

Introduction to the TechThermo library

Beta041206

Content

I Essentials of TechThermo

1 Introduction

- 1.1 Aims of TechThermo
- 1.2 About this Document
- 1.3 Notational Conventions
- 1.4 Demo Model

2 Basic Concepts in TechThermo

- 2.1 Structure of TechThermo
- 2.2 Control of Model Structure by Parameters – ‘switch_’ and ‘option_’
- 2.3 Thermophysical Properties

3 The Main Packages of TechThermo

- 3.1 Interface
 - 3.1.1 The sub-packages for the four different types of connectors
 - 3.1.2 Subpackage Adapter
- 3.2 Main package Source
 - 3.2.1 General Source Models
- 3.3 Main package Medium
 - 3.3.1 Structure of main package Medium and Icons used for graphical representation of medium models
 - 3.3.2 Basic concepts for physical property models in package Medium
 - 3.3.3 Example for property model: ideal gas law for air
 - 3.3.4 Combined property models
- 3.4 Main package Basis
 - 3.4.1 Subpackage HeatTransport
 - 3.4.2 Subpackage MassTransport
 - 3.4.3 Subpackage Compartment
 - 3.4.4 Subpackage Junction
 - 3.4.5 Subpackage BasicProcess

4 Examples

- Ex1: Heat conduction in a wall with varying boundary conditions
- Ex2: Heat losses of cup of coffee
- Ex3: Steam power plant with saturated steam
- Ex4: Compressed air energy storage

I Essentials of TechThermo

- **Main packages:** TechThermo includes six main packages and an additional package with examples in separate files. The six main packages are Interface (connectors), Source (boundary conditions), Medium (thermophysical properties), Basis (fundamental processes in technical thermodynamics), Component (basic technical devices) and Subsystem (simplified technical systems). The examples are included in package Xample.
- **Interdependencies:** The interdependencies between the main packages are that Interface can be used alone, Source demands Interface, Medium demands Interface and Source etc.
- **Internal structure:** TechThermo has a four-level structure: each main package contains subpackages (second level). The subpackages contain model which can be used without further modifications (third level). If necessary, the third level also contains Support folders with models which are not intended for direct use without additions or modifications (fourth level). The Support folders also include the functions. If necessary, the third level might also include Data-folders including data-records.
- **Structural parameters** are often used in TechThermo to modify models before compilation; in combination with if-expressions, structural parameters allow a quick modification of models. Structural parameters are either of type Boolean (only two alternatives) or Integer (more than two alternatives). Names of structural parameters start either with switch_ (Boolean) or with option_ (Integer).
- **Connectors:** TechThermo includes four different types of connectors. Three of these connector-types are related to energy flows: combined heat and mass flow, heat flow without mass transfer and pure exergy flow. The fourth connector type transfer information about the thermal state of the working medium
- **Boundary conditions** are defined at connectors by using models from package Source. Boundary conditions are either defined by parameters or by signal sources.
- **Thermophysical Properties:** If models require correlations between state variables to complete the set of equations, these models are first defined without specification of the working fluid. The working fluid is defined in a second step by connecting a model from main package medium which provides the correlations between the state variables for the selected working fluid.

1 Introduction

1.1 Aims of TechThermo

The availability of model libraries containing the base modules needed to build a system is essential for the efficient application of Modelica. Different libraries are already available. The Modelica library TechThermo provides descriptions for components needed in systems including thermodynamic processes. TechThermo is intended for engineering applications without being restricted to a certain thermodynamic application. The models included in this library can be divided into three groups:

- models representing the infrastructure needed in any Modelica simulation: connectors and models for imposing boundary conditions
- models describing physical processes which are important for the bulk of systems in technical thermodynamics
- models for basic technical devices used in thermodynamic systems

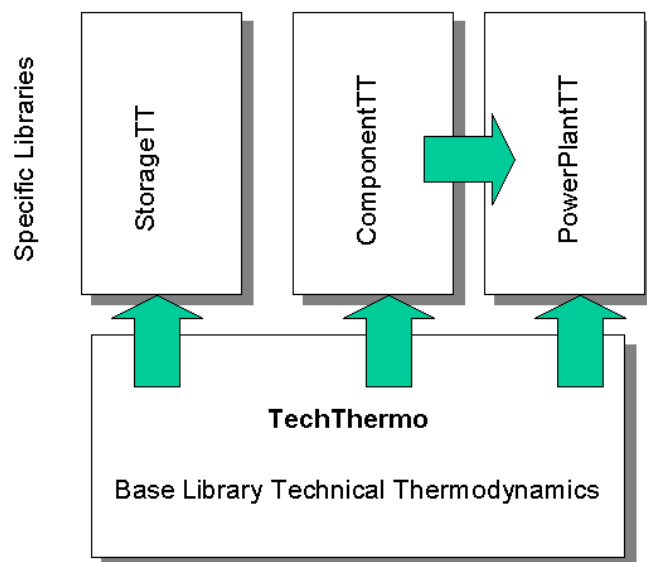


Fig.1: Example for application of TechThermo: libraries developed for specific problems use models from TechThermo; problem specific libraries also exchange models with other problem specific libraries using connectors defined in TechThermo.

There are different aspects how TechThermo can improve the efficiency of modelling activities:

- Thermodynamic systems should be modelled by composition of models representing physical and technical processes; by using models from TechThermo, only problem specific models must be newly implemented.
- Experienced users should profit from TechThermo primarily by extending the models provided by the library thus minimizing the extent of work spent on implementation of trivial equations needed for describing a physical process.
- by standardization of interfaces the cooperation between model-developers should be improved,

- the models provided by the library should allow a quick first analysis of thermodynamic system with only minor effort

Although it would be comfortable to have a universal thermodynamic library which allows the modelling of any system by combination of basic models without input of further model equations, this approach was not chosen for TechThermo since the implementation of such a library seems not to be feasible in practice. Instead, the aim of TechThermo is to minimize the effort for supplementary models for a wide range of application. Essential for the success of a model library is the acceptance by the users. This implies a limitation of the number of models included since

- time for the introduction of a base library is limited; users will develop their own solutions and will hesitate to accept a base library after a certain time
- given a limited time for development, quality of implementation and documentation usually decreases with an increasing number of models
- the willingness of experienced users to spend time on orientation in libraries is limited, selecting a model from the library must be more effective than implementing a new one
- users should know the complete library to have a clear understanding which additions are really necessary to solve a specific simulation problem
- after the initial release of a base library, modification should be reduced to a minimum. The frequency of such modifications corresponds to the number of models in the library.

TechThermo was developed parallel to modelling activities necessary for various research projects. Instead of developing the library from a theoretical point of view, models already utilized were slightly modified to increase the range of applicability and then collected in TechThermo.

1.2 About this Document

This document should introduce users to the application of TechThermo. The manual is organized in four chapters and five appendices:

- Chapter 1, Introduction; notation and demo-model
- Chapter 2, Explanation of basic concepts in TechThermo
- Chapter 3, Description of the models
- Chapter 4, Explanation of the examples included in the library
- Appendix A: Interdependency of the main packages
- Appendix B: Overview connectors and connector variables
- Appendix C: Tree structure of TechThermo
- Appendix D: Alphabetical index of models and short codes
- Appendix E: Literature

1.3 Notational Conventions

- A short version of the essentials of a chapter is presented in a solid frame at the beginning

Additional information, which is not absolutely necessary for understanding the library, is provided in a dashed frame; this information usually is intended for advanced Modelica users

! Here a short advice is given for avoiding trouble

EaS4

Here a listing of Modelica code is given; the combination of letters and numbers in the left upper corner is the short code of the TechThermo model including this listing (description of short code s. 2.1)
The listings in this document don't include the annotation lines.

Fragments of Modelica-code are indicated as `courier font`

Italics indicates a text that should be replaced by the user

1.4 Demo Model

Throughout the manual the TechThermo-model of a cooled compressor is used to demonstrate various features of the library. The listing is given here without commentary lines, the model will be explained in detail in the manual later.

The model `CoolCompressorNoProp` represents a compressor which can simulate either an isothermal compressions which demands cooling or an adiabatic polytropic compression without cooling. Fig. 2 shows the course of the specific volume for three different cases: isothermal compression, isentropic compression and polytropic compression. This model does not include a definition of the working medium.

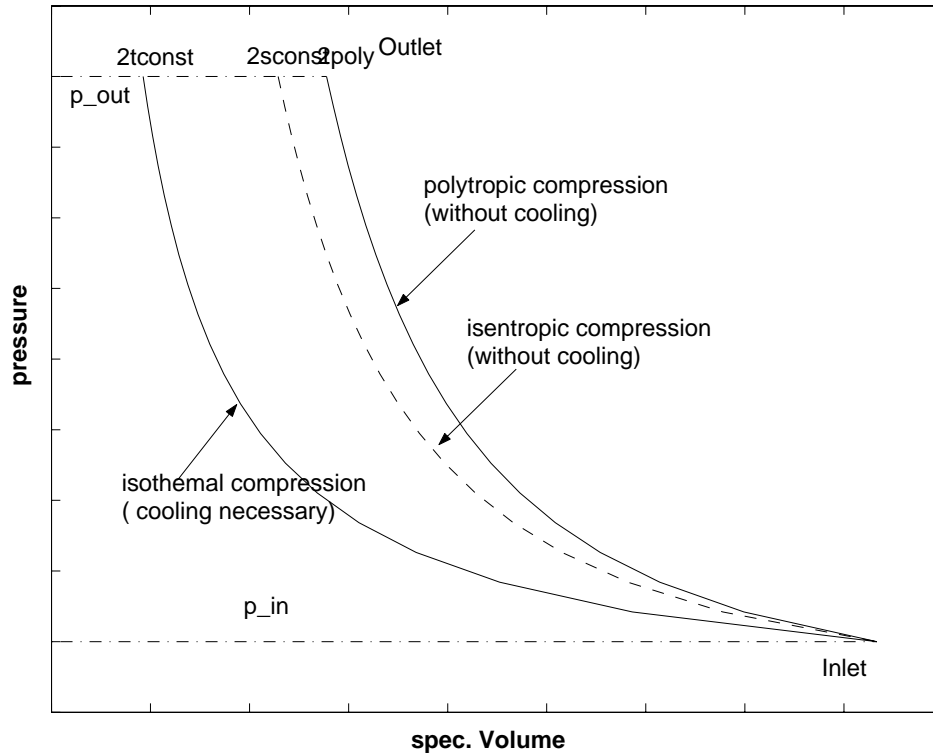


Fig. 2: Three different ways of compressing a gas: isothermal, isentropic or adiabatic polytropic

For the physical model the following variables are assumed:

h_{in}	spec. enthalpy gas at inlet
p_{in}	pressure gas at inlet
$Inlet_t$	temperature gas at inlet
$Inlet_s$	spec. entropy gas at inlet
m_{in_dot}	mass flow rate at inlet
h_{out}	spec. enthalpy gas at outlet
p_{out}	pressure gas at outlet
$RemoveHeat_qdot$	heat-load for isothermal compression
$RemoveHeat_t$	temperature of heat-load for isothermal compression
$Pmech_exergy_dot$	mechanical power needed for compression

The calculation demands the values of thermal state variables for a second thermal state. The physical meaning of this state depends on the decision whether the compression is isothermal or not. The variables for this second thermal state are

$State2_p$	pressure gas for state 2
$State2_h$	spec. enthalpy gas for state 2
$State2_t$	temperature gas for state 2
$State2_s$	spec. entropy gas for state 2

In case of an adiabatic polytropic compression, the isentropic efficiency η_{isentrop} describes the deviation from isentropic compression. For isentropic compression $\eta_{\text{isentrop}} = 1.0$.

With these variables, the following equations can be used for calculation of the compression process:

General:

Energy balance; the needed mechanical work for compression equals the difference between enthalpy flow at inlet and enthalpy flow at outlet and the removed heat:

$$\dot{W}_{\text{mech_exergy}} = -\dot{m}_{\text{in}} * (h_{\text{out}} - h_{\text{in}}) - \dot{Q}_{\text{RemoveHeat}}$$

The pressure for the second state is identical to the pressure at outlet:

$$p_{\text{State2}} = p_{\text{out}}$$

no storage of mass, absolute value mass flow rate at inlet and outlet is identical

Isothermal Compression:

State2 corresponds to the thermal state at the outlet

$$T_{\text{State2}} = T_{\text{Inlet}}$$

$$h_{\text{State2}} = h_{\text{out}}$$

$$\dot{Q}_{\text{RemoveHeat}} = -\dot{m}_{\text{in}} * (T_{\text{Inlet}} + 273.15) * (S_{\text{State2}} - S_{\text{Inlet}})$$

$$\dot{Q}_{\text{RemoveHeat}} = \dot{Q}_{\text{Inlet}}$$

Polytropic Adiabatic Compression:

State2 corresponds to the thermal state after an isentropic compression

$$S_{\text{Inlet}} = S_{\text{State2}}$$

$$\dot{Q}_{\text{RemoveHeat}} = 0.0$$

$$h_{\text{out}} = h_{\text{in}} + (h_{\text{State2}} - h_{\text{in}}) / \eta_{\text{isentrop}}$$

$\dot{Q}_{\text{RemoveHeat}} = -1$ (physically meaningless, since there's no heat flow)

The Modelica model CoolCompressorNoProp will be used to demonstrate various features of the TechThermo library. The explanations will be based on the following listing and line numbers:

EaS4

```

1  model CoolCompressorNoProp "compressor without specification of working fluid"
2  extends TTComponent.Compressor.Support.CoolCompressorCIM(
3  final switch_m_dot_const=true,
4  final switch_x_i_const=true,
5  final switch_h_const=false,
6  final switch_p_const=false);

7  TTInterface.ThermalState.In Inlet "Thermal state at Inlet";
8  TTInterface.ThermalState.Out State2 "Second thermal state";

9  parameter SIunits.Efficiency eta_const=0.8 "const. isentropic efficiency of
compressor";

10 parameter Boolean switch_eta_const=true "if switch_eta_const== true then
eta_isentrop=eta_const";

11 parameter Integer option_cooling=1 "1: isothermal compression, 2: adiabatic
compression";

12 SIunits.Efficiency eta_isentrop "isentropic efficiency of compressor";

13 equation

14 Inlet.h = h_in;
15 Inlet.p = p_in;

16 State2.p = p_out;

17 if switch_eta_const == true then
18 eta_isentrop = eta_const;
19 end if;

20 if (option_cooling == 1) then
21 Inlet.t = State2.t;

22 RemovedHeat.q_dot = -m_in_dot*(Inlet.t + 273.15)*(State2.s - Inlet.s);
23 RemovedHeat.t = Inlet.t;

24 State2.h = h_out;

25 end if;

26 if (option_cooling == 2) then

27 Inlet.s = State2.s;

28 RemovedHeat.q_dot = 0.0;
29 RemovedHeat.t = -1;
30 h_out = h_in + (State2.h - h_in)/eta_isentrop;

31 end if;

32 Pmech.exergy_dot = -m_in_dot*(h_out - h_in) - RemovedHeat.q_dot;

33 end CoolCompressorNoProp;

```

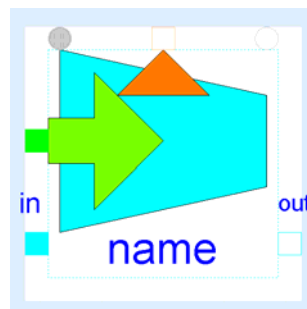


Fig.3: Icon representing model CoolCompressorNoProp

Model CoolCompressorNoProp extends model CoolCompressorCIM (line 2), which includes the definition of the mass flow-connectors and the icon.

2 Basic Concepts in TechThermo

2.1 Structure of TechThermo

- **TechThermo comprises six main packages in six separate Modelica-files and an additional package with examples**
- **There's a four level structure of packages and models: all models which can be used without further modifications are placed on the third level**
- **A short code consisting of either two letters and a number or three letters and a number allows a fast identification of models**

TechThermo is composed of seven main packages. These main packages are stored in separate Modelica files. The names of the corresponding Modelica files are composed of the main package name and the date of the last modification in form yymmdd.

Package Interface contains the definitions of the connectors and some fundamental models. Models needed for imposing boundary conditions are collected in package Source. Package Basis comprises components for calculation of heat and mass transfer processes and conservation laws. Correlations for the calculation of thermophysical properties of substances are collected in package Medium. Package Component contains description of the basic technical units used in thermodynamic systems. Simplified representations of subsystems are implemented in package Subsystem. Finally, package Example contains various examples for the application of models from TechThermo.

Name of main package	Short Identification	Content
Interface	A	connectors and general base models
Source	B	boundary conditions
Basis	C	heat and mass transfer, control volumes
Medium	D	thermophysical properties
Component	E	basic components
Subsystem	F	simplified models for thermodynamic systems
Example	G	examples for application of components from TechThermo

The interdependence of the main packages corresponds to the alphabetical order of the short identification letter (s. Appendix A): `Interface` needs no other package, `Source` demands `Interface`, `Basis` demands `Interface` and `Source` etc... Provided this interdependence is regarded, the selection of a subset of TechThermo-files is possible.

