WORKSHOP

Initialization and Debugging in Modelon Impact

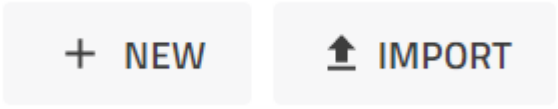
Contents

Introduction

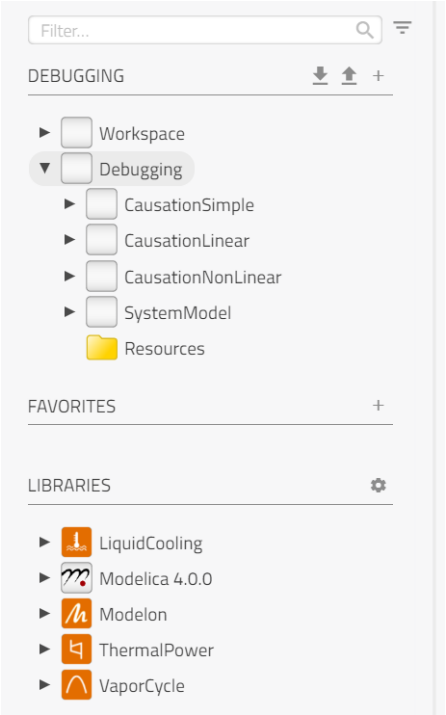
In this workshop, you will familiarize yourself with the some of the debugging tools available in Modelon Impact, that will allow you to understand the models states, linear and non-linear systems.

Import the workspace

1. In the training content you downloaded for Innovate, find the Debugging.zip workspace file.
2. On the Modelon Impact landing page, click “Import” and choose the Debugging.zip workspace file.



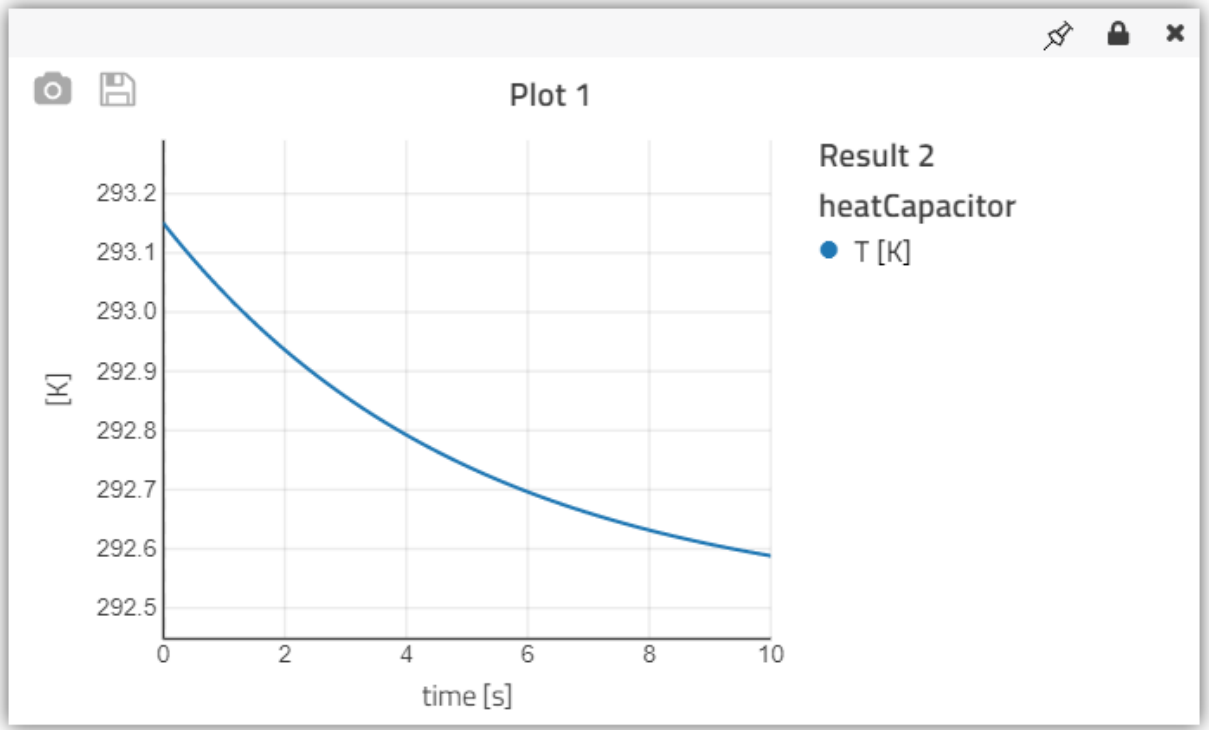
3. You are now set to start the workshop!



Simulating a dynamic model

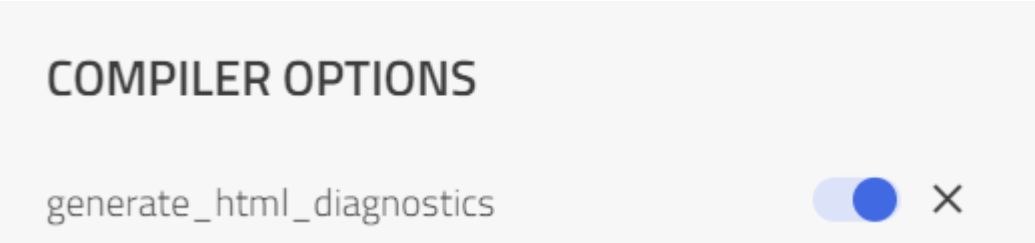
We will start by investigating a simple dynamic model.

