Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Modelling of air flows in automotive engines using Modelica

Alexander Stankovic

Reg nr: LiTH-ISY-EX-3065

August 30th, 2000

Modelling of air flows in automotive engines using Modelica

Master´s Thesis

Division of Vehicular Systems
Department of Electrical Engineering
Linköping University

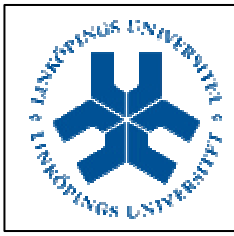
Alexander Stankovic

Reg nr: LiTH-ISY-EX-3065

Supervisor: Dr. Mattias Nyberg DaimlerChrysler AG

Examiner: Prof. Lars Nielsen University of Linköping, LiTH

Linköping, August 30‘th, 2000



ISBN

ISRN

Serietitel och serienummer ISSN
Title of series, numbering

LiTH-ISY-EX-3065

ABSTRACT

In the industry the need of simulation of complex systems, composed of subsystems from various domains, is increasing. A new standardized modelling language, called Modelica, has been developed for the purpose of efficient simulation. The language is object-oriented, non-causal, and multi-domain capable.

Automotive engines are mainly composed of thermodynamic subsystems. In this master's thesis three different modelling principles are used to analyse the performance of Modelica in automotive engine applications. The modelling principle is determined by the connector applied, i.e. the interaction between the components. Connectors composed of various combinations of variables describing pressure, temperature, mass flow and energy flow have been applied. During the analysis, the performance of a control volume, restrictor, and sources has been studied. The purpose is to decide a standard set of connector variables for the description of engine components.

None of the tried modelling principles has proved to be completely satisfactory. The problems have been: two control volumes connected to each other, bi-directional gas flow, and complete support of independent design of components. A conclusion is that these problems can be solved by a simulation engine that supports conditional equations without else-clauses, and requires only that the *relevant* number of equations and variables is equal.

SAMMANFATTNING

Inom industrin ökar behovet av simulering av komplexa system, bestående av subsystem från olika domäner. Ett nytt standardiserat modellerings språk, Modelica, är utvecklat med avsikt för effektiv simulering. Språket är objekt orienterat, icke kausalt och multidomän kapabelt.

Bil motorer består huvudsakligen av termodynamiska subsystem. I detta examens arbete har tre olika modelleringsprinciper använts för att analysera prestandan av Modelica i motor applikationer. Modelleringsprincipen bestäms av den applicerade connectorn, d.v.s. samverkan mellan komponenterna. Connectorer, bestående av diverse kombinationer av storheter som beskriver tryck, temperatur, massflöde och energiflöde har applicerats. I analysen har prestandan av en kontrol volym, restrictor och källor studerats. Syftet är att bestämma en standard uppsättning av connector-storheter för beskrivning av motor komponenter.

Ingen av de provade modelleringsprinciperna har visat sig fungera helt tillfredsställande. Problemen har varit: två ihopkopplade volymer, gasflöde i två riktningar, samt fullständigt stöd av oberoende design av komponenter. En slutsatts är att dessa problem kan lösas av en simuleringsmotor som stödjer villkorsekvationer utan else-satser, samt kräver att enbart *relevanta* antalet ekvationer och variabler är lika.

ACKNOWLEDGEMENTS

This master‘s thesis has been carried out at the research department FT2/EA, DaimlerChrysler AG, during the spring and summer of 2000. The work is a result of a co-operation between DaimlerChrysler AG, MathCore AB and the University of Linköping, division of Vehicular Systems.

Many people have been involved, not just the ones mentioned here. I would especially like to thank the following people:

My supervisor at DaimlerChrysler AG, Dr. Mattias Nyberg, for his support and guidance during my work. Many results and ideas have originated from questions and discussions with him.

My examiner at the University of Linköping, Prof. Lars Nielsen, and his staff for their support and help.

The staff at MathCore AB for always providing answers to my great number of questions.

My family and friends for their encouragement and support at all times.

Esslingen am Neckar, August 2000

Alexander Stankovic

NOMENCLATURE

Physical variables

t : time [s]

T : temperature [K]

T_a : temperature at gate a [K]

T_b : temperature at gate b [K]

$T_{ambient}$: temperature [K]

p : pressure [Pa]

$\dot{p}$: change of pressure during a short time interval $[\frac{\text{Pa}}{\text{s}}]$

p_a : pressure at gate a [Pa]

p_b : pressure at gate b [Pa]

Δp : pressure difference [Pa]

m : mass [kg]

$\dot{m}$: change of mass during a short time intervall $[\frac{\text{kg}}{\text{s}}]$

W : mass flow $[\frac{\text{kg}}{\text{s}}]$

W_a : mass flow at gate a $[\frac{\text{kg}}{\text{s}}]$

W_b : mass flow at gate b $[\frac{\text{kg}}{\text{s}}]$

$\dot{H}$: energy flow [W]

$\dot{H}_a$: energy flow at gate a [W]

$\dot{H}_b$: energy flow at gate b [W]

Constants

K : restrictor constant $[\frac{1}{\mathrm{m}^4}]$

V : volume $[\mathrm{m}^3]$

R : gas constant $[\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

Values: $R_{air} = 287\ [\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

$R_{exhaust} = 285\ [\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

c_p : specific heat capacity at constant pressure $[\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

Values: $c_{p,air} = 1005\ [\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

$c_{p,exhaust} = 1150\ [\frac{\mathrm{J}}{\mathrm{kg\,K}}]$

CONTENTS

Appendices

1 INTRODUCTION

1.1 Background

The use of computer simulation in the industry is rapidly increasing. Simulation is typically used to optimise products and to reduce product development cost and time. In the past it was considered sufficient to simulate subsystems separately. The current trend is to simulate increasingly complex physical systems composed of subsystems from different domains such as mechanic, electric, thermodynamic, and control system components. Automotive engines models tend to be highly complex systems, and there is an interoperability problem amongst the variety of modelling and simulation environments available for complex systems. The main cause of this problem is the absence of a standardised model representation. The goal of the Modelica Design Group [7] is to design a standardised modelling language, called Modelica, for the purpose of efficient simulation.

In this thesis a performance analysis of Modelica, applied in automotive engines, is done. The thesis has resulted from a co-operation between DaimlerChrysler AG [1], MathCore AB [5] and the University of Linköping, division of Vehicular Systems [10].

1.2 Basic product description

1.2.1 Modelica

The language called Modelica for hierarchical physical modelling is developed through an international effort, the Modelica Design Group. Modelica is an object oriented language for modelling physical systems. The language unifies and generalises previous object oriented modelling languages and is intended to become a de facto standard. Modelica offers three important features:

- Non-causal modelling based on differential and algebraic equations. Equations can be written explicitly, like $x = y$, or can be inherited from other classes.

- Multi-domain modelling, i.e. it is possible to combine electrical, mechanical, thermodynamic etc. model components within the same application model.

- A general type system that unifies object orientation, multiple inheritance and templates within a single class construct.

The Modelica language specification [8] and tutorial [9] on Modelica can be found at the Modelica Design Group website [7].

The intention of the Modelica Design Group is to create standard libraries of ready-to-use components. The library of thermodynamic components, ThermoFlow, is still under development and not yet ready to be used. The beta version of the library shows that the description of the thermodynamic components is general and very complex. The high level of detailed description of components is perhaps difficult to understand if one does not posses advanced knowledge in thermodynamic.

1.2.2 MathModelica

MathModelica is both a language and an environment developed by MathCore AB. The MathModelica language is an extension of the of the Modelica language targeted for work within the Mathematica environment, developed by Wolfram Research [11]. A specific feature of MathModelica is that models are normally not written as free formatted text. Instead Mathematica expression are used. These can be written in a tree like prefix form or entered using standard mathematical notation. To this date the released and analysed version of MathModelica is 0.8.7.

Note that the MathModelica language has the same abstract syntax and the same semantics as Modelica, but different concrete syntax. This essentially means that the same language constructions are written differently.

1.2.3 Dymola

The most developed simulation engine for Modelica is based on the Dymola system by Dynasim [2]. When using MathModelica and a simulation command is given, the expressions in Mathematica are translated to pure Modelica code by MathModelica. The Modelica code is then transferred to Dymola which generates a C-code that is compiled and the actual simulation occurs. The results of the simulation are sent to Mathematica for visualisation.

An analysis of Dymola can unfortunately not be done due to restrictions in the software license agreement.

1.3 Purpose of the report

The main purpose of this report is to gather the information obtained from the performance analysis of Modelica. The purpose of the information is to make plans and decisions on a future standard description of engine components. To be able to make a decision, both theoretical, and practical aspects of Modelica have to be considered.

1.4 Readers guide

The report is intended for both readers active in the industry and university students, introduced to Modelica, in modelling courses. Theoretical facts of Modelica and engines in general are intentionally not written in this report, instead some very good references, [4], [5], and [7], are presented in the Bibliography. Efforts are made to find the thesis interesting for both categories of readers.

This section provides the outline of the report chapter by chapter.

Chapter 1, Introduction, presents the background of this work and the products used.

Chapter 2, Analysis of Different Modelling Principles, describes the different modelling principles, and their performance used during this work.

Chapter 3, Conclusions, presents a summary of the results obtained from the analysis. A suggestion of improvements and future work is also given in this chapter.

Appendices A-I, presents various cases of simulations and simulation results. The appendices are formulated as the corresponding notebook in Mathematica.

Appendix J, presents the proof of relation (2.4) in section 2.3.1.

The *Bibliography* is found in the end of this report.

2 ANALYSIS OF DIFFERENT MODELLING PRINCIPLES

This chapter describes the different modelling principles used when analysing MathModelica. The modelling principle is decided by the connector applied, i.e. the interaction between the components.

During the analysis, the performance of one dynamic component, i.e. a control volume, and one static component, i.e. a restrictor, and sources has been studied. The control volume is dynamic in the sense that it contains states, i.e. mass and pressure. The components are limited to only two gates. To distinguish between the two gates they are denoted as *a* and *b* respectively.

The reason for studying a restrictor is that many engine components, e.g. filters, intercoolers, throttles etc., are basically modelled as a restrictor.

The equations describing the component, and the MathModelica code, will vary depending on what connector is applied. See Appendix A through H.

The purpose of the analysis of the modelling principles is to decide a standard set of variables for the description of engine components using Modelica.

Three different modelling principles,

- pWT-connector
- pWH-connector
- pWHT-connector

have been used with their own possibilities and limitations. The connectors are described in detail in the following sections of this chapter.

The choice of connector has determined how well the components are performing and what limitations the models must have. None of the connectors has proved to be completely satisfactory.

2.1 Basic Assumptions

To simplify the modelling of the components used, some restrictions and assumptions are made:

- The components are restricted to handle gas flows composed of only one gas.

- The gas is considered to be ideal. For example air, both dry and humid, is considered to be an
  ideal gas mixture [3].

- Pressure changes will only occur over the restrictor.

- The gas constant, R, and the specific heat capacity of the gas, c_p, are assumed to be constant.

2.2 Connectors

To promote compatibility and exchange of models between users, one can define a standard set of connectors for an application domain which is used throughout all components. Engine models are normally described by general thermodynamic variables.

In this analysis, the connectors have consisted of various combinations of the following variables:

- Pressure - p
- Temperature - T
- Mass flow - W
- Energy flow - $\dot{H}$

Combinations of the four variables above are not enough to describe all engine components, e.g. for a turbine and a compressor the connector must also contain the turbocharger speed. Consequently the models must have the possibility to be extended with suitable connectors.

2.3 pWT-connector

In engine applications, control engineers usually work with and measure pressure, mass flow and temperature. Therefore the choice of using pressure, mass flow and temperature as connector variables seems to be a logic and intuitive approach for the first modelling principle. The MathModelica code of the pWT-connector is

```
Connector @FlowCut,
 Pressure p;
 Flow MassFlow W ;
 Temperature T;
D
```

The principle of using the pWT-connector will lead to the most straight forward description of the components. Automotive literature usually describes the engine models in terms of pressure, mass flow and temperature making it very easy to design models of various components based on the description in the literature [4] and [5].

2.3.1 Control volume

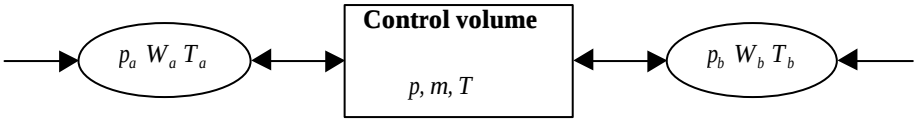


Figure 1: Control volume with pWT-connector borders.

A basic assumption, regarding the description of the control volume, is that the incoming gas is momentarily distributed and mixed with the already present gas. In the control volume, the pressure *p,* temperature *T,* and gas composition, are assumed homogeneous. The equations describing the dynamics of the gas are shown below.

The ideal gas law is assumed to apply to the gas in the control volume. Assuming a control volume with a constant volume V, and the mass of the gas is m the ideal gas law is given by

$$pV = mRT \tag{2.1}$$

where R is the gas constant.

The change of mass inside the control volume is obtained from the mass flows at gate *a* and *b*, i.e.

$$\dot{m} = W_a + W_b \tag{2.2}$$

A consequence of the basic assumptions, in section 2.1, is that the pressure is assumed to be uniform through the whole control volume giving

$$p = p_a = p_b \tag{2.3}$$

Assuming that the gas flow is directed into the control volume at gate *a* and out from the control volume at gate *b* the pressure variation is given by

$$\dot{p} = \frac{R c_p}{V (c_p - R)} (W_a T_a + W_b T) \tag{2.4}$$

where T is derived from the ideal gas law (2.1). The proof for relation (2.4) is given in Appendix I.

Relation (2.4) is limited to handle gas flows in one direction. If we would have gas flow into the control volume at gate *b* and out from the control volume at gate *a* equation (2.4) would be

$$\dot{p} = \frac{R c_p}{V (c_p - R)} (W_a T + W_b T_b) \tag{2.5}$$

Gas flow into the control volume and out from the control volume at both gates yield

$$\dot{p} = \frac{R\, c_p}{V\,(c_p - R)}(W_a\, T_a + W_b\, T_b) \qquad (2.6)$$

$$\dot{p} = \frac{R\, c_p}{V\,(c_p - R)}(W_a\, T + W_b\, T) \qquad (2.7)$$

respectively.

