Tools and Languages for Modeling and Optimization of Large-Scale Dynamical Systems

Johan Åkesson and Karl-Erik Årzen

{johan.akesson,karl-erik.arzen}@control.lth.se
Department of Automatic Control, Faculty of Engineering, Lund University

Keywords: Large Scale Dynamic Optimization, Modelica, Modeling of Complex Systems

1. Introduction

High-level modeling languages are receiving increased industrial and academic interest within several domains, such as chemical engineering, thermo-fluid systems and automotive systems. One such modeling language is Modelica, [1]. Modelica is an open language, specifically targeted at multi-domain modeling and model reuse. Key features of Modelica include object oriented modeling, declarative equation modeling, a component model enabling acausal connections of submodels, as well as support for hybrid/discrete behaviour. These features have proven very applicable to large-scale modeling problems in various fields.

While there exist very efficient software tools for simulation of Modelica models, tool support for static and dynamic optimization is generally weak. Furthermore, specification of optimization problems is not supported by Modelica. Since Modelica models represent an increasingly important asset for many companies, it is of interest to investigate how Modelica models can be used also for optimization.

This contribution gives an overview of a project targeted at *i)* defining an extension of Modelica, Optimica, which enables high-level formulation of optimization problems, *ii)* developing prototype tools for translating a Modelica model and a complementary Optimica description into a representation suited for numerical algorithms, and *iii)* performing case studies demonstrating the potential of the concept.

The project integrates dynamical modeling and optimization with computer science and numerical algorithms. One of the main benefits of the suggested approach is that the high-level descriptions are automatically translated into an intermediate representation by the compiler front-end. This intermediate representation can then be further translated to interface with different numerical algorithms. The user is therefore relieved from the burden of managing the often cumbersome API:s of numerical algorithms. The flexibility of the architecture also enable the user to select the algorithm most suitable for the problem at hand.

2. Optimica

A key issue is the definition of syntax and semantics of the Modelica extension, Optimica. Optimica should provide the user with language constructs that enables formulation of a wide range of optimization problems, such as parameter estimation, optimal control and state estimation based on Modelica models.

At the core of Optimica are the basic optimization elements such as cost functions and constraints. It is also possible to specify bounds on variables in the Modelica model as well as to mark variables and parameters as optimization quantities, i.e., to express what to optimize over. While this type of information represents a canonical optimization formulation, the user is often required to supply additional information, related to the numerical method which is used to solve the problem. In this category we have e.g., specification of transcription method, discretization of control variables and initial guesses. Optimica should also enable convenient specification of these quantities.

3. Software Tools

In order to demonstrate the proposed concept, prototype software tools are being developed.

In essence, the task of the software is to read the Modelica and Optimica source code and then translate, automatically, the model and optimization descriptions into a format which can be used by a numerical algorithm. The Modelica/Optimica compiler is developed using the Java-based compiler construction tool JastAdd, [5]. For an overview of the computer science aspects of the compiler implementations, see [7].

Currently, the front-end of the Modelica/Optimica compiler supports a subset of Modelica and a basic version of Optimica. In addition, a code-generation back-end for AMPL, [3], has been developed. AMPL is a language intended for formulation of algebraic optimization problems. Accordingly, the compiler performs automatic transcription of the original continuous-time problem into an algebraic formulation which can be encoded in AMPL. In the transcription procedure, the problem is discretized by means of a simultaneous optimization approach based on collocation over finite elements, see e.g., [2] for an overview. Finally, the automatically generated AMPL description may be executed and solved by a numerical NLP algorithm. For this purpose we have used IPOPT, [6].

4. A Case Study

The prototype tools have been used to formulate and solve a start-up problem for a plate reactor system. The plate reactor is conceptually a tubular reactor located inside a heat exchanger, and offers excellent flexibility, since it is reconfigurable and allows multiple injection points for chemicals, separate cooling/heating zones and easy mounting of temperature sensors. In this case study, an exothermic reaction, $A + B \rightarrow C$, was assumed. The reactor was fed with a fluid with a specified concentration of the reactant A. The reactant B was injected at two points along the reactor.

The primary objective of the start-up sequence was to transfer the state of the reactor from an operating point where no reaction takes place, to the desired point of operation. This problem is challenging, since the dynamics of the system is fast and unstable in in some operating conditions. In addition, the temperature in the reactor must be kept below a safety limit, in order not to damage the hardware.

Optimal control and state profiles were calculated off-line and then used as feedforward and feedback signals in a PID-based mid-ranging control system.

The experiences from using the Modelica/Optimica compiler in this project are promising, in that the tools enable the user to focus on *formulation* of the problem instead of, which is common, *encoding* of the problem. For more details on this case study, see [4].

5. Summary

This contribution gives an overview of a project targeted at extending the Modelica language to also support optimization. The goals of the project include specification of Optimica, development of prototype software tools and case studies. The results are promising, and encourage further development.

REFERENCES

[1] The Modelica Association, 2006. http://www.modelica.org.

[2] L.T. Biegler, A.M. Cervantes, and A Wächter. Advances in simultaneous strategies for dynamic optimization. *Chemical Engineering Science*, 57:575–593, 2002.

[3] R Fourer, D. Gay, and B Kernighan. *AMPL – A Modeling Language for Mathematical Programming*. Brooks/Cole — Thomson Learning, 2003.

[4] Staffan Haugwits and Johan Åkesson. Dynamic optimization of a plate reactor start-up supported by Modelica-based code generation software. In *8th International Symposium on Dynamics and Control of Process Systems*, 2007. Accepted for publication.

[5] Görel Hedin and Eva Magnusson. JastAdd: an aspect-oriented compiler construction system. *Science of Computer Programming*, 47(1):37–58, 2003.

[6] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–58, 2006.

[7] Johan Åkesson, Torbjörn Ekman, and Görel Hedin. Development of a Modelica compiler using JastAdd. In *Seventh Workshop on Language Descriptions, Tools and Applications*, 2007. Accepted for publication.