
Debugging

Need for Debugging Tools

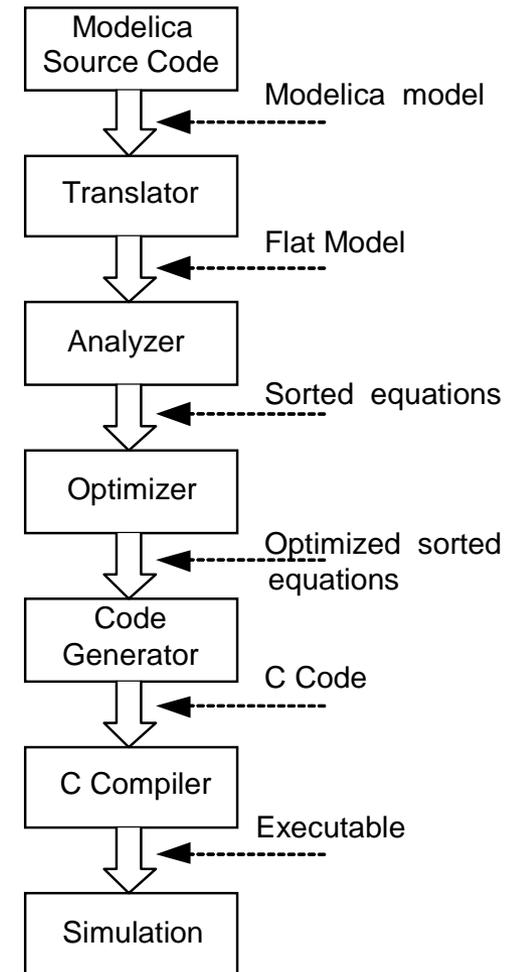
Map Low vs High Abstraction Level

- A **major part** of the total **cost** of software projects is due to testing and debugging
- US-Study 2002:
Software errors cost the US economy **annually ~60 Billion \$**
- **Problem: Large Gap in Abstraction Level**
from **Equations** to **Executable Code**
- Example error message (hard to understand)
Error solving nonlinear system 132
time = 0.002
residual[0] = 0.288956
x[0] = 1.105149
residual[1] = 17.000400
x[1] = 1.248448
...

Model Compiler Translation Phases Extended with Debugging

- Include debugging support within the translation process

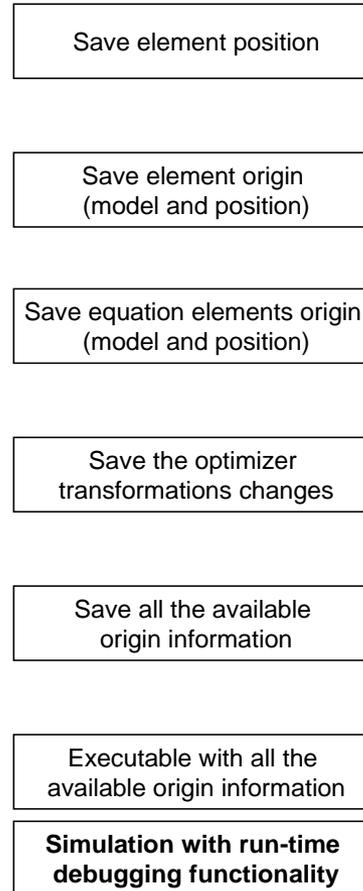
Normal Translation Process



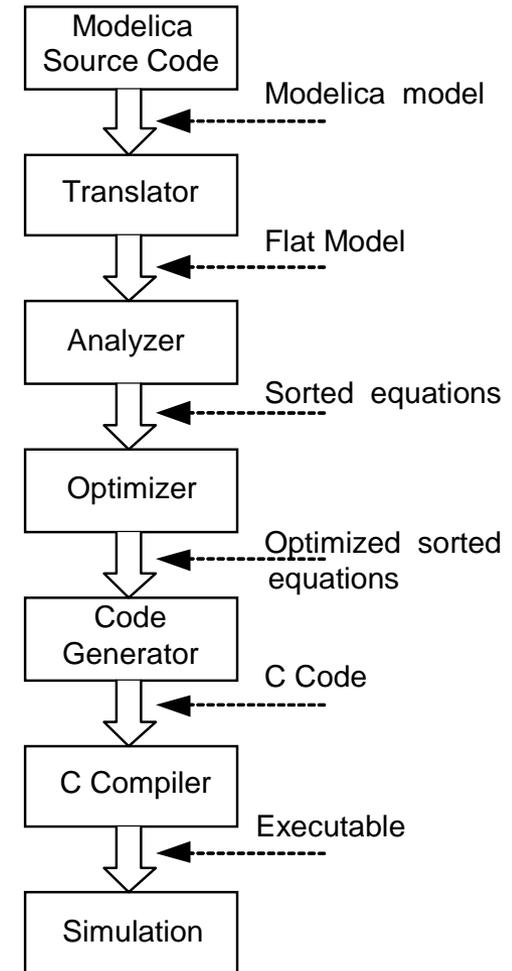
Model Compiler Translation Phases Extended with Debugging

