Fluid Simulation and Optimization using Open Source Tools

Kilian Link, Stephanie Vogel — Siemens AG
Ines Mynttinen — Technical University Ilmenau
kilian.link@siemens.com, vogel.stephanie@siemens.com, ines.mynttinen@tu-ilmenau.de

Abstract

Many open-source (OS) tools exist in the Modelica universe. OS tools have the benefit that they are freely available and fit well to Modelica as a non-proprietary language. However, the industrial usage of these tools seems to be very limited so far. Despite this fact, fluid modeling is possible if restrictions are taken into account.

In this contribution, we propose to use benchmark models for the systematic investigation of the accuracy and the performance of Modelica OS tools. The present implementation circumvents the limitations of OpenModelica making it possible to simulate the model system despite existing restrictions. Beside simulation tasks Modelica-based optimization is possible using the OS tool JModelica. However, care must be taken with respect to the model features. In particular instantaneous transitions within the system dynamics, such as phase transitions, switching of valves with discrete behaviour or flow reversal represent a severe obstacle for optimization. In this article, we present a parameter estimation problem including instantaneous changes of the flow direction. In addition, an example of model predictive control (MPC) for a control task difficult to solve with conventional methods is shown.

Keywords: Fluid Simulation; Optimization; OS Modelica Tools

1 Introduction

Modelica is the preferred modeling language for dynamic simulations within Siemens Energy [1] due to its applicability for multi-domain modeling of physical systems and the high degree of maintainability of Modelica models. The Modelica Libraries Modelica.Media and Modelica.Fluid provide basic elements to model pipe networks including, e.g., economizers, super heaters and evaporators which are essential parts of each power plant. However the flexible approach of Modelica.Fluid makes it unsuitable for daly buisness in a well defined application area. Thus well proven models of these components exist in the in-house library SiemensPower.

The commercial tool Dymola is used for modeling and simulation. The alternative tool OpenModelica [2] is an OS Modelica-based modeling and simulation environment intended for industrial and academic usage, which has the large benefit that it is freely available and fit well to Modelica as a non-proprietary language. However, the tool support for fluid modeling is limited due to some advanced Modelica features, e.g. the usage of Modelica.Media and Modelica.Fluid. Despite this fact, fluid modeling is possible if the functions missing in OpenModelica are called from external libraries. In order to measure the quality of OS Modelica tools compared to the established commercial software, e.g. Dymola with respect to the accuracy and the performance benchmark models are needed. In this way the systematic investigation of models with increasing size and complexity can reveal bottlenecks and shortcomings in OpenModelica. To go beyond simulation applications towards optimization, the Modelica-based open source platform for optimization, simulation and analysis of complex dynamic systems JModelica [3] is the most preferable choice. The main objective of the project is to create an industrially viable open source platform for optimization of Modelica models. The three-level structure of the user interface is probably its main advantage, since it allows for convenient implementation of user specified applications. The issues of compilation, data processing, setting up the algorithm and starting the optimization are well addressed in the Python script file. Furthermore, the Python script file serves to store the data and to do some customized plotting. These capabilities for Python scripting considerably reduce the effort to implement user applications. By means of the Optimica extension to Modelica, the optimization problem itself is formulated at the middle level implementing the objective function and the constraints using the special class optimization. At the lowest level of the JModelica user interface the dynamic model is defined. We used JModelica for solving the parameter estimation problem of a hybrid dynamic system as well as an off-line model predictive control (MPC) problem presented in sections 3 and 4, respectively.

2 Simulation of Fluid Models with OpenModelica

Fluid Modeling based on OpenModelica is not very common so far. This is mainly due to the following restrictions still present in OpenModelica. As mentioned above, the support of Modelica.Fluid and Modelica.Media is limited. On the other hand, the performance of the solver (the so called back-end) is very poor, i.e. solving real life problems is not possible up to now. However enormous improvements are on the way supported by joint efforts in the OPENPROD [11] project. In order to evaluate the improvements, the availability of benchmarks and realistic test cases is a natural and essential first need. The size and complexity of these benchmark models should be easily adaptable. Furthermore, the model is desired to be valid in different phase regions, since phase transitions are crucial in fluid dynamics. In addition, it is useful to build up the model from a set of components, which are also well established in real technical systems to facilitate the extension of the model from a sandbox example to real world application.

