Proceedings
of the 4th International Modelica Conference,
Hamburg, March 7-8, 2005,
Gerhard Schmitz (editor)

J. Ungethüm
German Aerospace Center, Stuttgart, Germany
Fuel Cell System Modeling for Real-time Simulation
pp. 67-74

Paper presented at the 4th International Modelica Conference, March 7-8, 2005,
Hamburg University of Technology, Hamburg-Harburg, Germany,
organized by The Modelica Association and the Department of Thermodynamics, Hamburg University of Technology

All papers of this conference can be downloaded from
http://www.Modelica.org/events/Conference2005/

Fuel Cell System Modeling for Real-time Simulation

Jörg Ungethüm*
German Aerospace Center, Institute of Vehicle Concepts
Pfaffenwaldring 38-40, 70569 Stuttgart

Abstract

In this paper a model of a subsystem of an automotive fuel cell power generation unit is presented. The subsystem model describes the cathode side of the fuel cell, containing the air supply system. As far as possible, standard libraries were used to accomplish a high level of compatibility with other models. The model runs in real-time on dSPACE hardware and is used in a Hardware-In-the-Loop (HIL) simulation environment.
Keywords: fuel cell, automotive simulation, real-time simulation, hardware-in-the-loop

1 Introduction

In the context of future vehicle development, models are needed for the simulation of the fuel cell system and its periphery. These models are used for off-line system simulation as well as for HIL simulation with respect to controller development. Most of the existing models are built upon MATLAB/Simulink. The model which is presented in this paper serves particularly in the evaluation of object-oriented modeling in Modelica as an alternative to modeling in MATLAB/Simulink. Apart from modeling potentialities, flexibility and simulation performance, the code administration is also relevant in this context.

In figure 1 a sketch of the modeled subsystem is shown, which covers the cathode side of a PEM fuel cell system. The fuel cell is supplied with compressed and cooled air. Due to the electro-chemical reaction in the stack, the rate of oxygen in the air is reduced and at the same time the air takes up the major part of the reaction water. Heat is rejected by means of a cooling medium. The remaining pressure difference between the stack outlet and the environment is used in an exhaust gas turbine. Therefore, and to recover reaction water, liquid water is dragged from the air in a separator. Using the recirculation valve, air can be led back directly to the entrance of the compressor. Compressor and exhaust gas turbine are mounted together with an electrical drive engine on a common shaft.

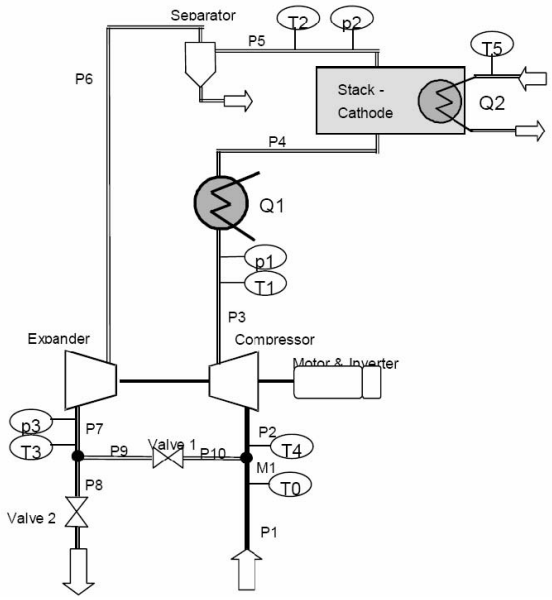

Figure 1: Sketch a the fuel cell subsystem to simulate

2 Model design

The model covers the process engineering part of the system, whereby the focus has been the description of the thermodynamic behavior. The mechanical part is built up with simple models, the fuel cell stack is realized as thermal inertia and a source of heat and humidity. The model realistically reproduces the most important influences on the dynamic behavior of the system. These are the volumes and thermal capacities of the components and the inertia of the common shaft of compressor, exhaust gas turbine and electrical machine. The top level of the model is shown in figure 2.