1. Open the model “Debugging.CausationSimple.ThermalWithState”.
2. Press the Simulate button.
3. Plot the heatCapacitor.T variable



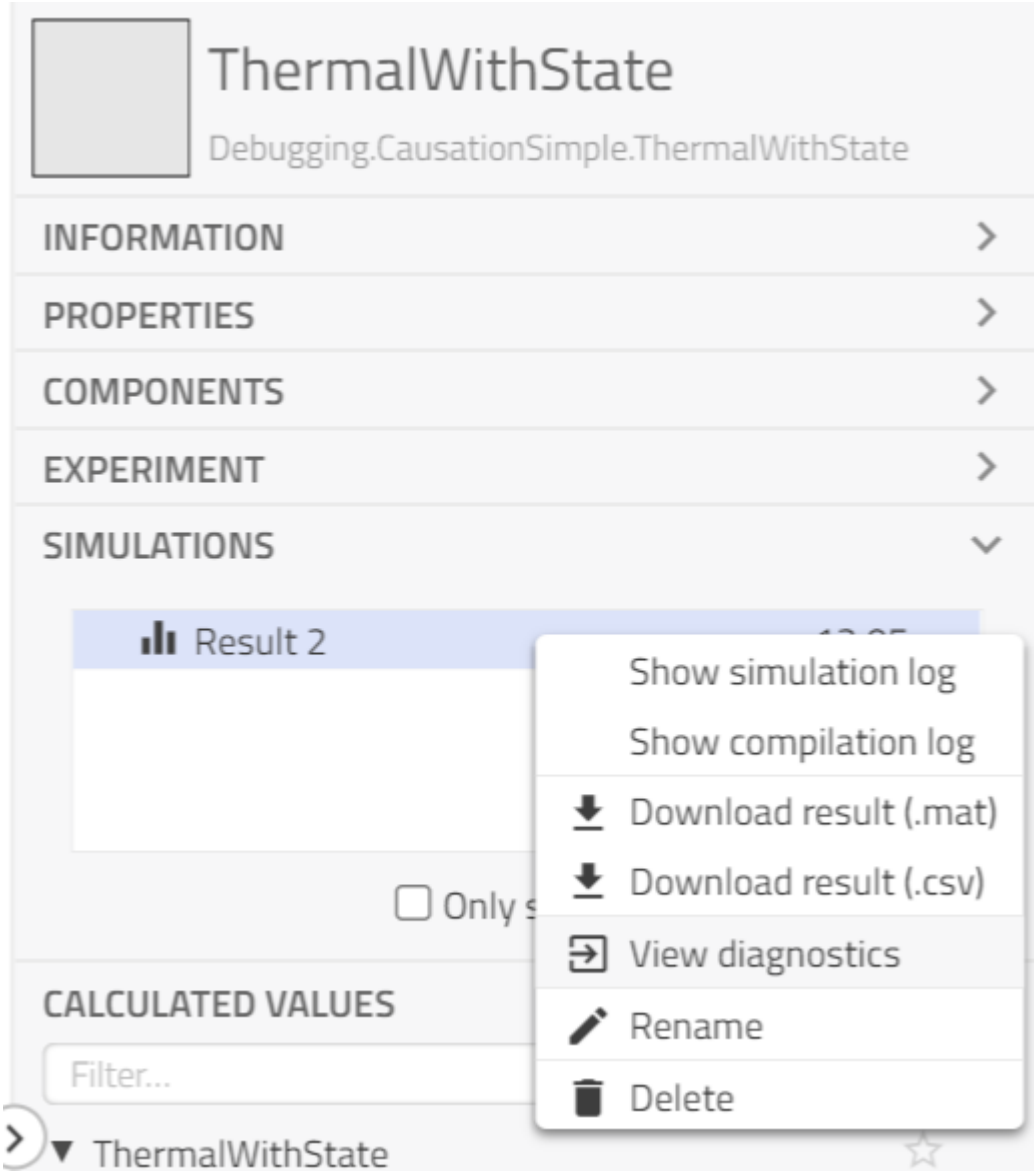
To understand the resulting mathematical system that Modelon Impact simulates, what state variables represent the dynamics, and how its initialized, we can use “generate_HTML_diagnostics” option to generate a model report.

4. Open the Settings and go to the “Execution” tab, and turn on the setting:



5. Re-run the model again to generate the report.

6. Open the report in the result pane, and right click the specific result:



7. Click “View diagnostics”

8. Open the State variables section:

Model Statistics for Debugging.CausationSimple.ThermalWithState

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

0 warnings 0 errors

▶ **Model Structure**

▼ **State variables** 1

Continuous 1

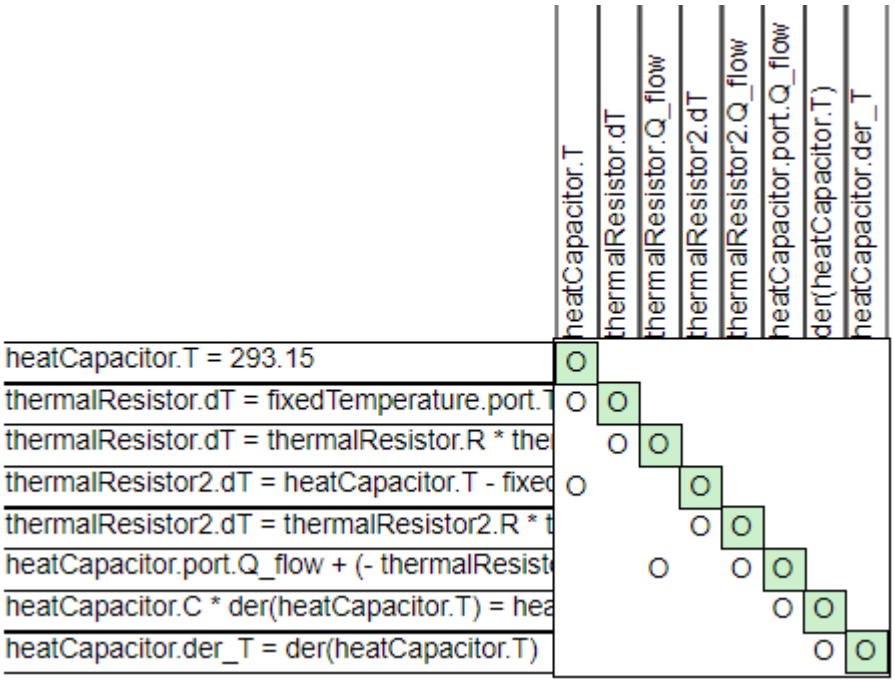
	State variable	nominal
0	heatCapacitor.T	300