In order to facilitate the orientation within the library a strict four level structure is used for the organisation of the packages and models of the library:

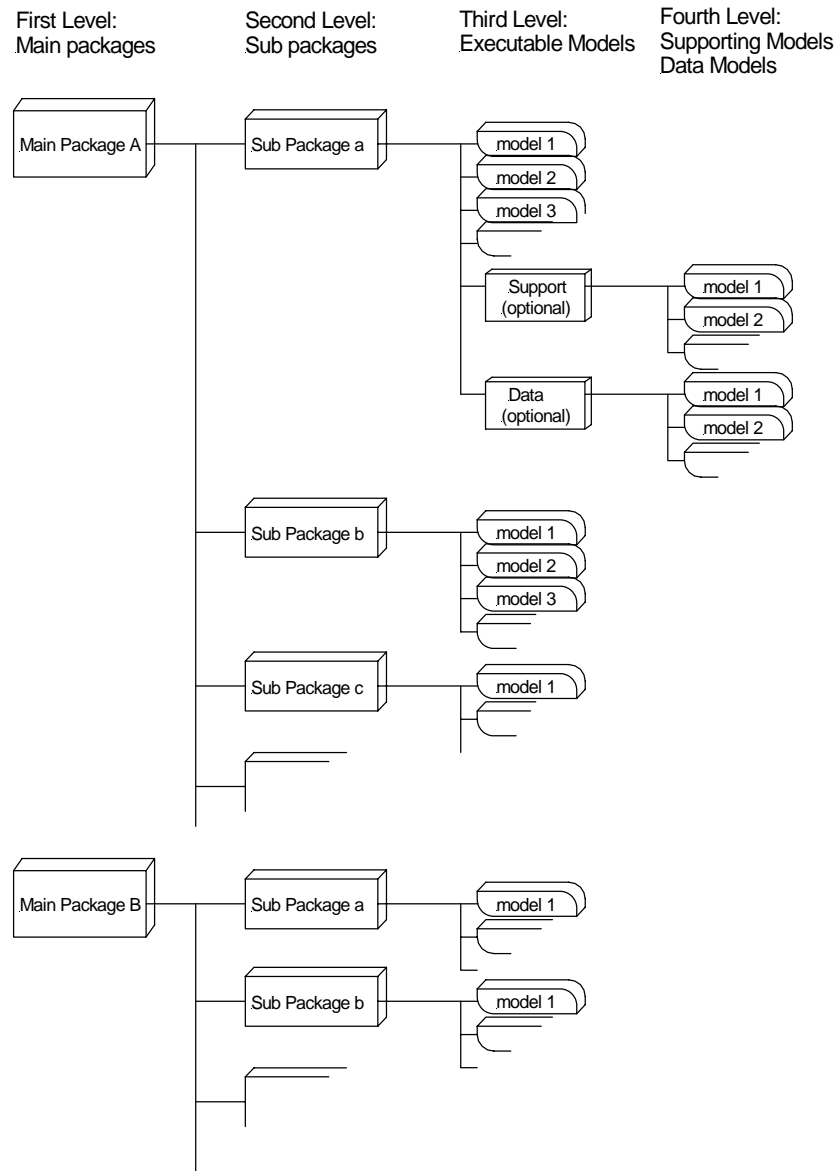


Fig.4.: Organisation of packages and models in TechThermo

A main package is a collection of sub packages which represent the second level of the structure. The sub packages collect a group of models which are related. A small letter is attributed to each sub package within a main package, starting always with "a" for the first package in the file.

All executable models are located on the third level inside the sub packages. On the main package level and sub package level there isn't any model.

Apart from the executable model on the third level, there are only two optional packages: package Support comprises models which demand further modifications before application. Examples for Support models are general descriptions for components or physical processes which demand the specification of the working fluid. Package Data include data models required for the application of models. The models included in packages Support and Data represent the fourth level of TechThermo.

Name of third level package	Short Identification	Content
Support	S	models demanding additional components before application

Data	D	data for further specifications of models
------	---	-------------------------------------------

The short identification of any model in TechThermo is now possible by grouping the corresponding letters for main package, sub package and addition of a number for a model within the sub package. For supporting models or collection of data an additional letter is necessary. The model `CoolCompressorNoProp` is located in main package `Component` (identification letter 'E'), in sub package `Compressor` (this is the first sub package in `Component`, so identification letter is 'a'). Since `CoolCompressorNoProp` demands the definition of the working medium, this model is located in the `Support-package` (letter 'S'), where it's the fourth model, so the complete short code for `CoolCompressorNoProp` is

EaS4.

This short code system allows a quick reference to models for documentation avoiding the necessity to give always the complete path. The short code is given in the initial comment after the name of the model

2.2 Control of Model Structure by Parameters – 'switch_' and 'option_'

- **parameters are used in combination with if-expressions to modify the structure of models before compilation**
- **names of parameters of type Boolean for model structure control start with the prefix 'switch_'**
- **names of parameters of type Integer for model structure control start with the prefix 'option_'**

In order to keep models flexible, not only replaceable models are used in TechThermo. Another possibility to control the structure of a model is the usage of parameters in combination with if-expressions to activate or deactivate certain parts of models. This technique reduces the number of components in the library while still offering the possibility of adapting models to different demands. This approach is preferred when variants of a model differ only in details. In such cases replaceable models might be cumbersome since users have to look up the models even if only a single line is affected. In order to distinguish parameters controlling the model structure from other parameters, names of these controlling-parameters start with `,switch_` (Boolean) or `,option_` (Integer).

Parameters of type Boolean are used when there are only two alternatives. An example for application of 'switch' parameters is the decision whether a variable can be regarded as constant or not depending on the application. This offers the option of a fast implementation of basic version of a model while preserving the potential of introducing more detailed models by extending the basic model and changing the value of the corresponding structural parameter.

In model `CoolCompressorNoProp` the Boolean parameter `switch_eta_const` is used to decide whether the isentropic efficiency `eta_isentrop` is constant or not (line 12). The corresponding if-expression is in line 17-19. The default-value of `switch_eta_const` is `true` (line 12), in this case `eta_isentrop` is identical to the parameter `eta_const`. By extending `CoolCompressorNoProp` and setting

`switch_eta_const=false` an equation for calculation of a variable efficiency can be introduced.

Control parameters of type Integer are used to choose between different options. Examples for these option-parameters are models offering different physical models for a process. The availability of different models enables a quick adaptation to the modelling tasks by selecting the appropriate model.

In model `CoolCompressorNoProp` the Integer parameter `option_cooling` is used to determine the course of the compression (line 11). If the value of `option_cooling` is 1, an isothermal compression is assumed (line 20-25). If `option_cooling` is 2, a polytropic compression is simulated (line 26-31). Usually, there are more than just two alternatives when using a parameter of type Integer for control of model structure. In the case of `CoolCompressorNoProp` the option to introduce additional possibilities in extended models should be preserved. For example, a non-isothermal cooling can be simulated by extending the model, setting `option_cooling = 3` and introducing the additional equations together with the expression

```
if (option_cooling = 3) then...
```

The concept of controlling model structure by parameters offers an attractive method of adapting the complexity of a model to the specific demands of a simulation problem. Starting with a basic version, the complexity of the model can be increased step by step by setting of the corresponding controlling parameters. This approach allows the implementation of very effective models, since the influence of specific model-assumptions can be determined by comparing simulation results.

2.3 Thermophysical Properties

- **implementing the physical model of a process in TechThermo is performed by separating equations for thermophysical properties from other parts of the physical model**
- **names of models including only connectors and the icon-graphics end with ‘CIM’, models are stored in Support-packages**
- **names of models demanding the addition of thermophysical property models for completion end with ‘NoProp’, models are stored in Support packages**
- **TechThermo does not include concepts for calculation of thermophysical properties which are only valid for a single substance; instead general models are preferred which allow the adaptation to a specific substance by variation of a limited number of parameters (scatter 3.4)**

The complete description of a process in technical thermodynamics usually demands correlations between the state variables. Regarding the compression process as an example, different cases can be distinguished:

- different gases are compressed, e.g. in one model air is the working medium, in another model hydrogen might be used
- depending on the specific problem, different degrees of accuracy might be reasonable for an efficient calculation; the complexity of property routines strongly affects the convergence behaviour of models; choosing routines with the highest precision often reduces the robustness while the benefit is neglectable.

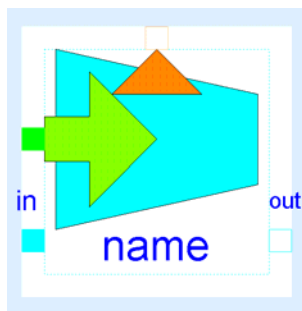
- different assumptions might be adequate: for a cooled process with high compression, the assumption for a constant specific heat capacity of the gas might be an acceptable, while the application of the ideal gas law results in significant errors. In the case of a non-cooled compression with significant changes in temperature, the assumption of a constant specific heat capacity can produce significant errors.
- the definition of a clear interface facilitates the introduction of new working media, no modification of the source code is necessary; the modification can be performed in the graphical environment.

These considerations show that the implementation of thermophysical correlations in separate models is advantageous. Model `CoolCompressorNoProp` includes all equations needed for the calculation of the compression process except for the correlations between the state variables. The names of all models demanding the addition of some models calculating thermophysical properties end with 'NoProp'. Since these models can't be used without modifications they're stored in Support-packages.

A complete model for the compressor demands the following correlations:

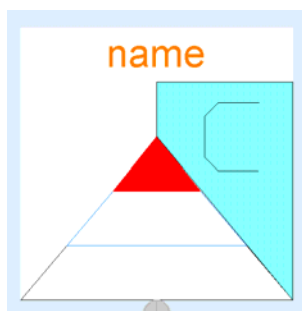
- correlation between temperature at inlet (T_{in}), pressure at inlet (p_{in}), spec. enthalpy at inlet (h_{in}) and entropy (s_{in})
- correlation between temperature, pressure, temperature and entropy at outlet

The implementation of a complete compressor-model should now be demonstrated assuming air as a working fluid. Apart from the `CoolCompressorNoProp` two other models are used:



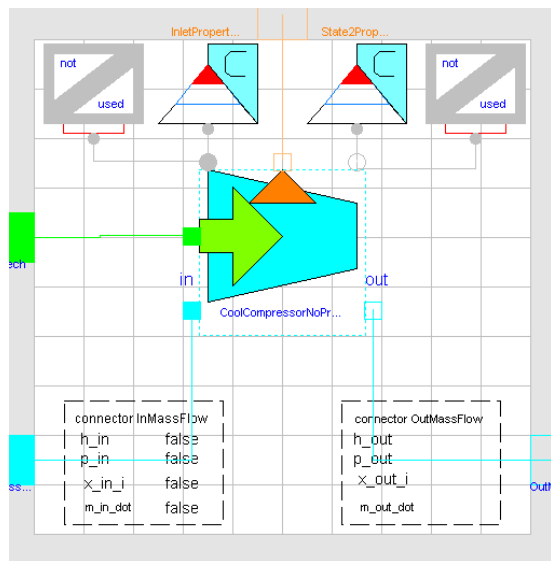
- the model `CoolCompressorCIM` (EaS3) includes only the connectors and the icon graphics for a cooled compressor without any further equations. It is used as a kind of container. (CIM = connector-icon-model, the names of all model of this kind have this ending)

The icon of `CoolCompressorCIM` almost looks identical to the icon of `CoolCompressorNoProp`, but a closer look shows that there are no `ThermalState` connectors (s. 3.1)



- the model `AirPerfectGasCaloric` gives the correlation between temperatures, spec. enthalpy, spec. enthalpy for air, constant spec. heat capacity assumed.

The implementation can be done by drag and drop:



create model AirCoolCompressor by extending CoolCompressorCIM

- Insert a model CoolCompressorNoProp
- Insert a model AirPerfectGasCaloric, name it InletProperties
- Connect InletProperties and CoolCompressorNoProp.Inlet
- Insert a model NotUsedVariables and name it InletNotUsed; set the corresponding parameters for the state variables of connector Inlet which are neither used by InletProperties nor by CoolCompressorNoProp1
- Insert a second model AirPerfectGasCaloric, name it State2Properties
- Connect AirState2 and CoolCompressorNoProp.State2
- Insert a model NotUsedVariables and name it State2NotUsed; set the corresponding parameters for the state variables of connector State2 which are neither used by State2Properties nor by CoolCompressorNoProp1
- connect the corresponding remaining connectors

3 The Main Packages of TechThermo

3.1 Interface



- there are four different types of connectors in TechThermo: **MassFlow** (**m_dot**, **p**, **h**, **x_i[n_comp]**), **HeatFlow** (**q_dot**, **t**), **ExergyFlow** (**exergy_dot**) and **ThermalState** (**h**,**p**,**rho**,**t**,**u**,**s**,**x**, **x_i[n_comp]**)
- application of model **NotUsedVariables** (Aa4) allows the elimination of variables in the **ThermalState** connector
- **MassFlow** connectors allow the definition of the thermal state by a minimal set of variables, **ThermalState** represents the definition of a thermal state by a maximum set of variables. Model **TwoPortThermalStateTerminal** correlates connector variables **h**,**p**,**x_i** of both connectors.







The definition of different types of connectors and corresponding connector variables represents an essential step in the implementation of a Modelica library. Package Interface contains the definitions of connectors and base models used as starting point for the development of models.

In technical thermodynamics three different kinds of energy flows can be distinguished:

- combined heat and mass transfer (e.g. mass flow through a pipe, mass flow in a turbine)
- heat transfer without mass transport (e.g. heat transfer by convection or conduction)
- transfer of exergy (work provided by a turbine, energy used for operation of an electrical heater)

Basically, a single connector could be defined as interface for exchanging all three kinds of energy flow. One drawback of this solution is the varying minimal number of variables necessary for definition for the different kinds of energy flows. While an exergy flow is defined by the exergy flow rate, the definition of combined heat- and mass transfer of a multicomponent fluid demands at least four independent variables. A unique connector would introduce unnecessary connector variables without a physical meaning. In TechThermo, separate connectors are defined for the different energy flows, which should also improve the clarity of a model.

Connectors defined in package Interface of TechThermo:				
Icon	model name TechThermo.Interface.	transferred information	connector variables	example for application
	MassFlow.In	energy transfer by combined heat- and massflow	MassFlowRate m_dot ; SpecificEnthalpy h ; Pressure p ; MassFraction x_i[n_comp] ;	Massflow into and out of turbine, pipe flow
	MassFlow.Out			

	HeatFlow.In	energy transfer by heatflow	HeatFlowRate q_dot ; CelsiusTemperature t ;	heat transfer by conduction or thermal radiation
	HeatFlow.Out			
	ExergyFlow.In	transfer of energy consisting completely of exergy	Power exergy_dot	mechanical power provided by turbine, electric power needed by an electrical heater
	ExergyFlow.Out			
	ThermalState.In	information about thermal state	SpecificEnthalpy h ; Pressure p ; Density rho ; SpecificEntropy s ; CelsiusTemperature t ; SpecificInternalEnergy u ; MassFraction x_i[n_comp] ; Real x ;	information about the thermal state of a working fluid
	ThermalState.Out			

The definition of a connector for heat transfer suggests itself in a thermodynamic library. The connector variables are

```
flow SIunits.HeatFlowRate q_dot
SIunits.CelsiusTemperature t;
```

The Celsius temperature seems to be of more practical use for technical applications than the Kelvin temperature.

The exergy-connector is used for transport of pure exergy. Possible applications of these connectors are components that involve non-thermal energy flow like turbines or compressors. In TechThermo, the exact kind of exergy is not defined, so a single connector variable is sufficient:

```
flow SIunits.Power exergy_dot;
```

A connector for describing energy transfer combined with mass transfer demands connector variables describing mass flow rate and composition in case of multicomponent flow which might be interesting for system involving chemical reactions or wet air. The mass flow rate is the sum for all components. The composition is defined by the mass-composition vector which contains the mass fraction of each component. The size of the vector depends on the number of components `n_comp`. The default value for `n_comp` is 1, so the user needn't to bother about the mass fraction vector for single component flow. The description of fluid flow

demands the clear definition of the thermal state by at least two state variables. In technical thermodynamics different variables are used for the definition of thermal state:

```
Specific Enthalpy h;  
Pressure p;  
Density rho;           Specific Entropy s;  
Celsius Temperature t;  
Specific Internal Energy u;  
Steam quality x
```

Not included in this list are the free energy and the free enthalpy, which are not used so often in technical applications. Only two of these variables are needed for the definition of the thermal state, but depending on the application, different variables are preferred, e.g.

- pipe flow demands the density of the fluid
- energy conservation for control volumes demands the internal energy
- calculation of turbines and compressors needs the specific entropy
- steam generation demands the steam quality
- measurements are often based on temperature and pressure

The selection of just two thermal state variables can't fulfil the demands of all applications. In addition to the three connectors for describing the transfer of energy, a fourth connector is defined for exchange of information about thermal state. With this fourth connector, the selection of two thermal state variables for the mass flow connector has just to fulfil the requirement of a clear definition of thermal state. The first chosen variable is pressure, which plays an important role in most applications. The specific enthalpy was chosen as second variable since it allows a fast implementation of energy conservation laws and is preferred in engineering practise. One drawback of specific enthalpy as connector variable is that it's neither an across variable nor a through variable, so the connection of more than two models by the mass flow connector demands the application of a mixing model.

The complete set of variables for the mass-flow connector is

```
parameter Integer n_comp=1 "number of components";  
flow SIunits.MassFlowRate m_dot;  
SIunits.SpecificEnthalpy h;  
SIunits.Pressure p;  
SIunits.MassFraction x_i[n_comp];
```

There's no definition of separate mass flow connectors for different species.