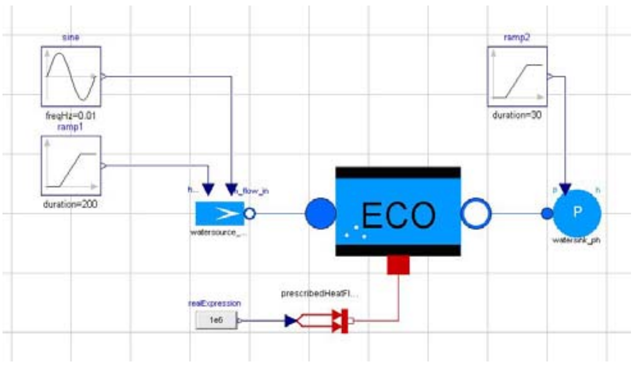

Figure 1: Heated pipe model

Figure 1 shows a model which can be used as such a benchmark. The central part of the model is a heated pipe which is connected to a water source and a heating element. The liquid channel of the pipe is discretized along the flow direction. The connections between these elements are represented by nodes.
The left hand node is connected to the water source which supplies the liquid flow. The heated metal wall of the pipe is modeled as cylindrical tube with the number of layers L in radial direction. The constant heat flow is distributed equally over all axial elements.

This quite simple model is very well suited as a benchmark model for the following reasons:

- The problem size can be easily adapted by changing the parameters for spatial discretization, see Table 1.
- Dynamic changes in the mass flow, enthalpy and pressure boundaries can be utilized to run through different phase regions.
- Based on basic components (i.e. tube plus boundaries) more complex models could be set up easily.

Discretization Parameters: N – number of nodes in flow direction L – number of tube wall layers	Continuous states
N = 3, L = 3	13
N = 10, L = 3	41
N = 30, L = 3	121
N = 200, L = 3	801
N = 100, L = 6	701

Table 1: Number of states depending on the spatial discretization of the heated pipe model

Although this model is very simple, some features of Modelica.Media which are not yet implemented in OpenModelica 1.6 are needed. To circumvent this problem, the model is rewritten such that direct functions compute all water properties without the use of the Media package of the Modelica Standard Library. The necessary water-steam functions have been substituted by external function calls of the TTSE (Tabular Taylor Series Expansion) [4] library. The TTSE uses a table of stored water properties and derivatives calculated with IAPWS (International Association for the Properties of Water and Steam) with pressure and enthalpy (p, h) or density and enthalpy (ρ, h) as variables. On each cell, the thermodynamic properties of water and steam are computed using the Taylor series expansion. In this way, TTSE offers fast computation with an acceptable accuracy for dynamic simulation. For the first investigation a small model corresponding to the first row in Table 1 has been used. The results from the well proven modeling environment Dymola in version 7.4 have been compared to those from OpenModelica version 1.6. In both tools 'dassl' with a tolerance of 1e-6 was used.
The comparison of the results shows that they are nearly identical. This suggests that the solver seems to solve an identical problem in both cases. However, the differences in performance are huge. While Dymola solves the problem in some milliseconds, OpenModelica spends several minutes to solve the

problem. Facing that enormous difference running larger models does not make much sense.

Primary investigations are carried out to identify the reasons for the poor performance of OpenModelica 1.6.0.

The pipe model was simulated with an increasing number of continuous states depending on the nodes in flow direction. Different simulation options were used to identify the reasons for the poor performance.

OpenModelica 1.6.0 uses 'dassl' as standard solver. Since 'dassl' is so slow, 'dassl2' has been tried. This solver is considerably faster than 'dassl', but it still takes at least seven times as long as Dymola 7.4 using 'dassl'. Table 2 shows the simulation time depending on simulation method. For higher discretization in the flow direction 'dassl2' is at least 100 times faster than 'dassl' generating the same results.

