D5.2.2 Prototype for optimisation toolchain in OpenModelica

WP 5.2 Optimisation Toolchain

Work Package 5 Optimized system operations

MODRIO (11004)

Version 2.0
Date 11/30/2015

Authors Willi Braun FH Bielefeld
Vitalij Ruge FH Bielefeld
Bernhard Bachmann FH Bielefeld

Executive summary

Efficient calculation of the solutions of nonlinear optimal control problems (NOCPs) is becoming more and more important for today's control engineers. The systems to be controlled are typically described using differential-algebraic equations (DAEs), which can be conveniently formulated in Modelica. The corresponding optimization problem can be expressed using the Modelica language extension developed in D5.1.1 [6], so Modelica tools like OpenModelica need the capability to solve such optimization problems. Therefore, we developed efficient solution algorithms based on collocation methods that are highly suitable for discretizing the underlying dynamic model formulation. The corresponding discretized optimization problem can be solved, e.g. by the interior-point optimizer Ipopt.

Summary

Here a newly developed OpenModelica-based tool chain is presented for solving nonlinear optimization control problems. The underlying dynamic model formulation is expressed in Modelica. The demonstrated solution method is based on orthogonal collocation methods, whereby the first interval is specially treated in order to consider control variables and their derivatives also at the initial time point. Special treatment of the matrices with focus on yielding optimal sparsity patterns with respect to block and cyclic structure is performed. The resulting optimization process turns out to be stable and efficient.

1 Introduction

The aim of this deliverable is to establish a new exhaustive tool chain implemented in OpenModelica [5] for dynamic optimization and focusing on nonlinear optimal control problems. This effort continues the development of the collocation approach already discussed in [2], which has been successfully tested using the algorithmic differentiation tool CasADi [8] and ADOLC [7]. Several enhancements, e.g. special treatment of the first collocation interval, efficient and stable calculation of all derivative information, have been realized therefore in OpenModelica.

The mathematical representation of the underlying nonlinear optimal control problem is discussed in report D5.1.6.(see [3]) and also the main idea of discretizing the NOCP based on orthogonal collocation principles are described there.

This report is organized as follows. In section 2 the capability of OpenModelica to formulate optimization problems is described. In section 3 the main features of the newly developed tool chain are shown and as an example the benchmark model DrumBoiler is demonstrated in section 4.

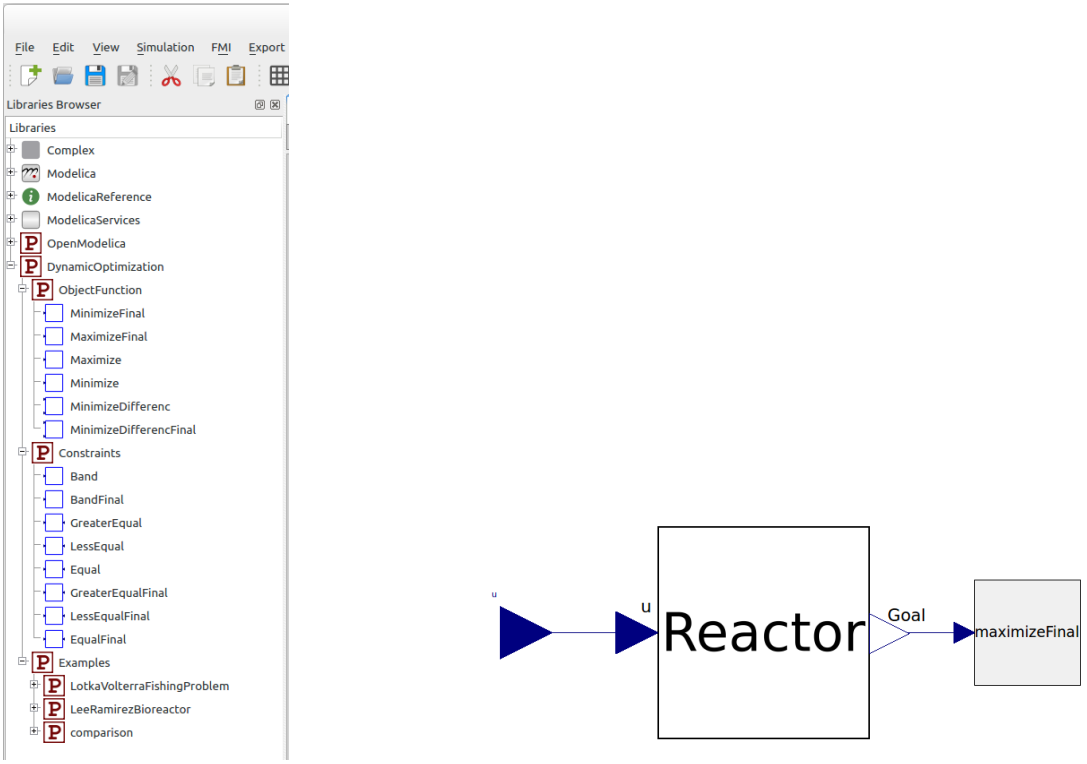
2 Formulation of the Optimization problem

The formulation of an optimization problem in OpenModelica was in the first iteration realized with a partial support of the optimization language extension Optimica [1]. In the next step the approach from MODRIO deliverable [6] was implemented in OpenModelica. Thereby an optimization library has been developed, that contains Modelica blocks, which allows to formulate an optimization problem by Drag'n'Drop (see figure 1a).

