



TROUBLESHOOTING

Lecture 2.4

Modelon

OVERVIEW



Development - Best practice



Troubleshooting



Diagnostics



DEVELOPMENT

BEST PRACTICE

SPECIFY

- When creating a new component or system, start by defining (at least) one test case that specifies how the model should function:
 - It should excite your model in a way you easily can verify
 - For a spring, plot the force compression curve
 - For some configurations, the result can be compared to an equivalent test using another model
 - If it is a nonlinear spring, parameterize as linear and compare with a standard linear spring.
 - It excites the whole operating region of the model
 - Make sure that the spring is tested for both rebound and compression
 - It verifies the behavior under different causality conditions
 - For a nonlinear gear, make sure that the results and the generated code is consistent, independent of which flange is dependent on the other.

IMPLEMENT

- Take advantage of the benefits of component oriented modeling, separate code into reusable primitive models
 - Use a base class that defines the relations between connectors, so that only the force-compression relation has to be given for each new spring model
- Use types, units, enumerations and a stringent nomenclature
 - “if spring_type == Types.Spring.NONLINEAR then ...” is easier to maintain and read than “if spring_type == 1 then ...”
 - For a spring compatible with Modelica.Mechanics.Translational, use s for position, v for velocity, a for acceleration and f for force.
- Define a model, either with components and connections, or with equations/algorithms.
 - A mix between the concepts makes it much harder to understand and debug

MAINTENANCE

- Version your models
 - Use a version management system, e.g. SVN, Git to be able to back-track changes
- Collect test cases and reference trajectories systematically
 - Once a model is finalized and the test results are fine, store the cases and the results for future reference
- Rerun the tests to detect unexpected changes/errors as early as possible
 - A good practice is to rerun test either on a regular basis or when a change has been made to the code
- There are tools to support automation of the maintenance
 - Modelon's Model Testing Toolkit - MTT (cross platform regression testing)
 - Integrate with Jenkins to automate the testing procedure



TROUBLESHOOTING

TRANSLATION PROBLEMS

- Case: Cannot translate the model.
- Common issues:
 - Missing class or parameter declaration
 - Modifiers on missing parameters or components
 - Missing connection
 - Number of equations and variables do not match
 - Modelica syntax problems
 - Index reduction problems
 - Overspecified number of initial values
 - Overspecified number of states

FINDING THE PROBLEM

- Read the Translation log
 - Be aware of any Warnings given in the Translation log
 - Try and see if you can locate the part of the model that does not work
- Check your model
 - Also try and check subparts of your model
- When you isolate the part of the model that does not translate, it might be a good idea to create a smaller test model for that part.
- If the error is related to Modelica syntax, refer to:
 - Modelica Language Specification

SIMULATION PROBLEMS

- Case: The model translates but does not run properly.
- Common issues:
 - Initialization problems
 - Convergence problems
 - Solver finds the wrong initialization point
 - Simulation does not finish due to:
 - Convergence problems in nonlinear systems
 - Singularities
 - Too stiff, stepsize goes to zero
 - Division by zero
 - Unphysical results
 - Missing parameter propagation
 - Using wrong units
 - Performance
 - Simulation speed too slow