Discrete 0

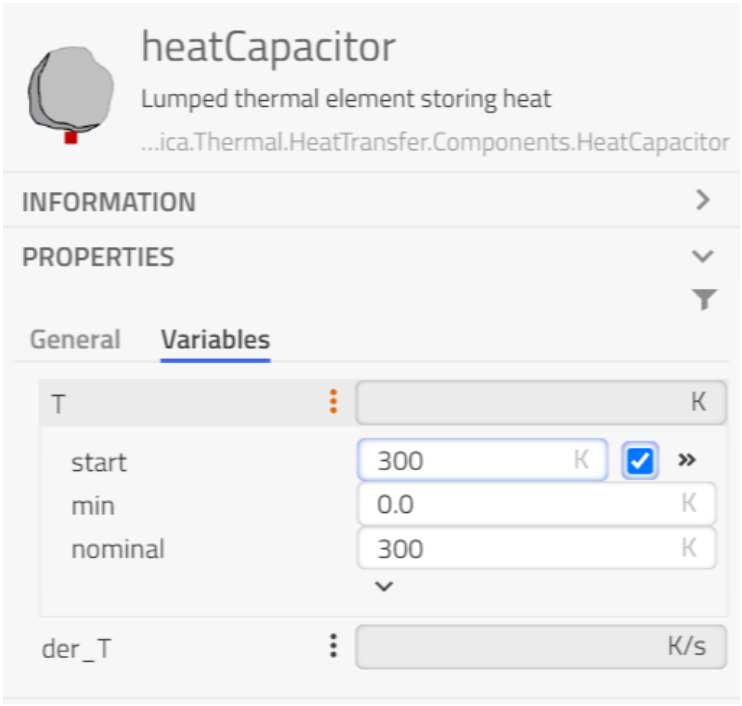
▶ **Initialization equation blocks** 0 linear / 0 non-linear

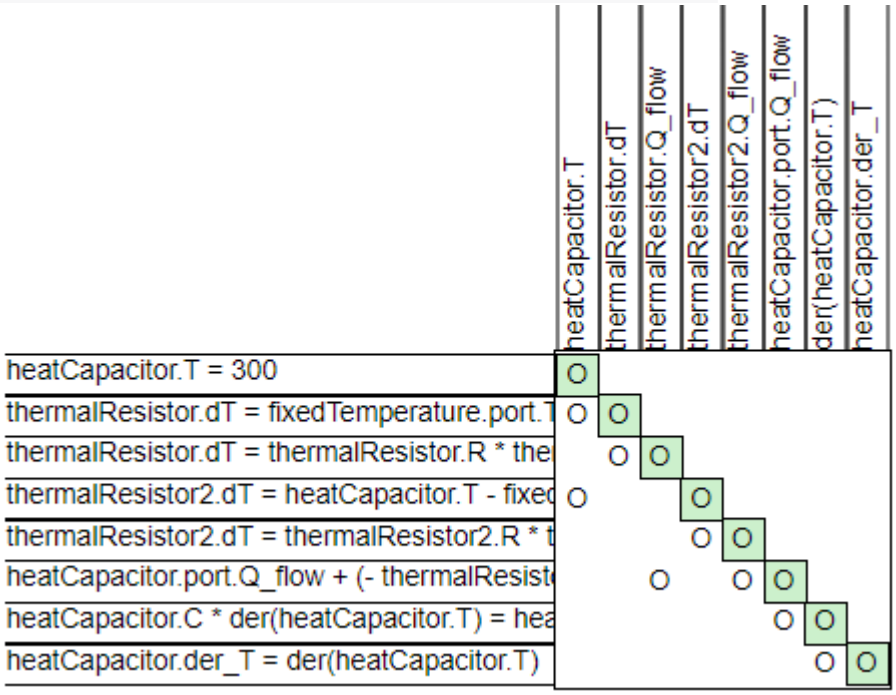
▶ **Equation blocks** 0 linear / 0 non-linear

9. Here we can see that the model has 1 state, heatCapacitor.T
10. Open the "Initalization equation blocks" and click "Interactive matrix" to see the BLT diagram, for solving the initial equations (note that there is a separate section for the dynamic equations)



11. From this view, we can see in the first line, that the initial equation for the state variable heatCapacitor.T = 293.15.
12. Go back into the model, make sure you are in "modeling mode", and change the initial condition for the state variable, and see if the initial equation is changed in the BLT diagram (you need to open the new "View diagnostics" report for the new result, they are re-generated per result).
    In the example below, we changed the initial condition to 300K:

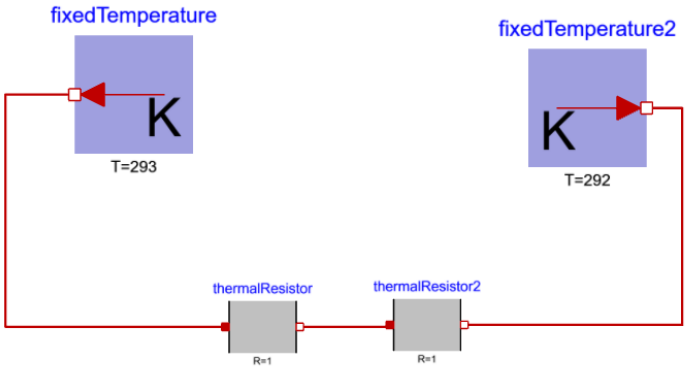




Model with linear systems

In the following example, we will see that we need to solve a linear system of equations to get the results.