*joerg.ungethuem@dlr.de

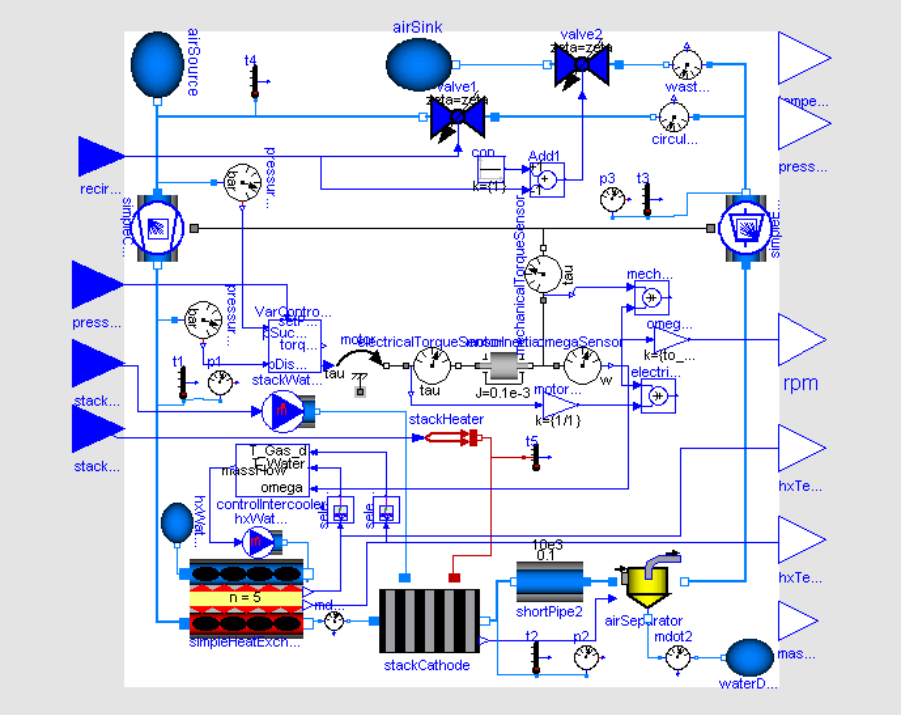

Figure 2: The Modelica system model

3 Aspects of real-time simulation

3.1 Toolchain

As the model is used in context of HIL simulation on a dSPACE system, it has to run in real-time on the target. To enable real-time simulation of potentially stiff models, Dymolas inline integration approach is used [1]. The model is embedded as a S-function into a simple Simulink wrapper model (figure 3) using Dymolas Simulink interface block. The MATLAB Real-Time Workshop and the dSPACE target compiler are needed to compile the compounded Simulink model. Visualization is realized using the dSPACE ControlDesk program.

3.2 Real-time model requirements

The central requirements of real-time modeling are deterministic computing time and high computing speed. In order to provide a deterministic computing time, iterative solution algorithms should be avoided. High computing speed is reached by keeping model equations as simple as possible.

3.2.1 Avoidance of nonlinear sets of equations

Nonlinear sets of equations must be solved by iteration, if they cannot be eliminated symbolically. However, this leads to bad performance of the simulation. In real-time simulation the situation is even worse, because the iteration can prevent the deterministic solution behavior which is required. On the other hand it is not strictly necessary to avoid any implicit equation. As long as the required number of iterations is moderate, real-time requirements can be fulfilled anyway.

3.2.2 Use of simple medium models

The evaluation of medium properties in thermodynamic models can take up a major part of the computing time. Therefore properties should not be formulated more complicated than absolutely necessary. In particular numeric problematic functions, e.g. logarithm, high polynomial degrees and broken exponents should be avoided. Within the implementation extraordinary care must be put on good performance and numeric stability. Using Horner's scheme instead of the `pow()` function for polynomial evaluation might be mentioned as an example.