DIAGNOSTICS

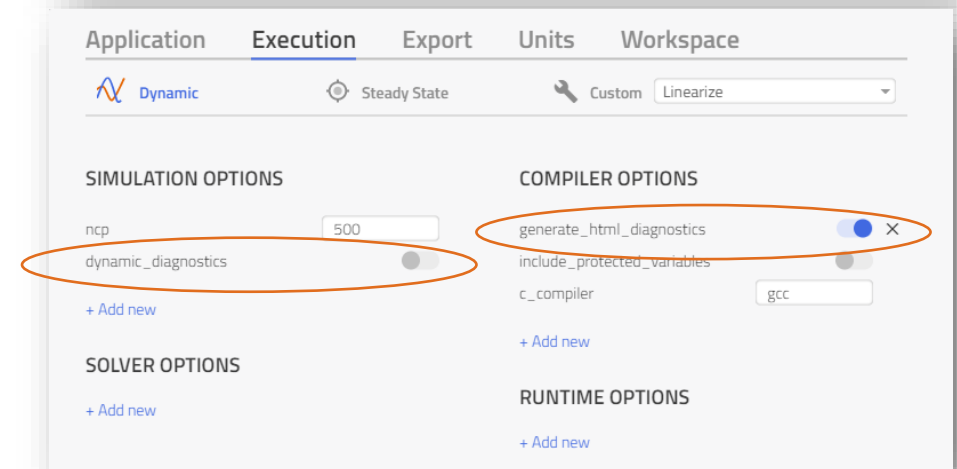
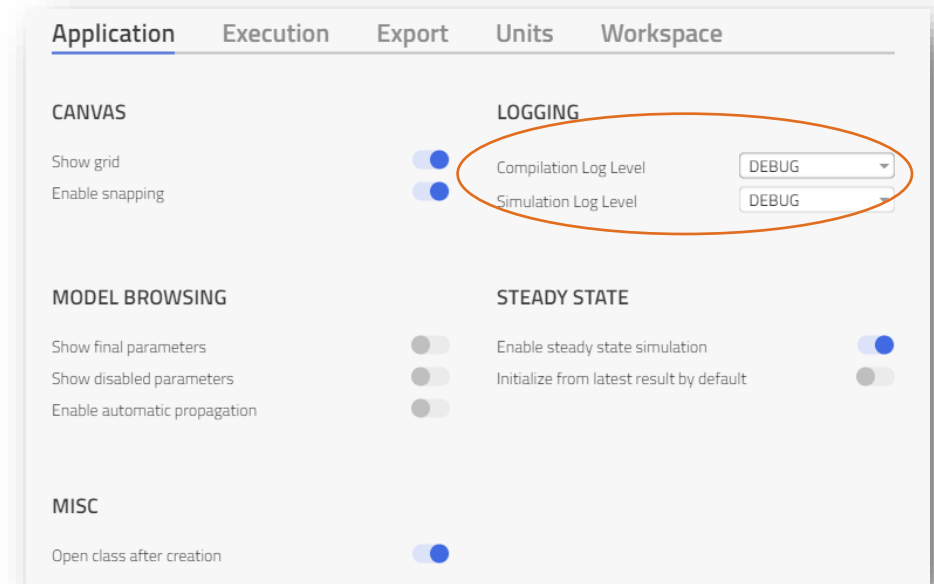
SETTINGS THAT CONTROL DIAGNOSTICS

Application tab

- “Compiler Log Level” and “Simulation Log Level”
 - Controls amount of information written to logs

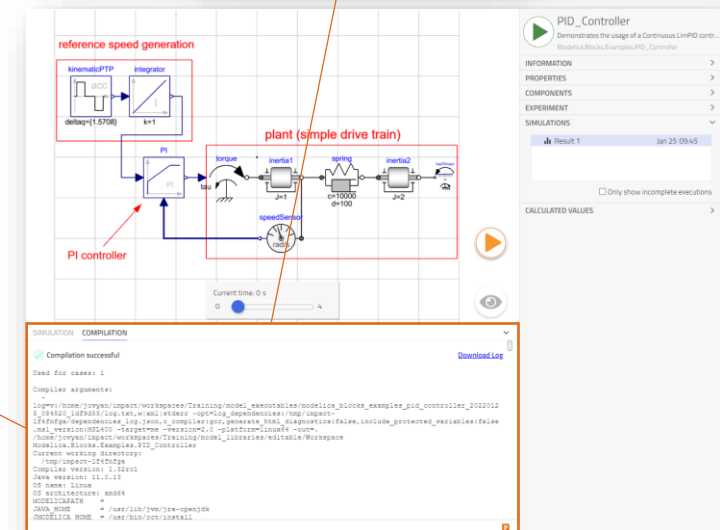
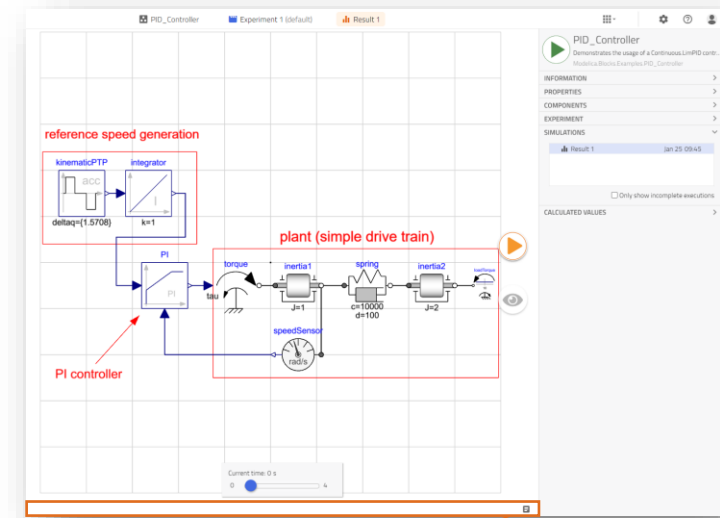
Execution tab