1. Open the model “Debugging.CausationLinear.Thermal”.



2. We now removed the heat capacitor, so there is no dynamic state in the model.
3. Simulate the model, and inspect the diagnostics:

Model Statistics for Debugging.CausationLinear.Thermal

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

0 warnings 0 errors

▶ Model Structure

▶ State variables 0

▼ Initialization equat

1 Linear initializa

Block sizes: [1]

0 Non-linear init

Block sizes: [ ]

Modelica text

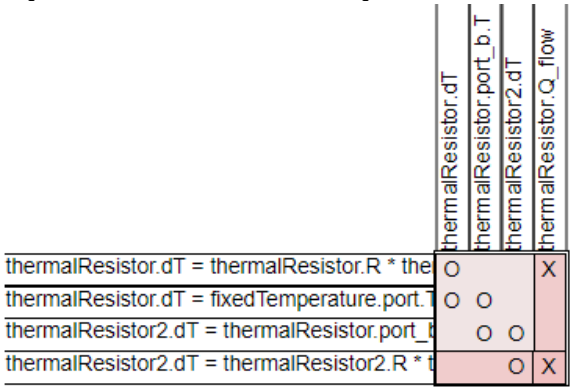
Interactive matrix

Linear Equation System Block (init)1

Iteration variable	start	min	max	nominal
0 thermalResistor.Q_flow	-	-	-	-

▶ Equation blocks 1 linear / 0 non-linear

4. We can now see the presence of a linear system, and the variable to be solved for is “thermalResistor.Q_flow”.
5. Inspect the BLT for the initial problem:

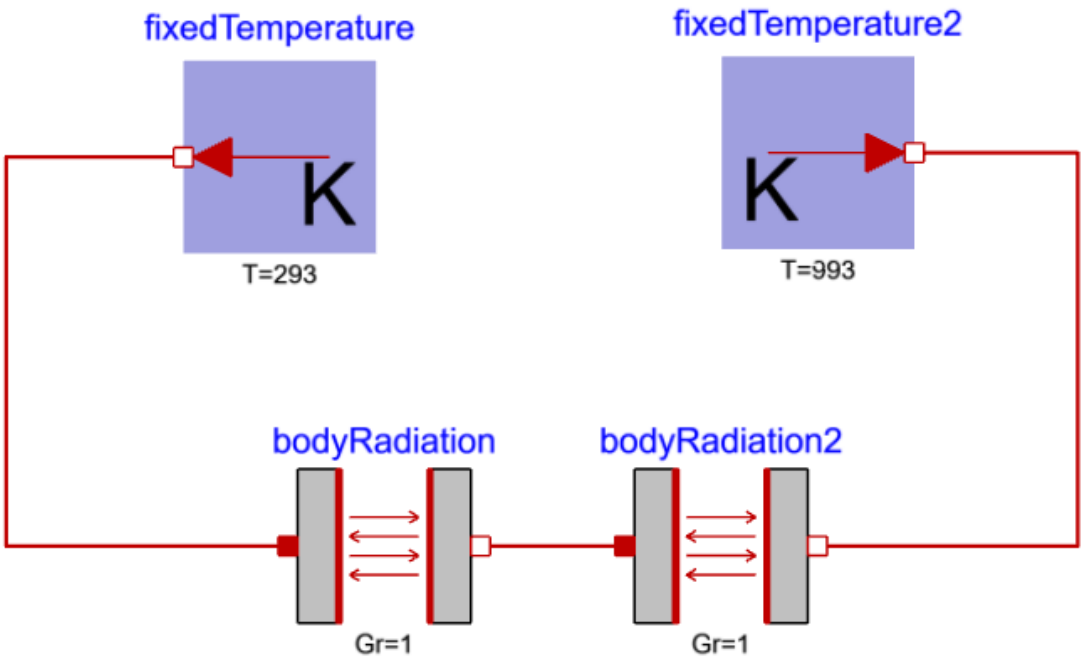


6. Here you can see that the BLT diagram has a “X” in the top right, so it needs to solve the complete linear system.

Model with non-linear systems

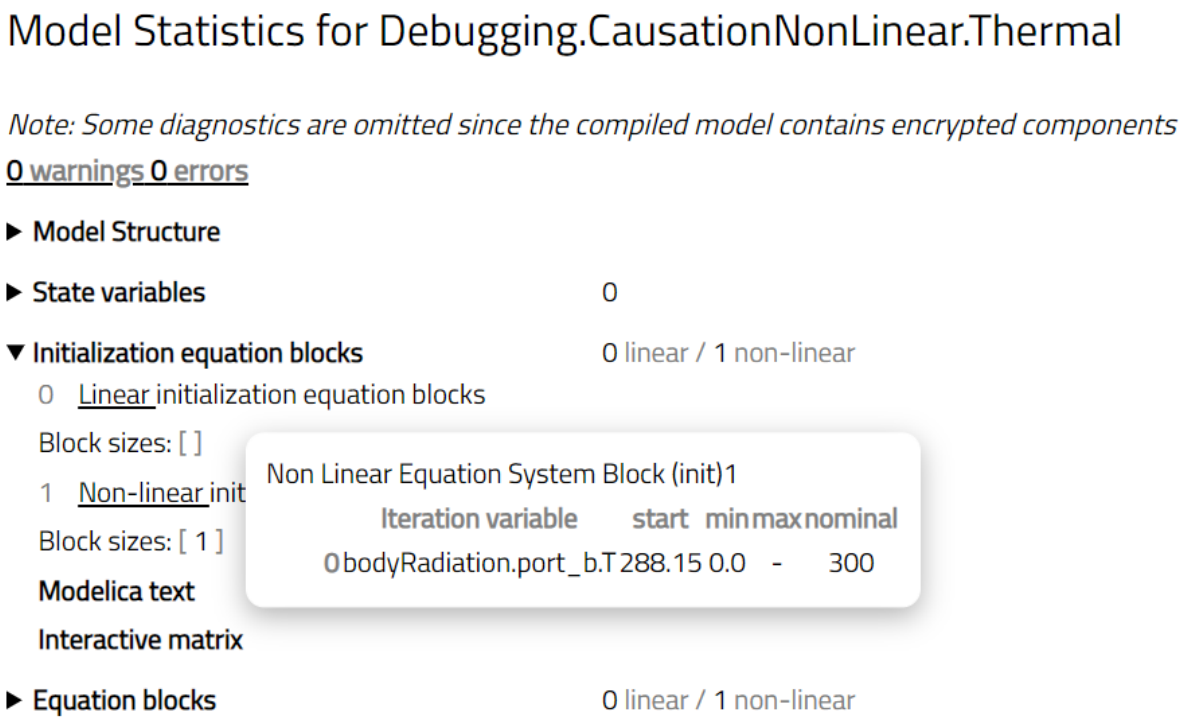
In the next section, we will investigate non-linear systems, and the possibility to influence the initial conditions of the iteration variables.