The relations (2.1)-(2.4) lead to the following MathModelica code of the control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E;  "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  T Š b.T;
  m' Š a.W + b.W;
  p' == R*cp / V*Hcp - RL *Ha.W*a.T + b.W*TL;
 E
E
```

Note that MathModelica uses dot-notation, e.g. the temperature at gate *a*, T_a, is written **a.T** in the code.

The first limitation in the model of the control volume above, when using the pWT-connector, is that gas flow is only supported in one direction. We must have gas flow into the control volume at one gate and outgoing gas flow at the other gate.

The problem is the temperature of the gas flow. When the gas flow is directed into the control volume at either one or both of the gates the temperature of the gas flow is determined by some of the other components or sources. When the gas flow is outgoing on either one or both of the gates the temperature is derived from the ideal gas law (2.1).

A natural approach to solve the temperature problem would be to introduce conditional equations like

```
If@a.W < 0, a.T Š TD;
If@b.W < 0, b.T Š TD;
```

to the existing MathModelica code of the control volume above. However this will not work. Every **If** clause must have an **else** clause and each branch must have the same number of equations. If this condition is violated the single assignment rule would not hold, because the number of equations, for the control volume, may change during simulation although the number of unknowns remains the same [8]. Note that the total number of equations are always the same. The only difference is from which component the equations origin. Modifying the conditional equations above by introducing a **else** clause

```
a.T Š If@a.W > 0, a.T, TD;
b.T Š If@b.W > 0, b.T, TD;
```

which branches have the same number of equations will not help either. The algebraic solver of MathModelica will interpret the trivial equations $T_a = T_a$, when $W_a > O$, and $T_b = T_b$, when $W_b > 0$, as two additional equations with no additional variables which will lead to a conflict. See Appendix A for a more complete illustration.

When the conditional equations above are used, the resulting total equation system obtained from the chain of components contains equal number of equations and unknowns. Therefore the equation system is algebraically solvable and a possible solution for the problems with the conditional equations would be to use other algorithms, in Dymola, for equation solving.

The conclusion is that the design of the control volume supports only one predetermined gas flow direction at a time, when the pWT-connector is applied.

2.3.2 Restrictor

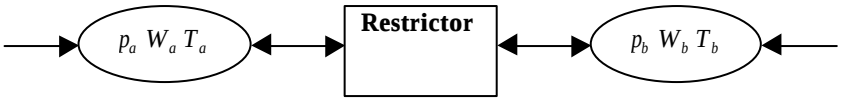


Figure 2: Restrictor with pWT-connector borders.

The restrictor is modelled using incompressible gas flow for a fixed restriction which gives the following static characteristics of the restrictor

$$W_a + W_b = 0 \tag{2.8}$$

Equation (2.8) tells us that all gas entering the restrictor will also exit the restrictor.

The temperature change across the restrictor is likely to be small, and so has been neglected leading to

$$T_a = T_b \tag{2.9}$$

Keeping in mind that the models are limited to have gas flows in one direction, and assuming that the gas flow is directed into the restrictor at gate *a* and out from gate *b*, the description of the pressure drop is

$$p_a - p_b = \frac{K W_a^2 R T_a}{p_a} \tag{2.10}$$

where K is a constant.

If we would have gas flow into the restrictor at gate *b* and out from the restrictor at gate *a*, equation (2.10) would be

$$p_b - p_a = \frac{K W_b^2 R T_b}{p_b} \tag{2.11}$$

If we try to use the expressions (2.10) and (2.11) written as they are, in MathModelica, we will run into problems with singularities because the denominator p_b will cause division by zero. See Appendix B for simulation results. When the numerical solver is trying to find a solution it will start the algorithm in $p_a = 0$ and $p_b = 0$ respectively causing model error. The start values, pa=0 and pb=0, of the algorithm are fixed by Dymola. A good function in Dymola would be a possibility to change the start values of the algorithms. The singularity problems above could then easily be avoided.

Rewriting equation (2.10) (correspondingly for (2.11) if used) into

$$p_a^2 - p_a p_b = K W_a^2 R T_a \qquad (2.12)$$

will not help either.

For example when simulating the chain of components in Figure 5 ,and the description (2.12) of the restrictor is used, the numerical solver will choose the negative root of equation (2.12). Note that the negative root is not mathematically incorrect but physically, and the simulation program does not produce a warning for multiple roots. The simulation is completed with the physically incorrect pressure. The resulting negative pressure at gate *a*, p_a, is shown in Figure 3. The corresponding pressure difference between the pressures at gate *a* and *b*, p_a-p_b, is shown in Figure 4. A negative pressure difference clearly shows that the restrictor does not function correctly. The function of the restrictor is that a pressure drop, i.e. $p_b < p_a$, shall occur when the direction of the gas flow is from gate *a* to gate *b*. See Appendix C for a complete illustration of the results of the simulation.

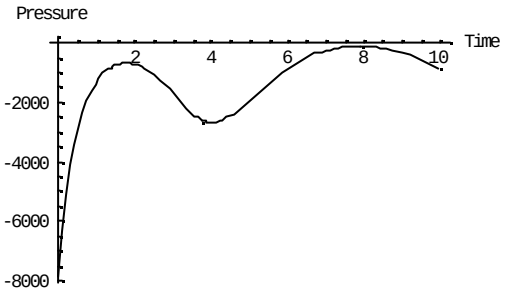


Figure 3: Pressure at gate *a*.

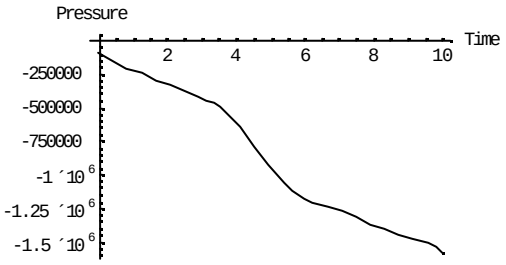


Figure 4: Pressure difference p_a-p_b.

One way to deal with the problem with negative roots would be to introduce conditions, e.g. minimum value, in the type definition of the pressure as shown below.

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
Type@MassFlow, Real@8Unit Š "kg•s"<DD;
Type@Pressure, Real@@Min Š 0DDD;H*8Unit Š"Pa"<*L
```

Unfortunately the present version of the MathModelica compiler, Dymola, does not support the condition when solving the equation numerically.

The only method to solve the problem of multiple roots is to force the numerical solver to choose the positive root by writing

$$p_a = \frac{p_b}{2} + \sqrt{\frac{p_b^2}{4} + K W_a^2 \, R \, T_a} \qquad (2.13)$$

However this method, by forcing the numerical solver to choose roots has it's limits. In this case it was easy to find the correct root manually. In other cases maybe it is not possible to find the exact root, and numerical algorithms has to be applied.

The resulting MathModelica code of the restrictor is

```
ModelARestrictor,
 FlowCut 8a, b<;
 PressureDp;
 Parameter Real K;
 Constant Real R Š 287;
 EquationA
  a.T Š b.T;
  a.W+ b.W Š 0;
  Dp Š a.p - b.p;
  a.p Š b.p/2 + $ Hb.pL^2/4 + K*Ha.WL^2*R*a.T;
 E
E
```

The introduction of the variable

$$\Delta p = p_a - p_b \qquad (2.14)$$

is for plotting simplification only.

2.3.3 Input and output sources

To be able to simulate and evaluate the restrictor and the control volume, some kind of components or sources that are "producing" mass flow are needed. Such components could be considered as fans blowing or sucking gas. In principle the inlet, and the exhaust, of a cylinder are modelled as fans. Another possibility would be to use so called reservoirs. This method has not been tried because the emphasis of this work is to analyse the performance of the engine components and not the sources.

Depending on how the chain of components is built up, the function of the sources will vary. In this analysis the configuration of the chain of components, shown in Figure 5, is to have one so called input source connected to a restrictor which in turn is connected to a control volume, ending the chain with a so called output source.

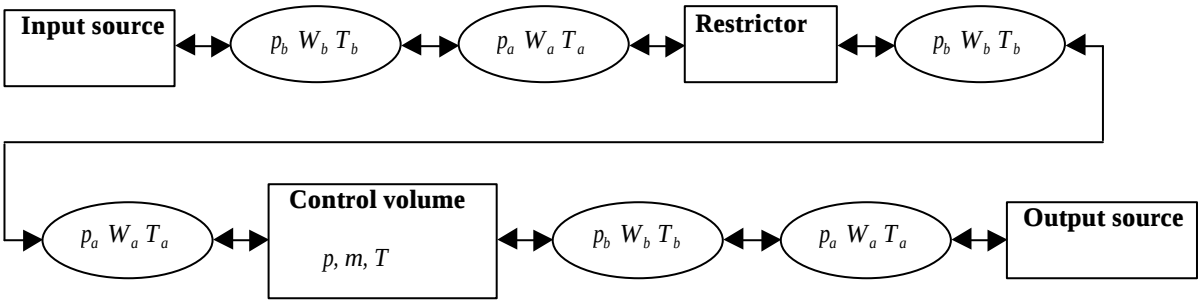


Figure 5: Chain of components with pWT-connector borders.

With this configuration of components, and keeping in mind that the pWT-connector only supports flow in one direction, the input and output source will have the following description.

Due to the chosen direction of the gas flow, the input source will always have a negative mass flow

$$W_b = -2 + \sin(t) \qquad (2.15)$$

That is the gas flow direction is out from the source and into the connected component. The input source will be blowing gas into the connected component and consequently the initial temperature of the gas has to be decided in the input source. The temperature is here chosen to

$$T_b = 300 \qquad (2.16)$$

There does not exist a gate *a* in the input source because this is the first component in the chain of components. The MathModelica code for the input source is as follows.

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.T Š 300;
  b.W Š -2+Sin@TimeD;
 D
D
```

The end of the chain of components will consist of the output source sucking gas from the control volume. Hence only a mass flow

$$W_a = \cos^2(t) \qquad (2.17)$$

that is always positive or zero, exists at gate *a*. The lack of a temperature function in the output source is explained by the function of the control volume described in section 2.3.1. In the control volume, the temperature of the gas at gate *b* is derived from the ideal gas law (2.1) and can therefore not be fixed by a function or value in the output source. Below is the MathModelica code of the output source.

```
ModelAOutputSource,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL^2;
 E
E
```

The mathematical functions of the temperature and mass flows in the input and output source can be chosen arbitrary as long as the gas flow from the input source is negative or zero and the flow to the output source is positive or zero. The sine and cosine functions are chosen because they will produce flows with varying intensity. A fixed temperature (2.16) together with the chosen mass flow functions, (2.15) and (2.17), will lead to easily verified simulation results.

Note that in this case the direction of the gas flow is fixed. Later on it will be shown how direction changes are supported, and then the input- and, the output source will both blow and suck gas. The names of the sources, input and output, are chosen because of the location in the chain of components, and not because of the function.

2.3.4 Simulation

Simulation of the configuration of the components, in Figure 5, yields the following plots of various variables. The chosen initial values, of the mass and the pressure of the control volume, are 2 kg. and 101300 Pa. respectively.

The pressure difference, Dp, of the restrictor is illustrated in Figure 6.

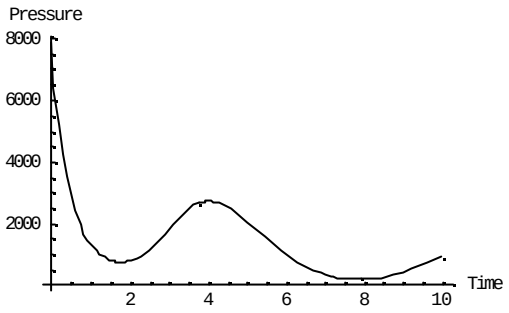


Figure 6: Pressure difference Dp.

The result in Figure 6 shows that the pressure difference is always positive which is reasonable and correct.

The pressure and the temperature of the gas inside the control volume are shown in Figure 7 and 8 respectively. The pressure and temperature inside the control volume are increasing due to the chosen input- and output source. The input source is blowing more gas into the system than the output source is sucking from the system. A more complete illustration of the simulation result is in Appendix D.

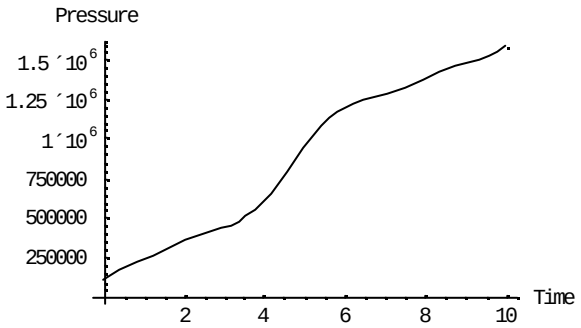


Figure 7: Pressure of the gas inside the control volume.

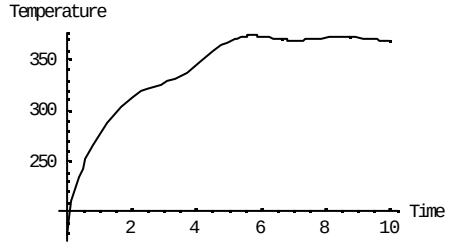


Figure 8: Temperature of the gas inside the control volume .

Another chain of components is also simulated. The chain of components consists of an input-, output source and two volumes connected to each other according to Figure 9. The initial values, of the masses and the pressures of both control volumes, are chosen to 2 kg. and 101300 Pa. respectively. The control volumes are both identical with a volume of 1.5 m^3.

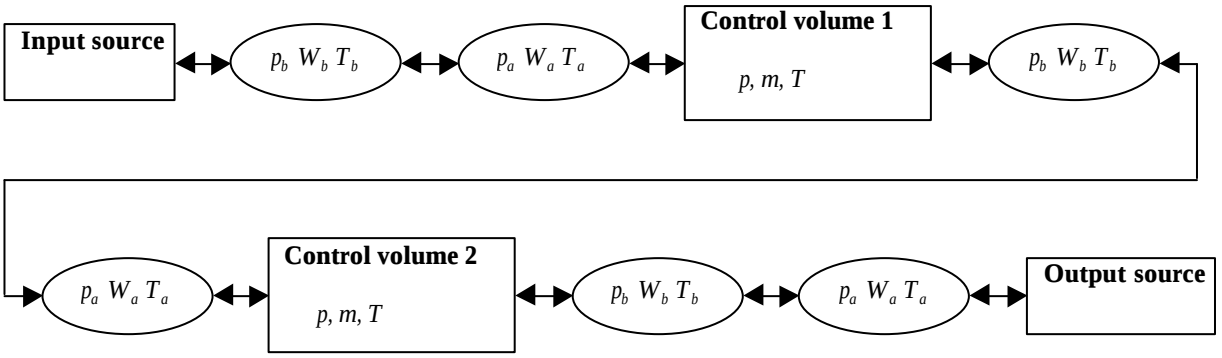


Figure 9: Chain of components with two control volumes.

To simulate two control volumes is interesting in the sense that there is no specified gas flow between the two control volumes and to investigate how two volumes are interpreted. Dymola managed to perform the simulation and the resulting mass flow at gate *b* of control volume 1 is shown in Figure 10.

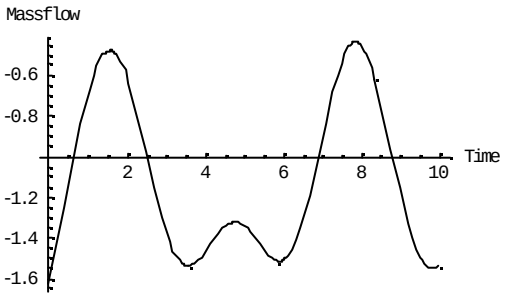


Figure 10: Massflow at gate *b* of control volume 1.

The two control volumes will have a different temperature of the gas, over 40 degrees difference, inside the respective control volume. The temperature plot is shown in Figure 11. This shows that the two connected control volumes are not interpreted as a single, large control volume. The complete simulation result can be studied in Appendix E.

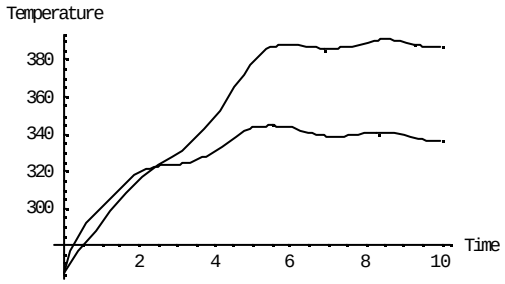


Figure 11: Temperature of the gas inside control volume 1 (lower curve) and control volume 2 (upper curve).

2.4 pWH-connector

The first principle supported a design of components that performs well if the gas flow was limited to only one direction. A natural step in the improvement and evolution of the components is to make them capable of handling changes in gas flow direction. As described in section 2.3.1, the variable that limited the direction changes of the gas flow in the principle of using pWT-connector is the temperature.

This section will describe how it is possible to avoid using the temperature in the connector by introducing a new flow variable, energy flow, in the connector. The MathModelica code of the connector is