3.2.3 Properties and state variables

In most cases, thermodynamic variables of state are also state variables of the model. The medium properties should be present as explicit, fast evaluable functions of the actual set of state variables. Therefore, the choice of the variables of state depends on the used

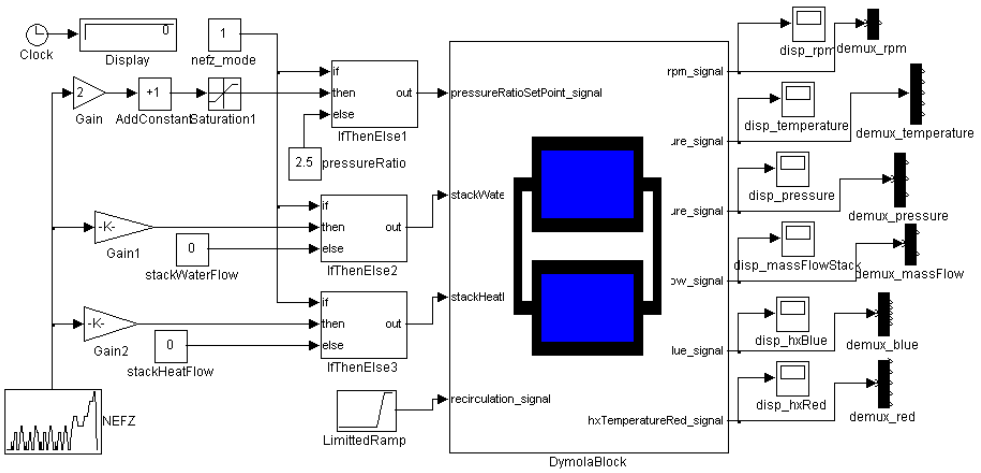

Figure 3: The Simulink wrapper model

states	derivatives
p, ρ	$\frac{\partial \rho}{\partial h}\|_p, \frac{\partial \rho}{\partial p}\|_h$
ρ, T	$\frac{\partial u}{\partial \rho}\|_T, \frac{\partial u}{\partial T}\|_\rho$

Table 1: Extra 2nd order derivatives needed in transformed balance equations

property model. Most property models are explicit in density ρ and temperature T or enthalpy h and pressure p. As a result the thermodynamic balance equations should be formulated in such a way that either density and temperature or enthalpy and pressure are computed by differential equations. Since in the primary form of mass and energy balance density ρ and internal energy u are calculated, these equations must be transformed. The transformed balance equations contain additional partial differentials of the thermodynamic state variables, which must be computed in the property model (table 1, [7]).

3.2.4 Avoidance of numeric Jacobians

Numerical Jacobian approximation is a common source of instability and inaccuracy. In Dymola numeric Jacobians are used only if the necessary partial derivatives cannot be computed symbolically. There are few cases where symbolic derivative generation is not possible yet:

- external functions (library calls, calls to routines written in C or Fortran)
- Modelica functions with the exception of one-liners (at least up to Dymola 5.2)

However, the necessary derivatives can be provided explicitly by the user [4].

3.2.5 Avoidance of numerically disadvantageous functions

Some mathematical built-in functions are problematic in numeric simulations. The square root function and the logarithm function have limited definition ranges, the derivation of the root function has an additional singularity at zero. These functions should be avoided or should be replaced by smooth approximations. Note, due to the symbolic equation treatment, the inverse functions of the above can also lead to trouble.

3.2.6 Avoidance of redundant events

Due to event propagation events require additional evaluations of the set of equations, which can lead to injury of the computing time restriction [2]. Events can be avoided if discontinuous equations and functions are replaced by continuous approximations.

4 Selection of Modelica libraries

An aim of the project is to use standard libraries as far as possible to achieve good compatibility with other models. Apart from proprietary developments [6] the ThermoFluid library [7] and in particular the new libraries Modelica.Fluid and Modelica.Media [3] are of special interest. Although the Modelica.Fluid library is still in an early development state, this library was selected as the base library. Modelica.Fluid itself is

based on the Modelica.Media library. As an unique feature Modelica.Fluid allows the implementation of models which are in fact medium-independent. In order to achieve good performance, the formulation of the thermodynamic balance equations must be adapted to the material property routines. In the ThermoFluid library this adaption is done explicitly by the user, as a suitable state transformation is selected. In the new Modelica.Fluid library this transformation takes place automatically via skillful use of the index reduction algorithm.
The Modelica.Fluid library is currently under construction. Substantially components like discretized pipes or control valves are still missing. Nevertheless the library is already usable. The interfaces (connectors), a control volume and throttling devices are already available. The Modelica.Media library is already developed further. Property models for ideal gases, several models for water (among other the IAPWS97 formulation [8]) are implemented. Mixtures are implemented likewise. However, these could not be used because of implementation problems in the version that was used. The library already offers a sufficiently good documentation including a tutorial, which makes the implementation of additional property models possible.