! Don't connect more than two MassFlow-connectors without using a mixing model

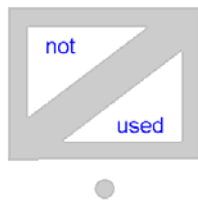
The thermal state connector uses all variables listed above:

```

parameter Integer n_comp=1 "number of components";
SIunits.SpecificEnthalpy h;
SIunits.Pressure p;
SIunits.Density rho;
SIunits.SpecificEntropy s;
SIunits.CelsiusTemperature t;
SIunits.SpecificInternalEnergy u;
SIunits.MassFraction x_i[n_comp];
Real x; // Steam quality

```

The thermal state connector is especially intended for the exchange of information between a model describing a physical process and a model for calculation of thermophysical properties. Usually, only a subset of the connector variables is really needed, but all connector variables must be defined. In this case it's sufficient to attribute constant values to the connector variables which are not used. TechThermo offers the application of the "NotUsedVariables"-model (Aa4):



```

model NotUsedVariables "state connector defining not used state
variables"

parameter Integer n_comp=1 "number of components";
parameter Real dummy_value=232323 "value for not-used variables";

//-----connector for thermal state-----
ThermalState.In StateCut(n_comp=n_comp);

//-----switch-parameters-----
parameter Boolean switch_h_notused=false
  "if switch_h_notused==true then StateCut.h=dummy_value ";
parameter Boolean switch_p_notused=false
  "if switch_p_notused==true then StateCut.p=dummy_value ";
parameter Boolean switch_rho_notused=false
  "if switch_rho_notused==true then StateCut.rho=dummy_value ";
parameter Boolean switch_s_notused=false
  "if switch_s_notused==true then StateCut.s=dummy_value";
parameter Boolean switch_t_notused=false
  "if switch_t_notused==true then StateCut.t=dummy_value";
parameter Boolean switch_u_notused=false
  "if switch_u_notused==true then StateCut.u=dummy_value";
parameter Boolean switch_x_notused=false
  "if switch_x_notused==true then StateCut.x=dummy_value ";
parameter Boolean switch_x_i_notused=false
  "if switch_x_i_notused==true then StateCut.x_i=dummy_i ";

parameter Real dummy_i[n_comp]=zeros(n_comp);
equation
  if switch_h_notused == true then
    StateCut.h = dummy_value;
  end if;
  if switch_p_notused == true then
    StateCut.p = dummy_value;
  end if;
  if switch_rho_notused == true then
    StateCut.rho = dummy_value;
  end if;
  if switch_s_notused == true then
    StateCut.s = dummy_value;
  end if;
  if switch_t_notused == true then
    StateCut.t = dummy_value;
  end if;
  if switch_u_notused == true then
    StateCut.u = dummy_value;
  end if;
  if switch_x_notused == true then
    StateCut.x = dummy_value;
  end if;
  if switch_x_i_notused == true then
    StateCut.x_i = dummy_i;
  end if;

```

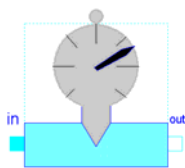
```
end NotUsedVariables;
```

This model can be connected to any ThermalState-connector. By setting the corresponding switch-parameter, state variables can be selected which are set equal to a parameter during the simulation.

While the MassFlow-connector represents the definition of thermal state by a minimal set of variables, the ThermalState-connector includes the maximal set of thermal state variables. These two types of connectors can be correlated by the model TwoPortThermalStateTerminal. This model includes two MassFlow-connectors and a ThermalState-connector. The average values for the common variables spec. enthalpy, pressure and composition are transferred from the MassFlow-connectors to the ThermalState-connector. The element can be used either serial or parallel to a mass-flow.

The serial and the parallel application of TwoPortThermalStateTerminal corresponds to two different situations:

- serial: the thermal state of a mass flow between two components should be determined; the inlet of TwoPortThermalStateTerminal is connected to the outlet of the first component while the outlet of TwoPortThermalStateTerminal is connected to the inlet of the second component; the (absolute) value of all connector variables at inlet and outlet of TwoPortThermalStateTerminal remain constant, so there's no difference between the connector values at MassFlow-connectors and the and the corresponding (average) values at the ThermalState-connector
- parallel: the average values of the state variables of two MassFlow-connectors are needed. TwoPortThermalStateTerminal connects these two connectors but the mass flow rate through TwoPortThermalStateTerminal is zero.



```
model TwoPortThermalStateTerminal
  "stationary mass flow element with state connector"
  extends MassFlow.TwoPort(
    switch_m_dot_const=true,
    switch_h_const=true,
    switch_p_const=true,
    switch_x_i_const=true);

    //-----connector for thermal state-----
    ThermalState.In StateCut(n_comp=n_comp) ;

  equation

    StateCut.h = (h_in + h_out)/2.0;
    StateCut.p = (p_in + p_out)/2.0;
    StateCut.x_i = (x_in_i + x_out_i)/2.0;

end TwoPortThermalStateTerminal;
```

3.1.1 The sub-packages for the four different types of connectors


Package Interface comprises a separate sub-package for each of the four different types of connectors (sub package MassFlow, HeatFlow, ThermalState, ExergyFlow) and an additional sub-package (sub-package Adapter) for connector-adapters. Most of the models included in the four connector-related sub-packages are similar.

For models in sub package MassFlow, HeatFlow, ThermalState, ExergyFlow which differ only in the used connector type identical model names are used; since these models are part of different sub-packages, the risk of confusion seems to be neglectible and regarding the complete path it seem not be necessary to add again the name of the connector to the model name, e.g. Interface.MassFlow.MassFlowIn instead of the name used in TechThermo Interface.MassFlow.In

For each type of connector two different connector models are given which differ only in the graphical representation. These connector-models are named In and Out.

Many processes in technical thermodynamics can be regarded as systems with an inflow and an outflow. In TechThermo for the four different connector types models including the definition of an In- and an Out-connector are provided. These TwoPort-models also offer the possibility to select connector-variables which remain constant by setting the corresponding switch-parameters. The following listing shows the implementation of the TwoPort model in package MassFlow:

Aa3



```

1      model TwoPort "  model mass flow element with two connectors"
2      parameter Integer n_comp=1 "number of components in fluid";
3      In InMassFlow(n_comp=n_comp) "connector for inlet mass flow"
4      Out OutMassFlow(n_comp=n_comp) "connector for outlet mass flow"

5      parameter Boolean switch_m_dot_const=false
6      "if switch_m_dot_const=true then m_in_dot+m_out_dot=0";
7      parameter Boolean switch_h_const=false
8      "if switch_h_const=true then h_in=h_out";
9      parameter Boolean switch_p_const=false
10     "if switch_p_const=true then p_in=p_out";
11     parameter Boolean switch_x_i_const=false
12     "if switch_x_i_const=true then x_in_i=x_out_i";

13     flow SIunits.MassFlowRate m_in_dot
14     "mass flow rate at connector InMassFlow";
15     SIunits.SpecificEnthalpy h_in "spec. enthalpy at connector
16     InMassFlow";
17     SIunits.Pressure p_in "pressure at connector InMassFlow";
18     SIunits.MassFraction x_in_i[n_comp]
19     "vector with mass-fractions at connector InMassFlow";
20     flow SIunits.MassFlowRate m_out_dot;
21     SIunits.SpecificEnthalpy h_out;
22     SIunits.Pressure p_out;
23     SIunits.MassFraction x_out_i[n_comp];
24     equation

25     h_in = InMassFlow.h;
26     m_in_dot = InMassFlow.m_dot;
27     p_in = InMassFlow.p;
28     x_in_i = InMassFlow.x_i;

29     h_out = OutMassFlow.h;
30     m_out_dot = OutMassFlow.m_dot;
31     p_out = OutMassFlow.p;
32     x_out_i = OutMassFlow.x_i;

33     if switch_m_dot_const then
34       = m_in_dot + m_out_dot;
35     end if;

36     if switch_h_const then
37       h_in = h_out;
38     end if;


39     if switch_p_const then
40       p_in = p_out;
41     end if;

42     if switch_x_i_const then
43       x_in_i = x_out_i;
44     end if;

45     end TwoPort;

```

In line 3 and 4 the two connectors are defined. Depending on the values of Boolean switch-parameters defined in lines 5-12, corresponding connector variables are set equal in lines 32-43. In lines 24-31 variables are defined from the connector variables which should help to reduce the extent of typing effort. Model `CoolCompressorCIM` extends the `TwoPort`-model of the `MassFlow` package. In model `CoolCompressorNoProp`, which extends `CoolCompressorCIM`, the absolute values of the mass flow rate at the inlet and outlet are equal, so `switch_m_dot_const` is true (`CoolCompressorNoProp` line 3, `TwoPort` line 32-34). The mass fraction remains also constant in the model, so `switch_x_i_const` is also true (`CoolCompressorNoProp` line 4, `TwoPort` line 35-37). Pressure and specific enthalpy usually are different at inlet and outlet, so `switch_h_const` and `switch_p_const` are false. (lines 5+6). Using the 'final' attribute for the definition of switch-parameters helps to reduce the number of parameters displayed and avoids errors by inexperienced users.

<p>EaS4</p> 	<pre> 1 model CoolCompressorNoProp "compressor without specification of working fluid" 2 3 extends TComponent.Compressor.Support.CoolCompressorCIM(4 final switch_m_dot_const=true, 5 final switch_x_i_const=true, 6 final switch_h_const=false, 7 final switch_p_const=false); </pre>
-----------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

When applying models extending `TwoPort` the multiple setting of switch-parameters must be avoided

! Don't use `TwoPort` switch-parameters in models which contain only icon-graphics and connector definitions (CIM-models).

The implementation of the compressor model for air in 2.3 shows the correct use of the `TwoPort` model; the complete model comprises four levels:

- 1 *TwoPort*
- 2 *CoolCompressorCIM*, extending *TwoPort*, definition of Icon graphics and additional connectors for heat flow (cooling) and exergy flow (mechanical power)
- 3 *CoolCompressorNoProp*, extending *CoolCompressorCIM*, addition of equations defining compression process
- 4 *AirCoolCompressor*, extending *CoolCompressorCIM* and including *CoolCompressorNoProp*; addition of property routines for air to complete set of equations.

The two massflow-connectors are identical in `TwoPort`, `CoolCompressorCIM` and `CoolCompressorNoProp`, the switch-parameters are set in `CoolCompressorNoProp`. `CoolCompressorNoProp` is integrated in `AirCoolCompressor`, the corresponding connectors are linked directly, so the switch-parameters in `AirCoolCompressor` mustn't be true, since this would introduce additional equations.

TwoPort models can also be used to transfer selected variables between two connectors by setting the corresponding switch-parameters

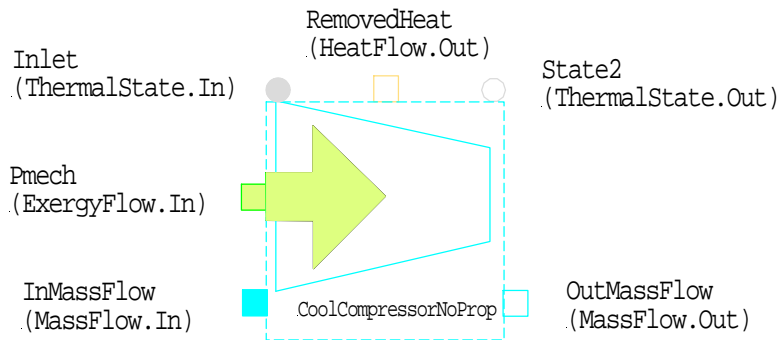
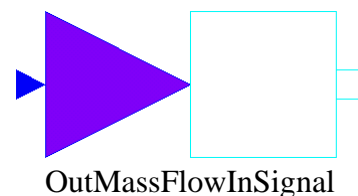
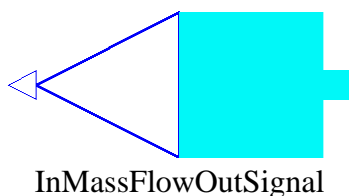


Fig.5: Connectors and application of model TwoPort in model CoolCompressorNoProp

3.1.2 Subpackage Adapter

- **models in Subpackage Adapter help to connect models from TechThermo to models from other Modelica libraries**

Subpackage Adapter includes models allowing the linkage between TechThermo models and models from other libraries. The values for corresponding variables are transferred between two different types of connectors. TechThermo offers for its four different types of connectors models allowing the linkage to elements using the InPort or OutPort from the Modelica Standard Library (Modelica.Blocks.Interface). The value of a selected TechThermo connector variable is set equal to the value of the InPort connector. The selection of the TechThermo connector variable depends on the value of parameter option_defsignal.



Subpackage Adapter: Examples for models linking TechThermo connectors to connector of other Modelica-libraries

3.2 Main package Source

- **Source comprises models to introduce boundary conditions at the connectors of the components of a system**
- **for each of the four connector types pack-age *Source* contains a subpackage with models to impose boundary conditions using a specific connector**
-
- **boundary conditions can be defined either by parameters or by (varying) external signals**

Models represent systems of limited extent and usually require the definition of a set of system variables to introduce the influence of the environment. Models from main package *Source* are used to impose these outer boundary conditions to the model. These models include at least a single connector and offer various options to define the connector variables of these connectors. Boundary conditions are always introduced at connectors of a system.




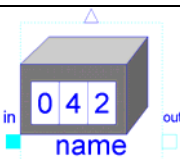
The definition of any connector variable should be possible either by parameters or by external signals. Regarding the four types of connectors defined in main package *Interface*, there are 15 different connector variables. One approach would be the definition of separate models for each of these connector variables. Instead, in order to limit the total number of models, only a single model is defined for each connector type. Which connector variables are used as boundary conditions is selected by switch-parameters.

3.2.1 General Source Models

For each connector, two different ways of defining boundary conditions are possible:

- the value(s) of the connector variable(s) used as boundary condition(s) is defined by parameter(s) which remain constant during the simulation
- the value of the connector variable used as boundary condition is defined by an external signal-source. Examples for models used as signal sources can be taken from the Modelica standard library

Again, there are separate sub packages for each of the four connector types and most of the models are similar in the different sub packages. Model *ParameterDefined* and *SignalDefined* are available for all four TechThermo connectors, models *FlowRateCtrl* and *Counter* are related to flow-variables and are available for MassFlow, HeatFlow and ExergyFlow, but not for ThermalState, since ThermalState connectors don't include flow variables.

Icon	Name	Purpose
	ParameterDefined	Definition of selected connector variables by parameters.
	SignalDefined	A selected connector variable is defined by an external signal; external signal source is linked to connector InSignal (type Modelica.Blocks.Interfaces.InPort)
	FlowRateCtrl	A flow variable is defined by an external signal between two models; external signal source is linked to connector InSignal (type Modelica.Blocks.Interfaces.InPort) This model is not implemented for ThermalState connectors, since this kind of connector doesn't include a flow variable
	Counter	Integration of the flow variable; the integrated value is available at connector OutSignal (type Modelica.Blocks.Interfaces.OutPort)

Model *ParameterDefined* is used to define selected connector variables by parameters. Provided the corresponding switch_parameter is true, the connector is set equal to a parameter value. The listing of model *ParameterDefined* (Bb1) for the HeatFlow-connector shows the structure of these models: depending on the number of connector variables (\dot{q} and t) two switch-parameters are defined (switch_q_dot_def and switch_t_def, lines 4-7), the default values for these Boolean parameters are false. Depending on the actual values in the

simulation (lines 13-18), the connector variables t and $q_{\dot{}}$ are set equal to the values of the parameters t_{para} or $q_{\dot{\text{para}}}$ (defined in lines 8-12). For this model, the number of variables can vary between zero (model is used as a sink) and two (both temperature and heat flow rate are defined by parameters)

Bb1	<pre> 1 model ParameterDefined 2 "heat-flow source with optional definition of heat-flow variables by parameters" 3 TTInterface.HeatFlow.Out OutHeatFlow 4 parameter Boolean switch_q_dot_def=false 5 "if switch_q_dot_def=true, OutHeatFlow.q_dot is determined by parameter q_dot_para" 6 parameter Boolean switch_t_def=false 7 "if switch_t_def=true, OutHeatFlow.t is determined by parameter t_para"; 8 parameter SIunits.HeatFlowRate q_dot_para=1.0 9 "value for heat-flow rate HeatFlowOut.q_dot at outlet if switch_q_dot_def=true"; 10 parameter SIunits.CelsiusTemperature t_para=25.0 11 "value for temperature at HeatFlowOut.t at outlet if switch_t_def=true"; 12 equation 13 if switch_q_dot_def then 14 OutHeatFlow.q_dot = q_dot_para; 15 end if; 16 if switch_t_def then 17 OutHeatFlow.t = t_para; 18 end if; 19 end ParameterDefined;</pre>
-----	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

If the boundary conditions don't remain constant during the simulation, the second type of model for imposing boundary conditions can be used: here the value of a single connector variable is defined by an external signal. The SignalDefined model has a connector InPort from package Modelica.Blocks.Interfaces to link signal sources to the model. Examples for signal sources can be found in Modelica.Blocks.Sources. The definition of more than a single connector variable is possible by combining a set of SignalDefined models; a combination with a ParameterDefined model is also possible, thus allowing both constant and transient boundary conditions.

With models ParameterDefined and SignalDefined the definition of flow variables between two models is difficult; in order to define flow variables like heat flow rate or mass flow rate between two models, model FlowRateCtrl are provided. The inlet of FlowRateCtrl is connected to the outlet of the first model while the outlet of FlowRateCtrl is connected to the inlet of the second model. The flow variable is then controlled by an external signal.

Finally, the Counter model is used to integrate flow variables. These elements help to determine the extend of mass, heat or exergy that flows through a connector during a simulation.