- Additional step to provide needed debugging information

Debugging Translation Process Additional Steps



Normal Translation Process



Example Symbolic Transformations with Compiler Debug Trace

- Complicated to understand source of some errors
- Efficient trace of transformations

Example: $0 = y + \text{der}(x * \text{time} * z); \quad z = 1.0;$

(1) substitution:

$y + \text{der}(x * (\text{time} * z))$

\Rightarrow

$y + \text{der}(x * (\text{time} * 1.0))$

(2) simplify:

$y + \text{der}(x * (\text{time} * 1.0))$

\Rightarrow

$y + \text{der}(x * \text{time})$

(3) expand derivative

(symbolic diff):

$y + \text{der}(x * \text{time})$

\Rightarrow

$y + (x + \text{der}(x) * \text{time})$

(4) solve:

$0.0 = y + (x + \text{der}(x) * \text{time})$

\Rightarrow

$\text{der}(x) = ((-y) - x) / \text{time}$

Properties of Transformation Trace

- Most equations have very **few** transformations on them
- Most of the interesting equations have a few
 - Still rather readable
- Some extra care to handle Modelica variable aliasing
- Very **efficient** implementation, max 1% overhead

MSL 3.1 MultiBody DoublePendulum

# Ops	Frequency	Comment
0	457	Parameters
1	89	Dummy eq & know var
2	720	Alias vars
3	479	Alias vars
4	124	Alias after simplify
5	25	Alias after simplify
6	99	Alias after simplify
7	55	Scalar eq
8	37	...
9	110	...
10	72	...
11	12	...
12	25	...
13	35	...
14	3	Known constant after many replacements
21	27	World object (3x3 matrix with many occurrences of aliased vars)

OpenModelica Equation Model Debugger

The screenshot displays the OMEdit - Transformational Debugger interface. It is divided into three main panes: Variables View, Equations View, and Source View.

- Variables View:** Shows a tree structure of variables (frame, boxBody1, body, frame_a, R, T) and their definitions in equations. It includes a search filter and expand/collapse buttons.
- Equations View:** Shows a list of equations with their indices and types. Below it, the 'Equation Operations' pane shows the current equation being processed, including substitution and simplification steps.
- Source View:** Shows the source code of the model, with line numbers 317 to 331. The code includes comments and mathematical expressions for frame transformations.

Showing equation transformations of a model:

```
0 = y + der(x * time * z); z = 1.0;
```

(1) **substitution:**
 $y + \text{der}(x * (\text{time} * z))$
 \Rightarrow
 $y + \text{der}(x * (\text{time} * 1.0))$

(2) **simplify:**
 $y + \text{der}(x * (\text{time} * 1.0))$
 \Rightarrow
 $y + \text{der}(x * \text{time})$

(3) **expand derivative (symbolic diff):**
 $y + \text{der}(x * \text{time})$
 $\Rightarrow y + (x + \text{der}(x) * \text{time})$

(4) **solve:**
 $0.0 = y + (x + \text{der}(x) * \text{time})$
 \Rightarrow
 $\text{der}(x) = ((-y) - x) / \text{time}$
 $\text{time} < 0$

Mapping run-time error to source model position

Transformations Browser – EngineV6 Overview (11 116 equations in model)

The screenshot displays the OMEdit - Transformational Debugger interface. The top bar shows the file path: /tmp/OpenModelica_marsj/OMEdit/Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6_info.xml. The interface is divided into several panels:

- Variables Browser:** Shows a list of variables including phi, phi_offset, Crank1, and body. The phi variable is selected.
- Defined in Equations:** A table showing the definition of the selected variable. It lists two equations: an initial condition at index 587 and a regular assignment at index 5016.
- Used In Equations:** A table showing where the selected variable is used. It lists several regular assignment equations involving cylinder3.B2.phi and its derivatives.
- Equations Browser:** A table listing all equations in the model. The equation for the derivative of phi is selected.
- Source Browser:** Shows the source code for the selected equation, including assertions for frame connectivity and rotation calculations.

