From system model to optimal control - A tool chain for the efficient solution of optimal control problems

Manuel Gräber[1] Jörg Fritzsche[2] Wilhelm Tegethoff[3]

[1]TLK Energy GmbH, Germany, manuel.graeber@tlk-energy.de
[2]Volkswagen AG, Germany
[3]Institut für Thermodynamik, TU Braunschweig, Germany

Abstract

Based on a specific application example - the thermal management system of an internal combustion engine - a toolchain is presented for formulating and solving of nonlinear optimal control problems. Starting from graphical modeling of the thermal management system with the Modelica library TIL, the model is exported to the standardized model exchange format Functional Mock-up Interface (FMI). Furthermore, it is imported to the optimal control software package MUSCOD-II. Python is used as scripting language for the problem formulation, the numerical solution and the processing of results. By using FMI as an interface, models from any simulation and modeling tools can be used if there is an FMI model export and the models fulfill certain mathematical requirements (smoothness).

Keywords: Optimal control, Functional mock-up interface, thermal management, cooling system

1 Introduction

When developing control concepts or superior operating strategies, frequently the question arises, what is the theoretically best possible control of a system. Questions of this kind can be mathematically expressed as *Optimal Control Problems* (OCP) describe. What is special about this class of optimization problems is the dynamics of the controlled system. Contrary to static optimization problems, not a finite number of parameters are free for optimization, but trajectories of system inputs. Therefore, an OCP is an infinite-dimensional optimization problem, which usually cannot be solved directly. However, different mathematical methods exist to determine approximated numerical solution. Detailed introductions in the theory of optimal control can be found in (Bryson and Ho 1979) and (Betts 2001). The *Direct Multiple Shooting Method* according to (H. G. Bock and Plitt 1984) used in this article is explained in Section 3.

Although these and other specialized OCP algorithms have existed for a long time, they have not yet made it into the broad industrial application. An exception to this is the aerospace industry, in which OCPs have been solved for optimal trajectory planning for decades. The largest (in our opinion) obstacle to a widespread industrial use of optimal control is the necessary time and knowledge-intensive effort. Successful work with existing software requires a high degree of expert knowledge. According to our experience, the by far largest time effort in optimization projects cannot be seen in performing the actual optimization calculations, but rather in the modeling of the system under consideration. On the one hand, derivative-based optimization algorithms require a certain numerical model quality (differentiability) that go beyond the requirements of pure simulation algorithms. On the other hand, it is important to depict the correct positive and negative effects, the superposition of which determines the optimum. The modeling process is almost always iterative. Reliable results can only be produced by repeatedly interpreting optimization results and changing modelling details.

Based on the described experiences and observations, we suggest a tool chain in this article to use optimal control efficiently. The thermal management system of a combustion engine serves as a continuous example. Starting with the modeling of the controlled system in Section 2, the model is exported as FMU (Blochwitz et al. 2012) and imported into to the specialized optimization package MUSCOD-II (H. G. Bock and Plitt 1984; Diehl 2001; Leineweber et al. 2003). For the problem formulation, the numerical solution and the processing of results Python is used as scripting language.

Other Modelica-related optimal control projects are described in (Åkesson et al. 2010), direct collocation (Imsland et al. 2010), single shooting, and (Franke 2002), multiple shooting.

2 Modelling of the controlled system

The most important part in the successful work with optimal control is the dynamic model of the system under consideration. Mathematically, the system model

is described as a system of ordinary differential equations:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) \qquad (1)$$

Here $\boldsymbol{x}$ denotes the vector differential states, $\boldsymbol{u}$ the vector of inputs, $\boldsymbol{p}$ the vector of parameters and t is time. It is also possible system models with additional algebraic implicit equations. Since this is not supported by the current FMI 2.0 standard and due to better readability, we limit ourselves to explicit ordinary differential equations (ODE) of the form shown above.

Equation systems of this type are the mathematical basis of various industrially used system simulators such as Simulink or Dymola. In order to be able to exchange models between different simulators, the open standard Functional Mock-up Interface (FMI) was developed. We use FMI to link system models to optimization algorithms. Thus, it is possible to develop the system model of an optimum control problem in the modeling tool of choice. The only requirement is the availability of FMI exports.

Models that are suitable for simulation need not yet be suitable for the use of derivative-based optimizers. Discontinuities in the model equations can lead to poor or even failing convergence behavior. In practice, however, the theoretical mathematical requirements for models must not be completely satisfied. Even if the continuous differentiability of all model equations is not fulfilled, for example by the linear interpolation of characteristic fields, reasonable results can be achieved with derivation-based optimizers.

