Comparison of Numerical Integration Methods in OpenModelica

Status and Plans on Integration methods

Willi Braun   Bernhard Bachmann

FH Bielefeld
University of
Applied Sciences

University of Applied Sciences Bielefeld
Bielefeld, Germany

February 6, 2017

Motivation

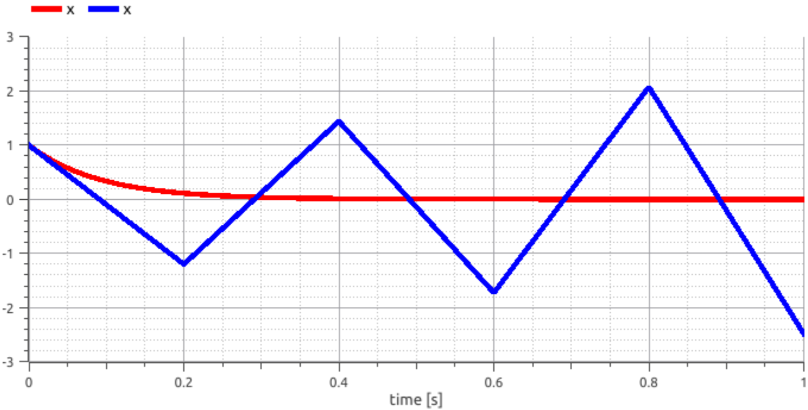
FH Bielefeld
University of
Applied Sciences

Basic criteria

Stability vs. Performance.

Basic criteria

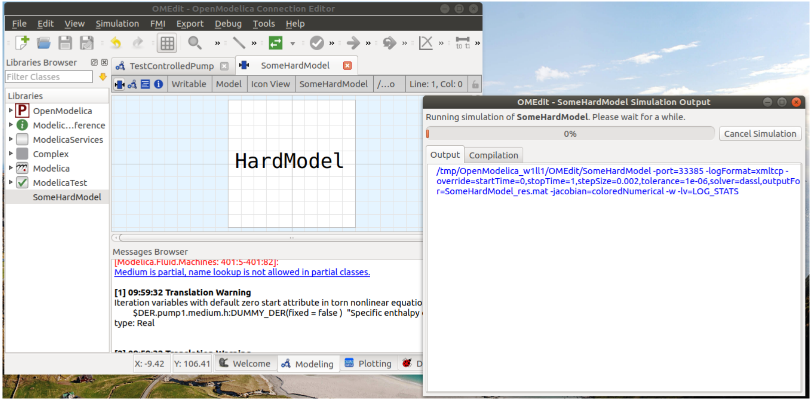
Stability vs. Performance.



Motivation

Basic criteria

Stability vs. Performance.



Motivation

We are 2 times slower, but we want to get 3 times faster.

Rüdiger

- Outline:
  - Overview of the current available solver
  - Comparision of IDA and DASSL
  - Improved Symbolic Inline Solver
  - Comparison of DAEMode vs. ODEMode

Motivation

We are 2 times slower, but we want to get 3 times faster.

Rüdiger

- Outline:
  - Overview of the current available solver
  - Comparision of IDA and DASSL
  - Improved Symbolic Inline Solver
  - Comparison of DAEMode vs. ODEMode

Solver in OpenModelica

$$\underline{0} = f(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{u}(t), t)$$

$$\downarrow$$

$$\underline{0} = f(\underline{x}(t), \underline{z}(t), \underline{u}(t), t),\ \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\underline{y}(t) = \underline{k}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

General Characteristic

- explicit vs. implicit
- higher order
- with step size control
- multi-step methods

FH Bielefeld
University of
Applied Sciences

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

FH Bielefeld
University of
Applied Sciences

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Explicit Euler

$$\dot{x} \approx \frac{x(t_{n+1}) - x(t_n)}{h_n}$$

$$x(t_{n+1}) = x(t_n) + h_n \cdot f(t_n, x(t_n))$$

- very cheap
- poor stability region

solver name: euler

FH Bielefeld
University of
Applied Sciences

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Implicit Euler

$$\dot{x} \approx \frac{x(t_n) - x(t_{n-1})}{h_n}$$

$$x(t_n) = x(t_{n-1}) + h_n \cdot f(t_n, x(t_n))$$

- very stable
- quite expensive
- non-linear loop solved by KINSOL

solver name: impeuler

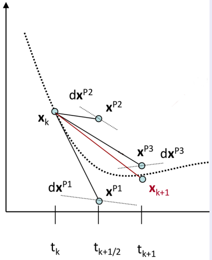
General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Explicit Runge-Kutta Methods

Butcher tableau :

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
$$



solver name: heun, rungekutta

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Explicit Runge-Kutta Methods

- orders 2 and 4
- good performace
- still small stability region

solver name: heun, rungekutta

FH Bielefeld
University of
Applied Sciences

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

implicit Runge-Kutta methods

Butcher tableau :

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

solver name: impeuler, trapzoide, imprungekutta

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

implicit Runge-Kutta methods

- order 1-6 (-impRKOrder=X)
- very stable
- quite expensice
- non-linear loop solved by KINSOL

solver name: impeuler, trapzoide, imprungekutta

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Explicit Runge-Kutta Step Size Control

Butcher tableau :

$$
\begin{array}{c|cccccc}
c_1 & 0 & 0 & 0 & \dots & 0 & 0 \\
c_2 & a_{21} & 0 & 0 & \dots & 0 & 0 \\
c_3 & a_{31} & a_{32} & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\
c_n & a_{n1} & a_{n2} & a_{n3} & \dots & a_{n(s-1)} & 0 \\
\hline
 & b_1 & b_2 & b_3 & \dots & b_{s-1} & b_s \\
 & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \dots & \hat{b}_{s-1} & \hat{b}_s
\end{array}
$$