1. Open the model “Debugging.CausationNonLinear.Thermal”:

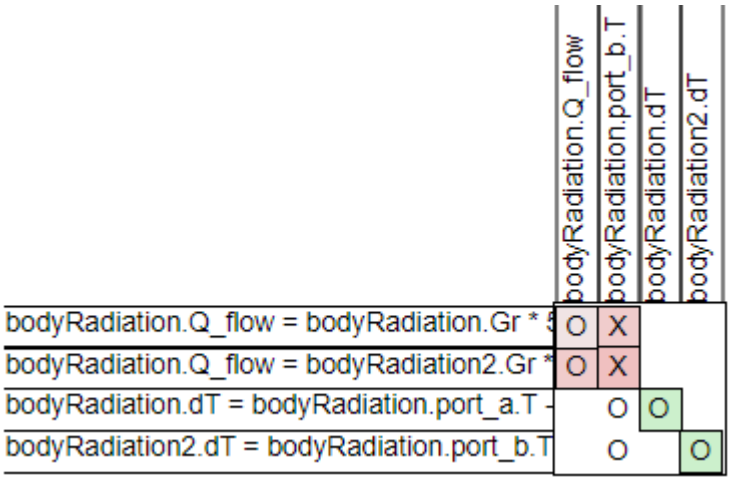


2. This time we have 2 radiation elements, and still no heatCapacitor in between with any state, so this will generate a system of non-linear equations.

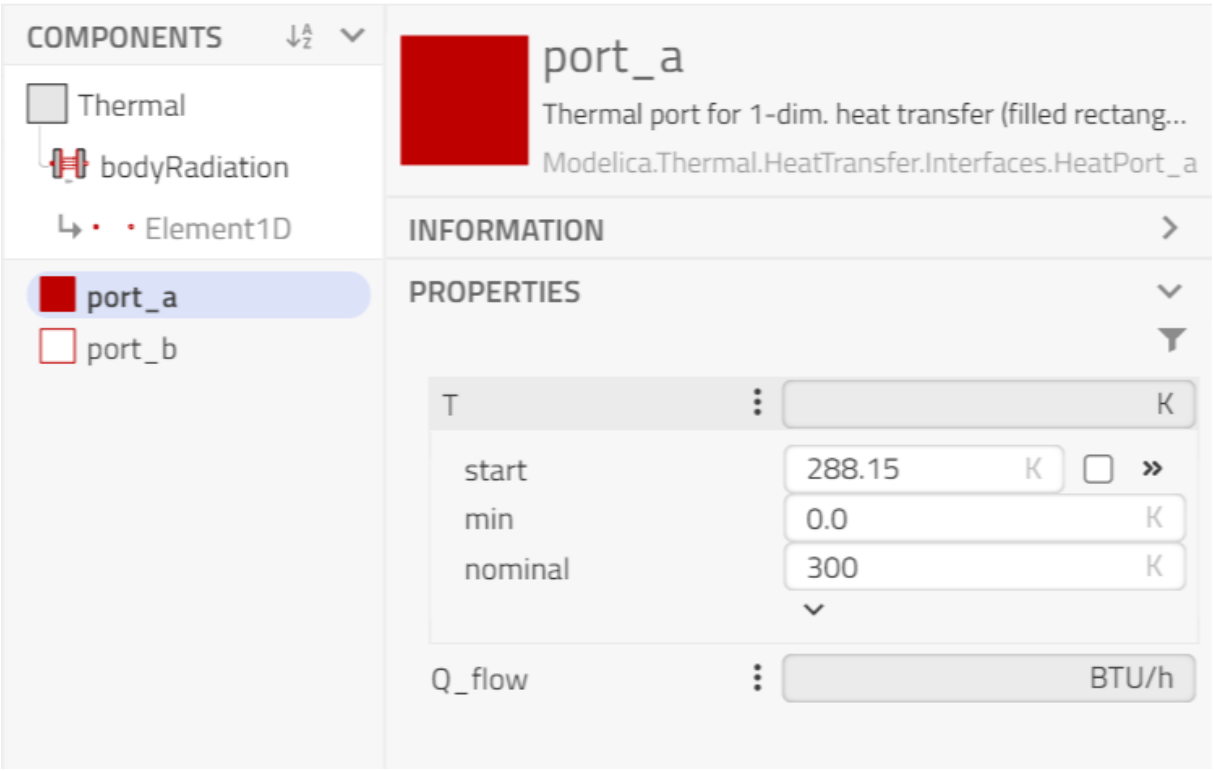
3. Simulate the model, and open the diagnostics:



4. Here we see that the model has 1 non-linear equation system. The iteration variable is bodyRadiation.port_b.T and the initial guess value is set to 288.15.
5. Open the BLT diagram, and verify the nonlinear system:



6. Go back into the model, and the modeling view.
7. Open the Components browser, and locate the iteration variable:

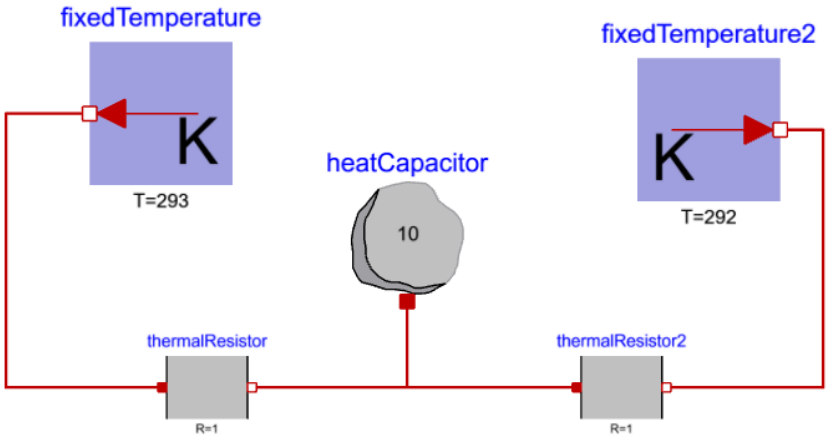


8. Change the initial guess value of the temperature, and see if there is any update to the "diagnostics".
9. Note: you cannot use fixed=true on an iteration variable, since that would enforce an initial equation stating T=T.start, and that is not possible since the solver needs to find T such that all equations are fulfilled.