Listing 1: Samll BatchReactor example

```
model BatchReactor
  Real A(start = 1, min = 0, max = 1);
  Real B(start = 0, min = 0, max = 1);
  Modelica.Blocks.Interfaces.RealOutput y a;
  Modelica.Blocks.Interfaces.RealInput u(min = 0, max = 5) a;
equation
  der(A) = -(u + u ^ 2 / 2) * A;
  der(B) = u * A;
  y = B;
  a;
end BatchReactor;
```

In the following the usage of this library is shown on the small example model BatchReactor 1. This model contains one input signal and one output signal, the input signal is the variable that should optimized and the output signal is used for the goal function. For the optimization problem formulation this model is used as a block and connected with the corresponding blocks from the Dynamic Optimization Library, as depicted in figure 1b.



(a) Dynamic Optimization Library (b) BatchReactor graphical example

Figure 1: Usage of the optimization library

3 Dynamic Optimization in OpenModelica

In order to start the optimization in the OpenModelica environment there is a new application program interface (API) call introduced `optimize(ModelName)`. This command runs immediately the optimization from OMShell, OMNotebook or MDT. The generated result file can be read in and visualized with OMEdit or within OMNotebook. The table 1 contains optimization related options for the API call optimize.

In case the optimization problem is formulated with custom annotations it is also possible to run the optimization with OMEdit, but therefore it is necessary to adjust the compiler settings. First it is important to add the following OMC compiler flags `+gDynOpt` in OMEdit options (see figure 2).

Options	Default value	Description
numberOfIntervals	500	collocation intervals
startTime	0	start time
stopTime	1.0	stop time
tolerance	1e-6	solver tolerance

Table 1: Options for the OpenModelica API call `optimize()`

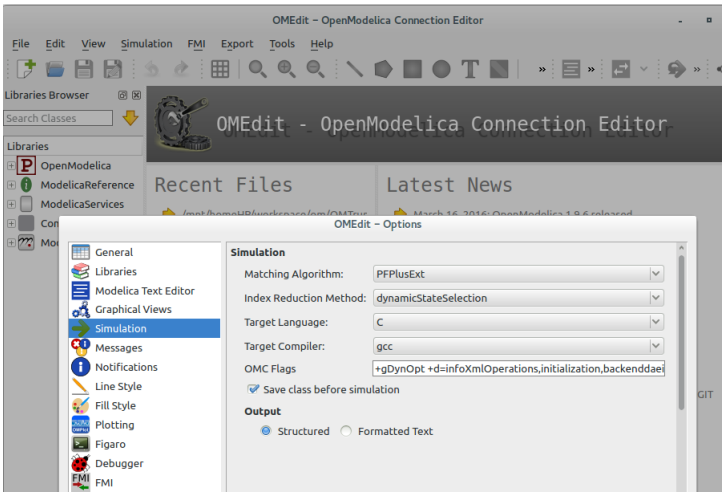


Figure 2: Configuration of OMC flags for optimization

This activates the corresponding tasks for the optimization while the model is symbolically instantiated by the compiler in order to get a single flat system of equations. The OpenModelica compiler frontend performs this, including syntax checking, semantics and type checking etc. are applied. This also involves the processing of the optimization related terms of optimica and custom annotations.

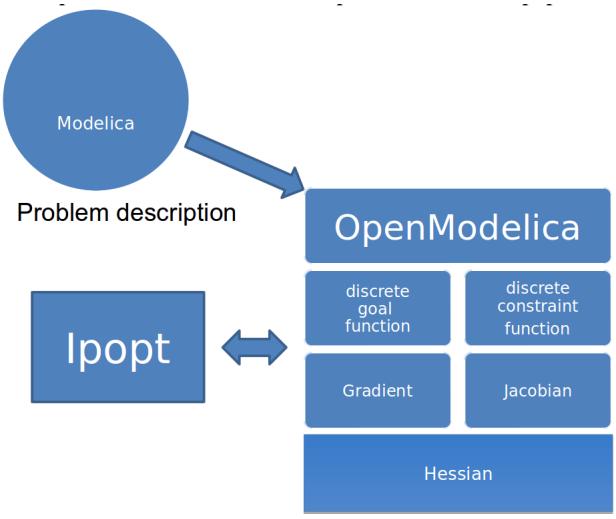


Figure 3: Diagram of the optimization formulation process

In the next step the compiler backend performs a lot symbolical preprocessing, like alias elimination, simplifications and index reduction. This also includes the generation of optimization related derivative matrices. After all these processes are completed an application is generated by the OpenModelica compiler, which can be used for initialization, simulation and optimization.