N (number of nodes)	solver	Simulation time
3	dassl	289.29 s
3	dassl2	2.52 s
10	dassl	803.81 s
10	dassl2	6.92 s
19	dassl	1523.63 s
19	dassl2	16.04 s

Table 2: Simulation time depending on solver and discretization

All test cases use the 'plt' output format, which is default and currently the only format capable of using plot functions. For testing it is better to use output format 'bin' to speed up calculation. For N>19 the compilation failed with an internal OpenModelica error. Probably the array sizes are too large.

We set up a special test case for external function calls to find out whether they cause considerable difference in the simulation time. We can exclude that the differences between Dymola 7.4 and OpenModelica 1.6.0 depend on external function calls, sincethe simulation time is quite similar for both simulation enviroments.

3 Parameter Estimation for a Hybrid System using JModelica

Parameter estimation is an important issue in many fields of industrial engineering [1], since it allows for efficient adaptation of system models.

Parameter estimation aims at extracting the best values of parameters determining the dynamics of the system under consideration, based on a series of measurements $x_{j\ell}^{(m)}$ of several state variables x_j , j = 1,…,M at different time points t_ℓ , ℓ = 1,…,N. Due to measurement error, the estimated parameters are subject to some uncertainty. Assuming that the measurement error is uncorrelated and normally distributed with variance σ_j^2, model parameters can be estimated by minimizing the weighted least-squares function

$$J(p) = \sum_j^M \sum_l^N \frac{\left(x_j(t_l) - x_{jl}^{(m)}\right)^2}{\sigma_j^2} \qquad (1)$$

subject to the DAE system representing the system dynamics as equality constraints and variable boundaries as inequality constraints.

In this study we consider a parameter estimation problem for a hybrid system. Hybrid systems possess a mixed continuous and discrete behavior due to instantaneous mode transitions. In fluid systems the latter arises very frequently as a result of valves with discrete behaviour, phase transitions or flow reversal. Solving these kinds of optimization problems remains a challenging task and the use of available solvers is limited. The major difficulties lie in the discontinuous function values and gradients.

To overcome the difficulties we studied reformulation methods for hybrid systems. In these methods the so-called switching condition Ψ determines the value of a newly introduced continuous switching variable $\varphi(\Psi)$. In order to force this variable to meaningful values to guarantee at least an approximate instantaneous switch either a relaxation or a penalization method can be used. In this study we apply the Smooth Step Function (SSF) Approach with

$$(2) \quad \varphi(\psi) = \frac{1}{1+\exp(-\tau\psi)}$$

as a relaxation method with relaxation parameter τ and the Penalization of Incomplete Switching (PICS) as a penalization method [6]. In order to examine the capabilities of reformulation methods in parameter estimation for hybrid systems, we consider a tank system similar to those used in [7], [8], and [9].

The system consists of three tanks in a row connected to each other (Figure 2). There are inflows Q_{zi}, i = 1, 3 to the left and the right tank. The parameter estimation problem can be stated as:

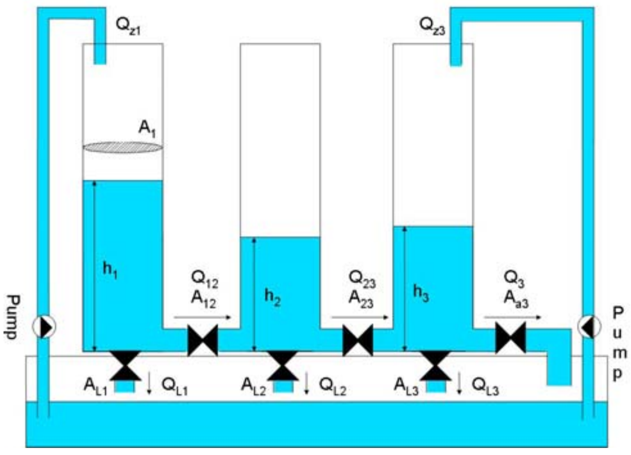

Figure 2: Three-tank system

$$\min_{A_{12},\,A_{23}} \; J(A_{12},A_{23}) = \sum_{j}^{N}\left(h(t_j) - h_j^{(m)}\right) \qquad \text{(2a)}$$

s.t.