```
Connector AFlowCut,
  Pressure p;
  Flow MassFlow W;
  Flow EnergyFlow H;
E
```

It is possible to use the energy flow instead of the temperature because neither the control volume nor the restrictor needs the temperature explicitly for the function of the components to be described. More on this in section 2.4.1 and 2.4.2.

However if the temperature of the gas flow necessarily wants to be studied it can be easily calculated from the following relation [3]

$$T = \frac{\dot{H}}{W c_p} \tag{2.18}$$

The reason for not introducing the energy flow from the beginning is that almost all automotive literature describes engines in terms of temperature instead of energy flow. Therefore introducing energy flow leads to a slightly abstract picture of the engine components in the eyes of control engineers. It is not even possible to measure the energy flow, making it more difficult to verify the performance of the components with measured data from a real engine.

2.4.1 Control volume

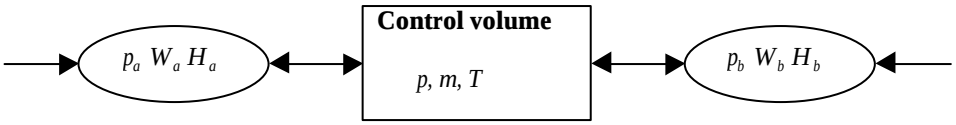


Figure 12: Control volume with pWH-connector borders.

The temperatures at gate *a* and *b*, expressed in terms of mass flow and energy flow yield

$$T_a = \frac{\dot{H}_a}{W_a\ c_p} \tag{2.19}$$

$$T_b = \frac{\dot{H}_b}{W_b\ c_p} \tag{2.20}$$

Inserting equations (2.19) and (2.20) into equation (2.6) gives the new expression for the pressure variation inside the control volume

$$\dot{p} = \frac{R}{V\ (c_p - R)}(\dot{H}_a + \dot{H}_b) \tag{2.21}$$

Note that direction specification of the energy flow inside the components is not needed because the keyword **Flow** in front of a variable in a connector automatically gives the direction. Consequently equation (2.21) handles gas flow in all directions which is desirable.

The characteristics, (2.1)-(2.3), are inherited by the new description of the control volume.

The ideal gas law (2.1) is only used to study the temperature inside the control volume and has nothing to do with the connector variables.

The resulting MathModelica code of the control volume is shown below.

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m^3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real c_p Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  m' Š a.W + b.W;
  p' == R/(V*Hc_p - RL) * Ia.H + b.HM;
 E
E
```

2.4.2 Restrictor

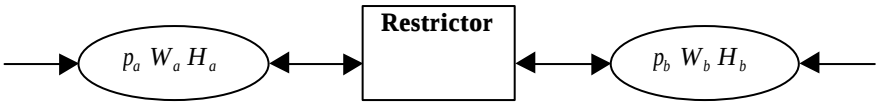


Figure 13: Restrictor with pWH-connector borders.

Based on the restrictor description in section 2.3.2, the introduction of energy flow is not as straightforward as replacing the temperature at gate *a* and *b* with the energy flow expression (2.18).

Assuming that no energy build-up or work occurs inside the restrictor, the static characteristics of the restrictor implies

$$\dot{H}_a + \dot{H}_b = 0 \qquad (2.22)$$

Adapting the restrictor to multi-directional gas flow one has to consider that a pressure drop occurs regardless of the direction of the gas flow. That is, if the flow is entering the restrictor at gate *a*, the pressure at gate *b* is lower than the pressure at gate *a*. If the flow suddenly changes direction and enters the restrictor at gate *b*, the pressure at gate *a* has to be lower than the pressure at gate *b*.

Keeping in mind the problem with multiple root equation described in section 2.3.2 and assuming a gas flow direction from gate *a* to gate *b*, leads to the following description of the pressure drop in the restrictor

$$p_a = \frac{p_b}{2} + \sqrt{\frac{p_b^2}{4} + \frac{K W_a R \dot{H}_a}{c_p}} \qquad (2.23)$$

If the gas flow direction is from gate *b* to gate *a* the pressure drop will be

$$p_b = \frac{p_a}{2} + \sqrt{\frac{p_a^2}{4} + \frac{K W_b R \dot{H}_b}{c_p}} \qquad (2.24)$$

When implementing equations (2.23) and (2.24) into MathModelica one has to specify when each equation is used. This is easily done by a simple If-condition on the massflow at one of the gates.

The resulting MathModelica code of the restrictor with bi-directional gas flow capability using pWH-connector can be studied below.

```
ModelARestrictor,
 FlowCut8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  a.H+ b.H Š 0;
  a.W+ b.W Š 0;
  Dp Š a.p- b.p;
  IfAa.W > 0, a.p Š b.p/2 + &(Hb.pL^2/4 + K*a.W*R*a.H/cp),
    b.p Š a.p/2 + &(Ha.pL^2/4 + K*b.W*R*b.H/cp) E;
 E
E
```

Note that it is not necessary to introduce an If-condition on the massflow at gate *b*, if there already is a condition at gate *a*. Through equation (2.8), conditions are implied from the mass flow at gate *b* when we have conditions on the mass flow at gate *a*.

2.4.3 Input and output sources

One limitation when using pWH-connector is that it is not possible to construct any input or output sources, that are *both* blowing and sucking gas. To be able to simulate the models we must have a value of the energy, i.e. an energy source. Since the energy flow is unmeasurable in engines we must state the temperature somewhere in the chain of components according to equation (2.18) to receive a value of the energy.

The conclusion is that the principle of using pWH-connector, where temperature is intentionally not used, makes it impossible to construct any sources. Without sources it is not possible to simulate the components in a correct way.

2.5 pWHT-connector

In the first principle we managed to design components that perform well but support flow in one direction only. The second principle managed to solve the problem of bi-directional gas flow, but we could not design any sources to simulate our components with bi-directional gas flow. Combining the two previous principles should result in a connector capable of handling changes in gas flow direction, and design of sources for simulation. The MathModelica code of the connector is

```
Connector AFlowCut,
 Pressure p;
 Flow MassFlow W ;
 Flow EnergyFlow H;
 Temperature T;
E
```

Due to the redundant information, energy flow and temperature, exchanged between the different components, the energy-temperature relation (2.18) must be introduced in the code for the various components used.

2.5.1 Control volume

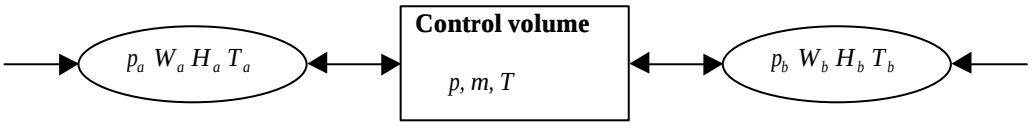


Figure 14: Control volume with pWHT-connector borders.

Based on the model of the control volume in section 2.4.1, and the temperature of the gas flow, the reasoning in section 2.3.1 requires the following conditions. When the gas flow at gate *a* is directed into the control volume, the temperature of the gas is determined by the preceding component. That is the temperature is T_a, otherwise the temperature of the gas is T, i.e. derived from the ideal gas law (2.1) inside the control volume. Further, the temperature of the gas at gate *b*, when the gas flow is directed into the control volume, is T_b otherwise the temperature is T. These conditions are easily implemented in MathModelica by conditional equations as shown below. Note that the energy flow and mass flow always have the same direction.

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E;  "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  m' Š a.W+ b.W;
  a.T== IfAa.W > 0, a.H / a.W * cp , TE;
  b.T == IfAb.W > 0, b.H / b.W * cp , TE;
  p' == R / V*Hcp - RL * Ia.H+ b.HM;
 E
E
```

One interesting observation is that, if the mass flow is directed into the control volume, i.e. positive. The conditional equation

```
a.T == IfAa.W > 0, a.H / a.W * cp , TE;
```

tells that the temperature at gate *a*, is

$$T_a = \frac{\dot{H}_a}{W_a \, c_p} \tag{2.25}$$

That is, equation (2.25) is actually equal to T_a but inserting this simplification in the conditional equation above does not work as described in section 2.3.1 and shown in Appendix A. The same is valid for the conditional equation describing the temperature and flow direction at gate *b*.

2.5.2 Restrictor

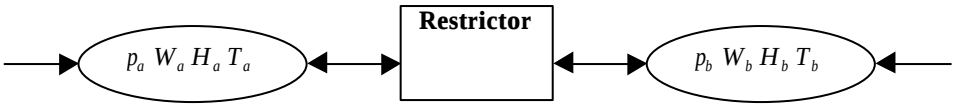


Figure 15: Restrictor with pWHT-connector borders

Introducing temperature to the model of the restrictor in section 2.4.2 is very easy and straight forward since it is assumed that no change of temperature occurs. Hence only the relation

$$T_a = T_b \tag{2.26}$$

has to be added. The resulting MathModelica code is shown below.

```
ModelARestrictor,
 FlowCut 8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  a.H+ b.H Š 0;
  a.W+ b.W Š 0;
  Dp Š a.p- b.p;
  IfAa.W > 0, a.p Š b.p/2 + & Hb.pL^2/4 + K*a.W*R*a.H/cp ,
    b.p Š a.p/2 + & Ha.pL^2/4 + K*b.W*R*b.H/cp E;
  a.T Š b.T;
 E
E
```

2.5.3 Input and output sources

The whole idea of introducing the redundant information, temperature, was to be able to design some kind of energy sources for simulation purposes.

The fundamental relation is still equation (2.18) where the energy flow will be determined by the mass flow and the temperature of the gas flow. Compared to section 2.3.3, the mathematical function of the mass flow can be chosen even more arbitrary. With the pWHT-connector there is no restriction on the sign, i.e. direction, of the gas flow. As in section 2.3.3, sine and cosine function are chosen to form the basis of the mathematical description of the various flows. Note that the input- and output source will have the same function, but different location in the chain of components.

A sine function with a small frequency- and phase difference is here chosen to describe the input source

$$W_b = \sin(2t + 0.6) \tag{2.27}$$

Because the input source will sometimes blow gas (during the negative period) into the connected component and, sometimes suck gas (during the positive period) from the connected component, the description of the energy flow will vary. When the input source is blowing gas into the connected component, the temperature of the gas flow is determined by the source itself, yielding the following expression for the energy flow:

$$\dot{H}_b = \frac{W_b}{c_p T_{ambient}} \tag{2.28}$$

where

$$T_{ambient} = 300 \tag{2.29}$$

The temperature $T_{ambient}$ must not necessarily be fixed, but here chosen so for easily verified simulation results.

During the positive period of the mass flow, the temperature of the gas flow is determined by the connected component and the following expression for the energy flow is used

$$\dot{H}_b = W_b c_p T_b \tag{2.30}$$

Relations (2.27)-(2.30) are implemented in MathModelica using the code below.

```
ModelAInputSource,
 FlowCut 8b<;
 Temperature Tambient;
 Constant Real cp Š 1005;
 EquationA
  Tambient Š 300;
  b.W Š Sin@2*Time + 0.6D;
  b.H Š If@b.W < 0, b.W*cp*Tambient, b.W*cp*b.TD;
 E
E
```

The output source will have a similar function as the input source and is therefore described in the same way as the input source. The different value on $T_{ambient}$, and cosine function instead of sine function is used for the sake of variation.