3.2.3 Examples for application of source models

The use of source models is now demonstrated for the definition of boundary conditions necessary to calculate the performance of a single cooled compressor with air as working medium. These models are included in sub-package Chapter3_2 in package Xample. Fig.6 shows the compressor model connected to source models.

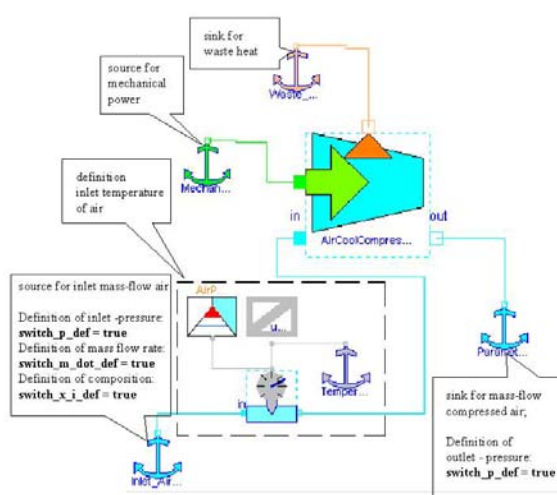


Fig.6: Definition of boundary conditions for a single compressor (description_example_3_2_1)

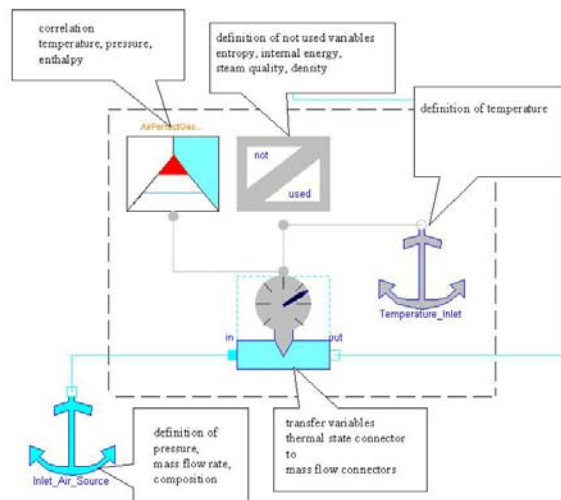


Fig.7: Components used to define inlet temperature for compressor-model shown in Fig.xx1.

For this example, the following boundary conditions are assumed:

- Inlet: definition of pressure, mass-flow rate and composition of air by parameters; these boundary conditions are defined by model *InletParameterDefined* of type *Source.MassFlow.ParameterDefined*. The Boolean parameters *switch_m_dot_def*, *switch_p_def* and *switch_x_i_def* are set from default value *false* to *true*, the corresponding parameters *m_dot_para*, *p_para*, *x_para_i* are modified.
- Inlet: definition of temperature of air; basically, the thermal state of the air at the inlet of the compressor could also be defined by pressure and spec. enthalpy, but for the user it's often more convenient to use the temperature as boundary conditions. Since the compressor uses a mass flow connector with spec. enthalpy and pressure as state variables, a correlation between temperature, pressure and spec. enthalpy is necessary. As described before, a model of type *Interface.MassFlow.TwoPortThermalStateTerminal* correlates spec. enthalpy and pressure of a mass flow connector to pressure, spec. enthalpy, temperature, spec. entropy, steam quality and density of a thermal state connector. By adding a source model for thermal state and a correlation between the state variables any variable of a thermal state connector can be used as boundary condition. In Fig.xx2 the application of this concept for the compressor is shown: model *TemperatureEnthalpyAirInlet* of type *Medium.Gas.AirPerfectGasCaloric* containing a correlation between spec. enthalpy and temperature is connected to the thermal state connector of *TwoPortThermalStateTerminal*, model *TemperatureInlet* of type *Source.ThermalState.ParameterDefined* is used to define the temperature by a parameter (*switch_t_def* = *true*). Model *NotUsedVariables* is used to define the remaining thermal state variables which are not relevant (ρ , u , s , x).
- Outlet: definition of pressure by model *OutAirSource* of type *Source.MassFlow.ParameterDefined* with parameter *switch_p_def* = *true*.

Models *WasteHeatSink* is used as sink for the heat produced during the compression; temperature and heat flow rate result from the other boundary conditions, so *Waste_Heat_Sink* introduces no additional boundary conditions. Model *Mechanical_Power_Source* is used as a exergy source for the mechanical power needed for compression. Since the mechanical power results from the other boundary conditions, no boundary conditions are introduced by *Mechanical_Power_Source*.

Alternative boundary conditions for compressor example

All described modifications refer to the basic version of the example

- Definition of mechanical Power:** Fig.8 shows the modifications of the source models if the mechanical power is defined and the mass flow rate should be calculated for a given pressure difference between inlet and outlet. Here, the Boolean parameter *switch_exergy_dot_def* of the model *MechanicalPowerSource* is set to true while the parameter *switch_m_dot_def* of *InletParameterDefined* is changed to false.
- Definition of mass-flow rate by a varying external signal:** Fig. 9 shows the diagram for a compressor with inlet mass flow rate controlled by an external signal. At the inlet, pressure and composition of air is defined by parameters in model *InletParameterDefined*. The inlet air flow goes through model *InletVariableFlowRateCtrl*; here the mass flow rate is defined by an external signal source connected to this element. The signal source used here is *Blocks.Source.Ramp* from the Modelica standard library.

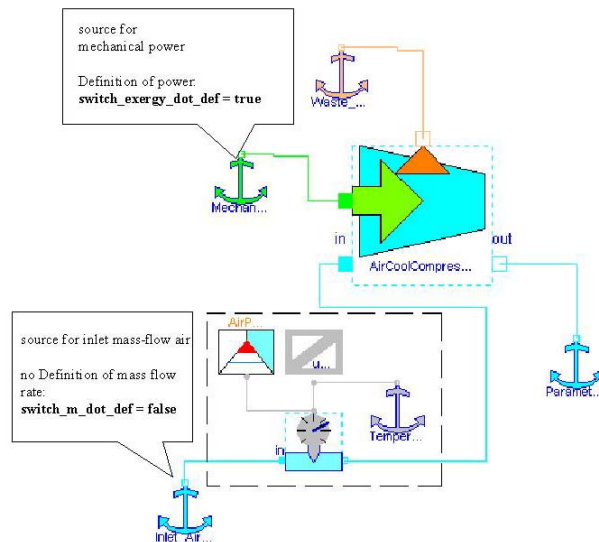


Fig.8: Modification of boundary conditions; mass flow rate results from prescribed mechanical power for compression and pressure difference.
(description_example_3_2_2)

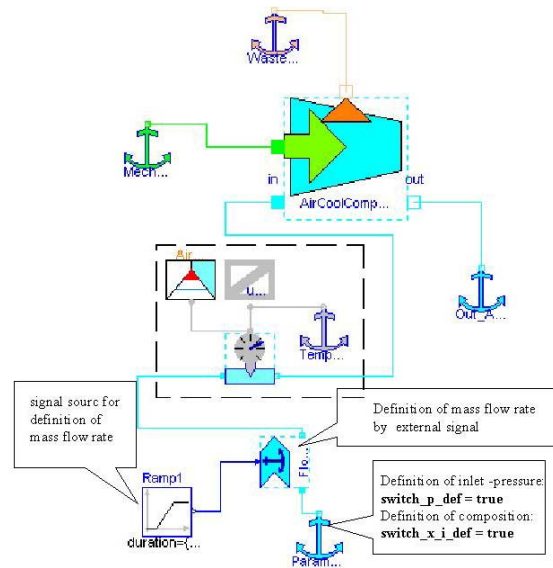


Fig.9: Definition of inlet air mass flow rate by an external signal source.
(description_example_3_2_3)

- Definition of mass flow rate and pressure at inlet by separate signal sources;** Fig. 10 shows the diagram of a compressor where not only the mass flow rate but also the pressure at the inlet is controlled by an external signal source. Model *InletPresssureSignalDefined* is connected to *Block.Source.Sine* from the Modelica standard library. Parameter *option_defsignal* of *InletPresssureSignalDefined* is set to 3, selecting pressure as variable defined by the external signal. *InletPresssureSignalDefined* introduces an additional mass flow variable, which is not relevant for the problem and is set to zero (parameter *switch_zero_m_dot* = true). The composition of the inlet air flow should remain constant and is defined by parameter *x_para_i* in model *InletParameterDefined*

-

The screenshot displays a Simulink model diagram for an air cooling system. The central component is the **AirCoolCompressor**, represented by a large blue arrow pointing right. It has two main inputs: **in** (from the left) and **out** (to the right). The **in** input is connected to a **Waste...** component (purple anchor icon) and a **Mechan...** component (green anchor icon). The **out** input is connected to an **AirCoolCompress...** component (blue box). The output of the **AirCoolCompress...** component is connected to a **Ramp1** component (graph icon). The **Ramp1** component's output is connected to the **duration(2)** input of the **SignalOutletVelocity** component (black circle icon). The **SignalOutletVelocity** component's output is connected to the **Out_Air...** component (blue anchor icon). A callout box points to the **model DensityCompressedAir:** correlation density, spec. enthalpy and pressure. Another callout box points to the **model OutletVelocity:** correlation mass flow rate/velocity. A third callout box points to the **model SignalOutletVelocity** signal source for velocity.

- the extend of variation for a state variable within a model; if the variation is limited, the application of simple linear property models may be sufficient without introducing

significant errors; e.g. if a simulation deals with a gas at room temperature at ambient pressure the application of the ideal gas law is often sufficient, using real gas property routines does not provide different results

- the accuracy of the property model should correspond to the accuracy of the other physical models; e.g. in two phase flow the results provided by models for pressure loss or heat transfer coefficients often show errors within the range of 30-50%, using complex models for calculating the density of the medium is not efficient in combination with models of limited accuracy.
- in dynamic simulations the assumption of thermal equilibrium in the working fluid may be not valid; the application of high accuracy property routines describing steady state systems does not improve the quality of the model compared to the real world.

For the evaluation of property routines different criteria can be applied:

- the accuracy of results compared to standardized values
- the consistency for any combination of independent state variables

Usually, a property model can't fulfil both of these criteria to the same degree; high accuracy routines are often based on polynomials which are optimized for a specific combinations of independent state variables, a change of the independent state variables requires the application of iterative solution procedures. Simple property routines often allow a symbolic manipulation of the equations for any combination of independent state variables. If the accuracy of the results is acceptable, models allowing a symbolic manipulation should be preferred.

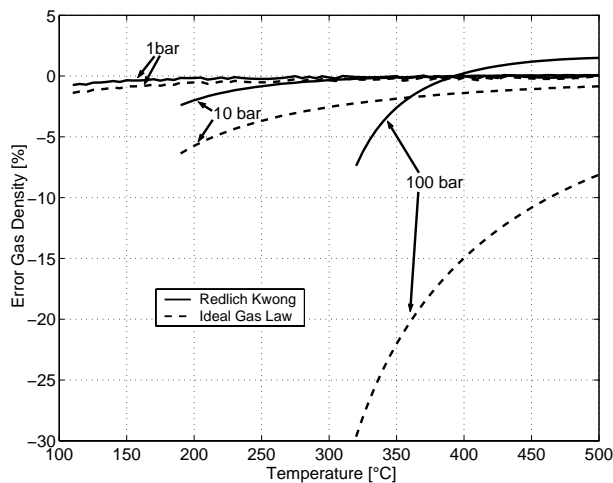


Fig.12: Accuracy of the results for density of steam provided by the Ideal Gas model and the Redlich-Kwong equation of different pressure levels.

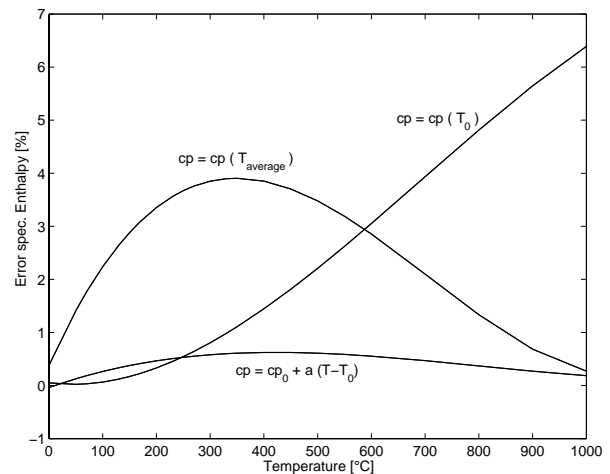


Fig.13: Accuracy of calculated spec. enthalpy of air for various approximations of the specific heat capacity.

3.3.1 Structure of main package Medium and Icons used for graphical representation of medium models

The models for calculation of thermophysical properties in package Medium are organised in five sub packages:

- Gas
- Liquid
- Solid

- MultiPhase
- MultiComponent

These five packages are completed by package

- MediumSpecificData

including fundamental data for various media. Package

- MathTool

provides mathematical routines needed by the property models.

A second criterium for classification are the physical properties calculated by a model. Three different groups can be distinguished:

- correlations for volumetric variables pressure, density and temperature
- correlations for caloric variables spec. enthalpy, spec. internal energy and spec. entropy
- correlations for transport properties like viscosity, heat conductivity or diffusion coefficients

The icons of the medium models should support a fast identification of the various models. The colours and filling style used for the segments in the basic symbol indicate

- range of validity (solid, liquid, gaseous)
- state variables used in the models (volumetric and / or caloric variables)
- medium

Fig.xx shows the meaning of the segments in the basic symbol: the triangle in the middle is divided in three horizontal areas representing the solid, liquid and gaseous state. If the model describes an entire region, the corresponding area is filled by the phase specific colour (black: solid, blue: liquid, red: gaseous). If the model is valid for the saturated state (e.g. boiling liquid, saturated gas) the corresponding area is filled in dashed style.

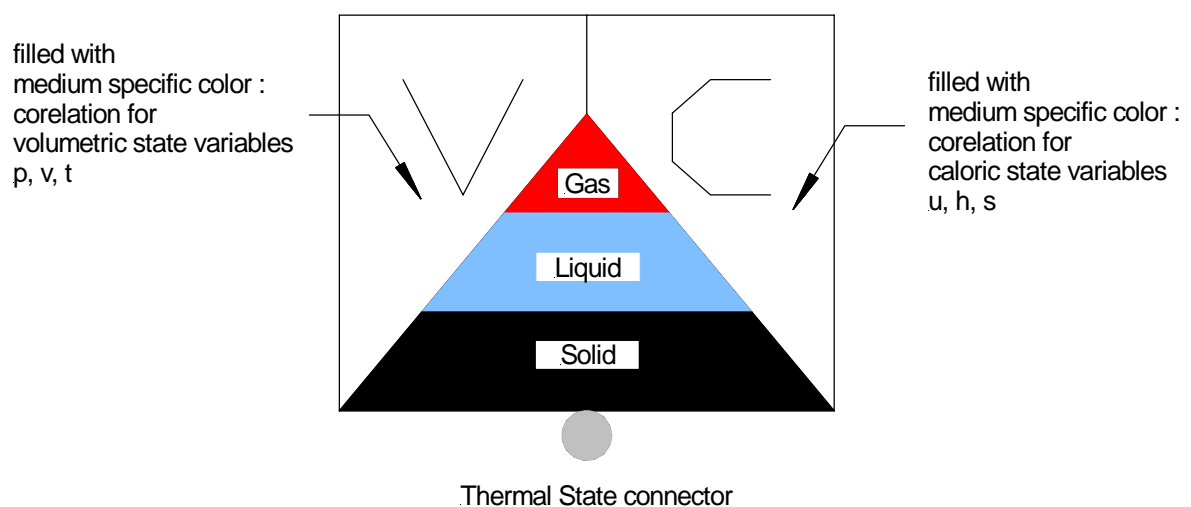


Fig.13: Meaning of segments in icon used for medium models.

Fig.xx-Fig.xx show icons for various medium models in package Medium; since TechThermo offers models of varying complexity for the calculation of thermophysical properties, identical icons may be used for various models.

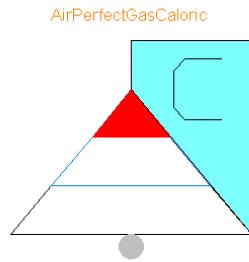


Fig.14: Icon of model with correlation for caloric state variables of gaseous air assuming constant c_p

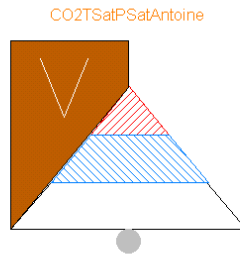


Fig.15: Icon for model with correlation between saturation temperature / saturation pressure for CO₂ for boiling liquid and saturated steam.

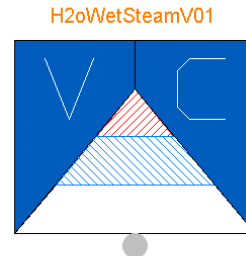


Fig.16: Icon for models with correlations between caloric and volumetric state variables of twophase region (liquid/gas) for H₂O.

