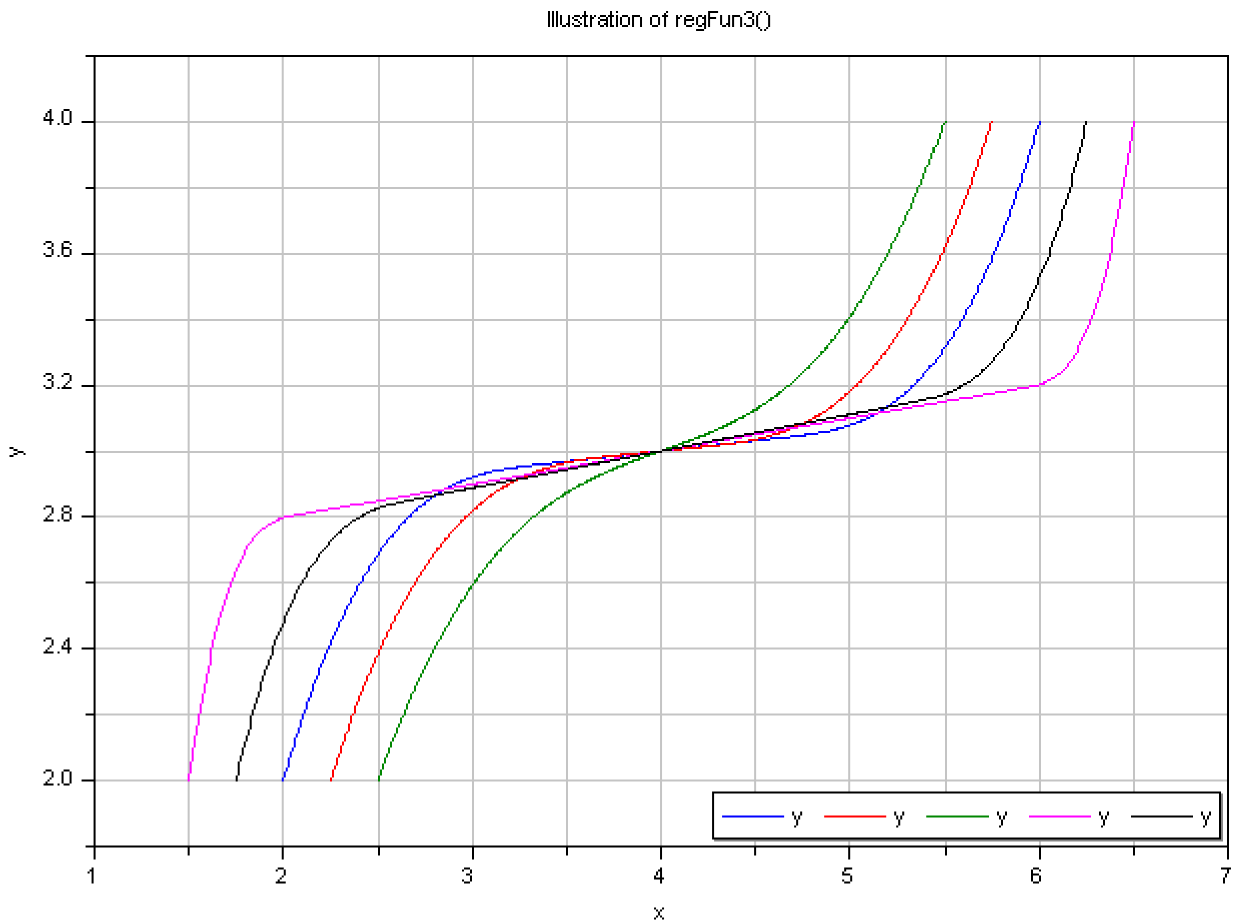
Approximates a function in a region between `x0` and `x1` such that

- The overall function is continuous with a continuous first derivative everywhere.
- The function is co-monotone with the given data points.