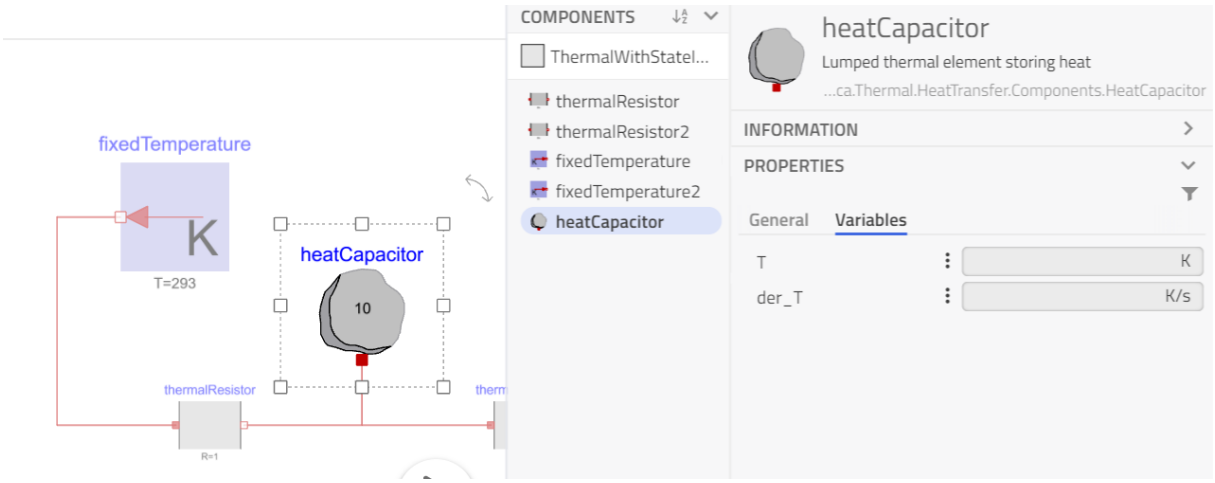
Initialize dynamic system in steady state

Lets go back and revisit the dynamic case, but set a steady state initial condition.

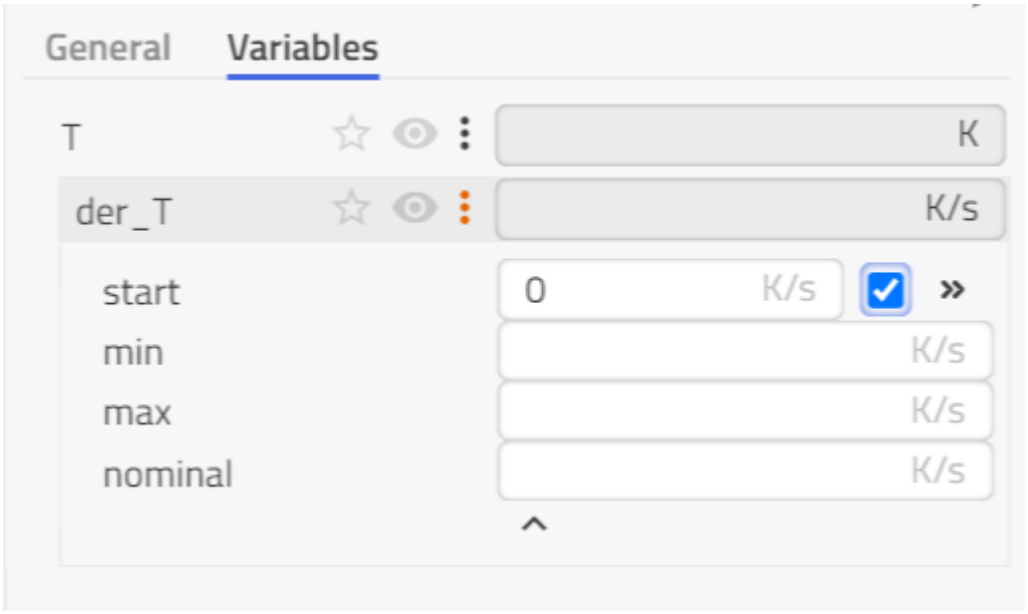
1. Open the model "Debugging.CausationSimple.ThermalWithStateInitSS":



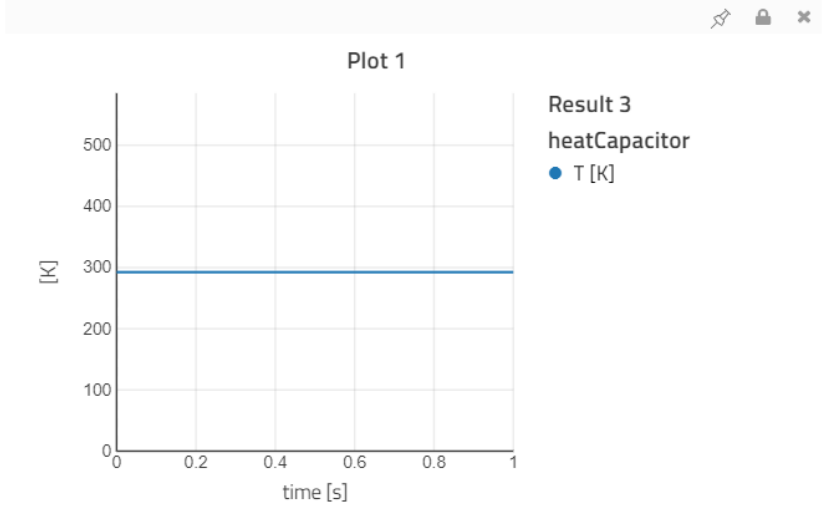
2. Double click the heatCapacitor and look at the "Variables" section:



