Dynamic Optimization of Modelica Models – Language Extensions and Tools

Johan Åkesson
Department of Automatic Control
Faculty of Engineering
Lund University

© Johan Åkesson 2007

Content

• Background
• Dynamic optimization
• Software tools
• Optimica

© Johan Åkesson 2007
Content

- **Background**
  - Dynamic optimization
  - Software tools
  - Optimica
Background

- Modelica is increasingly used in industry
  - Expert knowledge
  - Capital $$$
- Usages so far
  - Simulation (mainly)
- Other usages emerge
  - Sensitivity analysis
  - Optimization
  - Model reduction
  - System identification
  - Control design
- Usages reported so far
  - Cope with with software interfaces designed for simulation…

Dynamic Optimization

- New abstractions
  - Cost (objectives), constraints, optimization of parameters and functions, cases
  - Initial guesses
  - Transcription method (discretization)
- Requirements
  - Sensitivities
  - Derivatives
    - Jacobians
    - Hessians
  - Sparsity patterns
JModelica – Project Objectives

• Shift focus:
  – from encoding optimization problem
  – to problem formulation

• Enable dynamic optimization of Modelica models
  – State of the art numerical algorithms

• Develop a high level description language for optimization problems
  – Extension of the Modelica language

• Develop prototype tools
  – JModelica and The Optimica Compiler

• Case study
  – Plate reactor start-up optimization (see paper)

Content

• Background
• Dynamic optimization
• Software tools
• Optimica
A Dynamic Optimization Problem

\[
\min_{u,p} J(x, u, p, t_f) = \min_{u,p} \left\{ \int_0^{t_f} L(x, u, p) dt + \phi(x_f) \right\}
\]

subject to
\[
F(\dot{x}, x, u, p) = 0, \quad \text{(DAE dynamics)}
\]
\[
g_0(x_0) = 0 \quad \text{(initial conditions)}
\]
\[
g_f(x_f) = 0 \quad \text{(terminal constraint)}
\]
\[
r_i(x(t), u(t)) \leq 0 \quad \text{(inequality path constraints)}
\]
\[
r_e(x(t), u(t)) = 0 \quad \text{(equality path constraints)}
\]

Dynamic Optimization

- Many algorithms
  - Applicability highly model-dependent (ODE, DAE, PDE, hybrid...)
  - Active area of research
- User must specify additional information
  - Discretization mesh
  - Discretization scheme
- Heavy programming burden to use numerical algorithms
- Engineering need for high-level descriptions
  - Extend Modelica to support dynamic optimization
Key Abstractions in Optimization

- Optimization parameters and functions
- Cost – what to minimize
- Objectives (multi-objective optimization)
- Cases
- Bounds on variables – Not a good idea to hi-jack attributes
- Constraints
- Initial guesses
- Discretization information

Content

- Background
- Dynamic optimization
- **Software tools**
- Optimica
Software – Motivation

- Code generation requires syntax trees
  - "The final syntax tree before code generation"
  - Dymola – not available
  - OpenModelica – August 2005: RML…?
- Subset of Modelica sufficient initially
  - No "hybrid" constructs
  - Start in small scale
- Java
  - Safe language
  - Rapid development
  - Standard library
  - Not so slow…
- JastAdd
  - Compiler construction framework

The JModelica Project – Status

- Parsing of full Modelica 2.2
  - Standard library (2.2) parsed in approx 15 s
  - Name and type lookup + error check: 5 s
  - Improved error message generation
    • Dude, don’t use if-clauses. It is not good for ya…
- Partial support for instantiation
  - Classes, components, connections, modifications…
  - Partial support for arrays
  - Smoothing of discontinuities (min, max)
- Testing based on JUnit framework
- Prototype specification of Optimica
  - Description of an optimization problem
  - The Optimica Compiler: Code generation to AMPL/IPOPT
- Applications
  - Extensively used for start-up optimization of plate reactor
  - Two master’s thesis projects spring 2007: vehicle dynamics
JModelica Software Tools

MiniModelica (System model)

Optimica Light (OCP)

Initial guess

The Optimica Compiler

AMPL (Sim. Method)

Software Architecture – Re-use

Front-ends

Modelica AST builder
Optimica AST builder
Other AST builder

Translation optimization
Translation optimization
Translation optimization

Canonical data structures and API

Back-ends

Modelica
Optimica
Other

Translation
Translation
Translation

Interface/Code gen.
Interface/Code gen.
Interface/Code gen.

Bob’s solver
Pat’s solver
Linda’s solver
Wayne’s solver

© Johan Åkesson 2007
Content

• Background
• Dynamic optimization
• Software tools
  • Optimica

Language Extension Objections

• Use annotations?
  – Optimization problems have a rich structure – need for efficient language constructs
  – Annotations cannot be modified
• Language maintenance?
  – Modularization of specification
  – Modularization of implementation
    • JastAdd
  – Natural evolution of Modelica
    • Simulation → Optimization
Extension Concepts

- Enhanced class: `optmodel`
- Superimpose information on elements
  - Introduce new attributes for `Real`
  - New built-in package `Optimica`
  - Concept from aspect orientation
- New sections
  - `optimization`
    - Superimpose information
    - Meshes and discretization
  - `subject to`
    - Constraints
- New built-in functions
  - `minimize`: cost function
  - `integrate`: integration
  - `instantValue`: Instant time values of variables $x(t)$

Optimica – Basic Constructs

\[
\begin{align*}
\dot{x} &= v, \quad x(0) = 0 \\
\dot{v} &= u, \quad v(0) = 0
\end{align*}
\]

Optimica

```modelica
model DoubleIntegrator
  Real u;
  Real x(start=0);
  Real v(start=0);
  equation
    der(x) = v;
    der(v) = u;
end DoubleIntegrator
```

```optmodel OCP
  DoubleIntegrator di;
  optimization
    di.u(lowerBound=1, upperBound=-1);
  equation
    minimize(integrate(1));
  subject to
    terminal di.x=1;
    terminal di.v=0;
end DoubleIntegrator
```
Optimica – Specification of Discretization Scheme

package Optimica
record Mesh
    Real meshPoints[:];
...
end Mesh;
package Collocation
record LagrangeCollocation
    Real collocationPoints[:];
...
end LagrangeCollocation;
record DiscretizationSpec
    Mesh mesh;
    LagrangeCollocation lc;
end DiscretizationSpec;
end Collocation;
end Optimica

• Built-in package Optimica
  – Mesh
  – Collocation
• Essential information
  – Closer to algorithms
• Data structures
• Similar to annotations

Optimica – Specification of Discretization Scheme: Example

optmodel OCP
    DoubleIntegrator di;
    Optimica.Mesh mesh(...);
    Optimica.LagrangeCollocation lc(..);

optimization
    di.u(lowerBound=1,
        upperBound=-1);
    extends Optimica.Collocation.
    DiscretizationScheme(mesh=mesh,
        lc=lc);

equation
    minimize(integrate(1));
subject to
    terminal di.x=1;
    terminal di.v=0;
end DoubleIntegrator

© Johan Åkesson 2007
Summary

• JModelica and The Optimica Compiler
  – Extensible compiler
  – Front-end supporting subset of Modelica
  – Back-end for code generation to AMPL
• Optimica
  – Extension of Modelica for optimization
• Case study
  – See paper

Thank you!
Questions?