The Source Browser shows the following code snippet:

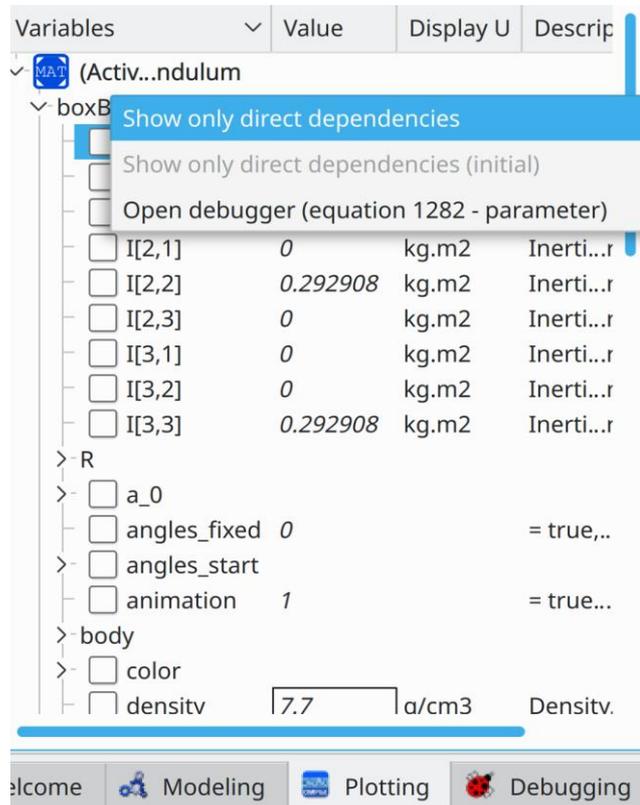
```
386 Connections.branch(frame_a.R,  
387 frame_b.R);  
388 assert(cardinality(frame_a) > 0,  
389 "Connector frame a of Revolute  
390 joint is not connected");  
391 assert(cardinality(frame_b) > 0,  
392 "Connector frame b of Revolute  
393 joint is not connected");  
394  
395 angle = phi_offset + phi;  
396 w = der(phi);  
397 a = der(w);  
398  
399 // relationships between quantities  
400 of frame_a and of frame_b  
401 frame_b.r_theta = frame_a.r_theta;  
402  
403 if rooted(frame_a.R) then  
404 R_rel = Frames.planarRotation(e,  
405 phi_offset + phi, w);  
406 frame_b.R =  
407 Frames.absoluteRotation(frame_a.R,  
408 R_rel);  
409 frame_a.f = -  
410 Frames.resolve(R_rel, frame_b.f);  
411 frame_a.t = -  
412 Frames.resolve(R_rel, frame_b.t);  
413 else  
414 R_rel = Frames.planarRotation(-e,  
415 phi_offset + phi, w);  
416 frame_a.R =  
417 Frames.absoluteRotation(frame_b.R,  
418 R_rel);  
419 frame_b.f = -  
420 Frames.resolve(R_rel, frame_a.f);  
421 frame_b.t = -  
422 Frames.resolve(R_rel, frame_a.t);  
423 end if;  
424  
425 // d'Alembert's principle  
426 tau = -frame_b.t*e;  
427  
428 // Connection to internal
```

Equation Model Debugger on Siemens Model (Siemens Evaporator test model, 1100 equations)

The screenshot displays the OMEdit - Transformational Debugger interface. The main window shows the source code of a Siemens evaporator test model, which consists of 1100 equations. A callout box with a black border and white background points to a specific equation in the source code: `y = u1/u2;`. The callout text reads: "Pointing out the buggy equation y = u1/u2; that gives division by zero". The source code is displayed in a monospaced font, and the equation is highlighted in blue. The interface also shows a Variables Browser on the left, an Equations Browser at the bottom, and a Source Browser on the right. The Variables Browser lists variables such as Scse1, Scse2, Scse3, Scse4, Scse5, Scse6, Evap, FGflow, FGINv, FGINVu, FGoutV, FGoutVu, FGsink, FGsource, FGtemp, FWinletV, FWinletVu, FWPump, FWPump_h, FWPump_p, SteamSink, and system. The Equations Browser lists equations such as Evap.Hex.Ln.C.us[1], Evap.Hex.Ln.C.flowModel.ms[1], Evap.Hex.Ln.C.mediums[1].sat.Ts, Evap.Hex.Ln.C.mediums[2].p_bar, Evap.Hex.Ln.C.statesFM[2].phase, Evap.Hex.Ln.C.heatTransfer.Ts[2], Evap.Hex.Ln.C.mediums[2].T_degC, Evap.Hex.Ln.C.statesFM[2].d, Evap.Hex.Ln.C.ms[2], Evap.Hex.Ln.C.fluidVolume, Evap.Hex.Ln.C.mediums[2].u, Evap.Hex.Ln.C.us[2], Evap.Hex.Ln.C.flowModel.ms[2], Evap.Hex.Ln.C.mediums[2].sat.Ts, Evap.Hex.Ln.C.flowModel.Fs.pl[1], Evap.Hex.Ln.C.flowModel.dps.fgl[1], Evap.Hex.Ln.C.ms.flows[2], Evap.Hex.Ln.C.flowModel.ms.act[1], Evap.Hex.Ln.C.ms.flows[2], Evap.Hex.Ln.C.H.flows[2], Evap.Hex.Ln.C.flowModel.lrhos.act[1], Evap.Hex.Ln.C.state_s.T, Evap.Hex.Ln.C.state_a.d, Evap.Hex.Ln.C.state_b.T, Evap.Hex.Ln.C.state_b.d, Evap.Hex.Ln.C.Arey.y, Evap.Hex.Ln.Q2.area.offse, Evap.Hex.Ln.Q2.division2.u2, Evap.Hex.Ln.Q2.kinn, Evap.Hex.Ln.Q2.add1.v, Evap.Hex.Ln.Q2.add1.k1.