5 Component library

To implement the fuel cell subsystem model outlined in chapter 1, component models like heat exchanger, mixer, separator, compressor and exhaust gas turbine are needed. These component models go beyond the scope of the Modelica.Fluid library. Therefore, a new component library ModelicaFluidX is built on basis of Modelica.Fluid. The structure of the library is shown in figure 4. Due to the level of development of the base libraries, structure and implementation are still subject of change. The library is developed aiming real-time applications.

5.1 The CommonFunctions folder

The folder CommonFunctions was taken over from the ThermoFluid library. The functions were partly redesigned as one-liners in order to ensure their automatic differentiability. As an example the function ThermoRoot is shown. The actual Modelica code was generated using the C Preprocessor:

```
function ThermoRoot
  "Square root function with cubic
```

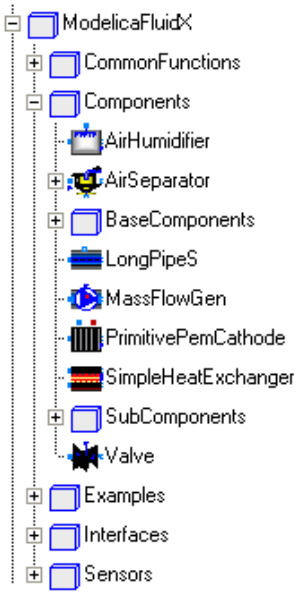

Figure 4: The ModelicaFluidX component library

```
   spline interpolation near 0"
  input Real x;
  input Real deltax;
  output Real y;
algorithm
/*
// pipe this through  'cpp -P -' to
// generate Modelica one-liner below
#define adeltax abs(deltax)
#define C1 (5/(4*adeltax^(0.5)))
#define C3 (-1/(4*adeltax^(2.5)))
algorithm
  y := noEvent(if (x > adeltax)
   then sqrt(x)
   else
   if (x < -adeltax)
   then -sqrt(-x)
   else (C1
    + C3*x*x)*x);
// EOF
*/
  y := noEvent(if (x > abs(deltax))
   then sqrt(x)
   else
   if (x < -abs(deltax))
   then -sqrt(-x)
   else ((5/(4*abs(deltax)^(0.5)))
    + (-1/(4*abs(deltax)^(2.5))) *x*x)*x);
end ThermoRoot;
```

The function extends the root function into the range of negative arguments and avoids the singularity of the 1st derivative in the origin (see figure 5). In contrast to the implementation in the ThermoFluid library, this version can be differentiated automatically. As side effect the restriction to constant approximation radii of the original implementation is void. Since