For the optimization capability a discretization of the ODE, the cost function and the constraints

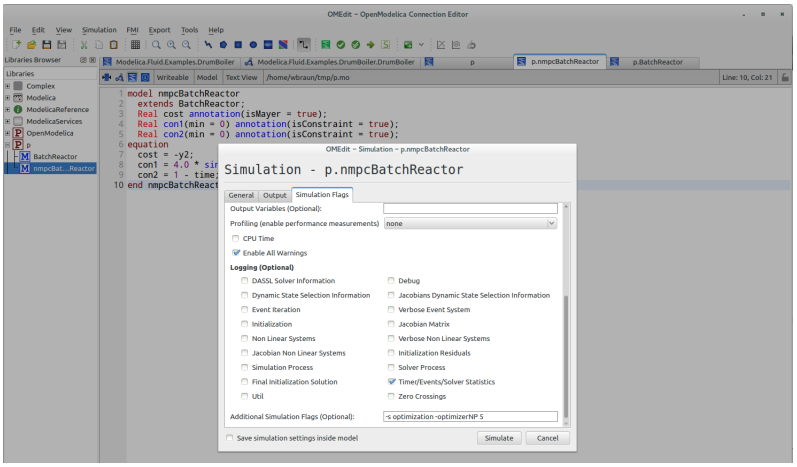


Figure 4: Set optimization flag.

Application flag	Example value	Description
-lv	LOG_IPOPT	console output
-ipopt_hesse	CONST,BFGS,NUM	hessian approximation
-ipopt_max_iter	10	maximal number of iteration for ipopt
-exInputFile <filename>	-exInputFile externalInput.csv	input guess of the inputs

Table 2: List of optimization related flags.

is performed based on collocation techniques (see [7]). With this discretization a nonlinear optimization problem (NLP) is generated, which is solved by the interior-point optimizer Ipopt. In order to start the optimization instead of the simulation it is necessary to set the following flag "-s optimization" (see figure 4). The table 2 contains all optimization related flags for the generated application.

The usage of nonlinear model predictive control (NMPC) in OpenModelica is realized by OpenModelica scripts and a low-level file interface. The idea is to use the result file from one optimization run and initialize the next optimization horizon with this result and use a modelica script to define the prediction horizon and the sampling points.

4 Optimization Example of the DrumBoiler

A comprehensive description of the DrumBoiler example can be found in [4]. Explanation of the formulation of the optimization problem with custom annotations for the DrumBoiler can be found in [6]. The model description was modified for handling $-25/60 \leq \dot{q}_F \leq 25/60$ and $q_F(0) = 0$ as

$$\min_{dq_F(t), Y_{Valve}(t)} \quad \int_0^{3600} \frac{(p_s(t) - 110)^2}{1000} + \frac{(qm_s(t) - 180)^2}{10000}\, dt$$

s.t.

$$\begin{aligned}
\sigma_D(t) &\geq -150 \\
0 &\leq Y_{Valve}(t) \leq 1 \\
0 &\leq q_F(t) \leq 500 \\
\frac{-25}{60} &\leq dq_F(t) \leq \frac{25}{60} \\
q_F(0) &= 0 \\
\dot{q}_F(t) &= dq_F(t) \\
\dot{x}(t) &= f\left(x(t), dq_F(t), Y_{Valve}(t), t\right)
\end{aligned}$$

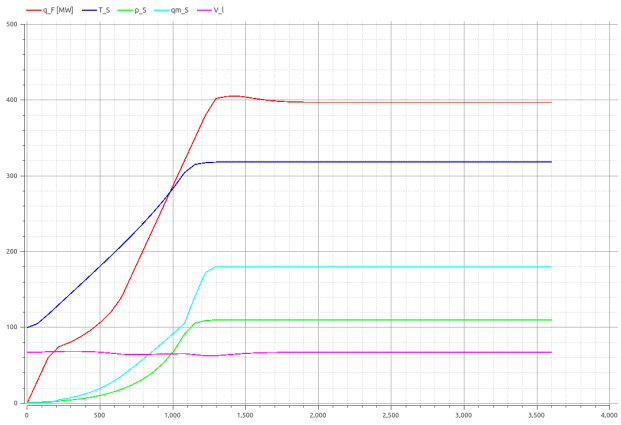
where $x(t) = (\text{controller}.x(t), \text{evaporator}.V_l(t), \text{evaporator}.p(t), q_F(t))^\top$ and $t \in [0, 3600]$.