3.3.2 Basic concepts for physical property models in package Medium

For the models in package Medium two basic rules apply:

- 1 only general correlations are used; the adaptation for a specific medium is done by modification of a small number of parameters.
- 2 the data needed to specify a correlation for a medium should be available easily,

The fundamental data specifying a medium is collected in a record extending the record MediumThermoFundamentalConstants in package Medium. MediumSpecificData.Data:

```
record MediumThermoFundamentalConstants
"record defining reference state for thermophysical properties TTcode:CfD1"

    parameter SIunits.MolarMass m_mol "molar mass";
    parameter SIunits.ThermodynamicTemperature t_critical
        "critical temperature";
    parameter SIunits.Pressure p_critical "critical pressure";
    parameter SIunits.Density rho_critical "critical density";
    parameter SIunits.SpecificHeatCapacityr_gas=
        GeneralConstants.R/m_mol "specific gas constant";
    parameter Real omega_acentric "acentric factor";

end MediumThermoFundamentalConstants;
```

This record includes the molar mass, the pressure, temperature and density at the critical point, the resulting gas constant and the acentric factor.

For example, the record containing these fundamental parameters for water is

H2oThermoFundamentalConstants in package
Medium. MediumSpecificData.Data:

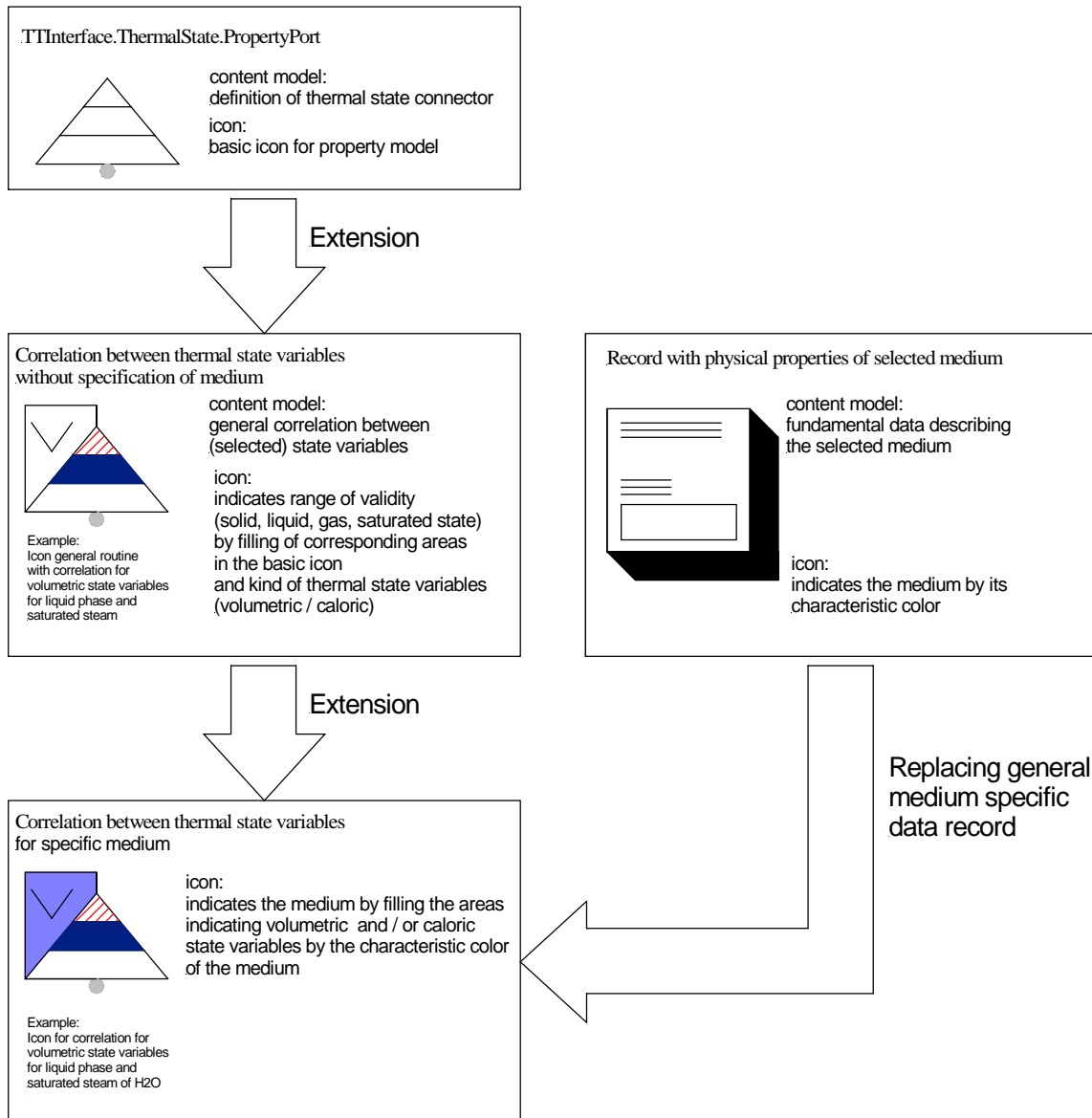


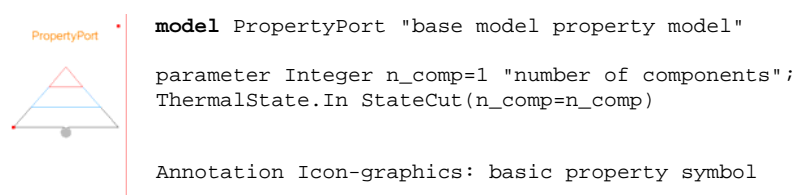
Fig.17: Basic concept for implementation of thermophysical property models in TechThermo

3.3.3 Example for property model: ideal gas law for air

Model `AirIdealGasVolumetric` provides the ideal gas law $p / \rho = RT$ for air. The values for the variables pressure p , density ρ and temperature $t = T - 273$ are exchanged with the environment by the thermal state connector `StateCut`; variables h , u , s , x , x_i are not used. The development of the model can be divided into three stages:

Model `PropertyPort` from main package `Interface` is taken as basis for the model.

This model provides the declaration of the thermal state connector `StateCut` and includes the basic property icon.




```
end PropertyPort;
```

Model `IdealGasVolumetricNoProp` extends `PropertyPort` and adds the correlation $\text{StateCut.p} = r_{\text{gas}} * (\text{StateCut.t} + 273.15) * \text{StateCut.rho}$ for the connector variables. The gas is specified by the gas constant r_{gas} . If the value of the switch-parameter `switch_r_const` is true, the value of r_{gas} is defined by the specific molar mass of the gas using the universal gas constant `GeneralConstants.R`. The molar mass is included in the record `SpecificConstants` that is introduced in line 1. The icon is modified to indicate the range of validity (gas phase) and the used state variables (volumetric variables p , ρ and t).



```

model IdealGasVolumetricNoProp "p/rho=RT"
  extends TTInterface.ThermalState.PropertyPort;

1    replaceable TTMedium.MediumSpecificData.Data.MediumThermoFundamentalConstants
    SpecificConstants "record with medium specific constants";
2    SIunits.SpecificHeatCapacity r_gas "spec. gas constant";
3    parameter Boolean switch_r_gas_const=true
4    "if switch_r_gas_const==true then specific gas constant r_gas is defined by
    parameter molar mass SpecificConstants.m_mol";

Annotation Icon-graphics: modification of the basic property icon included in
TTInterface.ThermalState.PropertyPort:
- The area representing the gas phase in the property symbol is filled
- The letter 'V' is added to the icon indicating that the model includes a
correlation for variables p,v,t

5    equation

6    if switch_r_gas_const==true then
7      r_gas = GeneralConstants.R/SpecificConstants.m_mol;
8    end if;

9    StateCut.p = r_gas*(StateCut.t + 273.15)*StateCut.rho;

10   end IdealGasVolumetricNoProp;

```

Model `AirIdealGasVolumetric` extends `IdealGasVolumetricNoProp`; a redeclare-statement is used to replaced the general record for fundametal properties by the record `TTMedium.MediumSpecificData.Data.AirThermoFundamentalConstants` containing the data for air. In the icon the left area is filled in light blue, indicating that the property routine is specified for dry air.



```

model AirIdealGasVolumetric "p/rho=RT for Air"
  extends TTMedium.Gas.Support.IdealGasVolumetricNoProp
  (redeclare TTMedium.MediumSpecificData.Data.AirThermoFundamentalConstants
  SpecificConstants);

```

Annotation Icon-graphics: modification of the property icon included in `TTInterface.ThermalState.PropertyPort`:

- The area representing the gas phase in the property symbol is filled
- The letter 'V' is added to the icon indicating that the model includes a correlation for variables p , v , t

```
end AirIdealGasVolumetric
```

3.3.4 Combined property models

Thermophysical property models in TechThermo are also implemented by combination of property routines. Models providing correlations for the complete set of state variables are composed of a model for calculation of the caloric state variables and a model for calculation of the volumetric.

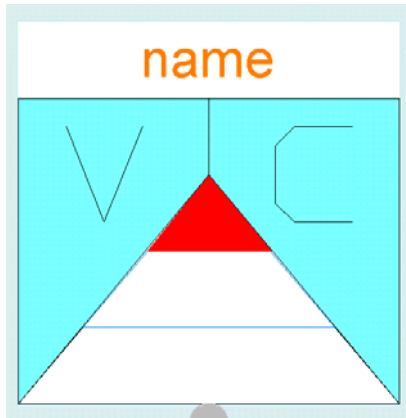


Fig.18: Icon of thermophysical property model AirPerfectGasCalVol

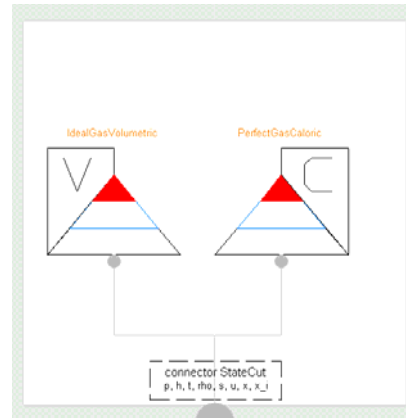


Fig.19: Diagram of model AirPerfectGasCalVol

This approach is especially used for models describing systems with more than a single phase or more than a single component. Fig.xx shows the internal structure of the model for calculation of the thermophysical properties of wet steam. The properties for saturated liquid and saturated steam are calculated by using models for the liquid phase and gas phase with saturation temperature and saturation pressure as input. This also avoids discontinuities at the borders between the two phase region and the single phase regions. Additional models provide correlations between the saturation temperature and the saturation pressure and the heat of evaporation. The model TwoPhaseMix calculates the properties of the two phase system from the information of the basic models including also the steam quality

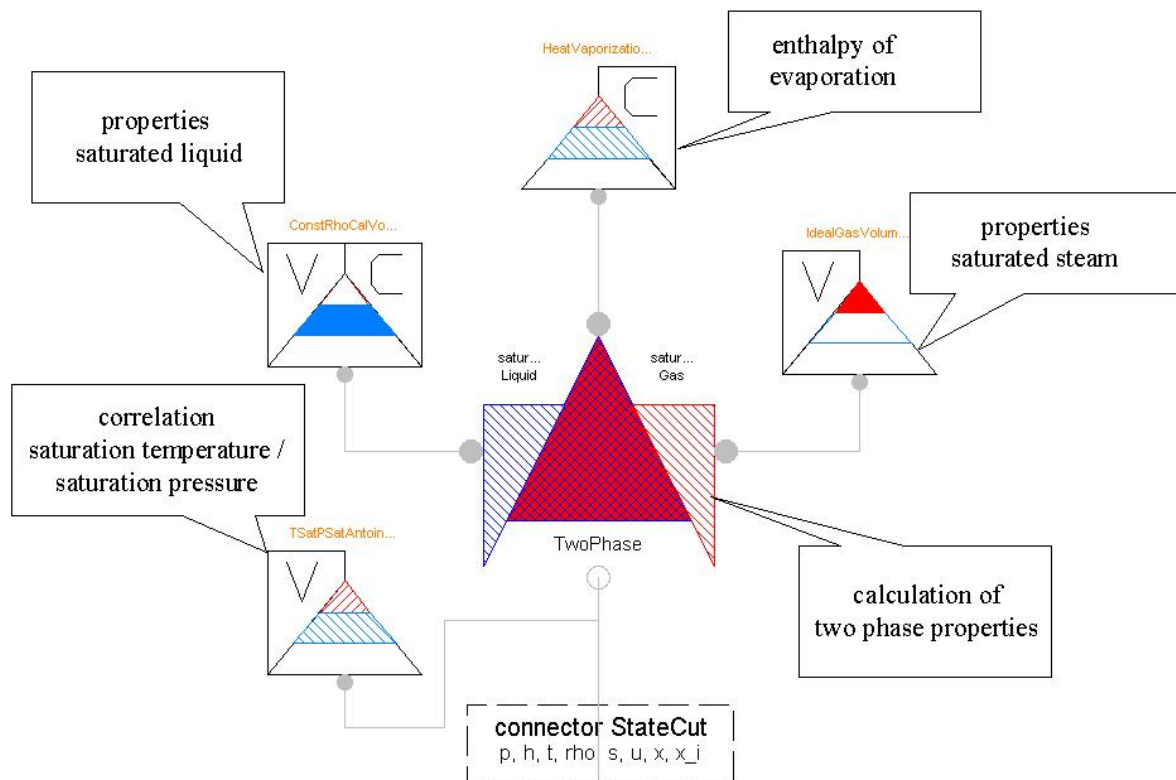


Fig.20: Internal structure of model for calculation thermophysical properties of wet steam

3.4 Main package Basis

The main package Basis comprises elements describing fundamental processes in thermodynamic systems. Four subpackages contain these models:

- HeatTransport;
descriptions of heat transport by conduction, convection or radiation
- MassTransport;
models describing transport of mass
- Compartment;
control volumes for heat and mass transport
- Junction;
junctions for heat and mass flows
- BasicProcess
basic thermodynamic processes

Subpackage HeatTransport

Models for heat conduction

Heat conduction in plates, cylinders and spheres is calculated by the models PlateHeatConducting, CylinderHeatconducting and SphereHeatConducting.

These models extend the TwoPort model from package Interface/HeatFlow. Depending on the values of structural parameters, different options are available:

- the models represent either a heat resistance without storage capacity (`switch_zero_mass = true`) or an element with a finite thermal capacity (`switch_zero_mass = false`)
- the models can be discretized in one direction, the parameter `n_segment` defines the number of segments. The results for the temperature of these segments are stored in an array.
- the material of the heat conducting element is described by the variables `k_thermal` (heat conductivity), `c_heat` (heat capacity, only relevant if `switch_zero_mass = false`) and `rho_material` (density, only relevant if `switch_zero_mass = false`). These values are defined by parameters, if the corresponding structural parameters `k`
- an internal heat source/heat sink can be included by the variable `q_internal_volumetric`. If the structural parameter `q_internal_const = true`, the value of `q_internal_volumetric` is defined by the parameter `q_internal_volumetric_const`
- the initial condition for the element with thermal capacity can be defined either by the initial heat flow rate = 0 (`option_initial = 1`) or by an initial temperature (`option_initial = 2`). The value for the initial temperature is defined by parameter `trinitities`

Models for convective heat transfer

The calculation of the heat transfer between a fluid and the surface of a solid body in contact with the fluid demands the consideration of different aspects:

- forced or free flow of the fluid
- geometry of the body
- extend of variation of fluid properties

Models for heat transfer by radiation

Models for heat transfer by thermal radiation are

- `RadiationHeatEmission`: calculation of heat emitted from a finite surface to an infinite environment by thermal radiation
- `RadiationHeatExchange`: model describing radiation heat transfer between two surfaces

Subpackage MassTransport

-Models for convective heat transport

The models in package MassTransport describe physical processes related to the transfer of mass.

- `Support.PressureLossPipeNoProp`: length specific pressure drop for mass flow in pipe; includes various models for the calculation of turbulent and laminar flow, selection by structural parameter. The thermophysical properties of the fluid must be specified to complete the model.

Subpackage Compartment

-Models including conservation laws

Subpackage Compartment includes models with conservation laws for heat and mass.

- `ThermalCapacity` is the transient thermal energy balance for a finite mass. The mass transfers energy to / from other models by a heatflow-connector

- `Support.SingleMassControlVolumeNoProp` includes the transient mass and energy balance for a medium in a finite volume with additional heat flow. The model includes a mass flow connector and a heat flow connector. The model requires correlations for the thermophysical properties of the medium which can be transferred by a thermal state connector. By a heat flow connector thermal energy can be transferred to or from the volume. This model is intended for the modelling of pressure vessels.
- `Support.TwoPortMassControlVolumeNoProp` is identical to `Support.SingleMassControlVolumeNoProp` except for a second mass flow connector. This model should be preferred for fluid flow elements.

Subpackage Junction

- `MassFlow3PortParaCtrl` is a node with three mass flows. One of the mass flow is defined as inflow, a second as outflow. The direction of the third mass flow is defined by a structural parameter
- `DividerMassFlow`, `DividerHeatFlow`, `DividerExergyFlow` are used for models where a single mass-, heat- or exergy flow is distributed into a parameter-defined number of identical parallel flows. These models include two connectors, one representing the variables of the single flow, the other variable includes the variables for one of the parallel flows.
- `MassFlowDouble`, `HeatFlowDouble`, `ExergyFlowDouble` are used to copy the connector variables of a mass-, heat- or exergy flow. These models are intended for analysis purposes, e.g. the calculation of efficiencies of a process

Subpackage BasicProcess

Subpackage `BasicProcess` contains models describing simple changes in the thermal state of a system and models needed for the analysis of thermal processes.

- `AirChangeState` is the model for a single step change in thermal state of dry air. The kind of change is defined by a structural parameter. `AirChangeState` extends `Support.ChangeStateNoProp` which is the basic model without specification of the working fluid.
- `ThermalEfficiency` calculates the thermal efficiency from an exergy flow and a heat flow

4 Examples

The examples should demonstrate the application of the TechThermo models. They are not intended as solutions for real problem, since the complexity of the example should be limited.