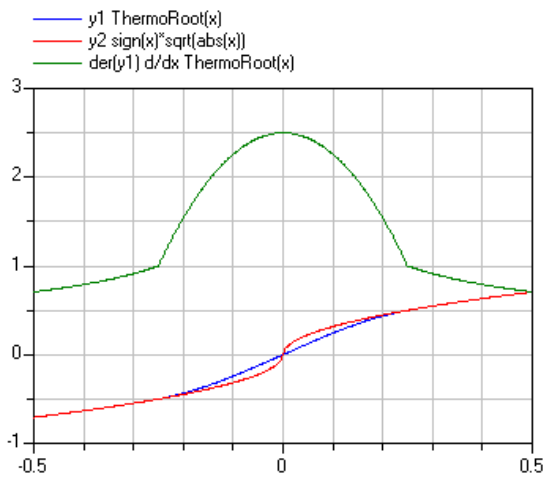

Figure 5: The ThermoRoot library function

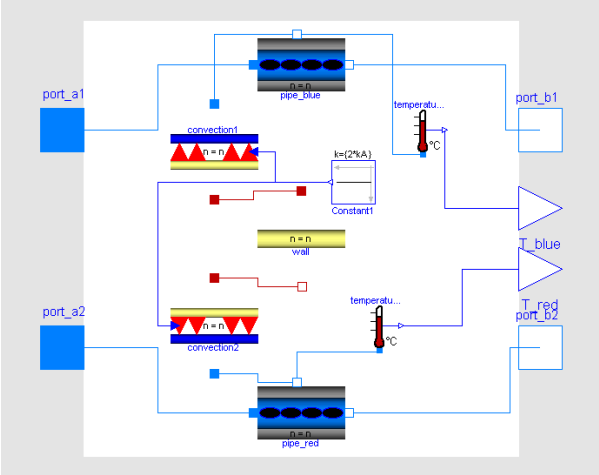

Figure 6: The SimpleHeatExchanger model

the function is continuously differentiable, the automatic event generation can be suppressed by using the `noEvent()` function.

5.2 The Interfaces folder

The library uses mainly the container models which are available in Modelica.Fluid. However these are not always sufficient, so additional interfaces must be provided. This applies especially for the discretized models, since Modelica.Fluid does not offer vectorized interfaces yet.

5.3 The Components folder

The Components folder contains the subfolder BaseComponents and SubComponents. The first contains abstract base models for components, the second contains component models, which are used only within other components. To give an idea of the level of implementation, some component models are discussed more explicitly.

5.3.1 The LongPipeS model

Hence Modelica.Fluid does not contain a useful model of a discretized pipe, a provisional model was implemented, which consists of a variable number alternating successively arranged control volumes and throttling devices. These two components are available as JunctionVolume and ShortPipe in Modelica.Fluid. In contrast to the more sophisticated implementation in the ThermoFluid library only the stationary impulse balance is implemented. For convenience, the model has an additional n-dimensional fluid connector. Using this connector, mass or heat can be transferred to each individual control volume.

5.3.2 The SimpleHeatExchanger model

The simple model of a heat exchanger consists of two LongPipeS models, the model of a massless wall and two very simple convection models. The models of the wall and of the convection are implemented in the subfolder SubComponents and can be replaced easily, if e.g. the thermal capacity of the wall has to be considered. On the other hand, the designs for heat exchangers are so various that generally appropriate abstract models can hardly be indicated.

5.3.3 The SimpleCompressor and SimpleExpander models

For compressors and exhaust gas turbines abstract base models are implemented. The models consider the volumes, the heat capacities and the rotating masses of the machines. The individual behavior of a machine is usually available as characteristic diagrams of mass flow and efficiency. These characteristics are implemented as replaceable class parameters, so that arbitrary machines can be modeled. The characteristic diagrams can take off either only the stationary or also the dynamic behavior of the machine. In most cases only stationary characteristics from measurements are available.

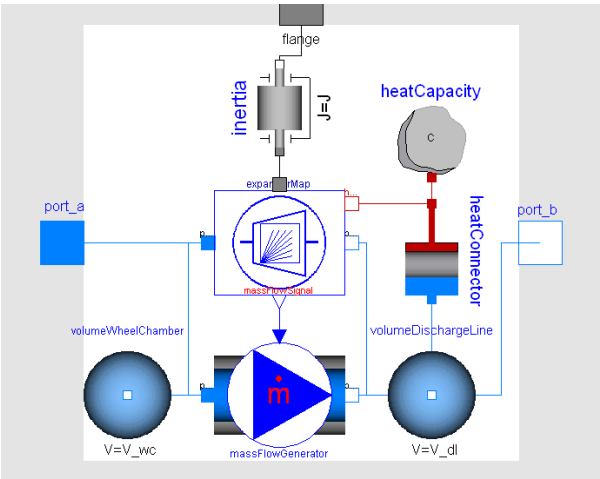

Figure 7: The BaseCompressor model

6 Medium properties

For the computation of the medium properties the Modelica.Media library is used. This library does not contain any model suitable for fuel cell systems computation yet. A model for humid air is needed, whereas both, the humidity and the gas composition are variable. The implementation of such a property model is already possible within the Modelica.Media library, this, however, did not succeed due to general problems concerning the mixture models. As a temporary solution the single component model SimpleAir is used, whereas the restriction of the temperature range was waived. This model is well suitable for real-time simulation in particular because of the simple implementation with constant heat capacity. With the exception of the compressor outlet the temperature of the air remains below 100 °C. In order to simulate the humidification and the drying process of air, a crude workaround is implemented into some component models. The heat of the condensation is computed directly in the component model and the absolute humidity of the medium is passed on as signal to the following component.

