Simulating MultiBody Applications with OpenModelica
– Current Status –

Christian Schubert, Dresden University

Linköping, 04.02.2013
1. Introduction

2. Feature Status

3. Performance by looking at Examples
   a. Examples
   b. Translation
   c. Compilation
   d. Simulation

4. Possible Future Directions
Models
Simulator
Applications

Toolchain

Modelica Model

OMC

FMU

SARTURIS

Motion Platform

Driver Cabin

Visuals

=> Performance is most important to us
## Required Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handling of Records, Vectors, Matrices</td>
<td>✔️</td>
</tr>
<tr>
<td>Overconstrained Connection Graph</td>
<td>✔️</td>
</tr>
<tr>
<td>Annotations: Inline, (<strong>Dymola</strong>)InlineAfterIndexReduction, derivative, noDerivative, Evaluate</td>
<td>✔️</td>
</tr>
<tr>
<td>Dynamic State Selection</td>
<td>🔄</td>
</tr>
<tr>
<td>Handling of replaceable gravityFunction</td>
<td>❌</td>
</tr>
</tbody>
</table>
### Performance/Comfort Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tearing</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Reducing size of sparse blocks</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Nonlinear Solver (Kinematic Loops)</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Analytic Jacobians</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Robust and fast Event Handling</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Support for large models</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
<tr>
<td>Visualization</td>
<td><img src="https://example.com/status.png" alt="Status" /></td>
</tr>
</tbody>
</table>
Necessary features implemented
Workaround for gravityFunction
⇒ Multibody should work
⇒ One test from MSL 3.2.1 fails: Elementary.UserDefinedGravityField

Performance/Comfort functions not finished

Let us see how they perform
## Examples

<table>
<thead>
<tr>
<th>Picture</th>
<th>RobotR3</th>
<th>EngineV6</th>
<th>Pendulum N</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Image" /></td>
<td><img src="image.png" alt="Image" /></td>
<td><img src="image.png" alt="Image" /></td>
<td><img src="image.png" alt="Image" /></td>
</tr>
<tr>
<td>#Vars</td>
<td>4921</td>
<td>12491</td>
<td>≈ 242 N</td>
</tr>
<tr>
<td>#States</td>
<td>36</td>
<td>4</td>
<td>2 N</td>
</tr>
<tr>
<td>#Nls</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Linear</td>
<td>137 -&gt; 6</td>
<td>322 -&gt; 31</td>
<td>≈ 12 N -&gt; N</td>
</tr>
</tbody>
</table>
model Pendulum_N
  constant Integer N = 2;
  inner World world;
  Revolute revolute[N];
  BodyCylinder bodyCylinder[N];
  Damper damper[N];
equation
  connect(world.frame_b, revolute[1].frame_a);
  connect(revolute.frame_b, bodyCylinder.frame_a);
  connect(damper.flange_a, revolute.support);
  connect(damper.flange_b, revolute.axis);
  connect(bodyCylinder[1:N-1].frame_b, revolute[2:N].frame_a);
end Pendulum_N;
Division into three steps

Translation → Compilation → Simulation

Modelica Model → C Source Code → Executable → Resultfile
Translation

RobotR3

EngineV6

Software

OMC 1.8.1  OMC 1.9.0.b3  Dymola

OMC 1.8.1  OMC 1.9.0.b3  Dymola

Software

 translateModel
 Templates
 Simcode
 Backend
 Frontend

≈

04.02.2013 Multibody simulation with OpenModelica
Normalized Time for Translation (with Tearing)

Time per body

Number of Bodies

0 0,2 0,4 0,6 0,8 1 1,2 1,4 1,6 1,8

10 30 50 70 90 110 130 150 170 190 210 230 250

Templates
Simcode
Backend
Frontend
• Translation lies within factor 10 of Dymola
• Translation scales with $O(N^2)$
• Problems have been identified and are being fixed
Compilation - Pendulum N

- OMC - Mingw
- OMC - Visual Studio
- Dymola - Visual Studio
• Compilation as long as translation
• Compilation limits model size
• gcc faster than Visual Studio
• much slower than Dymola
Tearing

- Deactivated by default for linear systems
  - \(+d=\) doLinearTearing
- Always treated as nonlinear system
  - \(\Rightarrow\) Numerical Jacobian
- Much faster/more robust with symbolic Jacobian
  - \(+d=\) doLinearTearing,NLSanalyticJacobian
Evaluating $x_{\text{dot}}=f(x,t)$

- Red: OMC
- Purple: OMC - Tearing
- Green: OMC - Tearing Analytical
- Blue: Dymola

Number of bodies vs. $t_{\text{RHS}}$ (ms)
EngineV6 \( f(t,x) \) equally fast

<table>
<thead>
<tr>
<th></th>
<th>OMC</th>
<th>Dymola</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time[s]</td>
<td>9.72</td>
<td>5.18</td>
</tr>
<tr>
<td>steps</td>
<td>8081</td>
<td>7912</td>
</tr>
<tr>
<td>events</td>
<td>544</td>
<td>335</td>
</tr>
<tr>
<td>F-Evaluations</td>
<td>12273</td>
<td>24548</td>
</tr>
<tr>
<td>Jacobians</td>
<td>3338</td>
<td>3215</td>
</tr>
</tbody>
</table>

Not everything is counted!
Event detection has to be improved
Pendulum with 40 bodies (f(t, x) 3x slower)

<table>
<thead>
<tr>
<th></th>
<th>OMC</th>
<th>Dymola</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time [s]</td>
<td>0.951</td>
<td>0.421</td>
</tr>
<tr>
<td>steps</td>
<td>257</td>
<td>240</td>
</tr>
<tr>
<td>events</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F-Evaluations</td>
<td>364</td>
<td>1837</td>
</tr>
<tr>
<td>Jacobians</td>
<td>14</td>
<td>17</td>
</tr>
</tbody>
</table>
Possible Future Directions

Short term goals: Finishing
• Dynamic State Selection
• Tearing
• Analytic Jacobians

Mid term goal:
• Visualization
• Clean up OpenModelica-Trac

Long term goal:

A complete redesign of the generated code should be considered
»Wissen schafft Brücken.«

Christian Schubert, Dipl.-Ing, M.Sc.
Dresden University of Technology
Institute of Processing Machines and Mobile Machines
E-Mail: christian.schubert@tu-dresden.de
Tel.: +49 351 463-39278