DAE Tools: An equation-oriented process modelling and optimization software

Introduction

Dragan Nikolic

University of Western Macedonia
Department of Mechanical Engineering
50100 Kozani, Greece

December 4, 2013

Outline

Outline

What is DAE Tools?

- Process modelling, simulation, optimization and parameter estimation software (www.daetools.com)
- Areas of application:
  - Initially: chemical process industry (mass, heat and momentum transfers, chemical reactions, separation processes, phase-equilibrium, thermodynamics)
  - Nowadays: multi-domain
- Hybrid approach between general-purpose programming languages (c++, Fortran, Java) and domain-specific modelling languages (Modelica, gPROMS...)

What can be done with DAE Tools?

- Simulation
  - Steady-State
  - Transient
- Optimization
  - NLP problems: IPOPT, NLOPT, OpenOpt, scipy.optimize
  - MINLP problems: BONMIN
- Parameter estimation: LevenbergMarquardt algorithm (scipy.optimize)

Types of systems that can be modelled

- Initial value problems of implicit form: systems of linear, non-linear, and (partial-)differential algebraic equations
- Index-1 DAE systems
- With lumped or distributed parameters: Finite Difference or Finite Elements Methods
- Steady-state or dynamic
- Continuous with some elements of event-driven systems (discontinuous equations, state transition networks and discrete events)

Why yet another software?

Advantages:

1. Hybrid approach betwen DSL and GPPL
2. Programmatical generation of models
3. Runtime modification of objcts/models (operating procedures)
4. Intropерability with 3rd party software packages/libraries
5. Code generation/Model exchange capabilities

Not a modelling language

- A set of software packages
- API for:
  - Model development
  - Results processing (plotting, various file formats)
  - Simulation, optimization and parameter estimation
  - Code generation for other DSLs and programming languages
  - Report generation (XML+MathML) and model exchange
- Large set of supported solvers (DAE, LA, NLP, MINLP)

Not a modelling language (cont'd)

- Allows easy interaction with other software libraries (two-way interoperability with other software, embedding in other software etc.)
- Free/Open source software (GNU GPL)
- Cross-platform (GNU/Linux, MacOS, Windows)
- Supports multiple architectures (32/64 bit x86, arm, any other with the GNU toolchain)
- Developed in c++ with Python bindings (Boost.Python)

Object-oriented modelling

- Everything is an object
- Models are classes derived from the base daeModel class
- Basically all OO concepts supported by the target language (c++, Python) allowed, except few exceptions
- Multiple inheritance is supported
- Models can be parametrized (using templates in c++)
- Derived classes always inherit all declared parameters, variables, equations etc. (polymorphism achieved through virtual functions where the declaration takes place)
- All parameters, variables, equations etc. remain public
- Hierarchical model decomposition

Equation-oriented (acausal) modelling

- Equations given in an implicit form (as a residual)
- Input-Output causality is not fixed
- Benefits:
- Increased model re-use
- Support for different simulation scenarios (based on a single model) by specifying different degrees of freedom
- For instance, equation given in the following form:
- can be used to determine either x1, x2 or x3 depending on what combination of variables is known:

Separation of models definition from operations on them

- The structure of the model (parameters, variables, equations etc.) is given in the model class while the runtime information in the simulation class
- Single model definition, but:
- One or more different simulation scenarios
- One or more optimization scenarios
- Hybrid continuous/discrete systems
- Continuous with some elements of event-driven systems (discontinuous equations, state transition networks and discrete events)

Code generation

- Model export from DAE Tools to other DSL or programming languages)
- Modelica
- ANSI C
- Support for Functional Mock-up Interface for Model Exchange and Co-Simulation (FMI): https://www.fmi-standard.org
- FMI a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code
- Still in experimental phase

Model reports

- Automatic model documentation
- XML + MathML format
- Two types:
- Model description report (contains model definition)
- Runtime report with all values and equations expanded (contains definition of the simulation)
- XSL transformation used to to generate HTML code and visualize reports

Outline

unnumbered lists

lists with pause

numbered lists

numbered lists with pause

Outline

Tables