7 Simulation of the model

The model which is shown in figure 1 consists of 1761 unknowns, 704 time-varying variables and 31 continuous time states. To enable offline tests, some simple controllers were added to the model. Thus a system startup was simulated using the Dassl variable step solver. The timetable of the system startup is shown in table 2. In figure 8, 9, 10 some results are arranged. For simulation of the startup process with the Dassl solver 1.01 s CPU time was used on a Intel Centrino 1400 MHz. During the entire starting process 16 state events occurred.

time	action
t = 0 s	The desired pressure ratio is set to 2.2, compressor and turbine starts.
t = 5 s	The stack begins to deliver heat and water.
t = 7 s	The recirculation valve begins to open.
t = 8 s	The recirculation valve reaches 25 % opening.
t = 10 s	Simulation stops.

Table 2: Simulation of the system startup

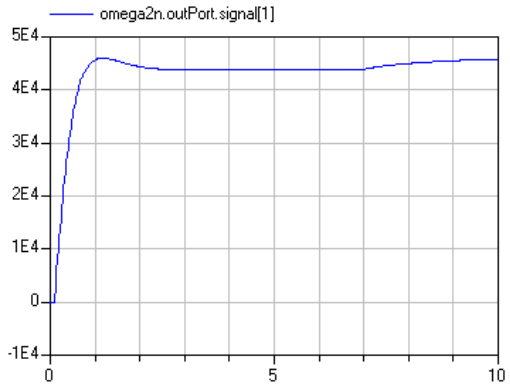

Figure 8: Simulation of system startup: angular velocity of common shaft

For comparison the same model was simulated using the mixed implicit/explicit Euler inline solver. The startup with angular velocity at zero is not possible in this case due to a division by zero. In figure 11 and 12 the deviations between the two simulation runs are shown. The difference between the simulation results is with exception of the very beginning less than approximately 2 %. Larger differences occur in the case of fast changes of the system states.

8 Real-time simulation on the dSPACE HIL target

The system model (figure 2) is inserted into a simple Simulink wrapper model (figure 3) and compiled with the help of the MATLAB Real-Time Workshop for a dSPACE target. The model runs with a stepsize of 2 ms on a dSPACE DS1005 PPC board in real-time. On the target, a certain number of overruns must be allowed, since in particular during the startup phase several overruns arises. Note, such adjustment is not pos-