Thermal systems such as the thermal management system considered here, can be graphically modelled with the Modelica library TIL (Schulze 2013; Gräber et al. 2010; Richter 2008). Large parts of TIL are directly suitable for use in optimizers. This includes circuits with compressible liquids and ideal gas mixtures. Two-phase fluid circuits modeled with TIL are not yet suitable for optimization. However, current research deals with this topic.

Figure 1 shows the thermal management system as a TIL model. The coolant (blue) is pumped through the engine block by an electrically driven pump. There, waste heat from the combustion engine is added as time-dependent heat flow. The upper circuit through the heating heat exchanger is constantly flowed through. The lower circuit for heat dissipation to the environment can be connected via an electrical valve as required.

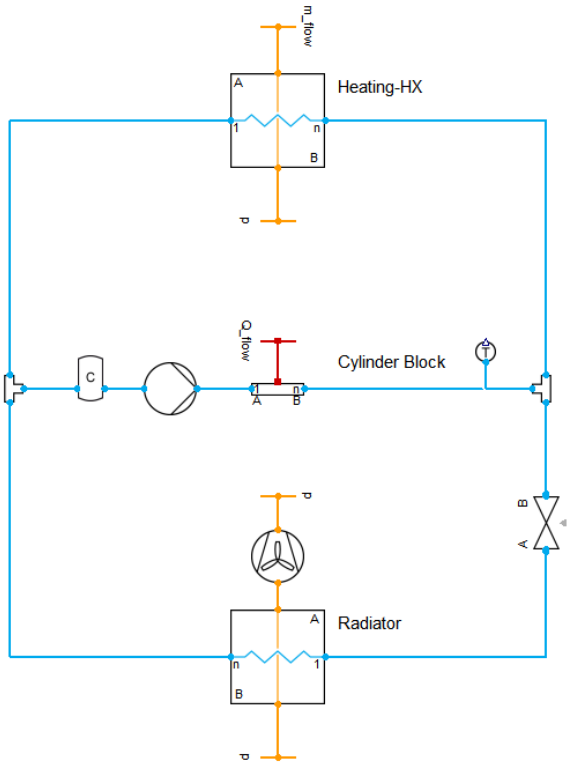

Figure 1. Sketch of the thermal management system. Screenshot of Modelica / TIL system models.

The manipulated variables of this system are:

- Water pump speed
- Cooling fan speed
- Opening degree of the valve

Both heat-exchangers are modelled according to the finite volume method with 5 discrete volumes. The system model has 36 differential states in total.

The primary control task is temperature control of the engine, which is achieved by demanding a setpoint of 90°C for the fluid temperature at the engine block outlet. Three manipulated variables and only one control variable leave two degrees of freedom. An open question is how to deal with these degrees of freedom. An obvious idea is to introduce the additional demand for the lowest possible energy consumption. Thus, the cost function to be minimized follows as:

$$C = \int_0^{t_\mathrm{f}} P_\mathrm{pump} + P_\mathrm{fan} + c(T - T_\mathrm{set})^2 \mathrm{d}t \qquad (2)$$

The electrical power consumption of pump and fan as well as a squared penalty term for setpoint deviations are integrated over a given period of time. The two control objectives can be weighted with the factor c. High values result in higher energy consumption but temperatures closer to the setpoint.

DOI 10.3384/ecp17132249

The input trajectories within the period under consideration are free for optimization. However, upper and lower bounds for all manipulated variables are taken into account:

$$\boldsymbol{u}_{\mathrm{lb}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\mathrm{ub}} \quad \forall\, t \in [0, t_{\mathrm{f}}] \tag{3}$$

In addition, the given initial state of the system model enters as equality constraint:

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{4}$$

The complete optimal control problem follows as:

$$\begin{aligned} &\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} C(\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}) \\ &\text{s.t.} \quad \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \quad \forall\, t \in [0, t_{\mathrm{f}}] \\ &\qquad \boldsymbol{u}_{\mathrm{lb}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\mathrm{ub}} \qquad \forall\, t \in [0, t_{\mathrm{f}}] \\ &\qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \end{aligned} \tag{5}$$

3 Numerical solution of optimal control problems

This section is based on (Gräber 2013) and attempts to explain the basic mathematical ideas behind the used numerical methods. For a deeper and more mathematical representation, references to further literature are given in several places.