- embedded Runge-Kutta formulas
- quite fast
- better stability region
- Current status: experimental

solver name: rungekuttaSsc

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Implicit Runge-Kutta Step Size Control

Butcher tableau :

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_n & a_{n1} & a_{n2} & \cdots & a_{ns} \\
\hline
 & b_1 & b_2 & \cdots & b_s \\
 & \hat{b} & \hat{b}_2 & \cdots & \hat{b}_s
\end{array}
$$

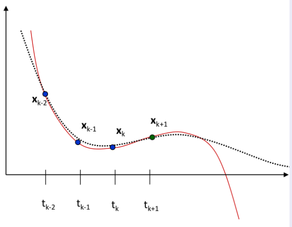
- Own implementation
- For now order 1-2
- Using own newton solver
- Current status: experimental

solver name: irksco

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

Multi-Step BDF method: DASSL



- implicit
- order control
- step size control

solver name: dassl, ida

Solver in OpenModelica

General Characteristic

FH Bielefeld
University of
Applied Sciences

General Characteristic:

- explicit vs. implicit
- higher order
- step size control
- multi-step methods

SUNDIALS IDA solver

- DASSL re-implementation in C
- Interface to fast linear solver (KLU)
- usable for large-scale models

solver name: dassl, ida

FH Bielefeld
University of
Applied Sciences

Selected compared models

model	solver	steps	evalF	time
fullRobot	dassl	5475	19363	3.114
	ida	5659	19533	3.154
HeatExhanger	dassl	158	1334	5.972
	ida	161	1374	6.181
EngineV6	dassl	15179	35622	15.0516
	ida	15509	35667	14.9201
Themal.Motor	dassl	896	722167	2.44322
	ida	920	722167	2.79349

ScaleableTestSuite DASSL vs. IDA

Get your own impression:

DASSL (2017-01-18) vs. IDA (2017-01-21)

Symbolic Inline Integration

FH Bielefeld
University of
Applied Sciences

Symbolic Inline

Replaces `der(states)`
by forward difference quitient:
`--symSolver=expEuler`
or by backward difference quitient:
`--symSolver=impEuler`

Symbolical Implications

- Result is a pure algebraic system
- Apply OpenModelica Backend(e.g. Tearing, symbolic simplification)
- Basic step size control available
- Current status: experimental

solver name: symSolver, symSolverSsc

Symbolic Inline Integration

Symbolic Inline

Replaces `der(states)`
by forward difference quitient:
`--symSolver=expEuler`
or by backward difference quitient:
`--symSolver=impEuler`

Symbolical Implications

- Result is a pure algebraic system
- Apply OpenModelica Backend(e.g. Tearing, symbolic simplification)
- Basic step size control available
- Current status: experimental

solver name: symSolver, symSolverSsc

DAE Integration

$$\underline{0} = f(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{u}(t), t)$$

$$\downarrow$$

$$\underline{0} = f(\underline{x}(t), \underline{z}(t), \underline{u}(t), t),\ \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\underline{y}(t) = \underline{k}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$\Rightarrow$ typical ODE transformation

DAE Integration

$$\underline{0} = f(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{u}(t), t)$$

$$\downarrow$$

$$\underline{0} = f(\underline{x}(t), \underline{z}(t), \underline{u}(t), t), \; \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\underline{y}(t) = \underline{k}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$\Rightarrow$ typical ODE transformation

Skip Matching and Sorting

DAE solution

- DAE solves also for $\dot{x}$, y
- No inner algebraic loops -> no tearing
- potentially faster compilation phase

DAE Integration

$$\underline{0} = f(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{u}(t), t)$$

$$\downarrow$$

$$\underline{0} = f(\underline{x}(t), \underline{z}(t), \underline{u}(t), t),\ \underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix}$$

$$\underline{z}(t) = \begin{pmatrix} \underline{\dot{x}}(t) \\ \underline{y}(t) \end{pmatrix} = \underline{g}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\downarrow$$

$$\underline{\dot{x}}(t) = \underline{h}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$$\underline{y}(t) = \underline{k}(\underline{x}(t), \underline{u}(t), \underline{p}, t)$$

$\Rightarrow$ typical ODE transformation

Skip Matching and Sorting

Current Status

- additional DAE code is generated (`simflags="-daeMode"`)
- Event handling and initialization require matching and sorting
- Two options: `--daeMode=[dynamic|all]`

DAE Integration

Selected compared models

model	solver	steps	evalF	time
CascadedFirstOrder_N_6400	dae	2510	2766	3.00101
	ode	2512	3268	5.78234
DistributionSystemLinear_N_10_M_10	dae	53	149	0.0759903
	ode	73	2493	5.01925

ScaleableTestSuite DAE vs. ODE

Get your own impression: ODE mode (2017-01-12) vs. DAE mode (2017-01-13)

Plans and Outlook

- Further improvements on the DAEMode
- Develop OSI (based on FMI) for the OM runtimes
- Include the available methods to FMI/CS
- Adding CVODE integrator from SUNDIALS suite
- Further development on irksco and symSolver

- Further improvements on the DAEMode
- Develop OSI (based on FMI) for the OM runtimes
- Include the available methods to FMI/CS
- Adding CVODE integrator from SUNDIALS suite
- Further development on irksco and symSolver