New OM Debug function that can trace (and plot) which variables and equations influence a variable

New menu choice to show direct dependencies

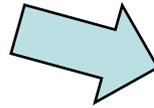


The screenshot shows the OpenModelica GUI with a context menu open over the 'boxBody1' component. The menu options are:

- Show only direct dependencies (highlighted)
- Show only direct dependencies (initial)
- Open debugger (equation 1282 - parameter)

The background table shows the following variables and their values:

Variables	Value	Display U	Descript
<input type="checkbox"/> I[2,1]	0	kg.m2	Inerti...r
<input type="checkbox"/> I[2,2]	0.292908	kg.m2	Inerti...r
<input type="checkbox"/> I[2,3]	0	kg.m2	Inerti...r
<input type="checkbox"/> I[3,1]	0	kg.m2	Inerti...r
<input type="checkbox"/> I[3,2]	0	kg.m2	Inerti...r
<input type="checkbox"/> I[3,3]	0.292908	kg.m2	Inerti...r
> R			
> <input type="checkbox"/> a_0			
<input type="checkbox"/> angles_fixed	0	= true...	
> <input type="checkbox"/> angles_start			
<input type="checkbox"/> animation	1	= true...	
> body			
> <input type="checkbox"/> color			
<input type="checkbox"/> densitiv	7.7	a/cm3	Densitiv.

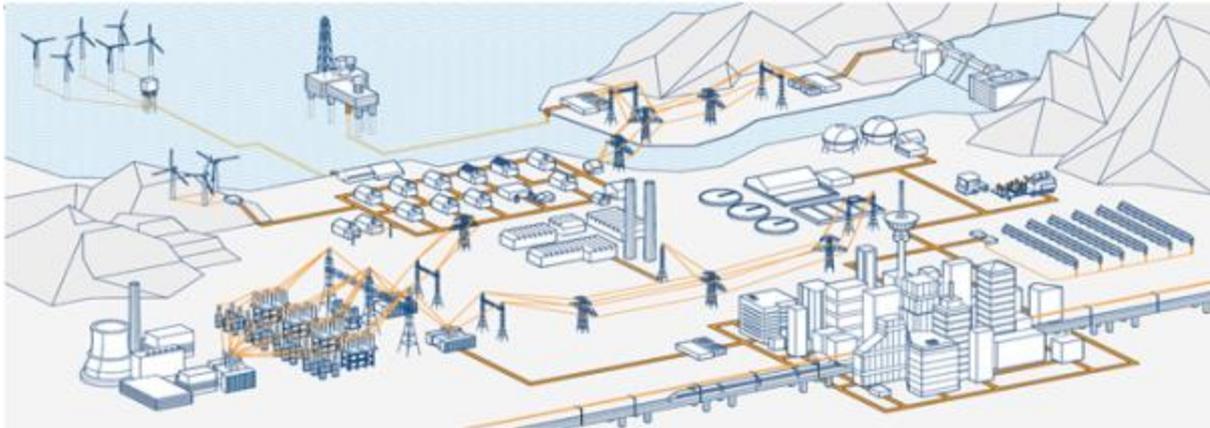


List of Variables directly influencing:

Variables	Value	Display U	Description
<input checked="" type="checkbox"/> (Activ...ndulum			
> <input type="checkbox"/> boxBody1			
<input type="checkbox"/> I[1,1]	0.008316	kg.m2	Inerti...ram
<input type="checkbox"/> height	0.06	m	Height of b
<input type="checkbox"/> innerHeight	0	m	Height...eig
<input type="checkbox"/> mi	0	kg	Mass o...of
<input type="checkbox"/> mo	13.86	kg	Mass o...t h
<input type="checkbox"/> width	0.06	m	Width of b
> <input type="checkbox"/> width...ction			

ABB Industry Use of OpenModelica Debugger

- ABB OPTIMAX® provides advanced model based control products for power generation and water utilities



- ABB: *“ABB uses several compatible Modelica tools, including OpenModelica, depending on specific application needs.”*
- ABB: *“OpenModelica provides outstanding debugging features that help to save a lot of time during model development.”*