Optimal control problems of the above-described form are not directly solvable by numerical methods. Considering the continuous trajectories sought as a set of infinitely many individual points, it becomes clear that an OCP is an infinite-dimensional optimization problem. Deriving analytical solutions is only possible for very simple subclasses. For most real problems, only an approximate numerical solution is possible.

Within the last decades, various methods have been developed to numerically solve optimal control problems. These can be divided into two large groups: indirect and direct methods. Indirect methods are based on *Pontryagin's Maximum Principle*. With the help of this necessary optimum condition, the OCP is analytically transformed into a boundary value problem with the original differential equations and additional adjoint equations. This boundary value problem can then be solved with various numerical methods. A frequently mentioned disadvantage of these methods is the difficult consideration of restrictions. Since it is necessary in many technical applications to limit state variables and controls to certain areas, this disadvantage is not insignificant.

Recent work deals almost exclusively with *direct* methods. The term comes from the fact that not a transformed problem, but directly the original OCP is used. By discretizing the trajectories, the infinite-dimensional OCP is approximated with a finite-dimensional *Nonlinear Program* (NLP). This NLP can then be solved with common numerical methods such as *Sequential Quadratic Programming* (SQP) or *Interior Point Method* (IP). Within direct methods, a distinction is made between sequential and simultaneous methods.

In direct sequential procedures, the control trajectories are described by piecewise defined functions – mostly polynomials. In the simplest case, the polynomials are of the order of zero, and the controls are piecewise constant functions over time. The coefficients of these polynomials are the free optimization variables. Using an ODE or DAE solver, the cost function can now be evaluated for given control trajectories and initial values. Coupled to an optimization algorithm, the OCP can be iteratively solved by repeatedly solving an initial value problem with different control trajectories. It has been shown, that particularly for ill-conditioned problems convergence and stability properties of such methods are not very good (Hans Georg Bock 1987; Albersmeyer and Diehl 2010).

Direct simultaneous methods go one step further and discretize not only the control but also the state trajectories. In the case of direct collocation, the trajectories of all state variables and controls are again described by piecewise defined functions. The continuous ODE is converted into a system of difference equations using a suitable scheme. This equation system is included as an equality constraint in the optimization problem. Leading to a very large but finite-dimensional NLP, which can be solved with conventional methods. In order to reduce the computation time, the special structure of the equation systems can be exploited. (Biegler 2007) provides an overview of current simultaneous methods.

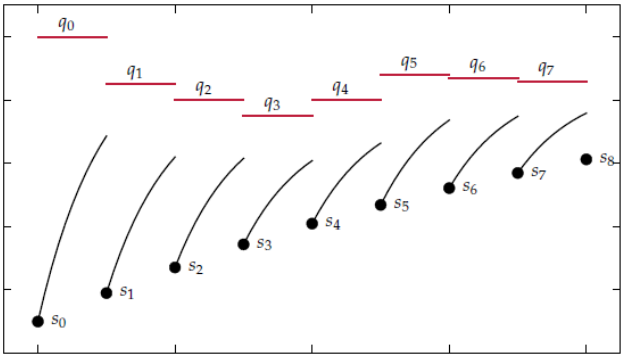

Figure 2. Multiple Shooting Method. Control trajectories (red) are discretized with piecewise constant functions and state trajectories (black) are discretized by solving independent initial value problems.

The direct multiple shooting method used in this work is usually seen as simultaneous method, but could also be interpreted as a mixed form between the sequential and simultaneous methods. Control trajectories are discretized analogously to the methods described so far. The state trajectories are also divided into individual sections. However, the path within these sections is not described by polynomials. Rather, initial values for the states are introduced at the nodes of the multiple shooting grid. At node *i*, these additional variables are designated as $\boldsymbol{s}_i$. Based on these initial values and the original ODE, the trajectories of the states are determined by solving several independent initial value problems. For an arbitrary choice of the initial values, the resultant total trajectories of the states have jumps, see Figure 2. Therefore, closing conditions are included in the OCP as additional equality constraints. The value of a state variable at the end of a section must be equal to the initial value of the next section.