$$A_1\dot{h}_1 = Q_{z1} - Q_{12} - Q_{L1} \qquad \text{(2b)}$$

$$A_2\dot{h}_2 = Q_{12} - Q_{23} - Q_{L2}$$

$$A_3\dot{h}_3 = Q_{z3} + Q_{23} - Q_{L3} - Q_3$$

$$Q_{ij} = A_{ij}\,\varphi_{ij}\sqrt{2g\,|\psi_{ij}|} \qquad \text{(2c)}$$

$$\psi_{ij} = h_i - h_j,\ (i,j) = \{(1,2),(2,3)\}$$

$$-1 \le \varphi_{ij} \le 1 \qquad \text{(2d)}$$

The objective function (Eq. (2a)) is to be minimized subject to the dynamic model equations and bounds of the switching variable φ (Eq. (2b-d)). The dynamics of the tank levels h_i (i = 1, 2, 3) are given by the mass balance of the tanks (Eq. (2b)). The outflows Q_{Li}, Q_3 and the flows between the tanks are modeled by Torricelli's law (Eq. (2c)). In the original formulation, a sign function switches the direction of the flow between two tanks abruptly from +1 to −1 or vice versa, when the condition $\Psi_{ij} = h_i - h_j = 0$ is passed. Since the gradient of the flow diverges to infinity at this point, in our reformulation the sign function in Eq. (2c) is replaced by the switching variable φ. The correct switching behavior should be ensured by the relaxation or penalization methods mentioned above.

We used JModelica for solving the nonlinear optimization problem. Here we exploit the three-level structure of JModelica's front end as described above. JModelica does not allow optimization problems including instantaneous mode transitions since in this case the gradients of the objective function or the constraints needed by the optimization algorithm are expected to be not well defined. However, the relaxation and penalization methods used in this study lead to differentiable objective functions and constraints. The reformulation method can easily be implemented in JModelica.

Our aim is to estimate via minimization of the objective function (Eq. (2a)) the flow parameters A_{ij} based on (simulated) measurement data $h_\ell^{(m)}$, ℓ = 1...10 of the tank levels taken equidistantly within the time horizon $(t_0, t_f) = (0, 20)$ s. The data are generated via simulation of the original model with added Gaussian noise. The optimal state trajectories found for the parameter estimation problem are shown in Figure 3).

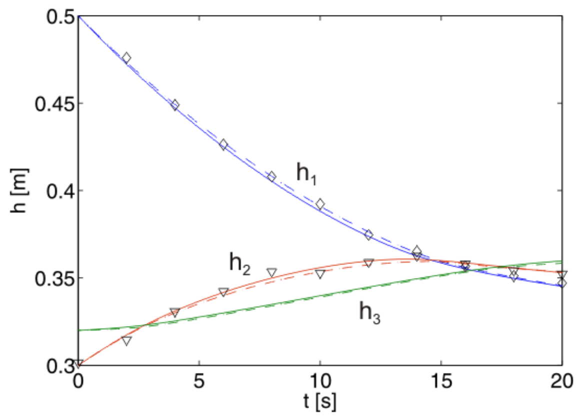

Figure 3: State trajectories SSF (solid) and PICS (dashed) as well as the measurements of h_1 (diamonds) and h_2 (triangles)

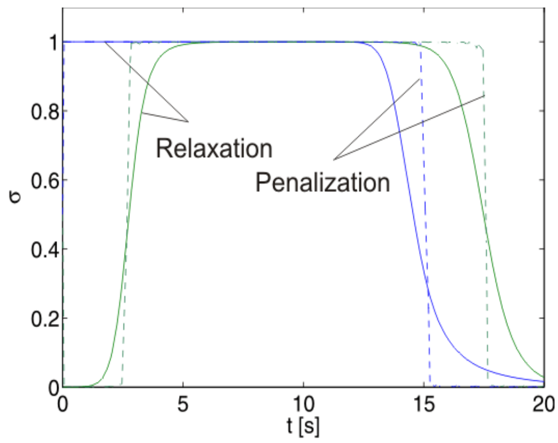

Figure 4: The corresponding switching variables φ_{12} (blue) and φ_{23} (green)