Ex1: Heat conduction in a wall with varying boundary conditions

- Definition of boundary conditions for heat flow by element
Source.HeatFlow.ParamterDefined and Source.HeatFlow.SignalDefined
- definition of model features by structural parameters

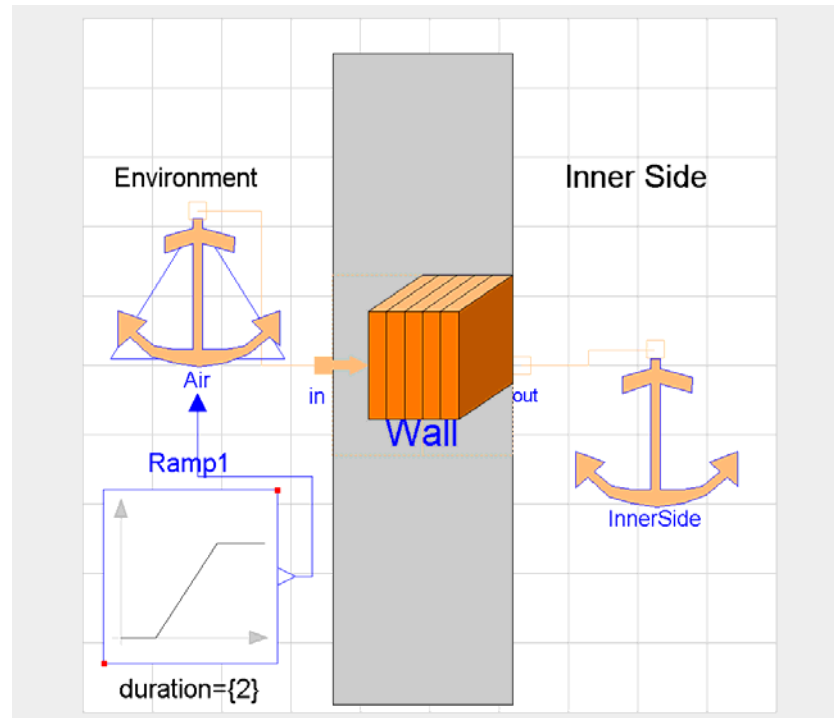


Fig.Ex1_1: Model for heat-conducting wall

Ex1a Steady State Heat Conduction

Heat conduction in a wall without thermal capacity, temperature varying on one side, temperature constant on the other side

Definition of boundary conditions

Inner side: **constant temperature of 15°C**

Name model:	InnerSide	
Parameter:	Value:	Comment:
switch_t_def	true	at connector OutHeatFlow of model InnerSide the variable t is defined by parameter t_para
t_para	15	parameter value for temperature at connector OutHeatFlow of model InnerSide

Environment: tempeature starts at 15°C, increases by 20°C and remains constant at 35°C

Source model controlled by external signal:

Name model:	Air	
Parameter:	Value:	Comment:
option_def_signal	2	at connector OutHeatFlow of model Air the variable t is defined by an external signal source connected to connector InSignal of model Air

The external signal source used to control the temperature at connector OutHeatFlow of model Air is taken from the Modelica Standard library (Modelica.Blocks.Sources.Ramp):

Name model:	Ramp1	
Parameter:	Value:	Comment:
height	20	the temperature of the environment increases by 20°C
offset	15	the temperature of the environment starts at 15°C
duration	2	the temperature of the environment is changed from 15°C to 35°C within 2 seconds.

The wall is represented by a heat conducting element

Name model:	Wall	
Parameter:	Value:	Comment:
n_segment	1	no spatial discretization in direction of heat flow
k_thermal_const	1.5	thermal heat conductivity of wall material, does not change during the simulation
dz_plate	0.5	thickness of plate in direction of heat conduction
cross_area	0.25	cross-sectional area for heat conduction through wall
switch_zero_mass	true	wall has no thermal capacity

The parameters `c_heat_const` (specific heat capacity of wall material), `rho_material` (spec. density of wall material) and `t_initial` (initial temperature wall) are not relevant here, since due to the setting of structural parameter `switch_zero_mass = true`, no energy is stored in the wall. The wall is regarded as a heat conducting element that transfers heat without any delay. Usually, this assumption should only be made for thin wall structures.

The check of model Ex1a provides the message
“DAE with 20 unknowns scalars and 20 scalar equations”

After compilation, the simulation time interval is set to 3 seconds in the Experiment Setup menu.

The simulation provides the following results for the temperature at the outer side of the wall (imposed by the signal source) and the heat flow at the inner side of the wall at connector OutHeatFlow of the model InnerSide:

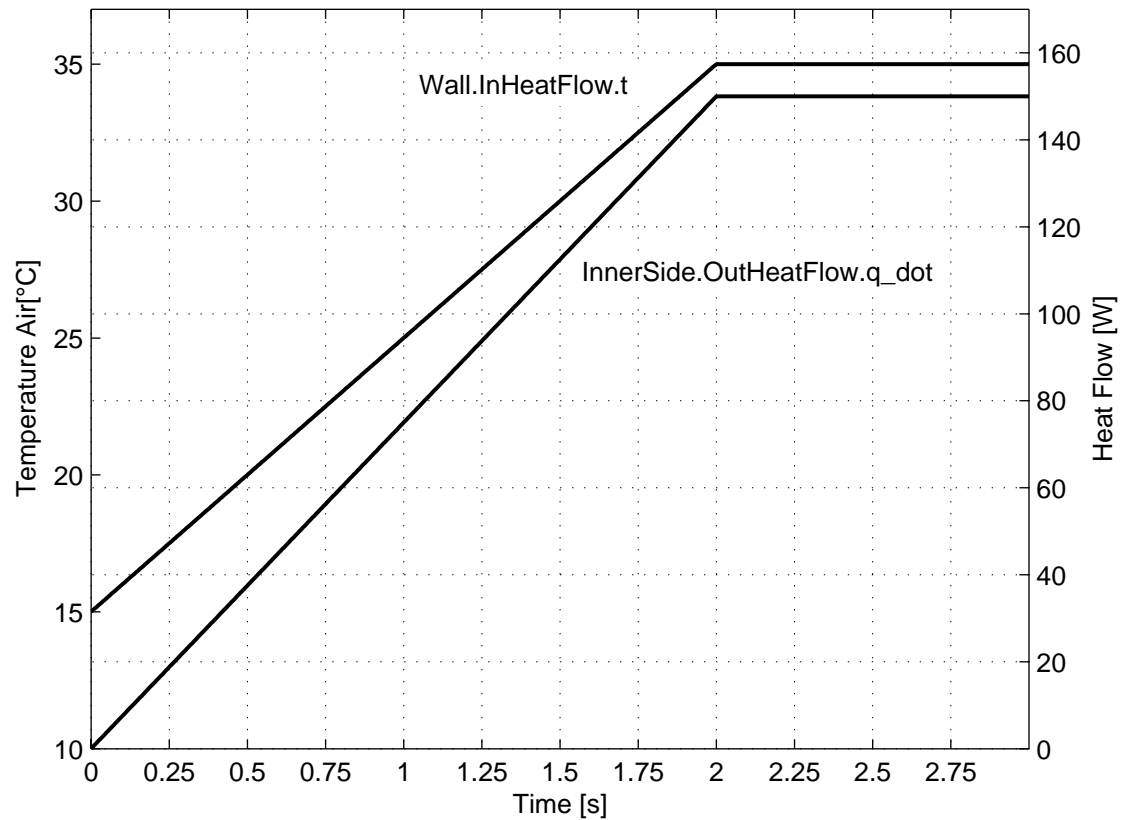


Fig.Ex1_2: Simulation results for model Ex1a, wall without thermal capacity.

The value for the heat flow at the end of the ramp (time >2seconds) is

$$q_{\dot{}} = \frac{(\text{Wall.InHeatFlow.t} - \text{Wall.OutHeatFlow.t}) \cdot k_{\text{thermal_const}} \cdot \text{cross_area}}{dz_{\text{plate}}}$$

$$= \frac{20 \cdot 1.5 \cdot 0.25}{0.05} \text{ W} = 150 \text{ W}$$

which corresponds to the value in Fig.2, taking the right y-axis. Since the thermal capacity of the wall is neglected, there's no delay between temperature and heat flow rate.

Ex1b- Transient Heat conduction with finite thermal capacity –sudden increase of surface temperature

In this example, we want to calculate the transient temperature in the wall at a position that is 0.036m away from the surface. One dimensional heat flow is assumed.

In model Wall the following parameter-values are used:

Name model:	Wall	
Parameter:	Value:	Comment:
n_segment	1	no spatial discretization in direction of heat flow
k_thermal_const	1.0	reduction of thermal heat conductivity
dz_plate	0.072	the center of the heat conducting element should be at 0.036m, so the thickness of plate in direction of heat conduction is 0.072m
switch_zero_mass	false	the wall has now a finite thermal capacity, in contrast to Ex1a

Important is the value of the structural parameter `switch_zero_mass`: since the thermal capacity of the wall should now be considered, `switch_zero_mass` is now false.

The thermal capacity of the Wall-component is now defined by parameters

Name model:	Wall	
Parameter:	Value:	Comment:
c_heat_const	1100	value for specific heat material of the wall, assumed constant during the simulation (<code>switch_c_const = true</code>)
rho_material	2000	density of the wall

The volume of the wall results from cross-sectional area `cross_area` and the thickness of the wall `dz_plate`.

At the inner side of the wall there should be no heat flow , but no definition of the temperature, so the parameters of component `InnerSide` are:

Name model:	InnerSide	
Parameter:	Value:	Comment:
switch_q_dot_def	true	the value (= 0.0) of the heat flow rate at the inner side of the wall is given by parameter <code>q_dot_para</code> .
switch_t_def	false	temperature is not given at the inners side of the wall
q_dot_para	0	parameter value for the heat flow at the inner side of the model.

The change in tempeature of the environment should be from 15°C to 20°, so the parameters in component `Ramp1` are

Name model:	Ramp1	
Parameter:	Value:	Comment:
height	5	the temperature of the environment increases by 5°C
offset	15	the temperature of the environment starts at 15°C
duration	2	the temperature of the

		environment is changed from 15°C to 35°C within 2 seconds.
--	--	------------------------------------------------------------

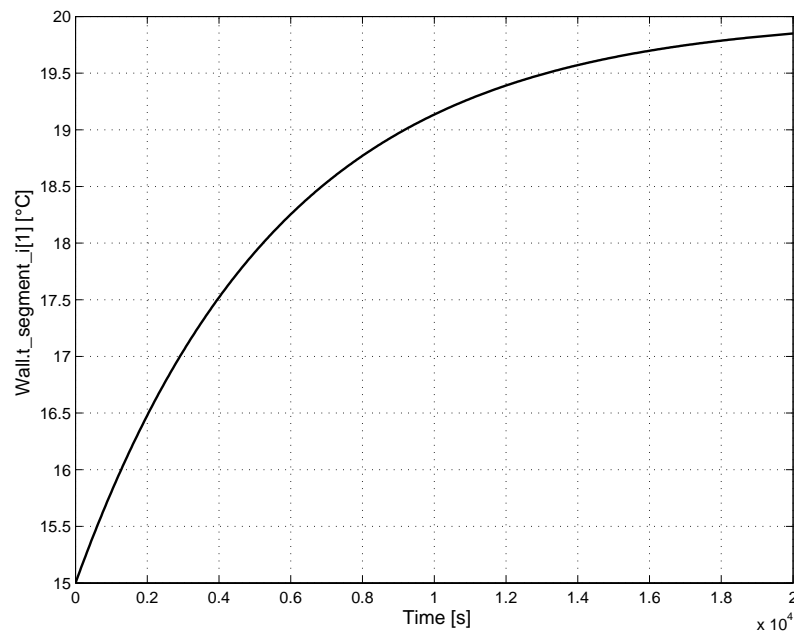
Initialization of Wall

Two options are available for initialization of model PlateHeatConducting used for component Wall:

- 1 steady conditions, the derivate of the wall temperature = 0.0
- 2 a temperature for the wall is prescribed by parameter t_initial

Which initialization is chosen depends on the value of option_initial. For all examples in Ex1 we assume steady conditions, so option_initial = 1; in Ex1b we could achieve the same results by option_initial = 2 and t_initial=15.

After compilation, the simulation provides the following results for the temperatur Wall.t_segment_i[1] during a period of 10000s.



Ex1c- Comparision Modelica finite difference solution to analytical solution

An analytical solution should be compared to the DYMOLA-results. This solution is valid for an infinite wall that is exposed to a sudden change in temperature. The temperature field is one-dimensional,

The analytical solution for the temperature T at distance z from the surface of the wall is at time t after the change of initial temperature T₀ to temperature T_s is:

$$\frac{T(z, t) - T_s}{T_0 - T_s} = \operatorname{erf}\left(\frac{z}{\sqrt{4at}}\right)$$

with a being the thermal diffusivity of the wall:

$$a = \frac{\text{Wall.k_thermal_const}}{\text{Wall.c_heat_const} \cdot \text{Wall.rho_material}}$$

One assumption for the analytical solution is that the heat flow rate becomes zero at an infinite distance from the surface of the wall exposed to the environment (boundary condition on the right side). This boundary condition can't be fulfilled by a finite difference approach, since here the maximum distance from the surface of the wall is limited.

For the first simulation, the wall is divided into two equal segments in direction of the heat flow. The position of the centers of the two segments are provided by vector `Wall.z_segment_i`.

Name model:	Wall	
Parameter:	Value:	Comment:
n_segment	2	heat conducting element is divided into two identical segments of length $dz_plate / 2 = 0.036\text{m}$
dz_plate	0.072	the centers of the two segments are at $z_segment_i[1] = 0.018\text{m}$ and $z_segment_i[2] = 0.054\text{m}$
switch_zero_mass	false	the wall has a finite thermal capacity

The simulation provides the temperature at the position of the centers of the segments in the vector `t_segment_i`.

In order to identify the influence of the spatial discretization, a second simulation is performed with a heat conducting plate divided into six segments:

Name model:	Wall	
Parameter:	Value:	Comment:
n_segment	6	heat conducting element is divided into six identical segments of length $dz_plate / 6 = 0.012\text{m}$
dz_plate	0.072	the centers of the segments are at $z_segment_i[1] = 0.006\text{m}$ $z_segment_i[2] = 0.018\text{m}$ $z_segment_i[3] = 0.030\text{m}$ $z_segment_i[4] = 0.042\text{m}$ $z_segment_i[5] = 0.054\text{m}$ $z_segment_i[6] = 0.066\text{m}$
switch_zero_mass	false	the wall has now a finite thermal capacity, in contrast to Ex1a

Fig.Ex1_4 shows the results for the two different discretizations together with the analytical solution. The results calculated with six segments match better the values from the analytical solution in the initial phase, later the differences in the boundary condition on the right side becomes visible.

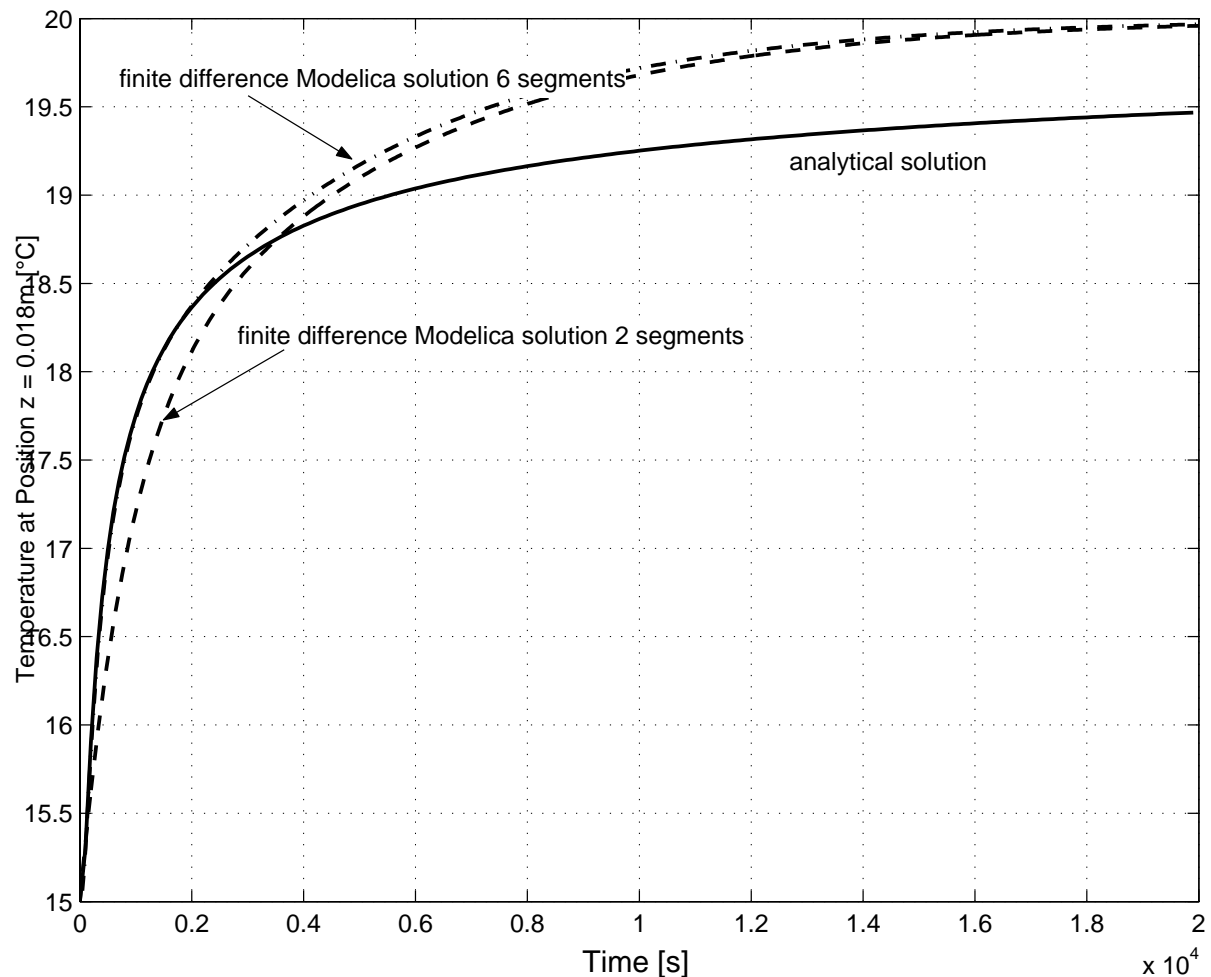


Fig.Ex1_4: Comparison of results from Ex1c for discretization of heat conducting element in 2 and 6 elements and analytical solution.