If an identical discretization grid with *n* intervals is chosen for controls and states and the controls are parameterized piecewise constant with the values $\boldsymbol{q}_i$, the following NLP results from OCP (5):

$$\begin{aligned} \min_{\substack{\boldsymbol{s}_0,\dots,\boldsymbol{s}_n \\ \boldsymbol{q}_0,\dots,\boldsymbol{q}_{n-1}}} & \sum_{i=0}^{n} k_i(\boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}) \\ \text{s.t.} \quad & \boldsymbol{s}_{i+1} = \boldsymbol{x}_i(t_{i+1}; t_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}) \qquad \forall i \in \{0,\dots,n-1\} \\ & \boldsymbol{u}_{\text{lb}} \le \boldsymbol{q}_i \le \boldsymbol{u}_{\text{ub}} \qquad \forall i \in \{0,\dots,n\} \\ & \boldsymbol{s}_0 = \boldsymbol{x}_0 \end{aligned} \tag{6}$$

It should be noted that the solution of an initial value problem is behind the evaluation of the cost functions $k_i(\boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p})$ and the determination of the states at the end of an interval $\boldsymbol{x}_i(t_{i+1}; t_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p})$. In the solution of this NLP with derivative-based methods, it is of great importance to determine the derivatives of these functions with respect to the free optimization variables accurately and efficiently. This is a non-trivial task when using variable step size integrators. An extensive discussion of this topic can be found in (Bauer 1999) and (Albersmeyer 2010).

To illustrate the multiple shooting method, the discretization for a simple example is shown in Figure 2. On each shooting interval *i*, an independent initial value problem is solved with the initial value s_i and the constant control q_i. The figure shows the result of an optimization iteration, that has not yet converged. The violation of the matching conditions for the state variables is clearly visible.

4 Optimal Control of a Thermal Management System

This section describes optimization results for the thermal management system described in section 2.

The system model is graphically generated and parameterized in Dymola using the library TIL. With the Dymola FMI export functionality, the model is exported as FMU for Model Exchange 2.0. Control inputs must be declared as top level inputs in Modelica and variables, which are used in the cost function, as top level outputs.

The coupling of the FMU to the optimizer MUSCOD-II, and the complete configuration of the calculations is done in the Python language. The Python code used her is shown in Figure 3.

```
import fmi_muscod as fm

# create an OCP object
ocp = fm.OptimalControlProblem()

# Load FMU as system model
ocp.loadFMU('ThermomanagementSystem.fmu')

# Configure time: 600s divided into 60 intervals
ocp.setTimeHorizon(60, 600.0)

# Estimate start values, bounds and scales
# based on a forward simulation.
# Inputs are set constant to given values.
ocp.estimateSettings([50.0, 50.0, 0.3])

# Overwrite bounds for inputs
ocp.setLowerBounds(uBounds=[10.0, 1.0, 0.01])
ocp.setUpperBounds(uBounds=[100.0, 100, 1.0])

# Scale cost function
ocp.setScaleValues(cScale = 1e5)

# Define cost function (Lagrange type)
lval = '(x33 - 90.0)*(x33 - 90.0) + 0.001*y0'
ocp.setCostFunction(lval)

# Start optimization
ocp.solve()

# Plot results for some states
# directly in Python
ocp.plotResult([1, 33, 34, 35, 31])

# Export result to Modelica
# for detailed post-processing
ocp.exportResultToModelica()
```

Figure 3. Python code for the numerical solution of the optimal control problem.

The scenario considered is a 10-minute drive up a mountain pass after a cold start at 20°C. This means that relatively much engine waste heat is introduced into the cooling circuit, which in turn has to be dissipated to the ambient air. Due to the comparatively low uphill speed the cooling fan has to be used more extensively. The

DOI 10.3384/ecp17132249

optimum control trajectories are calculated as piecewise constant curves with an interval length of 10s. A simulation of the thermal management system with a simple control concept is used as a basis for comparison For this purpose, the pump, such as a mechanical water pump, is operated at a rotational speed coupled to the motor speed. The valve is controlled by a P-controller and a setpoint of 85°C for the coolant temperature. While the fan controls the same temperature with a PI controller to the desired setpoint of 90°C.

Figure 4 shows the optimum and simulated (with PI controllers) controls. Obvious differences are:

- Maximum pump speed in the first minute of the optimal solution
- The fan becomes active in the optimal solution earlier.

The first difference is only to be explained by the fact that the electrical power of the pump is used to heat the coolant. At the beginning all temperatures are at 20°C. In order to reach the setpoint of 90°C as quickly as possible, it is worth (in the sense of the cost function) to use the electrical power of the pump to heat up the system.