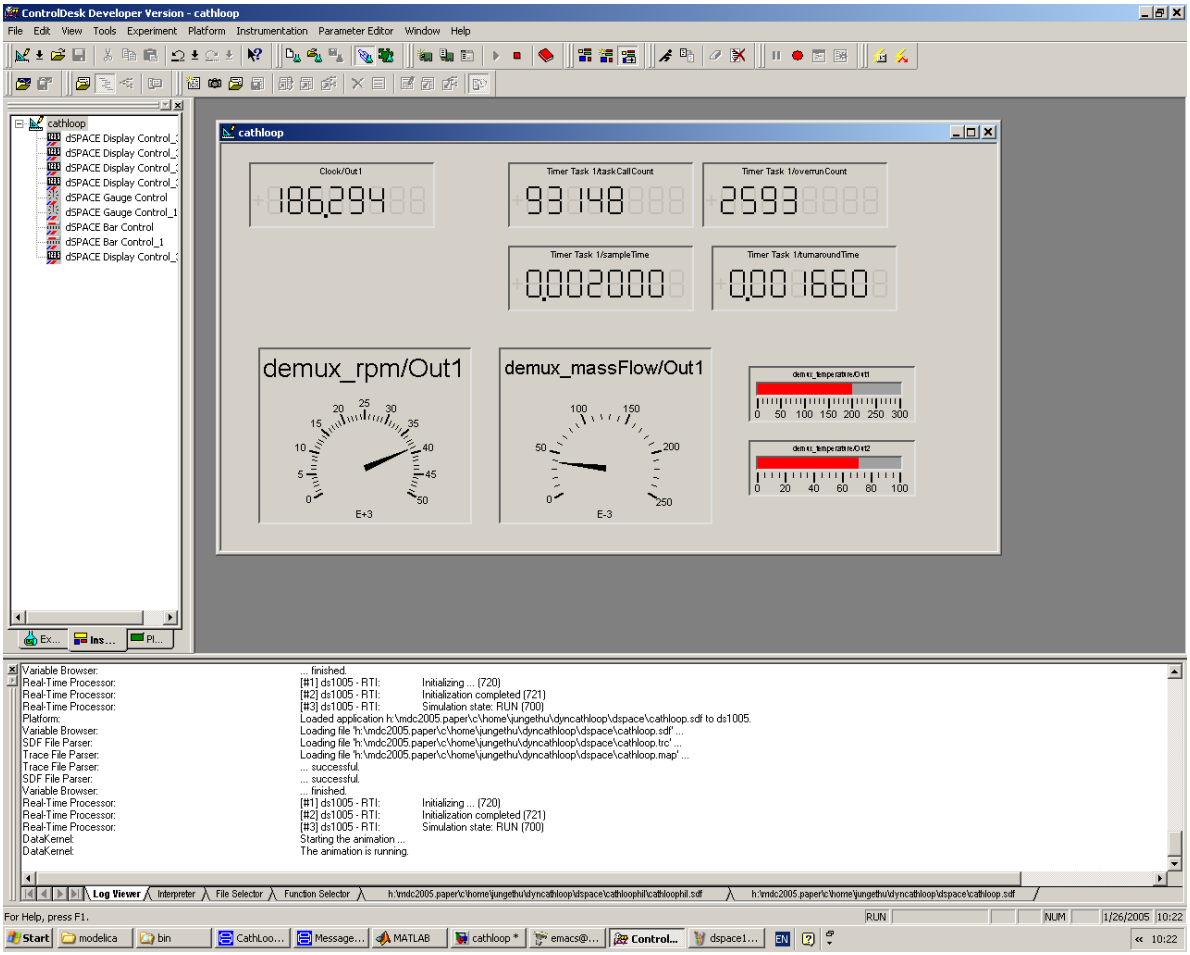

Figure 13: Screenshot of dSPACE ControlDesk with running simulation on a DS1005 target

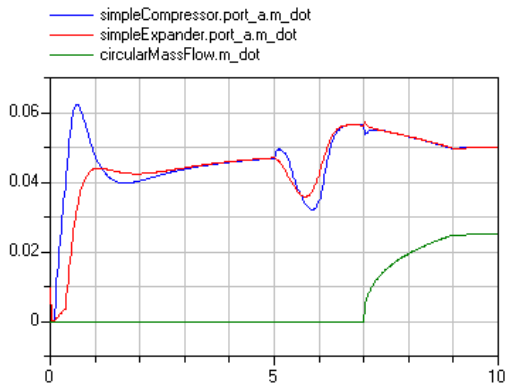

Figure 9: Simulation of system startup: mass flow rate through compressor, exhaust gas turbine and recirculation valve

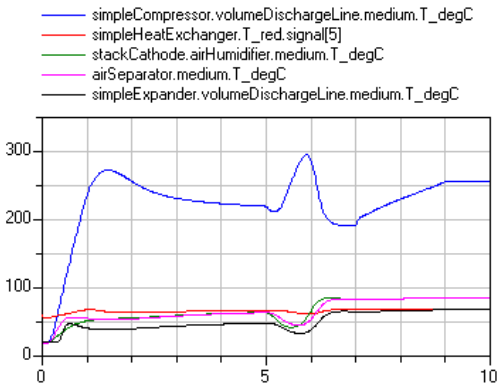

Figure 10: Simulation of system startup: air temperature at humidifier and separator, compressor, exhaust gas turbine and heat exchanger outlet

sible on any arbitrary real-time hardware. On various real-time platforms an overrun is a fatal error which aborts the simulation. In figure 13 a screen shot of the running simulation is shown. The two gages show the number of revolutions of the common shaft and the mass flow through the compressor. Right beside the temperature in the compressor discharge line and in the outlet of the fuel cell are shown. Above the task counter, the number of overruns, the sample time and the actual turnaround time are indicated. The simulation clock is shown in the upper left corner. Although