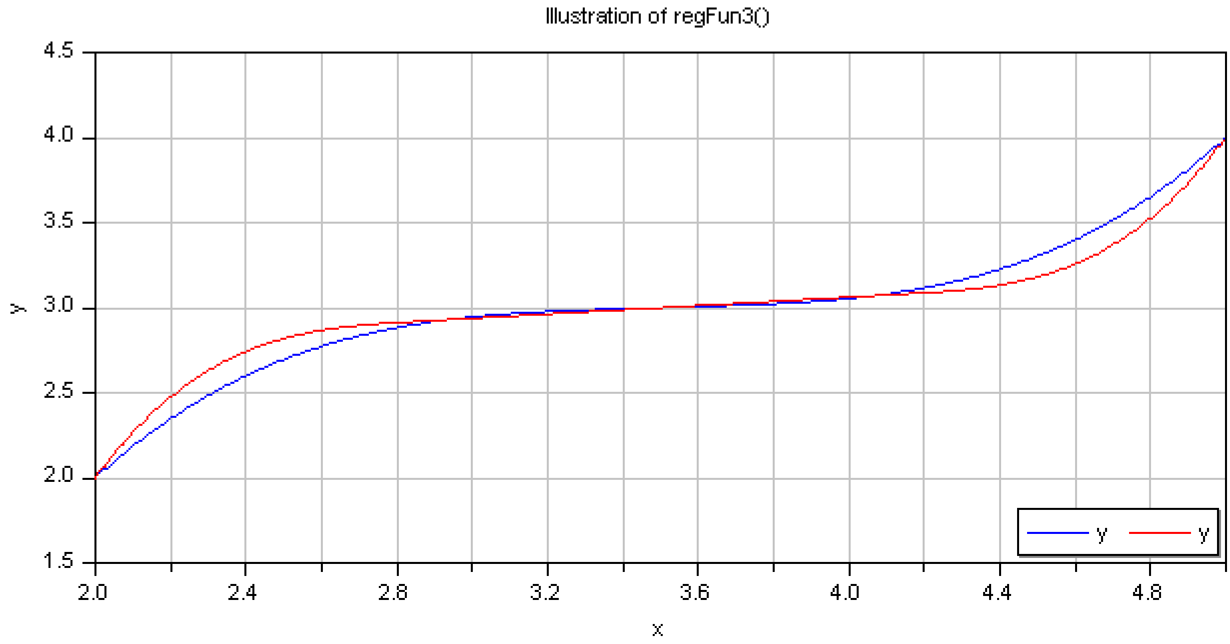
In this region, a continuation is constructed from the given points `(x0, y0)`, `(x1, y1)` and the respective derivatives. For this purpose, a single polynomial of third order or two cubic polynomials with a linear section in between are used *[Gasparo and Morandi, 1991]*. This algorithm was extended with two additional conditions to avoid saddle points with zero/infinite derivative that lead to integrator step size reduction to zero.