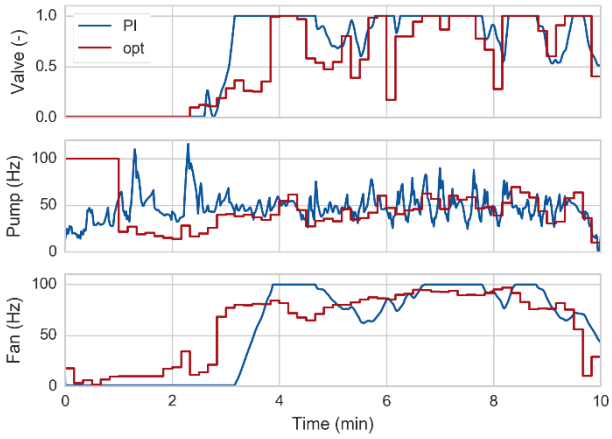

Figure 4. Controls from optimization and simulation. The optimum control uses the fan earlier and completely opens the valve later.

The second difference can be explained by looking at figure 5. While the PI control of the temperature does not become active until the setpoint is exceeded, the optimum control reacts earlier. With increased fan speed and valve opening, cooling is started before 90°C is reached. An exact approach of the setpoint can thus be achieved, without overshooting. In addition, it is avoided that the fan is operated with maximum speed and disproportionately high energy demand.

This second positive aspect is clearly visible in figure 6. The cumulative electrical energy consumption of both variants is shown, divided into fan and pump energy. The fan energy clearly dominates in both cases. In the case of PI control, the fan runs at a much higher speed compared to optimal control, especially between minutes 4 and 5. Within this time span, the PI controller reacts to the overshooting temperature. This has the consequence that the cumulative energy consumption increases sharply. Whereas the more uniform optimal control of the fan speed leads to total energy consumption reduced by 19%. The individual numerical values are listed in Table 1.

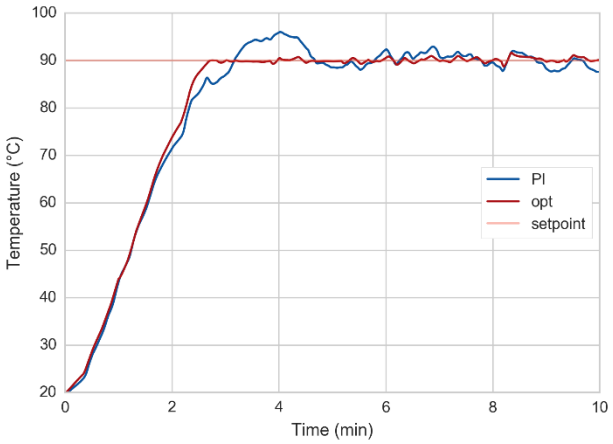

Figure 5. Results from optimization and simulation. The primary control objective of keeping the coolant temperature at 90 ° C is achieved in both cases. The optimum control achieves the setpoint value somewhat earlier, without overshooting.

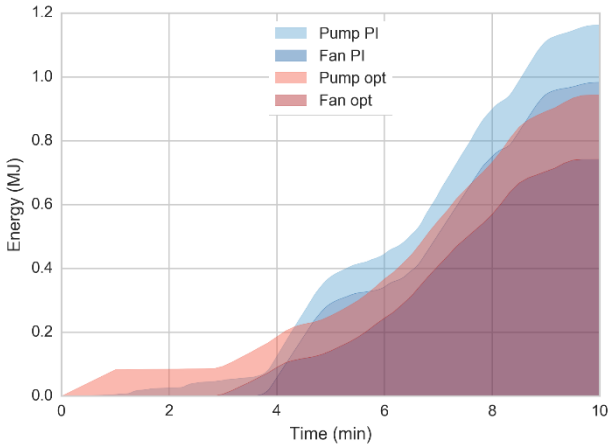

Figure 6. Cumulative electrical energy consumption from optimization and simulation. The optimum control achieves a 19% reduction in energy consumption.

For the described cold-start high-load scenario, the optimum control shows a significant reduction in energy consumption while at the same time better compliance with the setpoint for the coolant temperature. With the presented tool chain, such investigations can also be carried out for other systems and scenarios. Such optimal control results can be used for various purposes:

- as reference for control concepts
- finding heuristic (almost optimal) control laws
- for online optimization (NMPC)

Table 1. Total consumed electrical energy for both variants.