```
ModelAOutputSource,
 FlowCut 8a<;
 Temperature Tambient;
 Constant Real cp Š 1005;
 EquationA
  Tambient Š 350;
  a.W Š 0.25*Cos@TimeD;
  a.H Š If@a.W < 0, a.W*cp*Tambient, a.W*cp*a.TD;
 E
E
```

2.5.4 Simulation

Simulation of the configuration of the chain of components, shown in Figure 16, is performed. The initial values, of the mass and the pressure of the control volume, are 2 kg. and 101300 Pa. respectively.

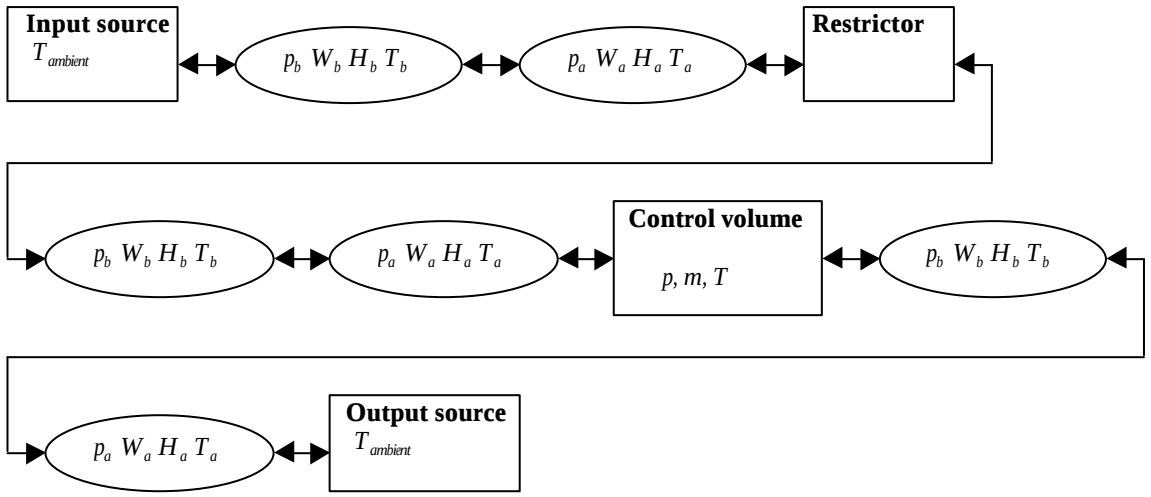


Figure 16: Chain of components with pWHT-connector borders.

The first characteristic plotted, is the pressure difference, Dp, of the restrictor.

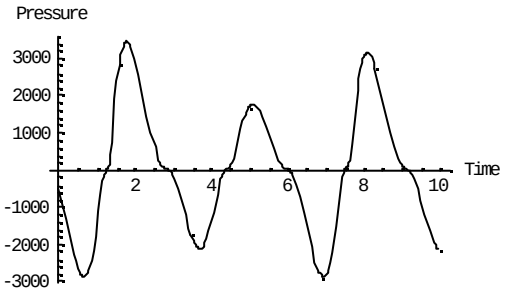


Figure 17: Pressure difference, Dp.

A negative pressure difference is received because the difference is calculated like p_a-p_b , regardless of the direction of the gas flow. That is, when the direction of the gas flow is from gate *b* to gate *a*, the pressure at gate *b* is higher than the pressure at gate *a*, resulting in a negative pressure difference.

The temperature of the gas inside the control volume, *T*, is shown in Figure 18. The control volume clearly supports bi-directional gas flow, which can be realised by studying the temperatures at gate *a* and gate *b*, shown in Figure 19 and 20 respectively. The sharp changes of the temperatures is due to the switching between the temperature of the gas inside the control volume, and the fixed temperatures of the gas in the sources. The pressure of the gas inside the control volume, *p*, is shown in Figure 21. As expected the pressure, *p*, both falls below and rises above the initial pressure level due to the decreasing and increasing amount of gas inside the control volume. A more complete simulation result can be studied in Appendix F.

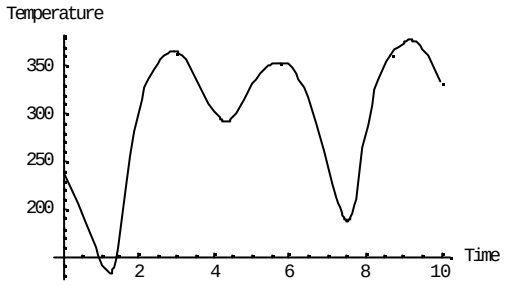


Figure 18: Temperature of the gas inside the control volume.

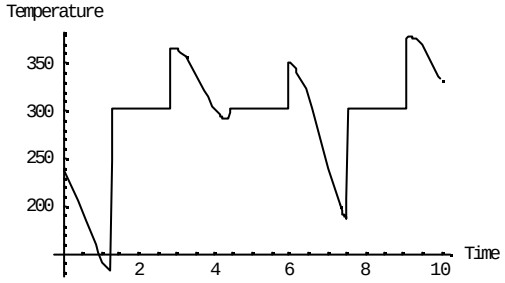


Figure 19: Temperature of the gas at gate *a* of the control volume.

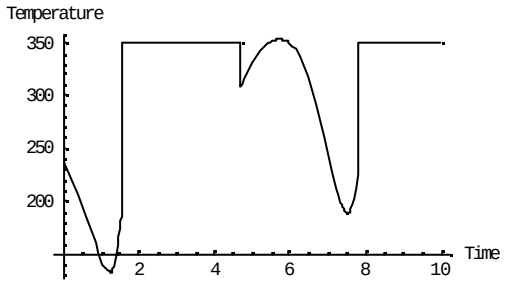


Figure 20: Temperature of the gas at gate *b* of the control volume.

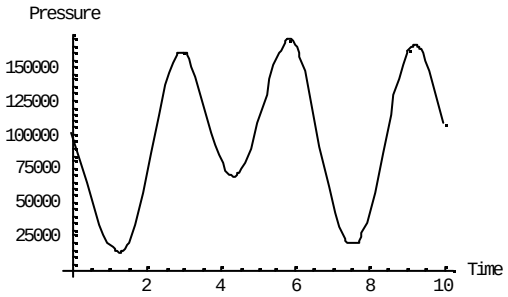


Figure 21: Pressure of the gas inside the control volume.

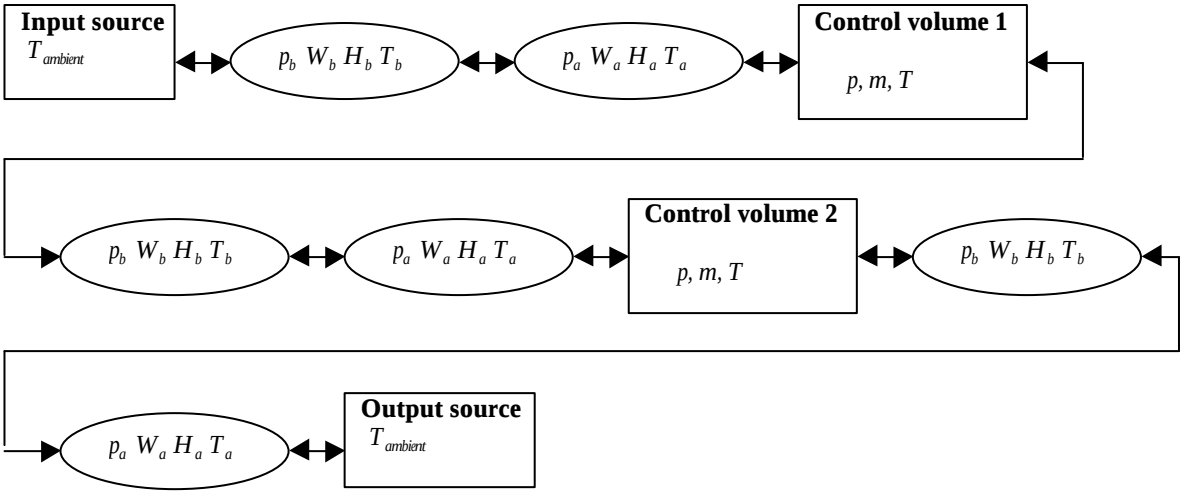


Figure 22: Chain of components with two control volumes.

The configuration of the chain of components with two control volumes and the sources, shown in Figure 22, has also been studied. The simulation resulted, surprisingly, in the following model error:

```
res = Simulate@Test, 80, 10<,
   InitialValues ® 8vol1.m Š 1.5, vol1.p Š 101300, vol2.m Š 2, vol2.p Š 101300<,
   IntervalLength ® 0.01D;

ReadMatlabData::noopen : Cannot open d:\temp\dsres.mat.
MatlabDataInterpolatingFunctionList::wrongargs :
 MatlabDataInterpolatingFunctionList called with 1
   arguments. Wrong number or nonmatching arguments: @$FailedD
```

```
!!dymolalg.txt
```

- translateModel("Test")
DAE with 32 unknown scalars and 32 scalar equations.
2 constants found.
0 parameter bound variables found.
15 alias variables found.
15 remaining time dependent variables.
Finished
- savelog

```
!!dslog.txt
```

Log-file of program .\dymosim
(generated: Mon Aug 07 11:40:15 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
The following error was detected:

Model error - division by zero: (vol1.b.U_2OverDot_H_1) / (1005*vol1.b.W) = (94755.4) / (0)

Figure 23: Model error message

According to Dymola the model error is caused by a division by zero. However according to the conditional equation

$$\texttt{b.T} == \texttt{If Ab.W} > 0, \frac{\texttt{b.H}}{\texttt{b.W} * c_p}, \texttt{TE};$$

this particular division should never occur. The conditional equation states that when W_b is zero or negative then T_b equals to T which is derived from the ideal gas law (2.1). Only when W_b is positive, i.e. not equal to zero the calculation

$$\frac{\dot{H}_b}{W_b c_p} \tag{2.31}$$

is needed to calculate the value of T_b. Why Dymola performs the division can unfortunately not be investigated due to restrictions in the software license agreement.

The complete simulation result can be studied in Appendix G.

2.5.5 Redundancy problems with pWHT-connector

As already mentioned the pWHT-connector contains redundant information. The following example will show how easy it is to run into problems when using a connector with redundant information.

Consider one wants to simulate a chain of components with a gas flow in only one direction. For example, if the gas flow is assumed to flow from gate *a* to gate *b* in the various components, an intuitive description of the sources would be (compare to section 2.3.3).

```
ModelAInputSource,
 FlowCut 8b<;
 Constant Realcp Š 1005;
 EquationA
  b.T Š 300;
  b.W Š -2+ Sin@TimeD;
  b.H Š b.W*cp*b.T;
 E
E

ModelAOutputSource,
 FlowCut 8a<;
 Constant Realcp Š 1005;
 EquationA
  a.W Š HCos@TimeDL^2;
  a.H Š a.W*cp*a.T;
 E
E
```

If the sources above are used to simulate the chain of components, shown in Figure 16, one will receive an error message. See Appendix H for an illustration. The problem is the temperature of the gas at gate *a* of the control volume. The mass flow, at gate *a* of the control volume, is positive because of the chosen input source. According to the conditional equation

$$\texttt{a.T} == \texttt{If a.W} > 0, \frac{\texttt{a.H}}{\texttt{a.W} * c_p}, \texttt{TE};$$

the temperature T_a, of the control volume, is supposed to be derived from equation (2.19). At the same time the connector, between the restrictor and the control volume, states that temperature T_a, of the control volume, shall have the same value as the temperature T_b, of the restrictor. This leads to a conflict in Dymola. This can be illustrated by studying the system of equations, (2.32)-(2.38), below.

Equations obtained from the restrictor:

$$T_b = 300 \tag{2.32}$$

$$W_b = -2 + \sin(t) \tag{2.33}$$

$$H_b = ((-2 + \sin(t)) \cdot c_p \cdot 300 \tag{2.34}$$

Equation obtained from the control volume:

$$T_a = \frac{\dot{H}_a}{W_a c_p} \tag{2.35}$$

Equations obtained from the connector:

$$W_b + W_a = 0 \tag{2.36}$$

That is the total number of equations is seven and the number of variables is six. However, in

$$\dot{H}_b + \dot{H}_a = 0 \tag{2.37}$$

$$T_b = T_a \tag{2.38}$$

this case, it is not a mathematical problem. The system of equations, (2.32)-(2.38), is mathematically solvable, but a fundamental demand of Dymola is that it requires the same number of equations as variables. This problem is comparable to the problems related to the conditional equations, described in section 2.3.1.

The solution for the problem above would be to remove, either equation

$$b.H \text{ Š } b.W * c_p * b.T;$$

from the input source, or equation

$$a.T == \text{If} A a.W > 0, \frac{a.H}{a.W * c_p}, TE;$$

from the control volume. See Appendix I for an illustration, where the equation from the input source is removed. The solution by removing equations from components is not good. Especially if the components are used, and exchanged, from a read-to-use library of components.

The problem above is perhaps the main problem with using a pWHT-connector, and can typically occur if several people are designing components and sources, and want to exchange them between each other. That is, when designing components, it is necessary for the designer to take into account, and to know, the detailed design of the neighbouring components.

3 CONCLUSIONS

In the previous chapter it is shown that the performances and limitations of the models are based on the connector used. This chapter will summarise the results and conclusions from the analysis and present guidelines and ideas for future design of engine components.

3.1 Connector selection

From the three principles analysed, the one using the pWH-connector is of no practical use. The control volume and restrictor would probably perform well if it was possible to simulate them with bi-directional gas flow. The components are only described by pressure, mass flow and energy flow. Without knowing the temperature it is impossible to know how much energy a certain amount of gas contains, which is needed for the sources.

The pWT-connector leads to the most straight forward description of the components. Applying the pWT-connector makes it possible to connect several control volumes in the component chain, see Appendix E. The disadvantage is that the pWT-connector only supports gas flow in one direction.

For bi-directional gas flow the pWHT-connector has to be applied. The disadvantages of the pWHT-connector are the slightly abstract description of the components due to the unmeasurable energy flow. The redundant information inhibits the exchange of models, because detailed knowledge of the design of the models is necessary. In the present version of MathModelica the chain of components is limited to one control volume. According to the simulation result in Appendix G a model error occurs when two control volumes connected to each other are simulated.

3.1.1 Conclusion

- pWH is of no practical use
- pWT and pWHT can be used under the described circumstances above
- A chain of components with several control volumes *and* bi-directional gas flow is not possible to design using the analysed principles and present version of MathModelica.

3.2 General comments on MathModelica

MathModelica as an environment is very pleasant to work with. With the Mathematica built-in functions, the MathModelica environment integrates most activities in simulation design and model manipulation such as coding, transformation of formulas, documentation, input and output visualisation. However some improvements have to be done to make MathModelica a really user friendly environment.

- First of all a manual is needed. With the help of a manual one could have maximum usage of the functions and many small mistakes could be avoided.

- The error messages received after a simulation failure are mostly of no help. To be able to find the error, or have a reasonable chance to understand what went wrong, one has to study the log files. Sometimes even the log files do not provide any help as in the case of the simulation of the two control volumes described in Appendix G.

• The MathModelica simulation engine and compiler, Dymola, does not support the language completely. One example is the minimum value condition in the type definition of the pressure in section 2.3.2. Another example is the problems related to the conditional equations described in section 2.3.1.

One note to the future user of MathModelica is that the initial values have to be stated explicitly on the states of the components. For example in the control volume, it is not possible to state the initial value of the pressure and the temperature, so that during simulation the ideal gas law (2.1) is applied to calculate the initial value of the mass. One has to state the initial values of the mass and pressure explicitly.

3.3 Improvements and future work

This thesis is far from a complete analyse of MathModelica from an automotive engine point of view. The author suggests the following steps of improvement in obtaining more facts to decide the full potential of MathModelica.

• Support gas mixtures

The components used in this thesis support only one type of a gas at the time. Analysing how gas mixtures can be modelled in MathModelica should be of interest to the automotive industry.

• Design more components

A complete model of an engine consists of more components than control volumes and restrictors. A natural progress would be to design and analyse more engine components. A lot of work in this area has already begun at the University of Linköping, division of Vehicular Systems.

• Analyse Dymola

A thorough investigation of Dymola is inevitable. Many errors and unexplained results is believed to be related to Dymola. An engineer using MathModelica must know what is limited by the language MathModelica and Dymola respectively. Initialisation and algorithms of Dymola have proved to be of importance to the simulation results. For example the problems with singularities, and multiple roots, described in section 2.3.3.

• Develop a users guide

To this date the product MathModelica is not released to the market meaning there is not much experience in using the product in industrial applications. Developing a “tips and tricks” guide, based on industrial experience, should provide aid for a future user of MathModelica.

APPENDIX A

Simulation of a control volume with conditional equations.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot, PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
```