They agree quite well with each other. In particular, the crossing points of the levels h_1 and h_2 and the levels h_2 and h_3 nearly coincide. This reflects the fact that the correct switching behavior is obtained in both cases as shown in Figure 4. It can be seen that using the relaxation method the switch is smooth and rather slow, which apparently has almost no impact on the trajectories. In contrast, when using the penalization approach the switch takes place almost instantaneously. Table 3compares the optimal parameter values estimated by the two methods. With penalization we obtain a very good value for the parameter A_{12}, but the deviation of A_{23} from the exact value is considerable. This is mainly due to the fact that the objective function is much more sensitive to A_{12} than to A_{23}. The relaxation method results in moderate deviations for both parameters. In summary, both approaches provide reasonably accurate results.

	exact	Relaxation	Penalization
A_{12} [10^{-5}m^2]	6.0	6.51	6.06
A_{23} [10^{-5}m^2]	2.0	2.29	2.98

Table 3: Optimal parameter values

We also studied the ability of the SSF algorithm to cope with random errors in the data set, which is inevitable in real data acquisition. The variance of the measurement is varied in the range $\sigma_M = [0.5, 10] \cdot 10^{-4}$ m. The parameter estimation is carried out for 50 series of h_2 for each σ_M and the mean parameter values as well as their variance σ_p are achieved (see Figure 5).

It can be seen that the mean values of the parameter stay constant over a wide range of random errors. Obviously, a higher variance of measured data leads to a higher variance of the estimated parameter value. A strict proportionality of σ_M and σ_p is expected in the case of the measured quantity (here h_2) linearly depending on the parameter (here A_{12}).

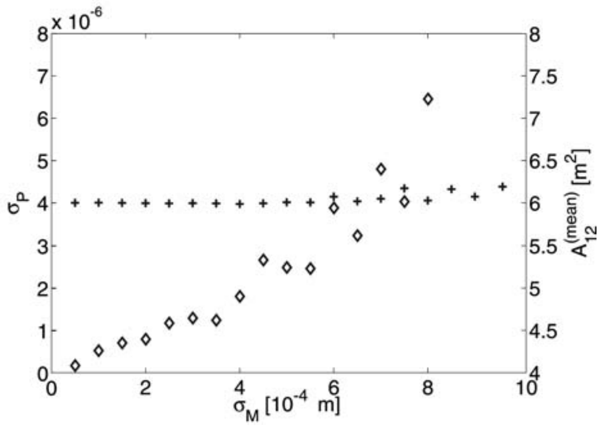

Figure 5: Mean parameter value A_{12} (crosses) and corresponding variance σ_p (diamonds) in dependence on the variance of measurement σ_M.

4 Nonlinear Model Predictive Control using JModelica

Nonlinear model predictive control (NMPC) is an advanced technique to solve challenging optimal control problems. In this method, the optimal control problem is formulated as constrained dynamic optimization problem, where the constraints are given by the dynamic model of the plant and the process restrictions. This problem is solved for the so-called prediction horizon T_p (see Figure 6). The resulting optimal controls within the so-called control horizon T_c are applied to the plant. At $t = T_c$, the time horizon of the optimal control problem is moved to this new initial point. The measurement of the present plant state serves as feedback. The initial conditions are accordingly updated and the problem is solved again. Since the optimal control problem is solved once per move of the time horizon, T_c is the CPU time available to solve the problem.