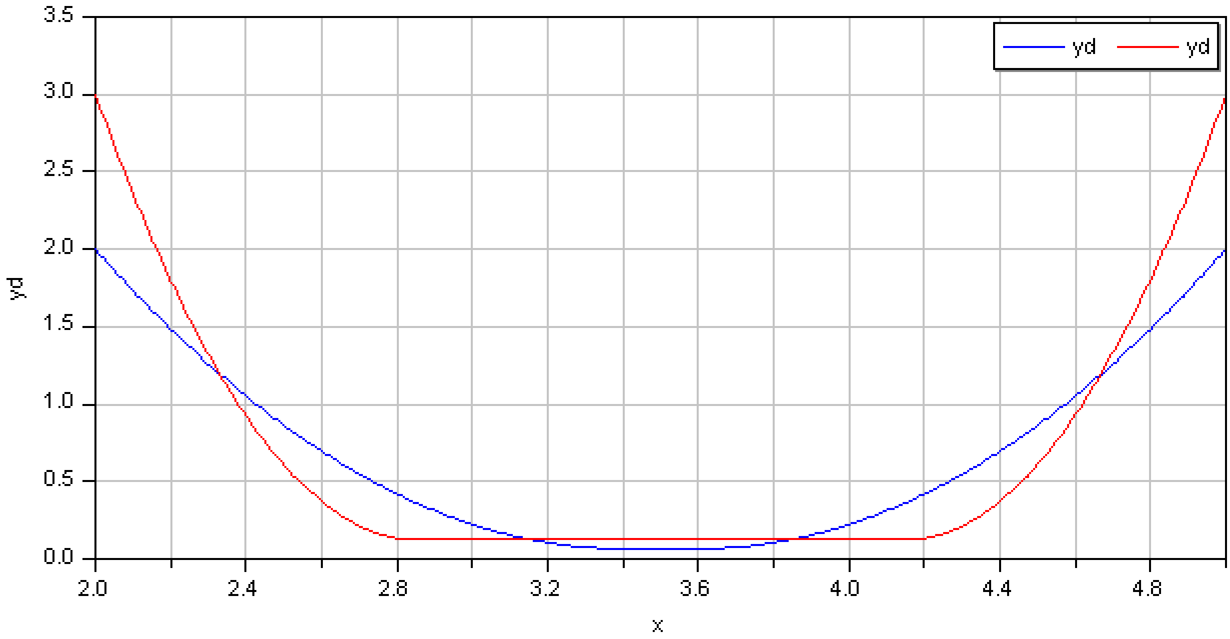
This function was developed for pressure loss correlations properly addressing the static head on top of the established requirements for monotonicity and smoothness. In this case, the present function allows to implement the exact solution in the limit of `x1-x0 -> 0` or `y1-y0 -> 0`.

Typical screenshots for two different configurations are shown below. The first one illustrates five different settings of `xi` and `yid`:



The second graph shows the continous derivative of this regularization function:





Literature

Gasparo M. G. and Morandi R. (1991):
> **Piecewise cubic monotone interpolation with assigned slopes**. Computing, Vol. 46, Issue 4, December 1991, pp. 355 - 365.

Inputs

Type	Name	Description
Real	x	Abscissa value
Real	x0	Lower abscissa value
Real	x1	Upper abscissa value

Real	y0	Ordinate value at lower ordinate value
Real	y1	Ordinate value at upper ordinate value
Real	y0d	Derivative at lower abscissa value
Real	y1d	Derivative at upper abscissa value

Outputs

Type	Name	Description
Real	y	Ordinate value
Real	c	Slope of linear section between two cubic polynomials or dummy linear section slope if single cubic is used

Utilities.cubicHermite

Evaluate a cubic Hermite spline

Inputs

Type	Name	Description
Real	x	Abscissa value
Real	x1	Lower abscissa value
Real	x2	Upper abscissa value
Real	y1	Lower ordinate value
Real	y2	Upper ordinate value
Real	y1d	Lower gradient
Real	y2d	Upper gradient

Outputs

Type	Name	Description
Real	y	Interpolated ordinate value

Utilities.cubicHermite_withDerivative

Evaluate a cubic Hermite spline, return value and derivative

Inputs

Type	Name	Description
Real	x	Abscissa value
Real	x1	Lower abscissa value
Real	x2	Upper abscissa value
Real	y1	Lower ordinate value
Real	y2	Upper ordinate value
Real	y1d	Lower gradient
Real	y2d	Upper gradient

Outputs

Type	Name	Description

Real	y	Interpolated ordinate value
Real	dy_dx	Derivative dy/dx at abscissa value x

Modelica_Fluid.Icons

Library of resuable icons

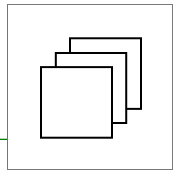
Information

Extends from Modelica.Icons.Library (Icon for library).

Package Content

Name	Description
VariantLibrary	Icon for a library that contains several variants of one component
BaseClassLibrary	Icon for library
ObsoleteFunction	Icon for an interal function

Icons.VariantLibrary

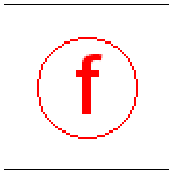


Icon for a library that contains several variants of one component

Icons.BaseClassLibrary

Icon for library

Icons.ObsoleteFunction



Icon for an interal function

Information

This icon is designed for a **function**