	Pump	*Fan*	*Total*
PI control	*0.18 MJ*	*0.98 MJ*	*1.16 MJ*
Optimal control	*0.20 MJ*	*0.74 MJ*	*0.94 MJ*
Difference	*+12%*	*-25%*	*-19%*

5 Summary

With the presented tool chain Modelica/TIL → FMI → MUSCOD-II thermal system can be modeled conveniently and optimal control trajectories can be calculated rapidly. For the example application, thermal management of an internal combustion engine, electrical energy savings of 19% (fan and pump) compared to PI control are achieved. By comparing the optimization and simulation results, the causes for energy savings can be explained.

Optimization calculations of this type can serve as a reference for control concepts to be developed. The interpretation of the optimal trajectories can also be used in finding heuristic (almost perfect) control laws. In principle, optimum control calculations are also suitable for online use in vehicles or other technical systems. Which is known as nonlinear model predictive control (NMPC). For prototypical NMPC applications on a Windows laptop, the presented tool chain can be used directly. However, it is not yet suitable for implementation on embedded control units.

References

Åkesson, Johan, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. 2010. "Modeling and Optimization with Optimica and JModelica.org--Languages and Tools for Solving Large-Scale Dynamic Optimization Problems." *Computers & Chemical Engineering* 34 (11): 1737–49. doi:10.1016/j.compchemeng.2009.11.011.

Albersmeyer, Jan. 2010. "Adjoint Based Algorithms and Numerical Methods for Sensitivity Generation and Optimization of Large Scale Dynamic Systems." Ruprecht-Karls-Universität Heidelberg.

Albersmeyer, Jan, and Moritz Diehl. 2010. "The Lifted Newton Method and Its Application in Optimization." *SIAM Journal on Optimization* 20 (3): 1655–84. http://epubs.siam.org/doi/abs/10.1137/080724885.

Bauer, Irene. 1999. "Numerische Verfahren Zur Lösung von Anfangswertaufgaben Und Zur Generierung von Ersten Und Zweiten Ableitungen Mit Anwendungen Bei Optimierungsaufgaben in Chemie Und Verfahrenstechnik." Universität Heidelberg. doi:10.1159/000328458.

Betts, John T. 2001. *Practical Methods for Optimal Control Using Nonlinear Programming*. Society for Industrial and Applied Mathematics.

Biegler, L. 2007. "An Overview of Simultaneous Strategies for Dynamic Optimization." *Chemical Engineering and Processing: Process Intensification* 46 (11): 1043–53.

Blochwitz, T., M. Otter, J. Akesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, et al. 2012. "Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models." In *9th International Modelica Conference*.

Bock, H. G., and K. J. Plitt. 1984. "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems." In *Proc. of the 9th IFAC World Congress Budapest*, 243–47. Pergamon Press.

Bock, Hans Georg. 1987. *Randwertproblemmethoden Zur Parameteridentifizierung in Systemen Nichtlinearer Differentialgleichungen*. Universität Bonn.

Bryson, Arthur E., and Yu-Chi Ho. 1979. *Applied Optimal Control: Optimization, Estimation, and Control*. John Wiley & Sons Inc.

Diehl, Moritz. 2001. "Real-Time Optimization for Large Scale Nonlinear Processes." Universität Heidelberg.

Franke, Rüdiger. 2002. "Formulation of Dynamic Optimization Problems Using Modelica and Their Efficient Solution." In *2nd International Modelica Conference*, 315–23. Oberpfaffenhofen.

Gräber, Manuel. 2013. "Energieoptimale Regelung von Kälteprozessen." TU Braunschweig.

Gräber, Manuel, Kai Kosowski, Christoph Richter, and Wilhelm Tegethoff. 2010. "Modelling of Heat Pumps with an Object-Oriented Model Library for Thermodynamic Systems." *Mathematical and Computer Modelling of Dynamical Systems* 16 (3): 195–209. doi:10.1080/13873954.2010.506799.

Imsland, L., P. Kittilsen, and T. S. Schei. 2010. "Model-Based Optimizing Control and Estimation Using Modelica Models." *Modeling, Identification and Control* 31 (3): 107–21. doi:10.4173/mic.2010.3.3.

Leineweber, D B, I Bauer, A A S Schäfer, H G Bock, and J P Schlöder. 2003. "An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II)." *Computers and Chemical Engineering* 27: 157–74.

Richter, Christoph. 2008. "Proposal of New Object-Oriented Equation-Based Model Libraries for Thermodynamic Systems." Technische Universität Braunschweig.

Schulze, C. 2013. "A Contribution to Numerically Efficient Modelling of Thermodynamic Systems." Technische Universität Braunschweig.