- “generate_html_diagnostics”
 - Generates browsable pages containing information about the compilation
- “dynamic_diagnostics”
 - Adds additional diagnostic variables to the result



ACCESSING LOGS

- After Compilation and Simulation have finished, compilation and simulation logs can be accessed by right clicking the result entry
- They also can be accessed from the logs bar at the bottom of the canvas



SIMULATION LOG

- Contains selected solver options and run statistics from the simulation if simulation run successfully.
- In case of simulation failure, it will give info about the error occurred.
- Errors found here includes:
 - Initialization failures
 - Convergence failures
 - Division by zero
 - Assertion errors

```
SIMULATION  COMPILATION
Simulation successful
Download Log

Log for: Case 1

Final Run Statistics: ---

Number of steps                : 710
Number of function evaluations  : 802
Number of Jacobian evaluations  : 16
Number of function eval. due to Jacobian eval. : 96
Number of error test failures   : 4
Number of nonlinear iterations  : 774
Number of nonlinear convergence failures : 0
Number of state function evaluations : 730
Number of state events          : 2
Number of time events           : 4

Solver options:

Solver                : CVode
Linear multistep method : BDF
Nonlinear solver       : Newton
Linear solver type     : DENSE
Maximal order          : 5
Tolerances (absolute) : [1.e-06 1.e-06 1.e-06 1.e-10 1.e-06 1.e-06]
Tolerances (relative)  : 1e-06

Simulation interval    : 0.0 - 4.0 seconds.
Elapsed simulation time: 0.0929769229987869 seconds.
```

```
SIMULATION  COMPILATION
Simulation failed
Download Log

Log for: Case 1

[Assertion Error]: msg = "
Temperature T (= 173.15 K) is not
in the allowed range (272.15 K <= T <= 403.15 K)
required from medium model "SimpleLiquidWater".
"

Initialization failed.
```

COMPILATION LOG

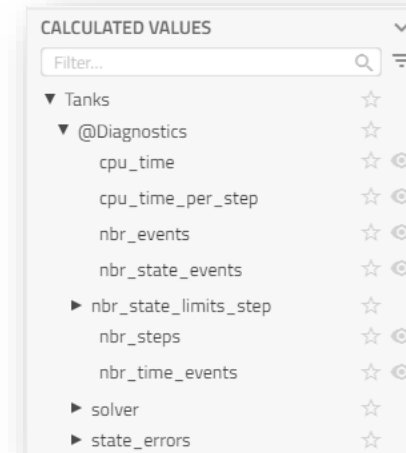
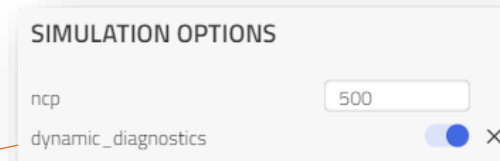
- Contains output from the compilation process according to log level set.
- Errors found here includes:
 - Syntax errors
 - Structural errors
 - Missing connections
 - Number of equations and variables do not match
 - Index reduction problems
 - Over specified number of initial values

```
SIMULATION  COMPILATION
! Compilation failed
Used for cases: 1
Error in flattened model:
  Index reduction failed: There were unmatched equations and/or variables left after index
  reduction.
Error in flattened model:
  The system is structurally singular. The following variable(s) could not be matched to any
  equation:
  der(spring.w_rel)
  PI.u_s
The following equation(s) could not be matched to any variable:
  spring.phi_rel = inertia2.phi - inertia1.phi
Download Log
```

DYNAMIC DIAGNOSTICS

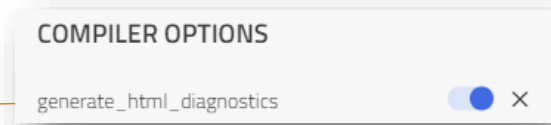
If “dynamic_diagnostics” is activated:

- Extra diagnostic variables are added to the result:
 - **cpu_time**: elapsed cpu time
 - **nbr_events**: number of events triggered
 - **nbr_state_limits_step**: number of times the error of a step limited the step size of the solver.
 - **state_errors**: error estimates of all state variables
- Like any other variable, drag the diagnostic variable to canvas to plot or add a sticky.