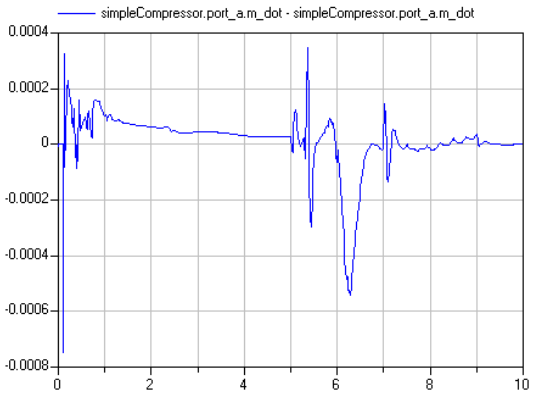

Figure 11: Comparison of simulation results using Dassl solver and fixed-step inline integration: difference of compressor mass flow

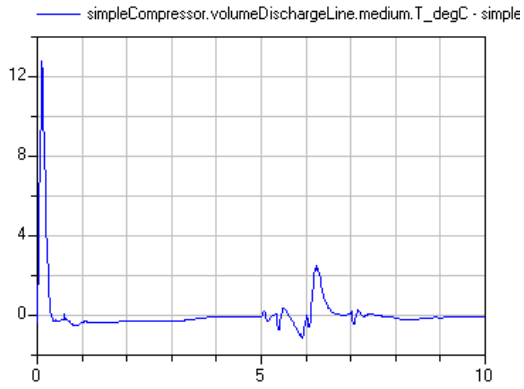

Figure 12: Comparison of simulation results using Dassl solver and fixed-step inline integration: difference of compressor outlet temperature

the mean turnaround time is more than 75 % (ca. 1.5 - 1.8 ms) of the sample time, overruns are rare after the startup phase.

9 Conclusion

On basis of a relatively simple model substantial requirements for real-time modeling of fuel cell systems in Modelica were worked out. Using the Dymola inline integration approach it is possible to use Modelica models in the HIL simulation on dSPACE hardware.

The generation and compilation of the target code with the help of the MATLAB Real-Time Workshop is a complex and expensive solution. Direct generation of the target executable without any MATLAB tools is already possible, but should be better supported by Dymola.

References

[1] Hilding Elmqvist, Sven Erik Mattsson, Hans Olsson. New Methods for Hardware-in-the-loop Simulation of Stiff Models. In: Proceedings of the 2th Modelica Conference 2002, Oberpfaffenhofen, Germany, Modelica Association, 18-19 March 2002.

[2] Hilding Elmqvist, Sven Erik Mattsson, Hans Olsson, Johan Andersson, Martin Otter, Christian Schweiger, Dag Brück. Real-time Simulation of Detailed Automotive Models. In: Proceedings of the 3rd Modelica Conference 2003, Linköping, Sweden, Modelica Association, 3-4 November 2003.

[3] Hilding Elmqvist, Hubertus Tummescheit, Martin Otter. Object-Oriented Modeling of Thermo-Fluid Systems. In: Proceedings of the 3rd Modelica Conference 2003, Linköping, Sweden, Modelica Association, 3-4 November 2003.

[4] Dymola User's Manual Version 5.1a. Dynasim AB, Research Park Ideon, SE-22370 Lund, Sweden.

[5] Peter Fritzson. Object-Oriented Modeling and Simulation with Modelica 2.1. John Wiley & Sons, Inc.

[6] Peter Treffinger, Martin Goedecke. Development of Fuel Cell Powered Drive Trains With Modelica. In: Proceedings of the 2nd Modelica Conference 2002, Oberpfaffenhofen, Germany, Modelica Association, 18-19 March 2002.

[7] Hubertus Tummescheit, Jonas Eborn, Falko Wagner. Development of a Modelica Base Library for Modeling of Thermo-hydraulic Systems. Modelica 2000 Workshop, Lund, Sweden. http://www.modelica.org/events/workshop2000

[8] Wolfgang Wagner, Alfred Kruse. Properties of water and steam. Berlin, Springer Verlag 1998.