Equation Debugging Summary

- Debugging **equation-based** models present new **challenges**
- **Equation** systems are **transformed** symbolically to a form hard for the user to understand
- Maintain and **explain** a **mapping** between the **low** level and the **high** level model
- **The first integrated static/dynamic debugger of any Modelica tool**

Debugging Example – Detecting Source of Chattering (excessive event switching) causing bad performance

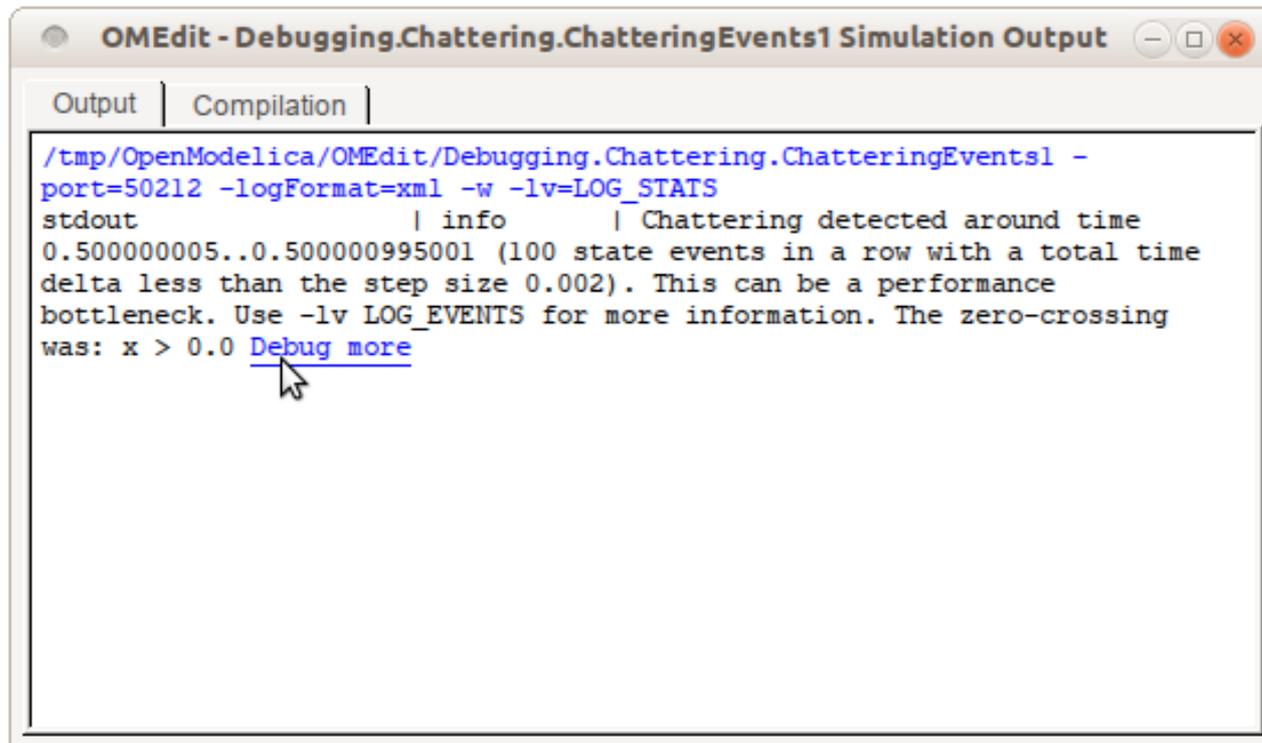
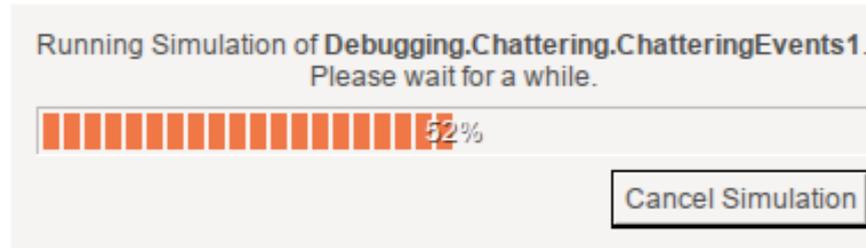
The screenshot displays the OMEdit - Transformational Debugger interface. The main window shows the source code of a model, with line 11 highlighted: `z = if x > 0 then -1`. A black arrow points from the 'Equations Browser' table to this line. The 'Equations Browser' table lists several equations, with equation 5 highlighted: `(assignment) z = if x > 0 then -1 else 1.0`. The 'Variables' table shows variables x, y, and z, with z highlighted. The 'Equation Operations' section shows the flattened equation: `solved: z = if x > 0 then -1.0 else 1.0` and `original: z = if x > 0 then -1 else 1; => flattened: z = if x > 0 then -1.0 else 1.0;`. The 'Source Browser' shows the model code, including the event-triggering line 11.