3. If we want to invoke a steady state condition for the initialization, we can simply set der_T(start=0,fixed=true).

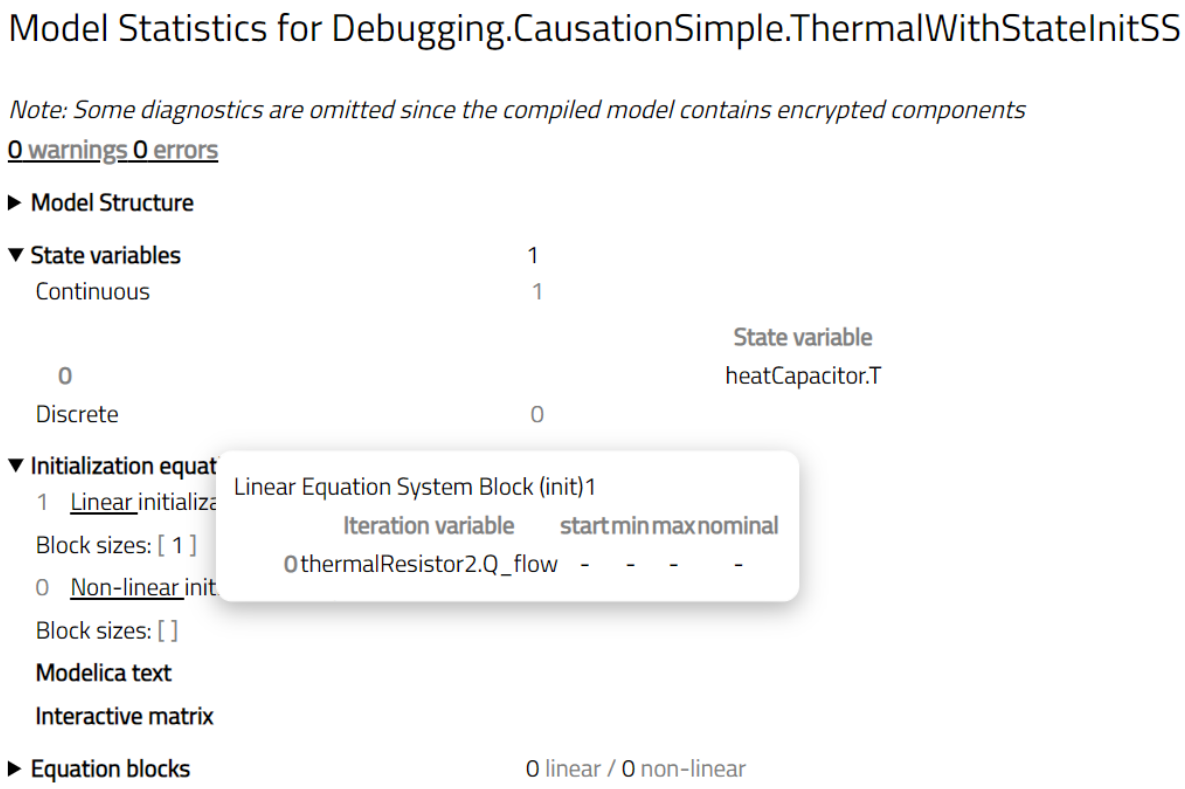


4. Run the model, and plot the heat capacitor temperature.



We can see that there are no transients present.

5. Open the diagnostics report:



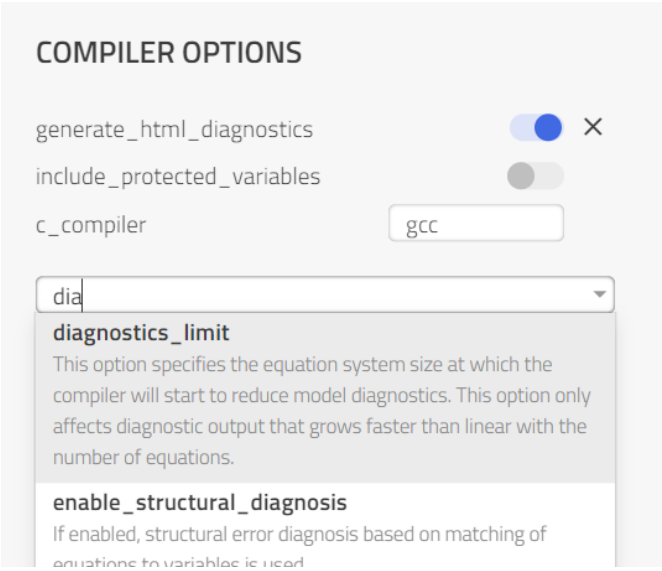
We can now identify that the model does have a state variable, but also need to solve a linear system of equations for the initialization problem, due to the steady state condition.

Also note that the dynamic problem is the same as before and does not include any linear or non-linear systems.

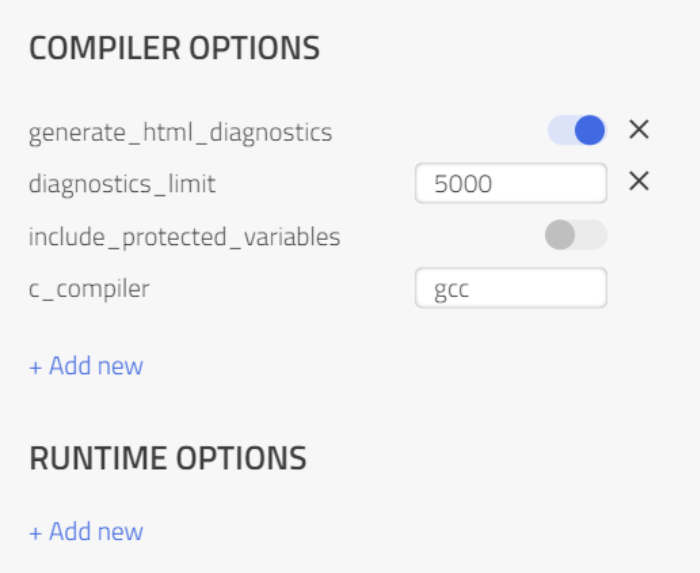
Closed Steam Cycle Example

Let's apply the knowledge we got in the simple examples on a more complex problem.