pWT-connector

```
Connector @FlowCut,
 Pressure p;
 Flow MassFlow W ;
 Temperature T;
D
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  m' Š a.W+ b.W;
  a.T Š If@a.W> 0, a.T, TD;
  b.T Š If@b.W> 0, b.T, TD;
  p' == R*cp / VHcp - RL * Ha.W*a.T+ b.W*b.TL;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.W Š -2+ Sin@TimeD;
  b.T Š 300;
 D
D
```

Output source

```
ModelAOutputValues,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL2;
  a.T Š 300;
 E
E
```

Input source, control volume and output soure connected together

```
Model@VolumeTest,
 ControlVolume vol@8V Š 1<D;
 InputSource source;
 OutputValues out;
 Equation@
  Connect@source.b, vol.aD;
  Connect@vol.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@VolumeTest, 80, 10<, InitialValues ® 8vol.m Š 1, vol.p Š 101300<D;

Simulate::trsmd : Simulate failed to translate model.
- translateModelH"VolumeTest"L
DAE with 15 unknown scalars and 17 scalar equations.
Error: Model is singular:
  The number of non-trivial HscalarL equations is 7.
  The number of HscalarL variables is 5.
  Additional equations:
    equation
    vol.a.W = 2-sinHtimeL;
    which was derived from
    source.b.W = H-2L+sinHtimeL;

    vol.b.W =  -powHcosHtimeL, 2L;
    which was derived from
    out.a.W = powHcosHtimeL, 2L;


See also VolumeTest.mof
Translation aborted.

See also
Translation aborted.
- savelog
```

APPENDIX B

Simulation of a restrictor with sigularity problems.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot, PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m^3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
```

pWT-connector

```
Connector @FlowCut,
 Pressure p;
 Flow MassFlow W ;
 Temperature T;
D
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R Š 287;
 EquationA
  a.T Š b.T;
  a.W + b.W Š 0;
  Dp Š a.p - b.p;
  a.p - b.p Š (K*Ha.WL^2*R*a.T)/a.p;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  T Š b.T;
  m' Š a.W + b.W;
  p' == R*cp / V*Hcp - RL * Ha.W*a.T + b.W*TL;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.T Š 300;
  b.W Š -2 + Sin@TimeD;
 D
D
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL2;
 E
E
```

Input source, restrictor, control volume and output source connected together

```
Model@Test,
 ControlVolume vol@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol.aD;
  Connect@vol.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<, InitialValues ® 8vol.m Š 2, vol.p Š 101300<,
   IntervalLength ® 0.01D;

ReadMatlabData::noopen : Cannot open d:\temp\dsres.mat.
MatlabDataInterpolatingFunctionList::wrongargs :
 MatlabDataInterpolatingFunctionList called with 1
   arguments. Wrong number or nonmatching arguments: @$FailedD
```

```
!!dymolalg.txt
```

- translateModel("Test")
DAE with 22 unknown scalars and 22 scalar equations.
4 constants found.
0 parameter bound variables found.
11 alias variables found.
7 remaining time dependent variables.
Finished
- savelog

```
!!dslog.txt
```

Log-file of program .\dymosim
(generated: Tue Jul 25 16:36:51 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
The following error was detected:

Model error - division by zero: (86100*R.K*pow(vol.a.W, 2)) / (R.a.p) = (8.61e+008) / (0)

APPENDIX C

Simulation of a restrictor with multiple roots problem.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot, PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
```

pWT-connector

```
Connector@FlowCut,
 Pressure p;
 Flow MassFlow W;
 Temperature T;
D
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 PressureDp;
 Parameter Real K;
 Constant Real R Š 287;
 EquationA
  a.T Š b.T;
  a.W + b.W Š 0;
  Dp Š a.p - b.p;
  Ha.pL2 - a.p * b.p Š K*Ha.WL2 * R*a.T;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  T Š b.T;
  m' Š a.W + b.W;
  p' == R*cp / V*Hcp - RL *Ha.W*a.T + b.W*TL;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.T Š 300;
  b.W Š -2 + Sin@TimeD;
 D
D
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL2;
 E
E
```

Input source, restrictor, control volume and output source connected together

```
Model@Test,
 ControlVolume vol@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol.aD;
  Connect@vol.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<, InitialValues ® 8vol.m Š 2, vol.p Š 101300<,
   IntervalLength ® 0.01D;

!!dslog.txt
```

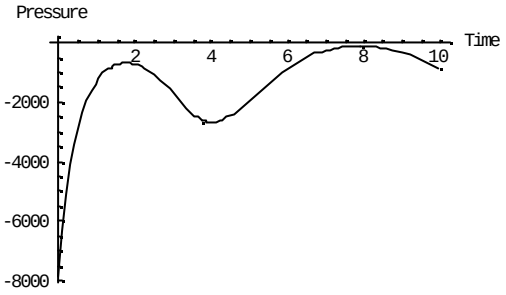
Log-file of program .\dymosim
(generated: Tue Jul 25 16:55:03 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
... "dsres.mat" creating (simulation result file)

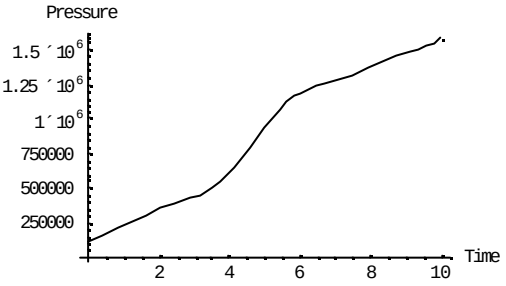
Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold))
Integration terminated successfully at T = 10
  CPU-time for integration : 0.431 seconds
  CPU-time for one GRID interval: 0.431 milli-seconds
  Number of result points : 1001
  Number of GRID points : 1001
  Number of (successful) steps : 73
  Number of F-evaluations : 168
  Number of Jacobian-evaluations: 15
  Number of (model) time events : 0
  Number of (U) time events : 0
  Number of state events : 0
  Number of step events : 0
  Minimum integration stepsize : 1e-005
  Maximum integration stepsize : 0.217
  Maximum integration order : 5
... "dsfinal.txt" creating (final states)

Plots

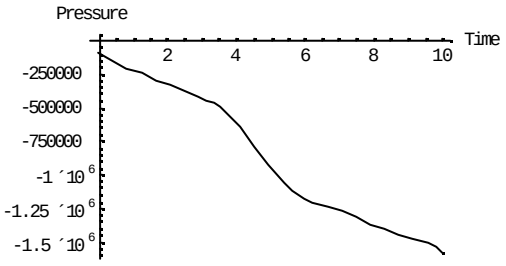
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.a.p , AxesLabel ® 8Time, Pressure<D;
```



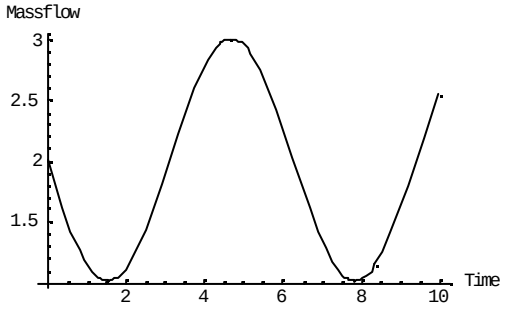

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.b.p , AxesLabel ® 8Time, Pressure<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.Dp , AxesLabel ® 8Time, Pressure<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.W<, AxesLabel ® 8Time , Massflow<D;
```



APPENDIX D

Simulation of an input source, restrictor, control volume and output source connected together with a pWT-connector.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot, PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m^3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
```

pWT-connector

```
Connector @FlowCut,
 Pressure p;
 Flow MassFlow W ;
 Temperature T;
D
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 PressureDp;
 Parameter Real K;
 Constant Real R Š 287;
 EquationA
  a.T Š b.T;
  a.W+ b.W Š 0;
  Dp Š a.p - b.p;
  a.p Š b.p/2 + $Hb.pL^2/4 + K*Ha.WL^2*R*a.T;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  T Š b.T;
  m' Š a.W + b.W;
  p' == R*cp / V*Hcp - RL * Ha.W*a.T + b.W*TL;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.T Š 300;
  b.W Š -2 + Sin@TimeD;
 D
D
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL2;
 E
E
```

Input source, restrictor, control volume and output source connected together

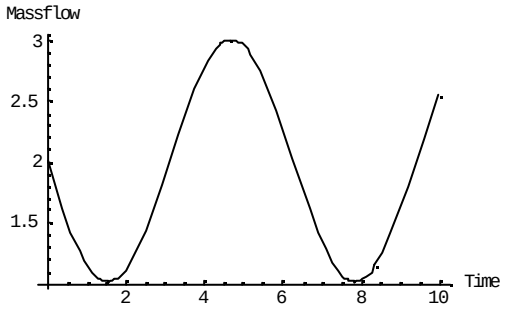
```
Model@Test,
 ControlVolume vol@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol.aD;
  Connect@vol.b, out.aD;
 D
D
```

Simulation

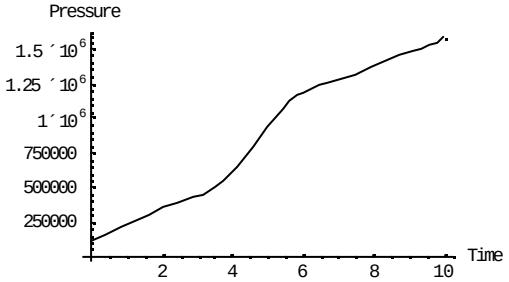
```
res = Simulate@Test, 80, 10<, InitialValues ® 8vol.m Š 2, vol.p Š 101300<,
   IntervalLength ® 0.01D;
```

Plots

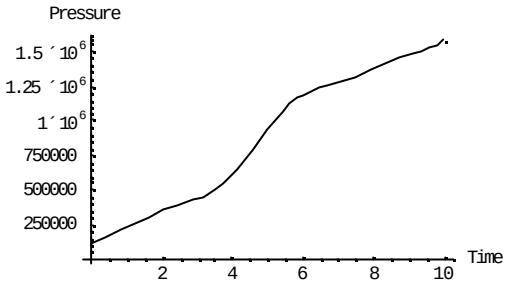
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.a.W, AxesLabel ® 8Time, Massflow<D;
```



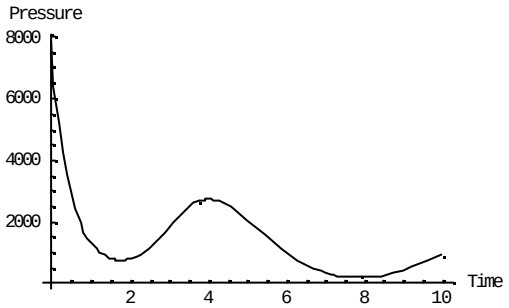

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.a.p, AxesLabel ® 8Time, Pressure<D;
```



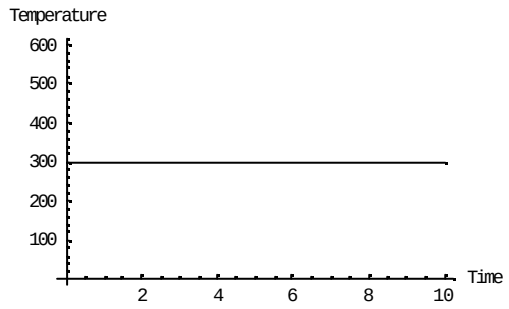
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.b.p, AxesLabel ® 8Time, Pressure<D;
```