Variables			Defined In Equations			Used In Equations		
Inc	Type	Equation	Inc	Type	Equation	Inc	Type	Equation
-2	initial	(assignment) z = 1.0	-3	initial	(assignment) y = 2.0 * z	-1	initial	(assignment) x = 1.0
-5	regular	(assignment) z = if x > 0 then -1 else 1.0	-6	regular	(assignment) y = 2.0 * z	-2	initial	(assignment) z = 1.0

Equations Browser			Defines		Depends	
Inc	Type	Equation	Variable	Variable	Variable	Variable
-1	initial	(assignment) x = 1.0	z	x		
-2	initial	(assignment) z = 1.0				
-3	initial	(assignment) y = 2.0 * z				
-4	initial	(assignment) der(x) = y				
-5	regular	(assignment) z = if x > 0 then -1 else 1.0				
-6	regular	(assignment) y = 2.0 * z				
-7	regular	(assignment) der(x) = y				

```
1 within ;
2 package Debugging "Test
3 cases for debugging of
4 declarative models"
5
6 package Chattering "Models
7 with chattering behaviour"
8 model ChatteringEvents1
9 "Exhibits chattering
10 after t = 0.5, with
11 generated events"
12 Real x(start=1,
13 fixed=true);
14 Real y;
15 Real z;
16 equation
17 z = if x > 0 then -1
18 else 1;
19 y = 2*z;
20 der(x) = y;
21 annotation
22 (Documentation(info="<html>
23 <p>After t = 0.5, chattering
24 takes place, due to the
25 discontinuity in the right
26 hand side of the first
27 equation.</p>
28 <p>Chattering can be
29 detected because lots of
30 tightly spaced events are
31 generated. The feedback to
32 the user should allow to
33 identify the equation from
34 which the zero crossing
35 function that generates the
36 events originates.</p>
37 </html>"),
38 experiment(StopTime=1));
39 end ChatteringEvents1;
40
41 model ChatteringEvents2
42 "Exhibits chattering
43 after t = 0.422, with
44 generated events"
```

Error Indication – Simulation Slows Down



Exercise – Equation-based Model Debugger

In the model ChatteringEvents1, chattering takes place after $t = 0.5$, due to the discontinuity in the right hand side of the first equation. Chattering can be detected because lots of tightly spaced events are generated. The debugger allows to identify the (faulty) equation that gives rise to all the zero crossing events.

```
model ChatteringEvents1
  Real x(start=1, fixed=true);
  Real y;
  Real z;
equation
  z = noEvent(if x > 0 then -1 else 1);
  y = 2*z;
  der(x) = y;
end ChatteringNoEvents1;
```

Uses 25% CPU

acrotray.exe *32	petfr27	00	976 K	A
AdobeARM.exe *32	petfr27	00	1,136 K	A
Bootcamp.exe	petfr27	00	1,448 K	B
conhost.exe	petfr27	00	1,300 K	C
csrss.exe		00	3,000 K	
DCSHelper.exe *32	petfr27	00	660 K	D
Debugging.Chattering...	petfr27	25	1,436 K	D
dllhost.exe	petfr27	00	2,224 K	C

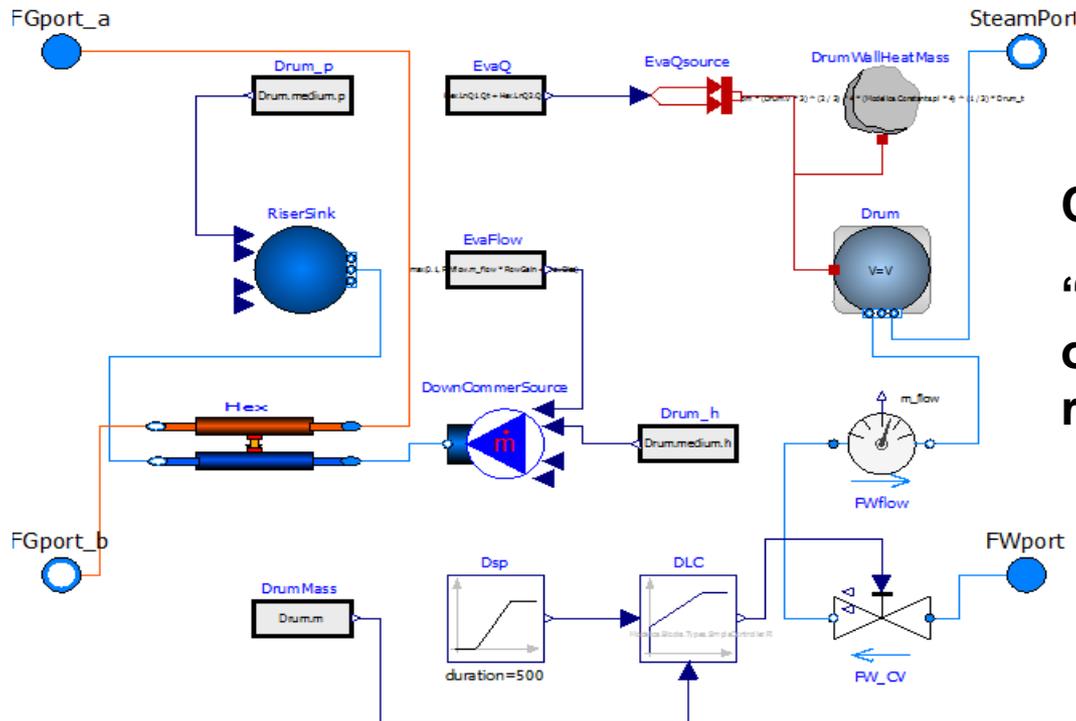
- Switch to OMEdit text view (click on text button upper left)
- Open the Debugging.mo package file using OMEdit
- Open subpackage Chattering, then open model ChatteringEvents1
- Simulate in debug mode (transformational debugger)
- Click on the button Debug more (see prev. slide)
- Possibly start task manager and look at CPU. Then click stop simulation button