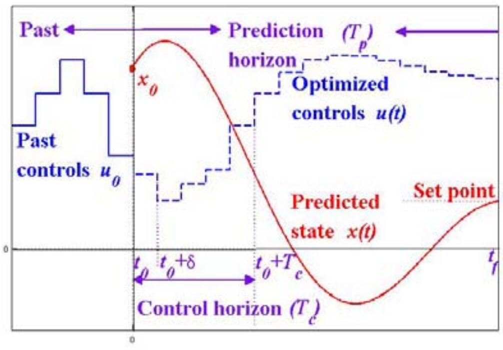

Figure 6: Principle of NMPC [9]

It is straight forward to carry out the NMPC based on power plant models primarily developed for dynamic simulation applications. The choice of test cases will naturally focus on optimal control problems which are difficult to solve by conventional controllers. Besides that, one has to take into account the size and complexity of the plant model as a limiting factor with respect to the computational effort.

In this study, the temperature control of live steam with intermediate water injection (see Figure 7) was chosen as a test case, since even this system although quite simple, is difficult to control.

The aim of the NMPC is to reach and to hold the setpoint. Thus, the objective function to be implemented in the Optimica class of JModelica

$$\min_{m(t)} \int_{t_0}^{t_{end}} (T_{ref} - T_{mix}(t))^2 \, dt$$

contains the squared deviation between the temperature set-point T_{ref} and the current temperature $T_{mix}(t)$ of the steam cooled by water. The objective is subject to the model equations and a maximum constraint for the injected water mass flow $m(t)$. The dynamic model is composed of proven components already existing in the library SiemensPower. To meet the model requirements of JModelica, the functions of the water and steam properties have been approximated.

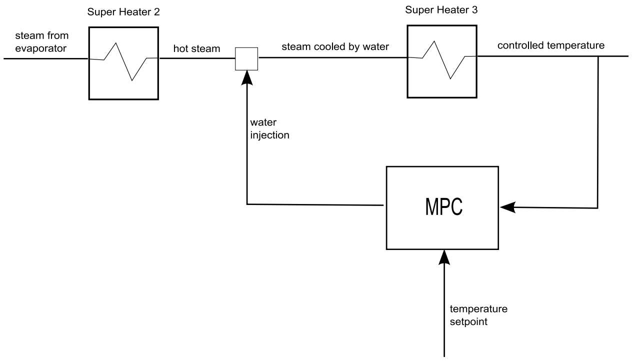

Figure 7: Temperature control of live steam with intermediate water injection.

For now, the NMPC problem has been implemented off-line. Instead of the real plant, the optimal control is applied to the plant model which simulates the behavior of the real process. This simulation provides the "measurements" of the states required for the feedback loop as described above.

Figure 8 shows the temperature profiles $T_{mix}(t)$ controlled by the NMPC (red) and a conventional controller typically implemented on present-day plants (blue). After some seconds the temperature controlled by NMPC reaches the set-point and is capable of maintaining this withnegligible deviations. In contrast, the classical control is not able to avoid remarkable deviations from the set-point. Clearly, the NMPC is far superior to the conventional control and its on-line implementation should be seriously considered. However, the real time criterion has to be met. Running the application on a standard desktop it is not yet satisfied in all cases (see Figure 9Figure 8).

In the example the control horizon is set to $T_c = 1$s, but the computation of the control profiles for one step needs more than one second, in particular for the former time horizons up to $n = 27$. As can be seen the computational effort for these early time horizons is higher than that of subsequent ones. This is presumably due to the fact that subsequent NMPC steps the temperature is already close to the set-point and the control input needs only slight modifications. Since for this quite simple model the computation time already represents a limitation, the computational effort will presumably be too large for more complicated or faster systems. Hence, in many cases some performance improvements will be needed to run the applications on-line. An efficient algorithm is presented in [8] but it requires a more elaborate implementation which may be realized in the future.

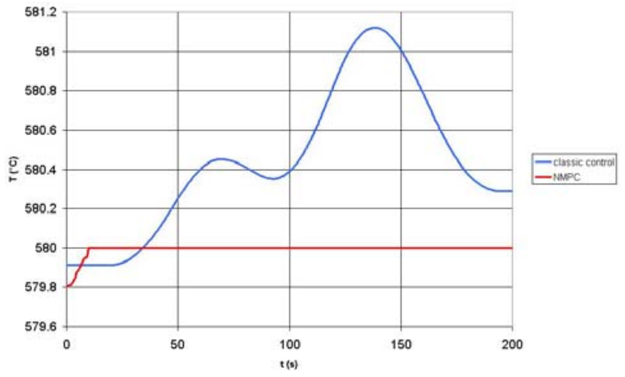

Figure 8: Temperature controlled by the NMPC and classic temperature control.