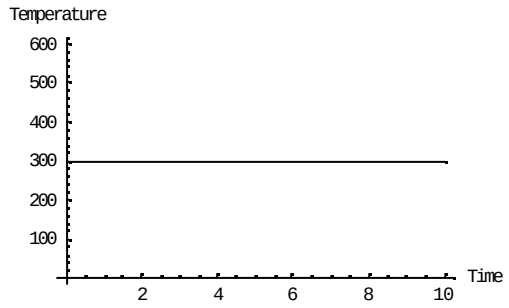

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.Dp, AxesLabel ® 8Time, Pressure<D;
```



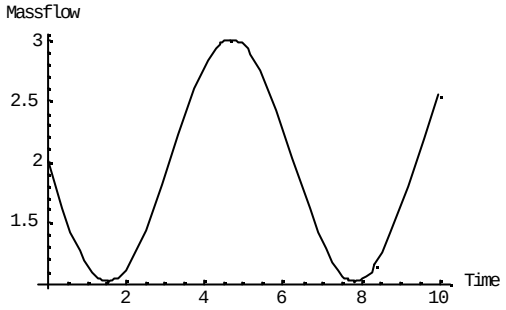

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.a.T, AxesLabel ® 8Time, Temperature<D;
```



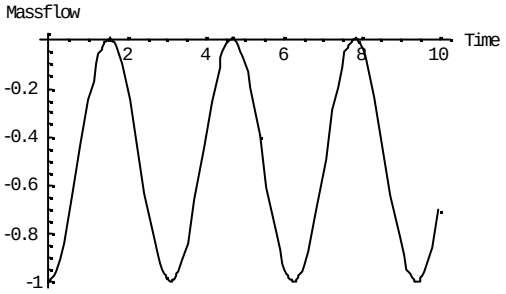

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® R.b.T, AxesLabel ® 8Time, Temperature<D;
```



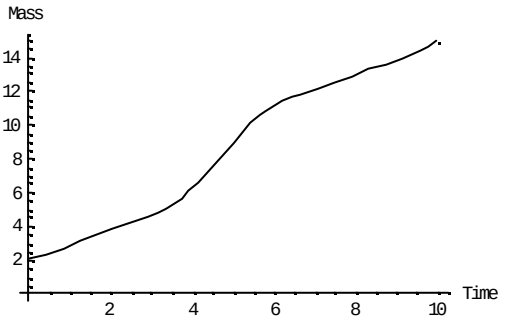
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.a.W, AxesLabel ® 8Time, Massflow<D;
```



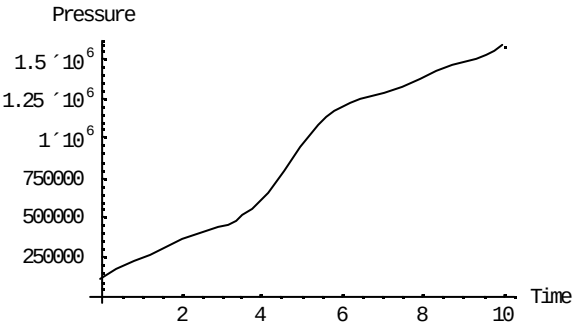

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.b.W, AxesLabel ® 8Time, Massflow<D;
```



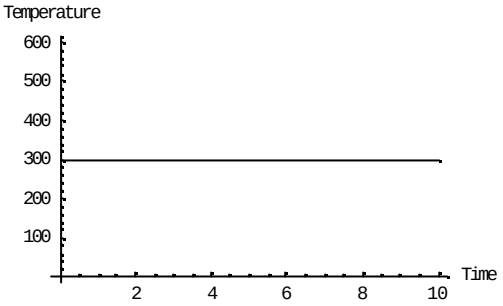

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.m, AxesLabel ® 8Time, Mass<D;
```



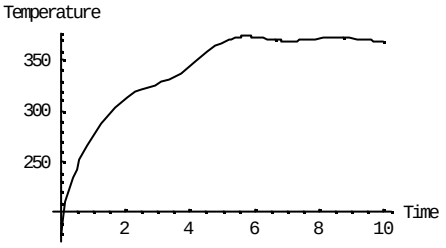

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.p, AxesLabel ® 8Time, Pressure<D;
```



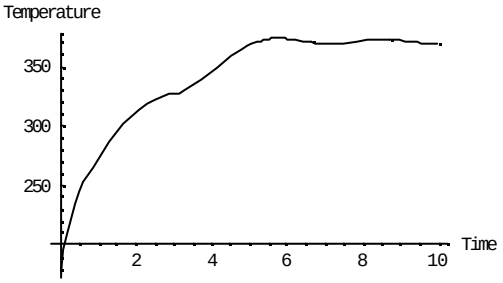
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.a.T, AxesLabel ® 8Time, Temperature<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.T, PlotRange ® All,
  AxesLabel ® 8Time, Temperature<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol.b.T, PlotRange ® All,
  AxesLabel ® 8Time, Temperature<D;
```



APPENDIX E

Simulation of two control volumes connected together with a pWT-connector.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot, PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
```

pWT-connector

```
Connector @FlowCut,
 Pressure p;
 Flow MassFlow W ;
 Temperature T;
D
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  T Š b.T;
  m' Š a.W + b.W;
  p' == R*cp / V*Hcp - RL *Ha.W*a.T + b.W*TL;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Equation@
  b.T Š 300;
  b.W Š -2 + Sin@TimeD;
 D
D
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 EquationA
  a.W Š HCos@TimeDL2;
 E
E
```

Input source, two control volumes and output source connected together

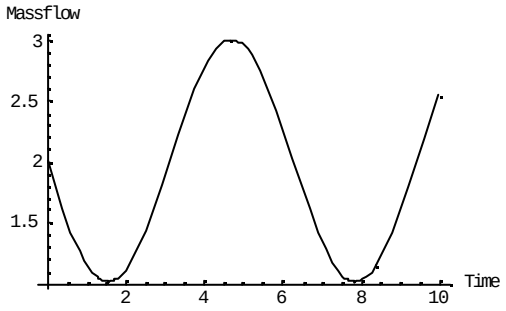
```
Model@Test,
 ControlVolume vol1@8V Š 1.5<D;
 ControlVolume vol2@8V Š 1.5<D;
 InputSource source;
 OutputValues out;
 Equation@
  Connect@source.b, vol1.aD;
  Connect@vol1.b, vol2.aD;
  Connect@vol2.b, out.aD;
 D
D
```

Simulation

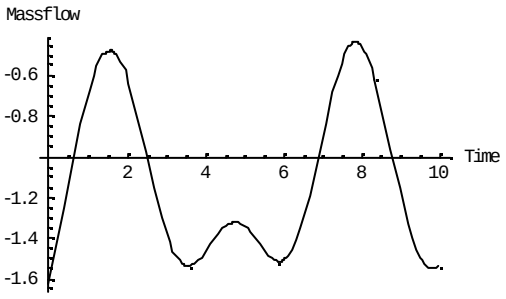
```
res = Simulate@Test, 80, 10<,
   InitialValues ® 8vol1.m Š 2, vol1.p Š 101300, vol2.m Š 2, vol2.p Š 101300<,
   IntervalLength ® 0.01D;
```

Plots

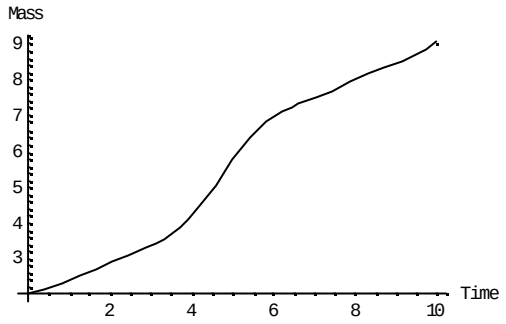
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol1.a.W, AxesLabel ® 8Time, Massflow<D;
```



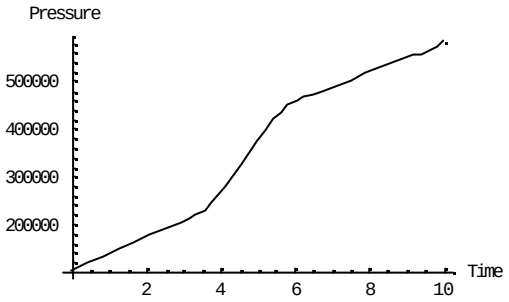

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol1.b.W, AxesLabel ® 8Time, Massflow<D;
```



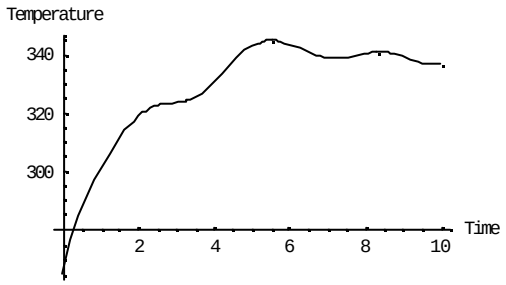
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol1.m, AxesLabel ® 8Time, Mass<D;
```



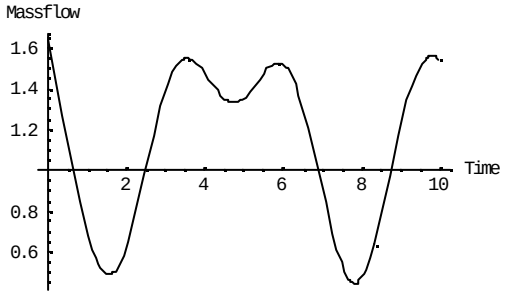

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol1.p, AxesLabel ® 8Time, Pressure<D;
```



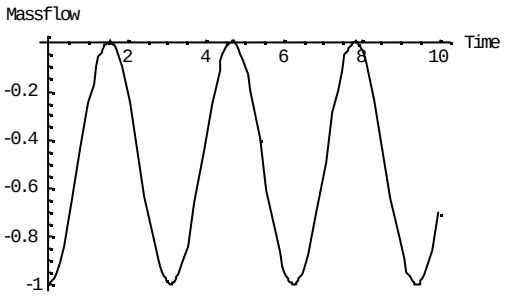

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol1.T, PlotRange ® All,
  AxesLabel ® 8Time, Temperature<D;
```



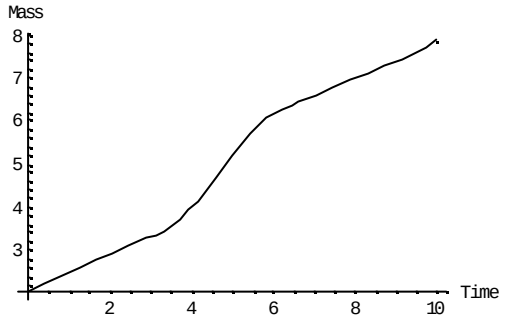

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol2.a.W, AxesLabel ® 8Time, Massflow<D;
```



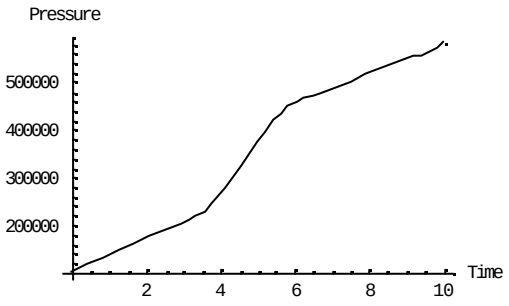
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol2.b.W, AxesLabel ® 8Time, Massflow<D;
```



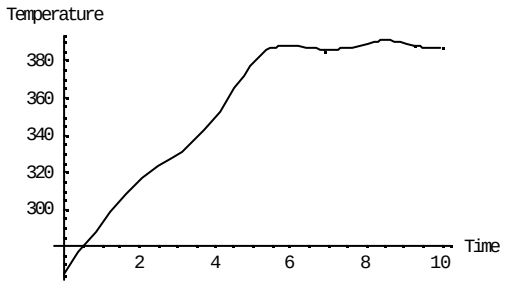

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol2.m, AxesLabel ® 8Time, Mass<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol2.p, AxesLabel ® 8Time, Pressure<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® vol2.T, PlotRange ® All,
  AxesLabel ® 8Time, Temperature<D;
```



APPENDIX F

Simulation of an input source, restrictor, control volume and output source connected together with a pWHT-connector.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot,
  PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
Type@EnergyFlow, Real@8Unit Š "W"<DD;
```

pWHT-connector

```
Connector AFlowCut,
 Pressure p;
 Flow MassFlow W ;
 Flow EnergyFlow H;
 Temperature T;
E
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  a.H + b.H Š 0;
  a.W + b.W Š 0;
  Dp Š a.p - b.p;
  IfAa.W > 0, a.p Š b.p/2 + &Hb.pL^2/4 + K*a.W*R*a.H/cp,
    b.p Š a.p/2 + &Ha.pL^2/4 + K*b.W*R*b.H/cp E;
  a.T Š b.T;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  m' Š a.W + b.W;
  a.T == IfAa.W > 0, a.H/(a.W * cp), TE;
  b.T == IfAb.W > 0, b.H/(b.W * cp), TE;
  p' == R/(V*Hcp - RL) * Ia.H + b.HM;
 E
E
```

Input source

```
ModelAInputSource,
 FlowCut 8b<;
 Temperature Tambient;
 Constant Real cp Š 1005;
 EquationA
  Tambient Š 300;
  b.W Š Sin@2*Time + 0.6D;
  b.H Š If@b.W < 0, b.W*cp*Tambient, b.W*cp*b.TD;
 E
E
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 Temperature Tambient;
 Constant Real cp Š 1005;
 EquationA
  Tambient Š 350;
  a.W Š 0.25*Cos@TimeD;
  a.H Š If@a.W < 0, a.W*cp*Tambient, a.W*cp*a.TD;
 E
E
```

Input source, restrictor, control volume and output source connected together

```
Model@Test,
 ControlVolume vol1@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol1.aD;
  Connect@vol1.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<, InitialValues ® 8vol1.m Š 1.5, vol1.p Š 101300<,
   IntervalLength ® 0.01D;
```

```
!!dslog.txt
```

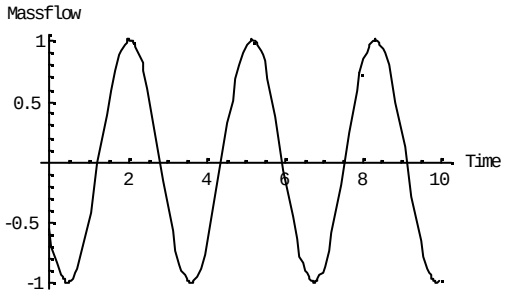
Log-file of program .\dymosim
(generated: Tue Aug 01 15:22:43 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
... "dsres.mat" creating (simulation result file)