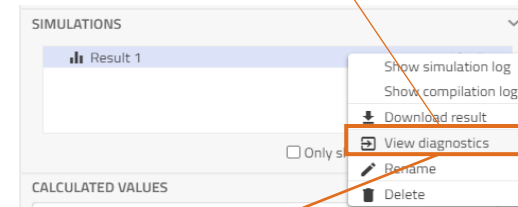


ADVANCED MODEL DIAGNOSTICS

If “generate_html_diagnostics” is activated:



- Additional compilation diagnostics and information is available from “View diagnostics”:



Model Statistics for Workspace.Tanks

Note: Some diagnostics are omitted since the compiled model contains encrypted components

6 warnings 0 errors

- ▶ Model Structure
- ▶ State variables 4
- ▶ Initialization equation blocks 0 linear / 1 non-linear
- ▶ Equation blocks 0 linear / 1 non-linear

ADVANCED MODEL DIAGNOSTICS: OVERVIEW

- Lists all compiler errors and warnings that occurred during compilation (same as compilation log).
- Information about the model structure such as number of equations, variables etc.
- List of selected state variables
- Information about initialization equation system
- Information about equation systems in model

Model Statistics for Workspace.Tanks

Note: Some diagnostics are omitted since the compiled model contains encrypted components

6 warnings 0 errors

▶ Model Structure	
▶ State variables	4
▶ Initialization equation blocks	0 linear / 1 non-linear
▶ Equation blocks	0 linear / 1 non-linear

ADVANCED MODEL DIAGNOSTICS: MODEL STRUCTURE

Model Structure:

- Summary statistics of the compiled model after transformation

Full statistics:

- Full model statistics from before and after transformation.

Model before transformation:

- Summarizes model statistics of the flattened model.

Model after transformation:

- Summarizes the statistics of the model after the compiler has completed the transformation work (alias elimination etc.)

Flattened model code:

- Displayed flattened model code. The flattening process includes:
 - Expanding inherited classes (extend statements)
 - Replacing connect statements with actual equations
 - Etc.

▼ Model Structure

Total size	219
Constants	10
Parameters	143
Variables	66
Full statistics	
Flattened model code	

ADVANCED MODEL DIAGNOSTICS: STATE VARIABLES

- Lists selected state variables
- Check so that the states selected are the expected ones. See lecture 2.2 for some advice about state selection.

▼ State variables		4
Continuous		4
	State variable	nominal
0	tank.level	-
1	tank.medium.T	300
2	tank2.level	-
3	tank2.medium.T	300
Discrete		0

ADVANCED MODEL DIAGNOSTICS: EQUATION BLOCKS

(Initialization) Equation blocks:

- Displays detailed information about existing equation blocks in the model.
 - Non-linear equation blocks require iterative algorithms to be solved which effect both simulation speed and robustness (might not be able to find solution)
 - If possible, these should be kept to a minimum

Modelica text & Interactive matrix

- All equations and equation blocks are listed and sorted in the order in which they are calculated, i.e., by
 - Can be viewed both in text form (**Modelica text**) and block-lower triangular form (**Interactive matrix**)

▼ Initialization equation blocks 0 linear / 1 non-linear
0 [Linear](#) initialization equation blocks
Block sizes: []
1 [Non-linear](#) initialization equation blocks
Block sizes: [5]
Modelica text
Interactive matrix

▼ Equation blocks 0 linear / 1 non-linear
0 [Linear](#) equation blocks
Block sizes: []
1 [Non-linear](#) equation blocks
Block sizes: [5]
Modelica text
Interactive matrix

Hover to get details about equation block

▼ Equation blocks Non Linear Equation System Block 1

Iteration variable	start	min	max	nominal
0 tank2.s[1]	tank2.fluidLevel_max	-	-	-
1 tank.s[1]	tank.fluidLevel_max	-	-	-
2 pipe.port_b.p	100000.0	0	1.0E8	100000.0
3 pipe.port_a.p	100000.0	0	1.0E8	100000.0
4 pipe2.port_b.p	100000.0	0	1.0E8	100000.0

WORKSHOP 2.4

In this workshop you will:

- Look at a simple steady state example
- Inspect the model using “HTML diagnostics”