Listing 2: DrumBoiler optimization problem with custom annotations

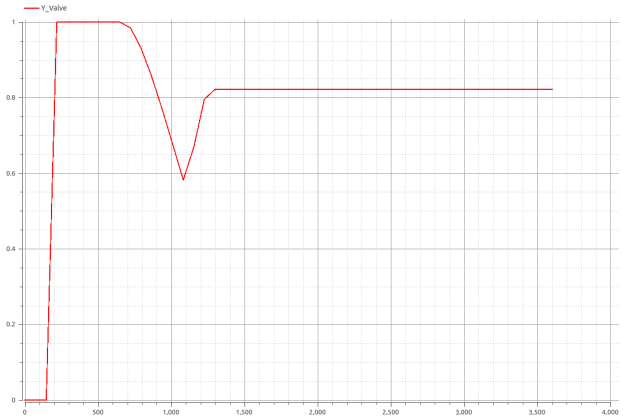
```
block LagrangeTerm
 input Real signal annotation(isLagrange = true);
end LagrangeTerm;

block Constraint
 input Real signal annotation(isConstraint = true);
end Constraint;

model optDrumBoiler
"
 see:
  On-line Optimization of DrumBoilerStartup
  Ruediger Franke, Manfred Rode and Klaus Krueger
  Paper presented at the 3rd International Modelica Conference
"
  extends DrumBoiler(
                     Y_Valve(min = 0, max = 1, nominal = 1, start = 0.5),
                     q_F(min = 0, max = 500, start = 0, fixed = true , nominal = 400),
                     controller.x(nominal =10),
                     use_inputs = true
                     );
  LagrangeTerm cost_q_S(signal(nominal = 1e3) = (p_S - 110)^2);
  LagrangeTerm cost_qm_S(signal(nominal = 1e4) = (qm_S - 180)^2);
  Constraint sigma_D(
    signal(min = -150) = (-1.0e3 * der(evaporator.T_D)) + 1.0e-05 * evaporator.p);
  input Real dq_F(min = -25/60, max=25/60,start = 0.1);
equation
  der(q_F) = dq_F;
end optDrumBoiler;
```

Consequences to the Modelica description of this problem with respect to the OpenModelica implementation can be found in listing 2, where the `annotation` are hidden inside blocks, in order to receive a more user friendly representation.

The results are visualized in figure 5a and 5b, which coincide with the reference `https://modrio.org/svn/MODRIO/trunk/WP5%20Optimization/WP5.1%20Model%20Predictive%20Control/D5.1.1/`



(a) DrumBoiler results for `q_F`, `T_S`, `p_S`, `qm_S`, `sigma_D`, `V_l`



(b) DrumBoiler results for `Y_Valve`

Figure 5: Results of DrumBoiler example

`Benchmarks/DrumBoiler/TrajectoryOptimizationReferenceSolution.png`.

References

[1] J. Åkesson. *Languages and Tools for Optimization of Large-Scale Systems*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund University, 2007.

[2] B. Bachmann, L. Ochel, V. Ruge, M. Gebremedhin, P. Fritzson, V. Nezhadali, L. Eriksson, and M. Sivertsson. Parallel multiple-shooting and collocation optimization with openmodelica. In *9th International Modelica Conference*, 2012.

[3] W. Braun, V. Ruge, and B. Bachmann. D5.1.6 first offline version of large-scale test case application in power generation. *ITEA2 project MODRIO(11004) deliverable*, 2014.

[4] R. Franke, M. Rode, and K. Krueger. On-line optimization of drumboilerstartup. *3th International Modelica Conference*, pages 287–296, 2003.

[5] P. Fritzson, P. Aronsson, H. Lundvall, K. Nyström, A. Pop, L. Saldamli, and D. Broman. Openmodelica - a free open-source environment for system modeling, simulation, and teaching. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 1588 –1595, oct. 2006.

[6] A. Pfeiffer, M. Otter, V. Ruge, B. Bachmann, J. Åkesson, T. Henngingsson, and H. Elmqvist. D5.1.1: Extension of modelica with language elements to describe optimization problems, and interfaces to execute nmpc in a virtual environment. *ITEA2 project MODRIO(11004) deliverable*, 2014.

[7] V. Ruge, W. Braun, B. Bachmann, , A. Walther, and K. Kulshreshtha. Efficient implementation of collocation methods for optimization using openmodelica and adol-c. In *10th International Modelica Conference*, 2014.

[8] A. Shitahun, V. Ruge, M. Gebremedhin, B. Bachmann, L. Eriksson, J. Andersson, M. Diehl, and P. Fritzson. Model-based dynamic optimization with openmodelica and casadi. *7th IFAC Symposium on Advances in Automotive Control*, pages 446–451, 2013.