Ex1d- Transient Heat Conduction with Heat Flux as boundary condition

In this example, the area specific heat flux at the surface of the semi-infinite wall should be used as boundary condition. An analytical solution is available for a sudden variation of the heat flow at the surface.

The boundary condition by component `Air` on the left side is now the heat flow rate so the structural parameter `option_def_signal` is set to 1:

Name model:	Air	
Parameter:	Value:	Comment:
<code>option_def_signal</code>	1	at connector <code>OutHeatFlow</code> of model <code>Air</code> the variable <code>q_dot</code> is defined by an external signal source connected to connector <code>InSignal</code> of model <code>Air</code>

The cross sectional area of the wall is set to 1m²; the boundary condition for the analytical solution on the right side is again $q_{\dot{}} = 0.0$ for position $z = \infty$, so `dz_plate` is extended to better fulfill this condition. The temperature in the wall should be used as initial condition so structural parameter `option_initial` is set to 2, the value for the initial temperature should be 15° defined by parameter `t_initial`.

Name model:	Wall	
Parameter:	Value:	Comment:
<code>n_segment</code>	10	wall is discetized into 10 elements in direction of heat

		flow
k_thermal_const	1.0	thermal heat conductivity of wall material, does not change during the simulation
dz_plate	3	thickness of plate in direction of heat conduction
cross_area	1.0	cross-sectional area for heat conduction through wall
t_initial	15	initial value for temperature of wall is 15°C
switch_zero_mass	false	wall has hermal capacity
option_initial	2	temperature of wall is initial condition

The heatflux at the surface of the wall is defined by Ramp1. The area specific heat flow rate should be increased from 0 to 100 W/m² within 1 second. Since Ramp1 defines the heat flow rate at connector OutHeatFlow of component Air, the value of height in Ramp1 must be negative.

Name model:	Ramp1	
Parameter:	Value:	Comment:
height	-100	the heat flow rate leaving component Air (positive at surface of wall). Since the area of the wall is 1m², the value corresponds to the area specific heat flow rate.
offset	0	initially no heat flow at surface of wall
duration	1	the heat flow rate should be increased from 0 W/m to -100 W/m within a second.

In case of a sudden increase of the surface heat flow from 0 W/m² to \dot{q}_s the analytical solution for the temperature T at position z (initially at T_0) after t seconds is:

$$T(z, t) - T_0 = \frac{\dot{q}_s}{\lambda} \left[\frac{4at}{\pi} e^{-\frac{z^2}{4at}} - z \cdot \operatorname{erfc}\left(\frac{z}{\sqrt{4at}}\right) \right]$$

Fig.Ex1_5 shows the results of the Modelica simulation and the results provided by the analytical solution at the position $z = 0.15\text{m}$ (= Wall.t_segment_i[1]).

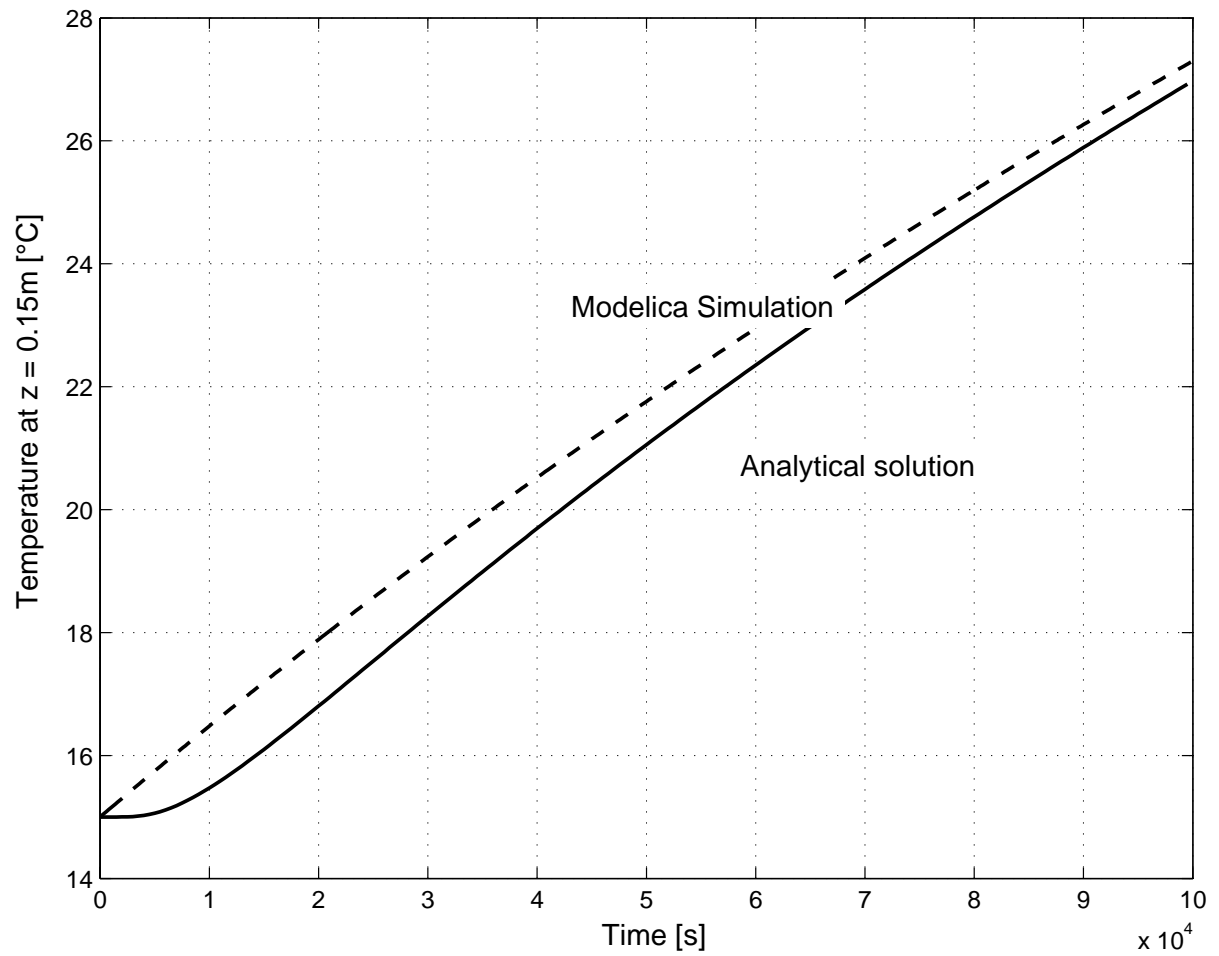


Fig.Ex1_5: Example Ex1d: temperature in wall after sudden increase of surface heat flux, results from Modelica simulation and analytical solution.

Ex2: Heat losses of cup of coffee

In the following examples, heat losses from a cup of coffee to the environment are calculated. The examples consider the following heat transfer mechanism:

- Ex2a: Heat conduction
- Ex2b: Heat conduction and natural convection
- Ex2c: Heat conduction, natural convection and radiation

The geometry of the problem is shown in Fig.Ex2_1: the cylinder-shaped cup is filled completely with coffee and is located on a large table.

The simulation starts immediately after the coffee has been filled into the coffee. The initial temperatures are:

- coffee: 90°C
- cup: 20°C
- table and environment: 20°C

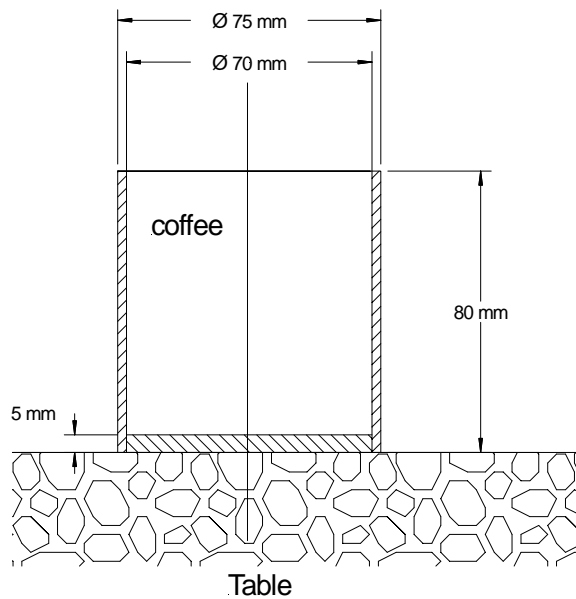


Fig. Ex2_1: Physical model of cup of coffee

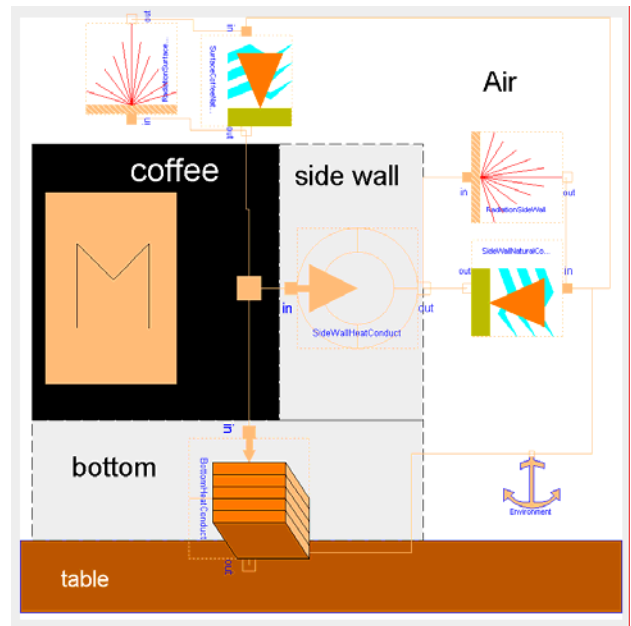


Fig. Ex2_2: Modelica model Ex2c with heat losses from cup of coffee to environment by heat conduction to table, heat transfer by natural convection and thermal radiation.

The following simplifications are assumed:

- the temperature in the coffee volume is homogenous
- the temperature in the side wall of the cylinder is homogenous
- the temperature in the bottom of the cup is homogenous
- no mass transfer between coffee volume and environment
- the temperature of the table is constant
- material properties are not temperature-dependent
- the thermal resistance between the coffee and the inner surface of the cup is neglected

Ex2a: Heat losses due to conduction

In the first example, heat is only lost to the table by thermal conduction through the bottom of the cup. Heat is transferred from the coffee to the walls of the cup, since the initial temperature of the coffee is higher than the initial temperature of the walls.

coffee volume: **ThermalCapacity**

Since there's no mass transfer the coffee is simply represented by a model of type **ThermalCapacity** from sub-package **Basis / Compartment**. The density of the coffee is assumed to be 965 kg/m^3 (constant), the volume of the coffee is $2.88 \times 10^{-4} \text{ m}^3$, so the mass of the coffee is 0.2785 kg. The specific heat capacity of the coffee is 4200 J/kg/K . The initial condition is defined by the temperature of the coffee, so `option_initial = 2`, `t_initial` is 90.

side walls of cup: **SideWallHeatConduct**

The side walls of the cup are represented by the component **SideWallHeatConduct** of type **CylinderHeatConducting** from sub-package **Basic / HeatTransport**. The values for `d_inner` and `d_outer` are taken from Fig. Ex2_1, the density of the material `rho_material_const` is 1500 kg/m^3 , the heat capacity `c_heat_const` is 900 J/kg/K , the heat conductivity `k_thermal_const` is 1.2 W/m/K . Since the mass of the cylinder should be considered, `switch_zero_mass` is false.

bottom of the cup: **PlateHeatConducting**

The bottom of the cup is represented by a model of type **PlateHeatConducting** from sub-package **Basis / HeatTransport**. The thickness `dz_plate` is taken from Fig. Ex2_1, the surface area orthogonal to the direction of heat flow is 0.0038 m^2 . No heat resistance is assumed between the bottom of the cup and the surface of the table.

Boundary conditions: **Environment and Isolation**

Model **BottomHeatConduct** is connected to model **Environment** of type **Source/HeatFlow/ParameterDefined** which defines the temperature of the environment and is also used as sink for the heat losses of the cup. **SideWallHeatConduct** is connected to model **Isolation** which defines `heatflow = 0.0`.

Fig. Ex2_3 shows the temperature of the coffee (variable `Coffee.Volume.HeatCut.t`) during a time interval of 3600s.

Ex2b: Heat losses due to conduction and natural convection

In Ex2b additional heat losses due to natural convection are included. The losses of the surface of the coffee are calculated by model **SurfaceCoffeeNaturalConvection**, assuming an empirical correlation for the heat transfer from a horizontal plate. The surface area is 0.0038 m^2 , the perimeter is 0.22m. The properties of air are defined by heat conductivity `lambda_fluid_const` = 0.0257 W/m/K , kinematic viscosity `nu_fluid_const` = $1.511 \times 10^{-5} \text{ m}^2/\text{s}$ and Prandtl-number = 0.713.

The heat losses of the outer wall of the cup are calculated by the component **SideWallNaturalConvection** using an empirical correlation for external natural convection. The cylinder geometry is selected by setting the value of `option_geometry` to 2, the size is defined by the outer diameter in parameter `d_body` and the height of the cup as value of parameter `l_parallel_flow_direction`. The two models for convective heat transfer are connected to the component **Environment**, which provides the temperature of the air and is used as a sink for the lost thermal energy.

The results for the temperature of the coffee are included in Fig. Ex2_3. Due to the additional heat losses, the temperature declines faster.

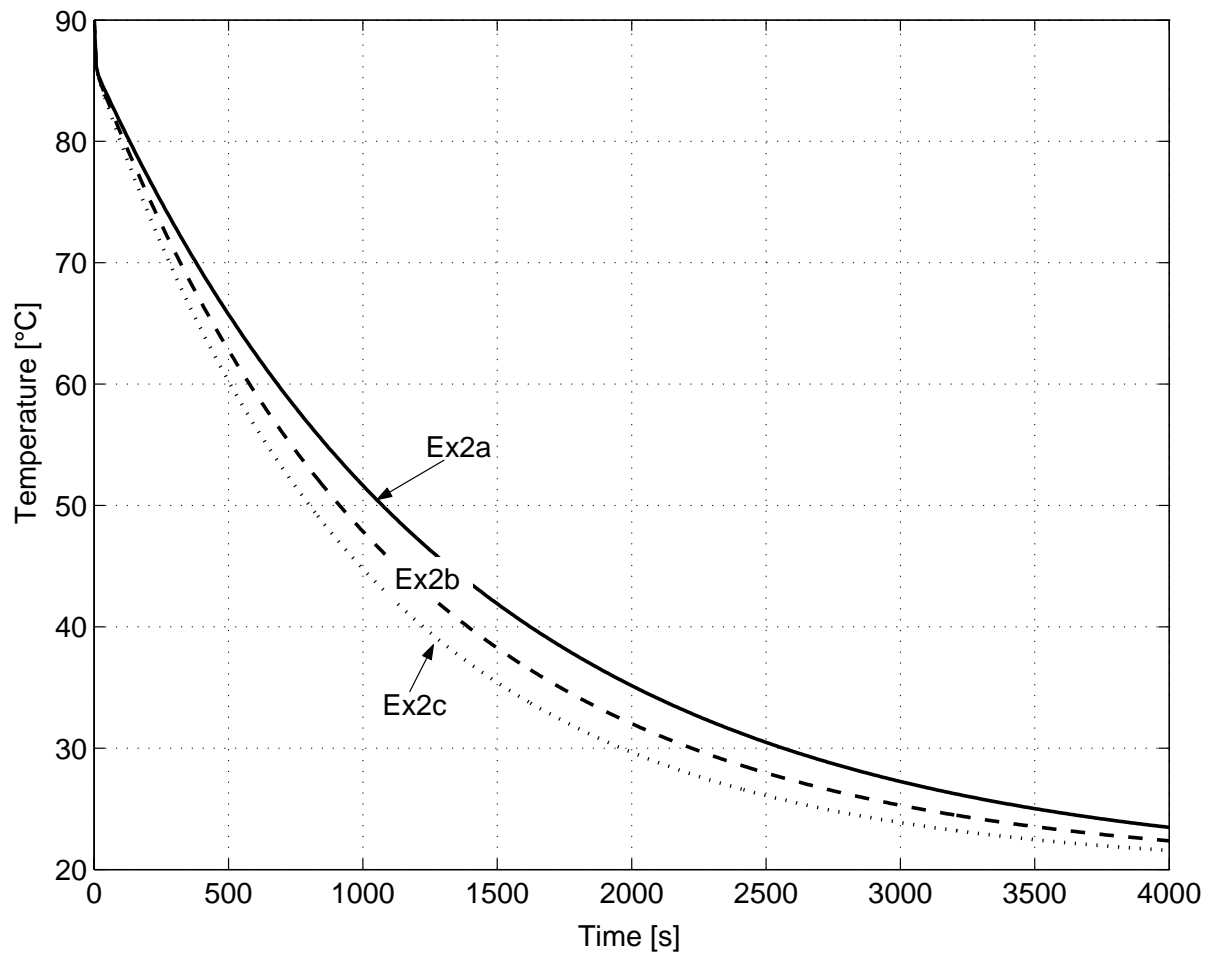


Fig. Ex2_3: Temperature of coffee for the examples Ex2a, Ex2b, Ex2c

Ex2c: Additional Heat losses due to thermal radiation

In Ex2c two components describing heat transfer by thermal radiation are added: `RadiationSurfaceCoffee` calculates the heat emitted by the surface of the coffee, `RadiationSideWall` determines the thermal radiation of the side walls. Both models require the definition of the surface and the emission coefficient by parameters. According to Fig Ex2_3,

Ex3: Steam power plant with saturated steam

Simple steam power plants are simulated in these examples. Transient effects are not considered since the components used here don't show thermal capacities. Working medium is liquid water and wet steam. Part load behaviour of components is neglected.