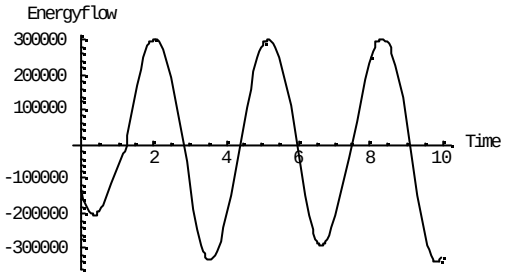
Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold))
Integration terminated successfully at T = 10
  CPU-time for integration : 0.451 seconds
  CPU-time for one GRID interval: 0.451 milli-seconds
  Number of result points : 1019
  Number of GRID points : 1001
  Number of (successful) steps : 293
  Number of F-evaluations : 771
  Number of H-evaluations : 1361
  Number of Jacobian-evaluations: 139
  Number of (model) time events : 0
  Number of (U) time events : 0
  Number of state events : 9
  Number of step events : 0
  Minimum integration stepsize : 5.22e-006
  Maximum integration stepsize : 0.167
  Maximum integration order : 5
... "dsfinal.txt" creating (final states)

Plots

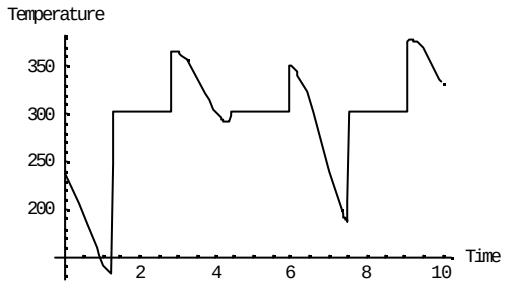
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.W<, AxesLabel ® 8Time, Massflow<D;
```



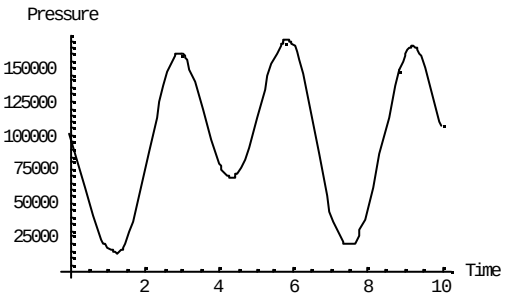
```
PlotSimulationAres, 8t, 0, 10<, PlotVariables ® 9R.a.H=, AxesLabel ® 8Time, Energyflow<E;
```



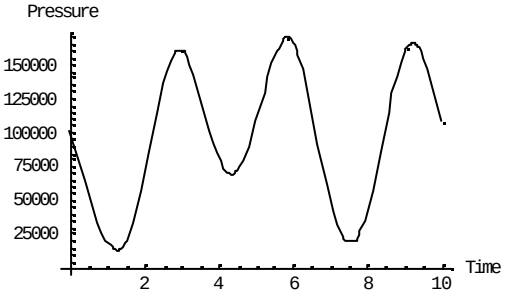

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.T<, AxesLabel ® 8Time, Temperature<D;
```



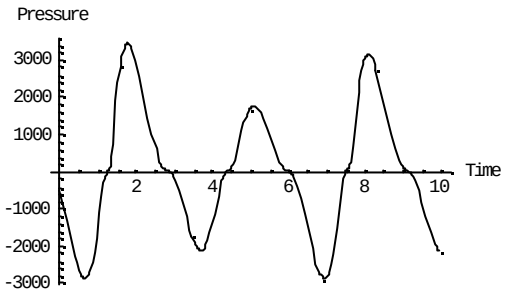

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.p<, AxesLabel ® 8Time, Pressure<D;
```



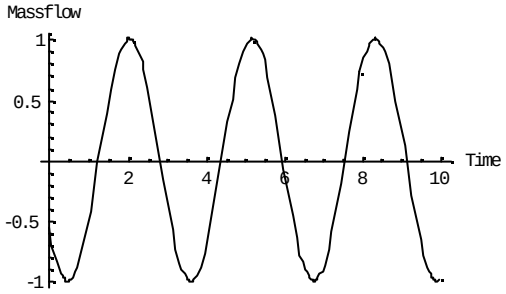

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.b.p<, AxesLabel ® 8Time, Pressure<D;
```



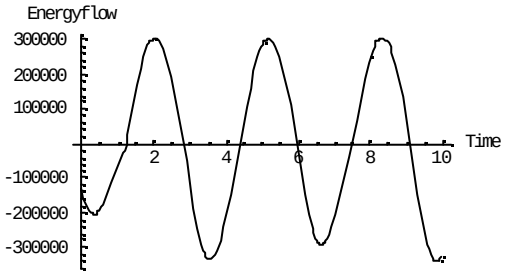
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.Dp<, AxesLabel ® 8Time, Pressure<D;
```



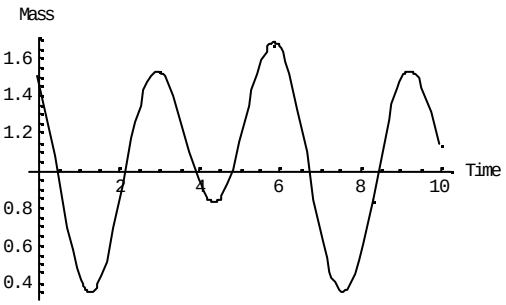

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.a.W<, AxesLabel® 8Time, Massflow<D;
```



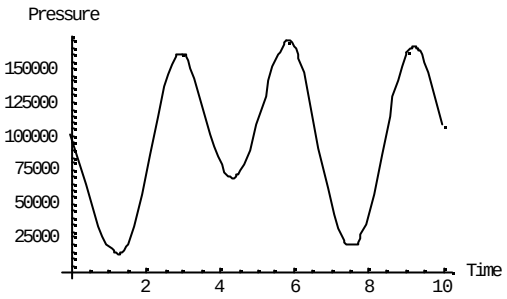

```
PlotSimulationAres, 8t, 0, 10<, PlotVariables ® 9vol1.a.H=, AxesLabel ® 8Time, Energyflow<E;
```



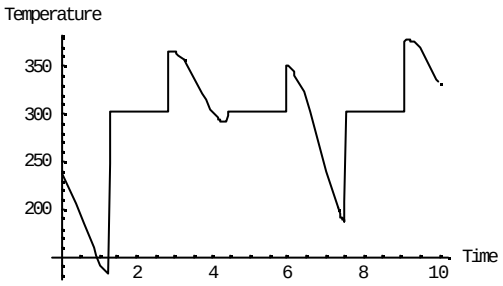

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.m<, AxesLabel ® 8Time, Mass<D;
```



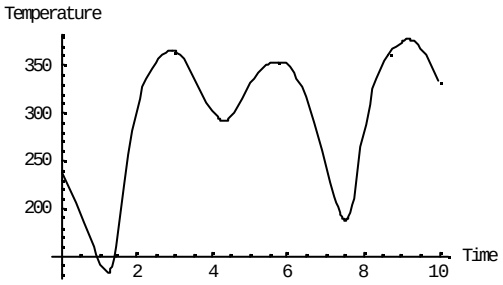
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.p<, AxesLabel ® 8Time, Pressure<D;
```



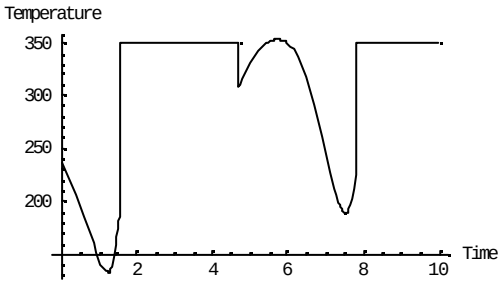

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables® 8vol1.a.T<, AxesLabel ® 8Time, Temperature<D;
```



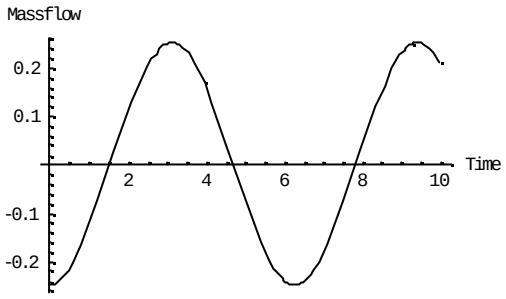

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.T<, AxesLabel® 8Time, Temperature<D;
```



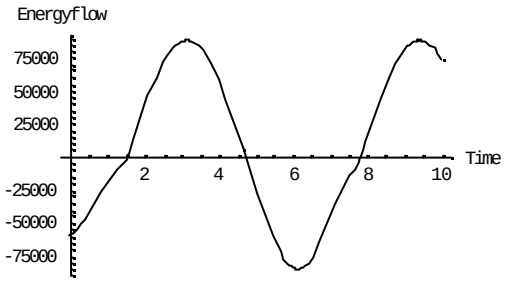

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.b.T<, AxesLabel® 8Time, Temperature<D;
```



```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8vol1.b.W<, AxesLabel ® 8Time, Massflow<D;
```




```
PlotSimulationAres, 8t, 0, 10<, PlotVariables ® 9vol1.b.H=, AxesLabel ® 8Time, Energyflow<E;
```



APPENDIX G

Simulation of two control volumes connected together with a pWHT-connector.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot,
  PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
Type@EnergyFlow, Real@8Unit Š "W"<DD;
```

pWHT-connector

```
Connector AFlowCut,
 Pressure p;
 Flow MassFlow W ;
 Flow EnergyFlow H;
 Temperature T;
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit Š "m3"=E; "Volume";
 Constant Real R Š 287;
 Constant Real cp Š 1005;
 EquationA
  p Š a.p;
  b.p Š p;
  p*V Š m*R*T;
  m' Š a.W+b.W;
  a.T== IfAa.W>0, a.H/(a.W*cp), TE;
  b.T== IfAb.W>0, b.H/(b.W*cp), TE;
  p'== R/(V*Hcp-RL)*Ia.H+b.HM;
 E
E
```

Input source

```
ModelAInputSource,
 FlowCut 8b<;
 Temperature Tfix;
 Constant Real cp Š 1005;
 EquationA
  Tfix Š 300;
  b.W Š Sin@2*Time+0.6D;
  b.H Š If@b.W<0, b.W*cp*Tfix, b.W*cp*b.TD;
 E
E
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 Temperature Tfix;
 Constant Real cp Š 1005;
 EquationA
  Tfix Š 350;
  a.W Š 0.25*Cos@TimeD;
  a.H Š If@a.W<0, a.W*cp*Tfix, a.W*cp*a.TD;
 E
E
```

Input source, two control volumes and output source connected together

```
Model@Test,
 ControlVolume vol1@8V Š 1<D;
 ControlVolume vol2@8V Š 1.5<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, vol1.aD;
  Connect@vol1.b, vol2.aD;
  Connect@vol2.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<,
   InitialValues ® 8vol1.m Š 1.5, vol1.p Š 101300, vol2.m Š 2, vol2.p Š 101300<,
   IntervalLength ® 0.01D;

ReadMatlabData::noopen : Cannot open d:\temp\dsres.mat.
MatlabDataInterpolatingFunctionList::wrongargs :
 MatlabDataInterpolatingFunctionList called with 1
   arguments. Wrong number or nonmatching arguments: @$FailedD
```

```
!!dymolalg.txt
```

- translateModel("Test")
DAE with 32 unknown scalars and 32 scalar equations.
2 constants found.
0 parameter bound variables found.
15 alias variables found.
15 remaining time dependent variables.
Finished
- savelog

```
!!dslog.txt
```

Log-file of program .\dymosim
(generated: Mon Aug 07 11:40:15 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
The following error was detected:

Model error - division by zero: (vol1.b.U_2OverDot_H_1) / (1005*vol1.b.W) = (94755.4) / (0)

APPENDIX H

Simulation with redundancy problems in a pWHT-connector.

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot,
  PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m^3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
Type@EnergyFlow, Real@8Unit Š "W"<DD;
```

pWHT-connector

```
Connector AFlowCut,
 Pressure p;
 Flow MassFlow W ;
 Flow EnergyFlow H;
 Temperature T;
E
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R = 287;
 Constant Real cp = 1005;
 EquationA
  a.H + b.H = 0;
  a.W + b.W = 0;
  Dp = a.p - b.p;
  IfAa.W > 0, a.p = b.p/2 + &(Hb.pL^2/4 + K*a.W*R*a.H/cp),
    b.p = a.p/2 + &(Ha.pL^2/4 + K*b.W*R*b.H/cp) E;
  a.T = b.T;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit = "m^3"=E; "Volume";
 Constant Real R = 287;
 Constant Real cp = 1005;
 EquationA
  p = a.p;
  b.p = p;
  p*V = m*R*T;
  m' = a.W + b.W;
  a.T == IfAa.W > 0, a.H/(a.W*cp), TE;
  b.T == IfAb.W > 0, b.H/(b.W*cp), TE;
  p' == R/(V*Hcp - RL) * Ia.H + b.HM;
 E
E
```

Input source

```
ModelAInputSource,
 FlowCut 8b<;
 Constant Realcp Š 1005;
 EquationA
  b.T Š 300;
  b.W Š -2+ Sin@TimeD;
  b.H Š b.W*cp* b.T;
 E
E
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 Constant Realcp Š 1005;
 EquationA
  a.W Š HCos@TimeDL2;
  a.H Š a.W*cp* a.T;
 E
E
```

Input source, restrictor, control volume and output source connected together

```
Model@Test,
 ControlVolumevol1@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol1.aD;
  Connect@vol1.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<, InitialValues ® 8vol1.m Š 1.5, vol1.p Š 101300<,
   IntervalLength ® 0.01D;
```

```
Simulate::trsmd : Simulate failed to translate model.

- translateModelH"Test"L
DAE with 28 unknown scalars and 29 scalar equations.
Error: Model is singular:
  The number of non-trivial HscalarL equations is 11.
  The number of HscalarL variables is 10.
  Additional equations:
    equation
    vol1.b.U_2OverDot_H_1 = 1005*vol1.b.W*vol1.b.T;
    which was derived from
    out.a.U_2OverDot_H_1 = out.a.W*out.U_2Subscript_c_p*out.a.T;


See also Test.mof
Translation aborted.