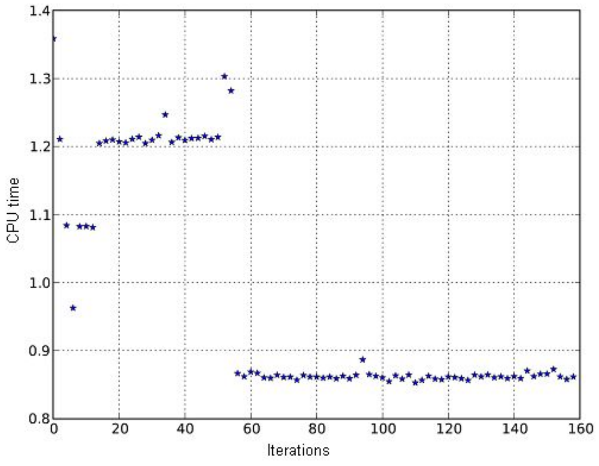

Figure 9: CPU time needed to solve the dynamic optimization problem for each moving prediction horizon with initial time $t_{init}=nT_c$.

5 Conclusions

In order to develop benchmark and realistic test cases, we implemented simple models for the systematic investigation of performance improvements of the OpenModelica tool chain. Some primary investigations have been carried out. Besides that the restricted functionality of OpenModelica with respect to water and steam properties has been supplemented by the usage of TTSE.

Parameter estimation and nonlinear model predictive control for fluid systems have been solved.

In fluid dynamics, we often have to deal with valves with discrete behaviour, flow reversal and phase transitions which considerably complicate the optimization of such problems. To overcome these difficulties we studied reformulation methods for hybrid systems and showed their capability of tackling the problem of discontinuity of state trajectories and gradients which occur due to instantaneous transitions.

For nonlinear model predictive control a simple offline example has been implemented. The model-based control proved to be superior to the recently applied classic control. Thus, the online application of the described solution should be seriously considered.

The support by German Ministry BMBF (BMBF Förderkennzeichen: 01IS09029C) within the ITEA project OPENPROD is gratefully acknowledged.

References

[1] www.energy.siemens.com

[2] www.openmodelica.org

[3] www.jmodelica.org

[4] Miyagawa, K., Hill, P.G., Tabular Taylor Series Expansion (TTSE) Method Based on IAPWS-IF97 Double Precision Enthalpy-Density Version, IAPWS Task Group TTSE, 2001-07-13

I. Weber, K. Knobloch, I. Kodl, H.-J. Kretzschmar, Test Report Of „Documentation and Software of TTSE Method applied to IAPWS-98 as an Example", TTSE Evaluation Task Group, July 2002

[5] Tummescheit H. Design and Implementation of Object-Oriented Model Libraries using Modelica. Lund, Sweden: PhD thesis, Department of Automatic control, Lund Institute of Technology, 2002.

[6] Mynttinen, I. and Li, P.: A Reformulation Scheme for Parameter Estimation of Hybrid Systems. ESCAPE 2010, submitted

[7] J. Till, S. Engell, S.Panek, O. Stursberg , "Applied hybrid system optimization: An empirical investigation of complexity," *Control Eng. Pract.*, vol. 12, pp. 1291–1303, 2004.

[8] B. T. Baumrucker, L. T. Biegler, "MPEC strategies for optimization of a class of hybrid dynamic systems," *J. Process Contr.*, vol. 19, pp. 1248–1256, 2009.

[9] J. M. M. Tamimi. Development of Efficient Algorithm for Model predictive Control of Fast Systems, PhD thesis, Technical University Ilmenau, 2011

[10] JModelica 1.4.0 User Guide, http://www.jmodelica.org/api-docs/usersguide/1.4.0, Modelon AB 2010.

[11] www.openprod.org