Ex3a: Basic steam cycle

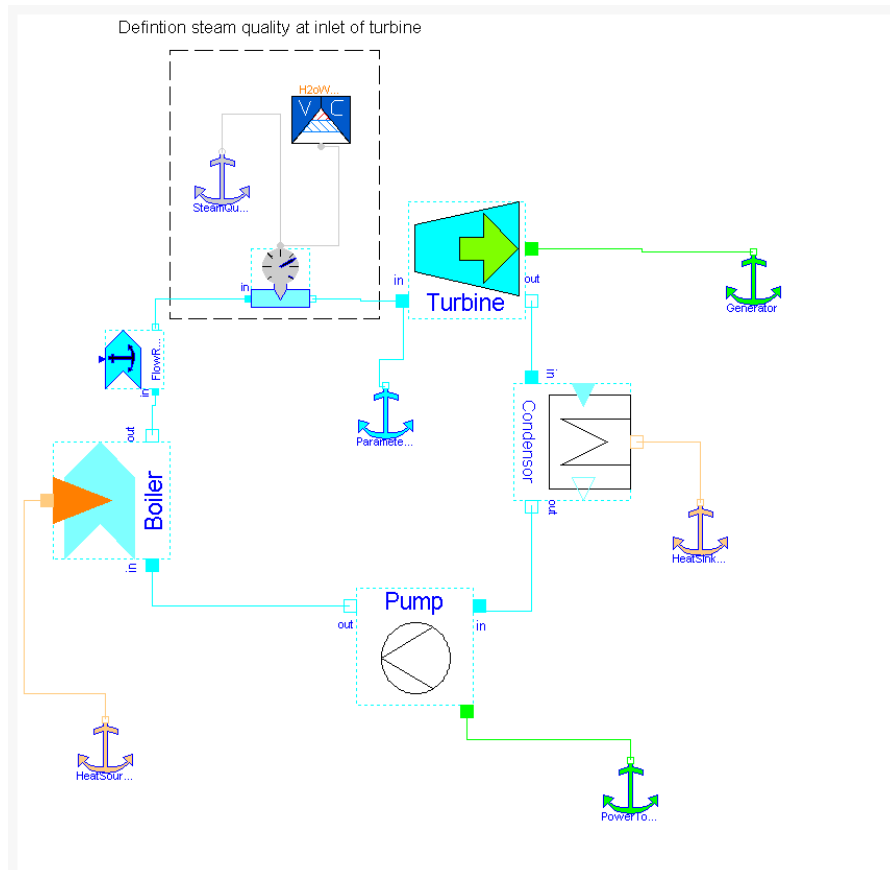


Fig. Ex3_1: Diagram layer of model Ex3a

The basic cycle consists of four main components:

Component	Model type	Function
Turbine	Component.Turbine.H2oWetSteamTurbine	generation of exergy by expansion of steam
Condensor	Component.HeatEx.H2OCondensor	condensates wet steam from turbine
Pump	Component.Pump.H2oLiquidPump	compression of water from condensation pressure to turbine inlet pressure
Boiler	Component.Heater.SimpleHeater	generates steam

The necessary boundary conditions are defined by the following components:

Component	Model type	Boundary condition
SteamQualityPressureInletTurbine	Source.ThermalState. ParameterDefined	steam quality inlet turbine = 1.0, saturated steam
ParameterTurbineInlet	Source.MassFlow. ParameterDefined	pressure at inlet of turbine, x_i
FlowRate	Source.MassFlow. FlowRateCtrl	mass flow rate in steam cycle
HeatSinkEnvironment	Source.HeatFlow. ParameterDefined	temperature of condensation in condensor
HeatSource	Source.HeatFlow. ParameterDefined	temperature of heat transferred to boiler

The heat flow provided by component HeatSource results by the change of specific enthalpy and the mass flow rate. The boiler model is based on an energy balance, the definition of the temperature of the provided heat is only necessary to complete the equations. Saturated steam should be fed to the turbine, the enthalpy is calculated by the medium model H2oWetSteamInlet for the pressure defined in ParameterTurbineInlet. The pressure at the outlet of the turbine results from the condensation temperature in the condensor; in model Component.HeatEx.H2OCondensor a constant temperature difference is assumed between the temperature provided at the heat flow connector from component HeatSinkEnvironment.

Some components are necessary as sinks and sources without defining connector variables:

Component	Model type	Function
Generator	Source.ExergyFlow.ParameterDefined	mechanical power provided by steam turbine
PowerToPump	Source.ExergyFlow.ParameterDefined	mechanical power needed for compression of feedwater

Interruption of closed loop for mass flow rate and composition vector x_i

The components of the steam power plant build a closed loop; if all elements included relations for mass flow rate and composition vector x_i at inlet and outlet connectors too many equations would be introduced. In order to avoid this, component FlowRate doesn't include correlations between the mass flow rate and x_i at the In and Out connector, the corresponding switch parameters switch_m_dot_const = false and switch_x_i_const = false.

Ex3b: Basic steam cycle with modified boundary conditions and calculation of thermal efficiency

Ex2b includes following modifications:

- variable pressure at inlet of turbine: component PressureInletDefinition defines the pressure depending on the signal provided by component PressureInletSignal, the composition vector x_i at the inlet of the turbine is defined by model CompositionDefinition.
- in steam turbines the minimum steam quality is limited, a certain fraction of liquid water must not be exceeded; hence the steam conditions at the exit of the turbine are now determined by the steam quality defined by parameter in model SteamQualityOutlet, the corresponding spec. enthalpy is provided by model H2oWetSteamOutlet.
- the thermal efficiency of the process is calculated by model ThermalEfficiency (variable ThermalEfficiency.eta). The thermal efficiency is defined as the ratio of the thermal power consumed by the process to the mechanical power provided by the process. This information is provided by the models HeatFlowDouble and ExergyFlowDouble to the model ThermalEfficiency.

Ex3c: Basic steam cycle with two turbinstages and extraction of liquid

Decreasing the minimal temperature in the condensor means increasing the efficiency of the process. Extracting liquid during the expansion represents one option to lower the pressure at the exit of the steam turbine while fulfilling the steam quality boundary condition at the exit.

This approach is used in Ex2c:

- two sequential turbines are used, Stage1Turbine and Stage2Turbine. The pressure at the exit of Stage1Turbine is determined by parameter in ExitPressureTurbine1.
- H2oPhaseSeparator divides the wet steam flow from Stage1Turbine into two single phase flows. The gas phase flows to the inlet of Stage2Turbine for further expansion.
- In model MixingMassFlows the liquid phase flow from the separator is mixed with the flow from the pump.

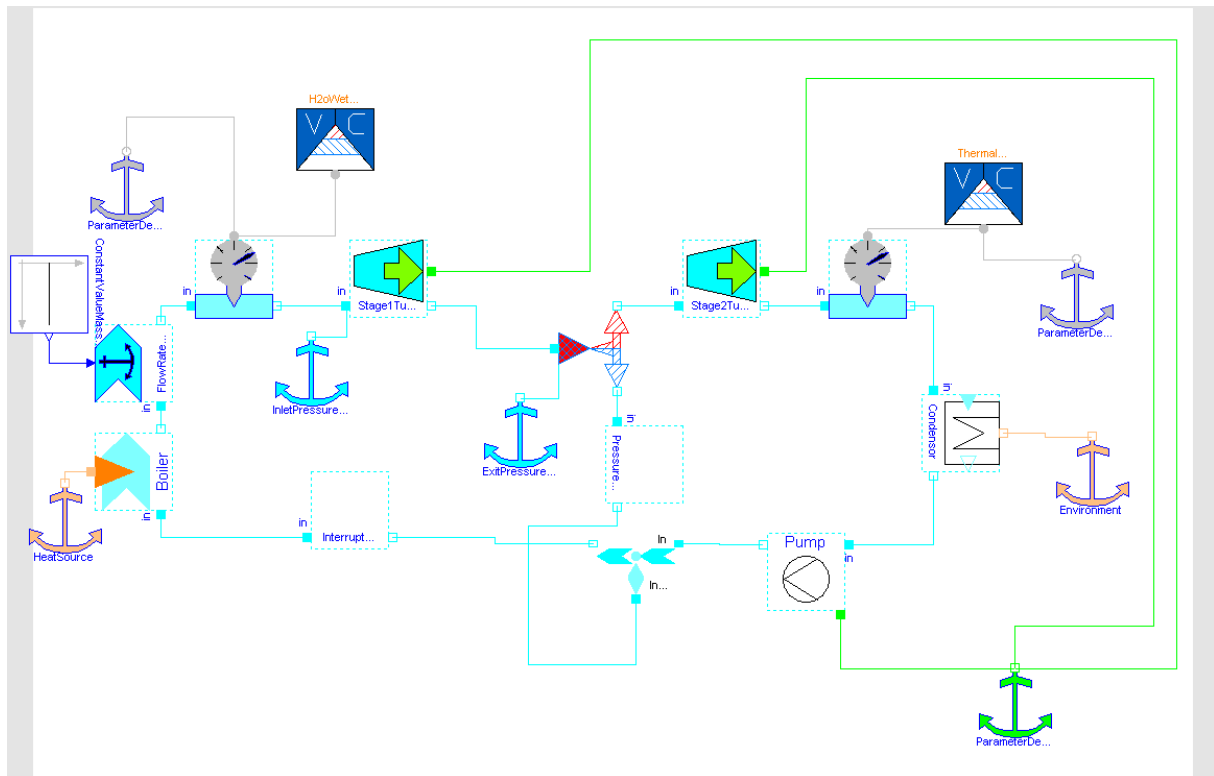


Fig. Ex3_2: Diagram layer of model Ex3c

Ex3d: Basic steam cycle with two turbine stages, extraction of liquid and calculation of thermal efficiency of steam cycle

In Ex3d, the thermal efficiency of the process is calculated with varying steam pressure at the inlet of the first turbine. Fig. Ex3_3 shows the results for the thermal efficiency for the single stage steam cycle from Ex3b and the two stage steam cycle from Ex3d.

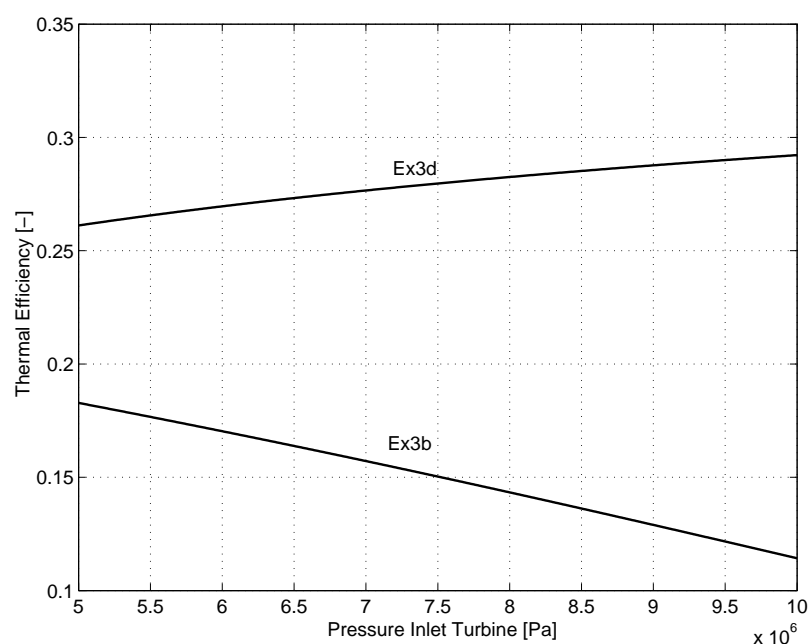


Fig. Ex3_2: Thermal efficiency ThermalEfficiency.eta for single-stage steam cycle (Ex3b) and two-stage steam cycle (Ex3d), constant steam quality at outlet of turbine.

Ex4: Compressed air energy storage

One option for large scale storage of electric energy is the use of compressed air. If more electric power is produced than needed (e.g. from wind power) air is compressed and stored. Since large storage volumes are needed, caverns are often used. After compression, the air is cooled down before flowing into the storage volume. During the discharge cycle, compressed air is extracted from the storage volume. A burner is used to add heat to the air before it is expanded in a turbine driving a generator.

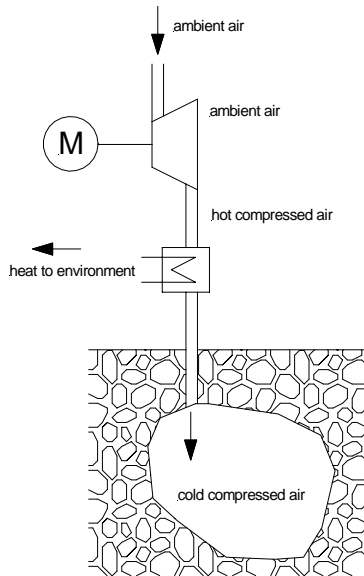


Fig. Ex4_1: Charging process for compressed air storage system: compressed air is filled into cavern.

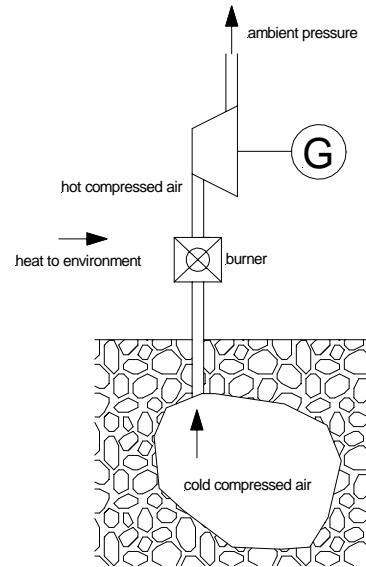


Fig. Ex4_2: Discharging process for compressed air storage system: compressed air is extracted from cavern.

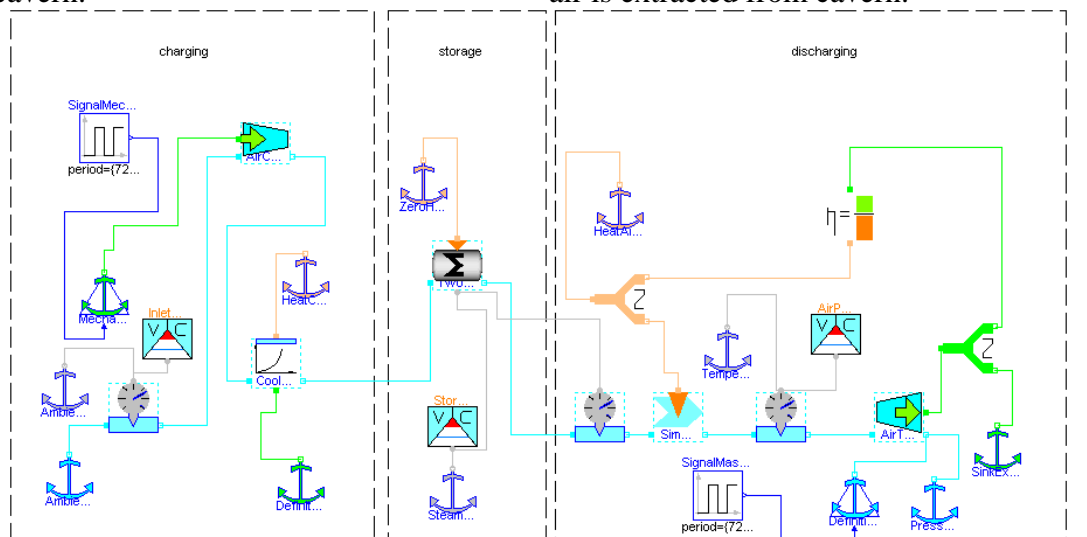


Fig. Ex4_3: Modelica model compressed air storage system; simulation time 7200s includes a charging and discharging process.