Performance Analysis

Performance Profiling for faster Simulation

(Here: Profiling equations of Siemens Drum boiler model with evaporator)

- Measuring **performance** of equation blocks to find bottlenecks
 - Useful as input before model simplification for real-time applications
- Integrated with the debugger to **point out the slow equations**
- Suitable for **real-time profiling** (collect less information), or a complete view of all equation blocks and function calls

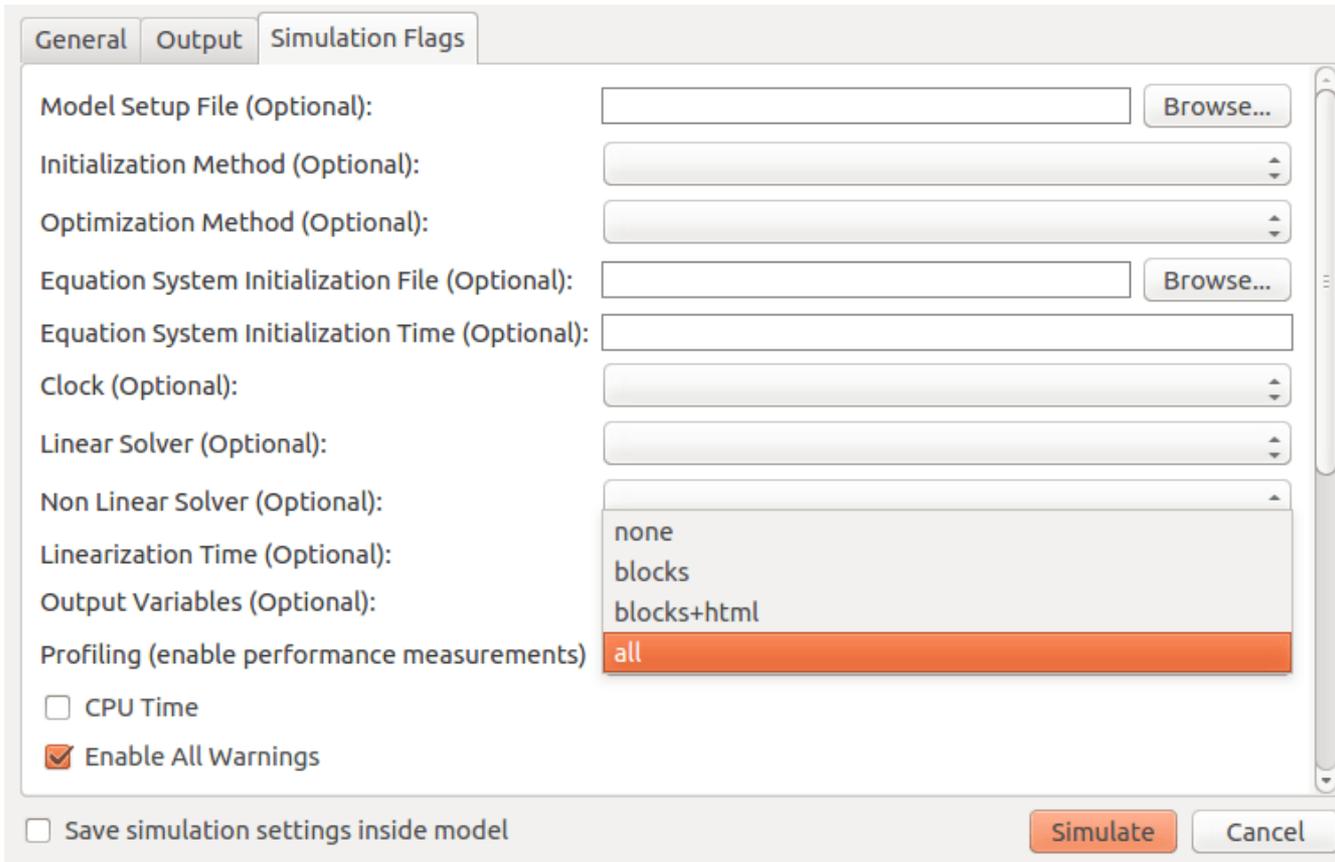


Conclusion from the evaluation:

“...the profiler makes the process of performance optimization radically shorter.”

Using the Performance Profiler on DoublePendulum

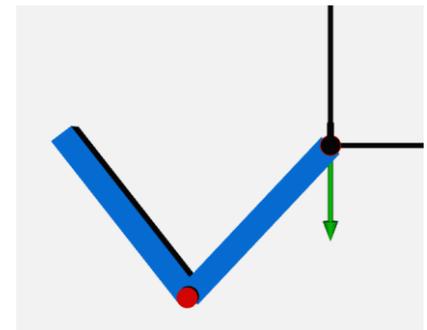
- When running a simulation from OMEdit, it is possible to enable profiling information, which can be combined with the [transformations browser](#).



Set this in
simulation
Setup



DoublePendulum in
MultiBody library



Using the Performance Profiler on the DoublePendulum model

- When profiling the DoublePendulum example from MSL, the following output below is a typical result. This information clearly shows which system takes longest to simulate (a linear system, where most of the time overhead probably comes from initializing LAPACK over and over).

Equations Browser							Defines
Index	Type	Equation	Executions	Max time	Time	Fraction ▲	Variable ▼
+ 876	regular	linear, size 2	4602	0.000199	0.0582	86.2%	damper.a_rel revolute2.frame_b.f[2]
- 836	regular	(assignment) revolute2.R_rel.T[2,2] = cos(revolute2.phi)	1534	8.25e-05	0.000491	0.728%	
- 837	regular	(assignment) revolute2.R_rel.T[2,1] = -sin(revolute2.phi)	1534	7.29e-05	0.000422	0.625%	