See also
Translation aborted.
- savelog
```

APPENDIX I

Simulation with solved redundancy problems

Initialization

```
Needs@"MathModelica`"D
Needs@"Graphics`Colors`"D
SetDirectory@"d:\\temp"D;
SetOptions@Plot,
  PlotStyle ® 8Red, Blue, Green, Cyan, Magenta, Yellow, Red<D;
```

Type definition

```
Type@Temperature, Real@8Unit Š "K"<DD;
Type@Mass, Real@8Unit Š "kg"<DD;
TypeAVolume, RealA9Unit Š "m3"=EE;
TypeAMassFlow, RealA9Unit Š "kg/s"=EE;
Type@Pressure, Real@8Unit Š "Pa"<DD;
Type@EnergyFlow, Real@8Unit Š "W"<DD;
```

pWHT-connector

```
Connector AFlowCut,
 Pressure p;
 Flow MassFlow W ;
 Flow EnergyFlow H;
 Temperature T;
E
```

Restrictor

```
ModelARestrictor,
 FlowCut 8a, b<;
 Pressure Dp;
 Parameter Real K;
 Constant Real R = 287;
 Constant Real cp = 1005;
 EquationA
  a.H + b.H = 0;
  a.W + b.W = 0;
  Dp = a.p - b.p;
  IfAa.W > 0, a.p = b.p/2 + &(Hb.pL^2/4 + K*a.W*R*a.H/cp),
    b.p = a.p/2 + &(Ha.pL^2/4 + K*b.W*R*b.H/cp) E;
  a.T = b.T;
 E
E
```

Control volume

```
ModelAControlVolume,
 FlowCut 8a, b<;
 Temperature T;
 Mass m;
 Pressure p;
 Parameter Real VA9Unit = "m^3"=E; "Volume";
 Constant Real R = 287;
 Constant Real cp = 1005;
 EquationA
  p = a.p;
  b.p = p;
  p*V = m*R*T;
  m' = a.W + b.W;
  a.T == IfAa.W > 0, a.H/(a.W*cp), TE;
  b.T == IfAb.W > 0, b.H/(b.W*cp), TE;
  p' == R/(V*Hcp - RL) * Ia.H + b.HM;
 E
E
```

Input source

```
Model@InputSource,
 FlowCut 8b<;
 Constant Realcp Š 1005;
 Equation@
  b.T Š 300;
  b.W Š -2+ Sin@TimeD;
 D
D
```

Output source

```
ModelAOutputSource,
 FlowCut 8a<;
 Constant Realcp Š 1005;
 EquationA
  a.W Š HCos@TimeDL^2;
  a.H Š a.W*cp* a.T;
 E
E
```

Input source, restrictor, control volume and output source connected together

```
Model@Test,
 ControlVolumevol1@8V Š 1<D;
 Restrictor R@8K Š 2500<D;
 InputSource source;
 OutputSource out;
 Equation@
  Connect@source.b, R.aD;
  Connect@R.b, vol1.aD;
  Connect@vol1.b, out.aD;
 D
D
```

Simulation

```
res = Simulate@Test, 80, 10<,
   InitialValues ® 8vol1.m Š 1.5, vol1.p Š 101300, vol2.m Š 2, vol2.p Š 101300<,
   IntervalLength ® 0.01D;
```

```
!!dslog.txt
```

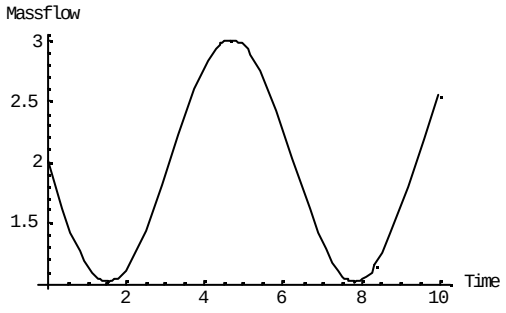
Log-file of program .\dymosim
(generated: Tue Aug 22 15:45:25 2000)

dymosim started (dymosim version 4.4, Nov 16, 1999)
... "dsin.txt" loading (dymosim input file)
... "dsres.mat" creating (simulation result file)

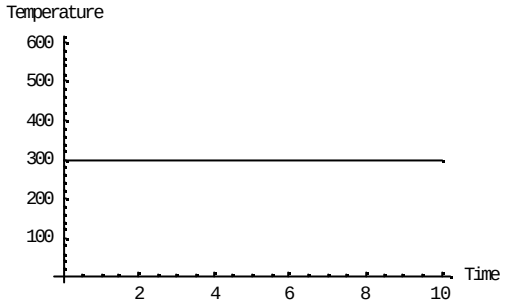
Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold))
Integration terminated successfully at T = 10
  CPU-time for integration : 0.491 seconds
  CPU-time for one GRID interval: 0.491 milli-seconds
  Number of result points : 1001
  Number of GRID points : 1001
  Number of (successful) steps : 73
  Number of F-evaluations : 166
  Number of H-evaluations : 1073
  Number of Jacobian-evaluations: 15
  Number of (model) time events : 0
  Number of (U) time events : 0
  Number of state events : 0
  Number of step events : 0
  Minimum integration stepsize : 1e-005
  Maximum integration stepsize : 0.286
  Maximum integration order : 5
... "dsfinal.txt" creating (final states)

Plots

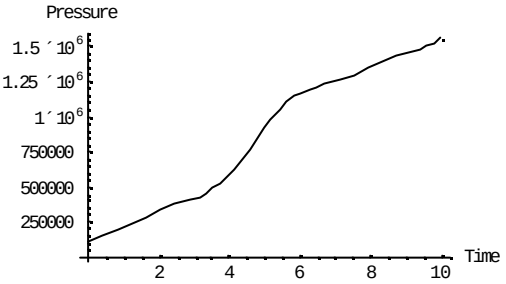
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.W<, AxesLabel ® 8Time, Massflow<D;
```



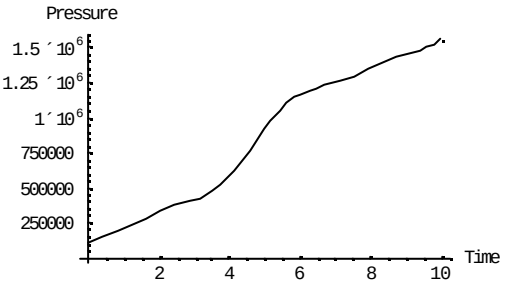
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.T<, AxesLabel ® 8Time, Temperature<D;
```



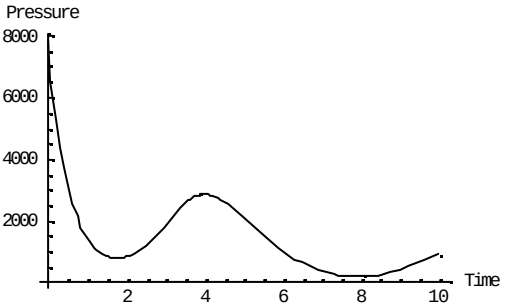

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.a.p<, AxesLabel ® 8Time, Pressure<D;
```



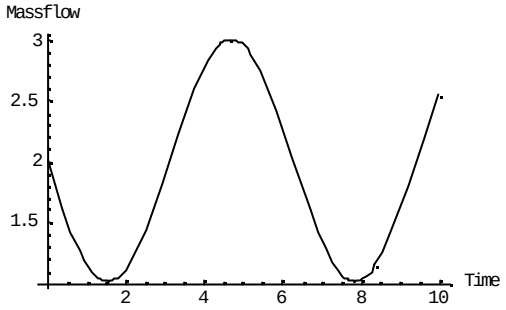

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.b.p<, AxesLabel ® 8Time, Pressure<D;
```



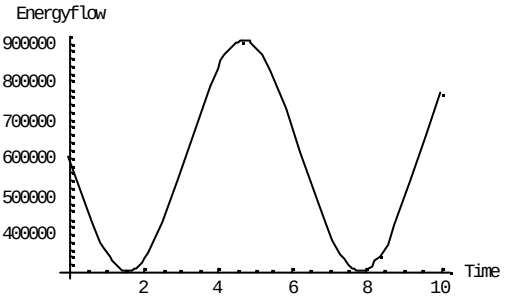

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables ® 8R.Dp<, AxesLabel ® 8Time, Pressure<D;
```



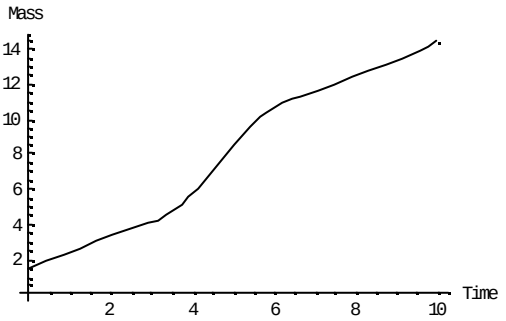
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.a.W<, AxesLabel®8Time, Massflow<D;
```



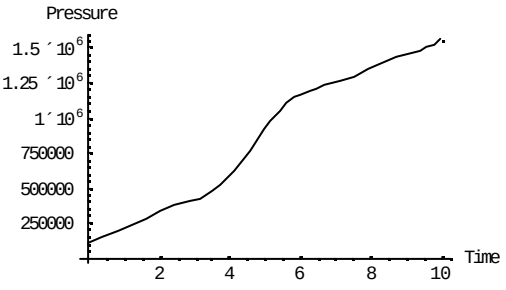

```
PlotSimulationAres, 8t, 0, 10<, PlotVariables®9vol1.a.H=, AxesLabel®8Time, Energyflow<E;
```



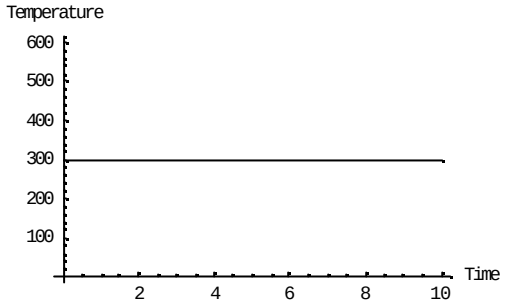

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.m<, AxesLabel®8Time, Mass<D;
```



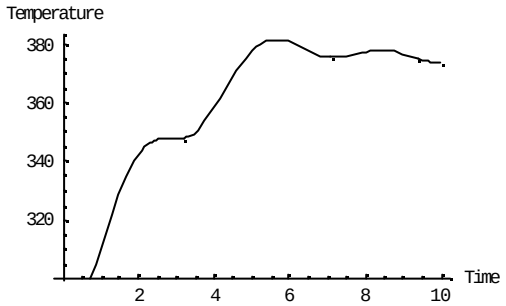

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.p<, AxesLabel®8Time, Pressure<D;
```



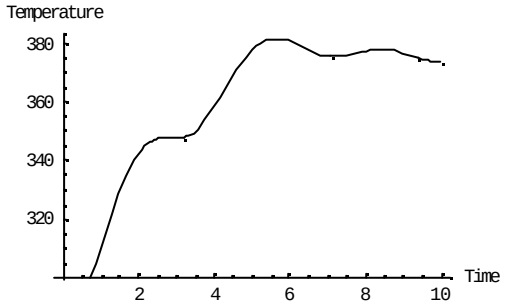
```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.a.T<, AxesLabel®8Time, Temperature<D;
```



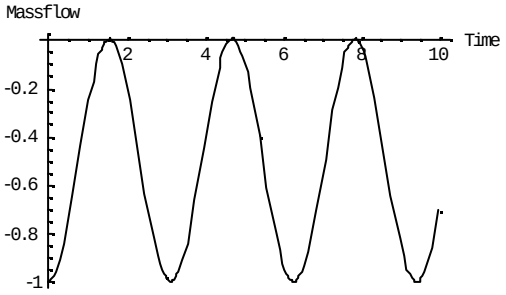

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.T<, AxesLabel®8Time, Temperature<D;
```



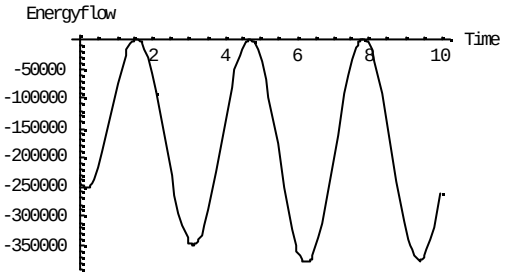

```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.b.T<, AxesLabel®8Time, Temperature<D;
```




```
PlotSimulation@res, 8t, 0, 10<, PlotVariables®8vol1.b.W<, AxesLabel®8Time, Massflow<D;
```



```
PlotSimulationAres, 8t, 0, 10<, PlotVariables ® 9vol1.b.H=, AxesLabel ® 8Time, Energyflow<E;
```



BIBLIOGRAPHY

[1] DaimlerChrysler AG
http://www.daimlerchrysler.com
Information about DaimlerChrysler AG.

[2] Dynasim
http://www.dynasim.se
Information about Dymola.

[3] Ekroth, I. and Granryd, E.
Tillämpad termodynamik
Institutionen för Energiteknik
Avdelningen för Tillämpad termodynamik
Kungliga Tekniska Högskolan, Stockholm 1994
ISBN: 91-7170-067-6

[4] Eriksson, L. and Nielsen, L.
Course material Vehicular Systems
Linköpings Universitet, Linköping 1999

[5] Kiencke, U. and Nielsen, L.
Automotive Control Systems
Springer-Verlag, Berlin Heidelberg New York, 2000
ISBN: 3-540-66922-1

[6] MathCore AB
http://www.mathcore.se
Information about MathModelica

[7] Modelica Design Group
http://www.modelica.org
Information about the Modelica language

[8] Modelica Design Group
Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, 1999
http://www.modelica.org/current/ModelicaSpec13norev.pdf

[9] Modelica Design Group
Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, Tutorial and Rationale, 1999
http://www.modelica.org/current/ModelicaRationale13norev.pdf

[10] University of Linköping, Sweden
http://www.fs.isy.liu.se
Information about the University of Linköping, division of Vehicular Systems

[11] Wolfram Research Inc.
http://www.wolfram.com
Information about Mathematica.