- 841	regular	(assignment) boxBody1.frame_...[2,1] = -sin(damper.phi_rel)	1534	7.1e-05	0.000395	0.585%	
- 840	regular	(assignment) boxBody1.frame_...T[2,2] = cos(damper.phi_rel)	1534	7.08e-05	0.000361	0.535%	
- 839	regular	(assignment) revolute2.R_rel.T[1,1] = cos(revolute2.phi)	1534	7.33e-05	0.000303	0.449%	
- 842	regular	(assignment) boxBody1.frame_b.R.T[1,2] = sin(damper.phi_rel)	1534	7.45e-05	0.000303	0.449%	
- 838	regular	(assignment) revolute2.R_rel.T[1,2] = sin(revolute2.phi)	1534	7.11e-05	0.0003	0.444%	
- 849	regular	(assignment) boxBody1.frame_...T[1,1] = cos(damper.phi_rel)	1534	7.29e-05	0.000286	0.424%	
- 827	regular	(assignment) revolute1.tau = (-damper.d) * revolute1.w	1534	6.84e-05	0.000274	0.406%	

Performance Profiler Exercise

- Try the profiler on this model. Results in Equations Browser, enlarge the window, click on Fraction to sort in ascending/descending order.

```
model ProfilingTest
  function f
    input Real r;
    output Real o = sin(r);
  end f;
  String s = "abc";
  Real x = f(x) "This is x";
  Real y(start=1);
  Real z1 = cos(z2);
  Real z2 = sin(z1);
equation
  der(y) = time;
end ProfilingTest;
```

Index	Type	Equation	Execution	Max time	Time	Fraction
21	regular	non-linear, unkno...tion variables: 1	1006	7.24e-05	0.000664	44.6%
19	regular	non-linear (torn),...ation variables: 1	1006	7.87e-05	0.000635	42.7%
22	regular	(assign) der(y) := time	540	8e-07	1.71e-05	1.15%
9	initial	non-linear (torn),...ation variables: 1	8	5.3e-06	5.3e-06	0.356%
11	initial	non-linear, unkno...tion variables: 1	3	1.5e-06	1.5e-06	0.101%
2	initial	(assign) s := "abc"	2	2e-07	2e-07	0.0134%
1	initial	(assign) y := \$START.y	1	1e-07	1e-07	0.00672%
12	initial	(alias) 22	0	0	0	0%
23	parameter	(alias) 2	0	0	0	0%