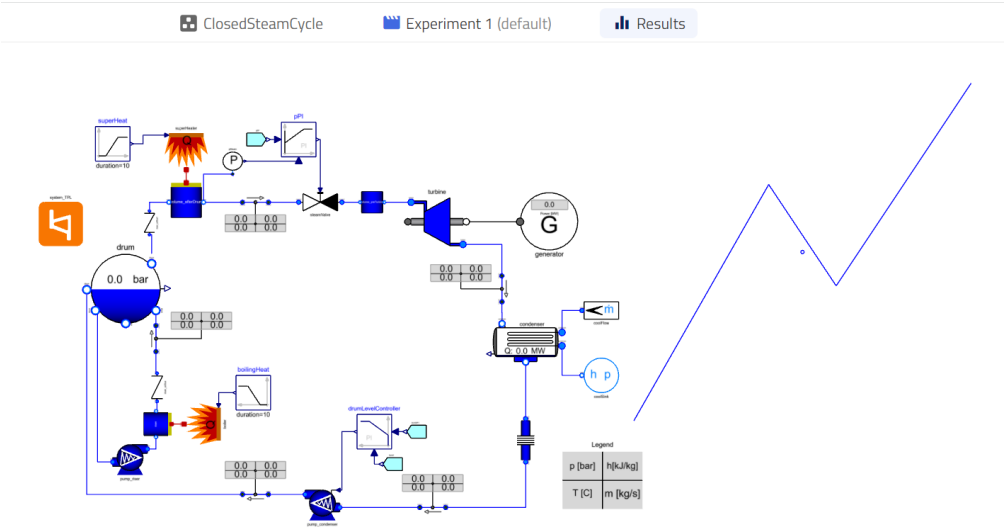
1. Note: There is a cap for how large a model is allowed to generate the diagnostics, and for large models, it might be necessary to increase the allowed model size in the compiler settings:



2. Set the diagnostic_limit to 5000

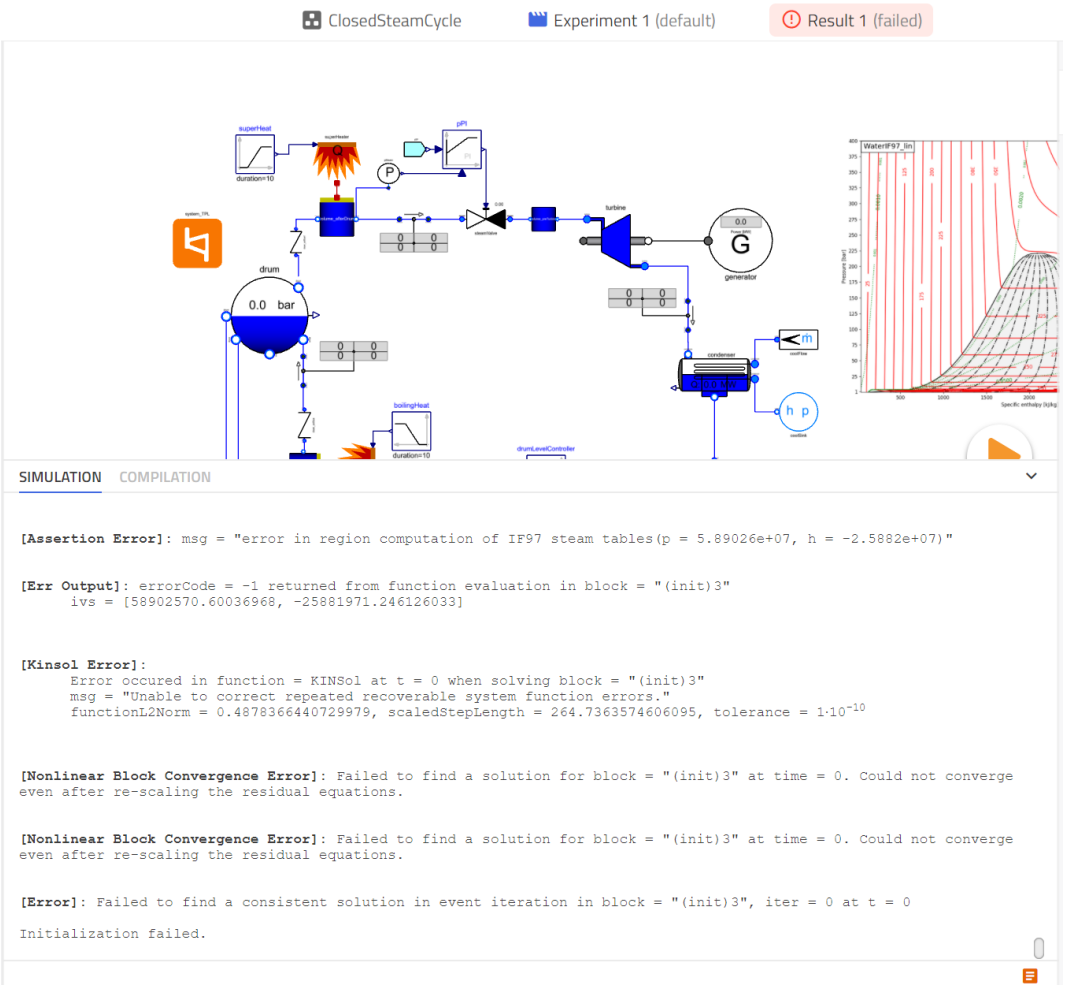


3. Open the model “Debugging.SystemModel.ClosedSteamCycle”:

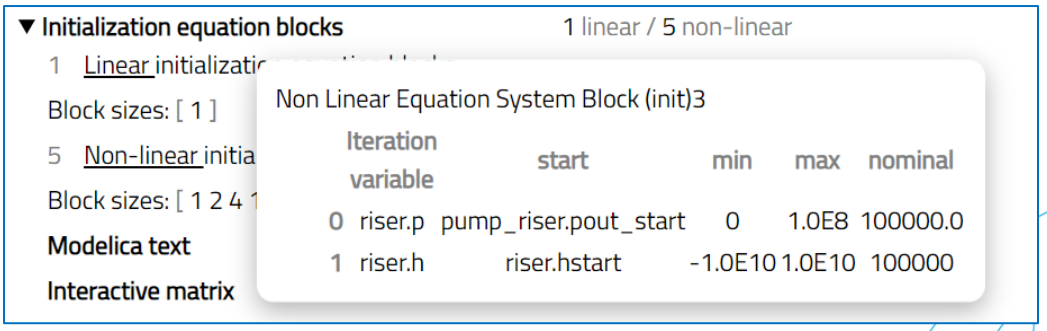


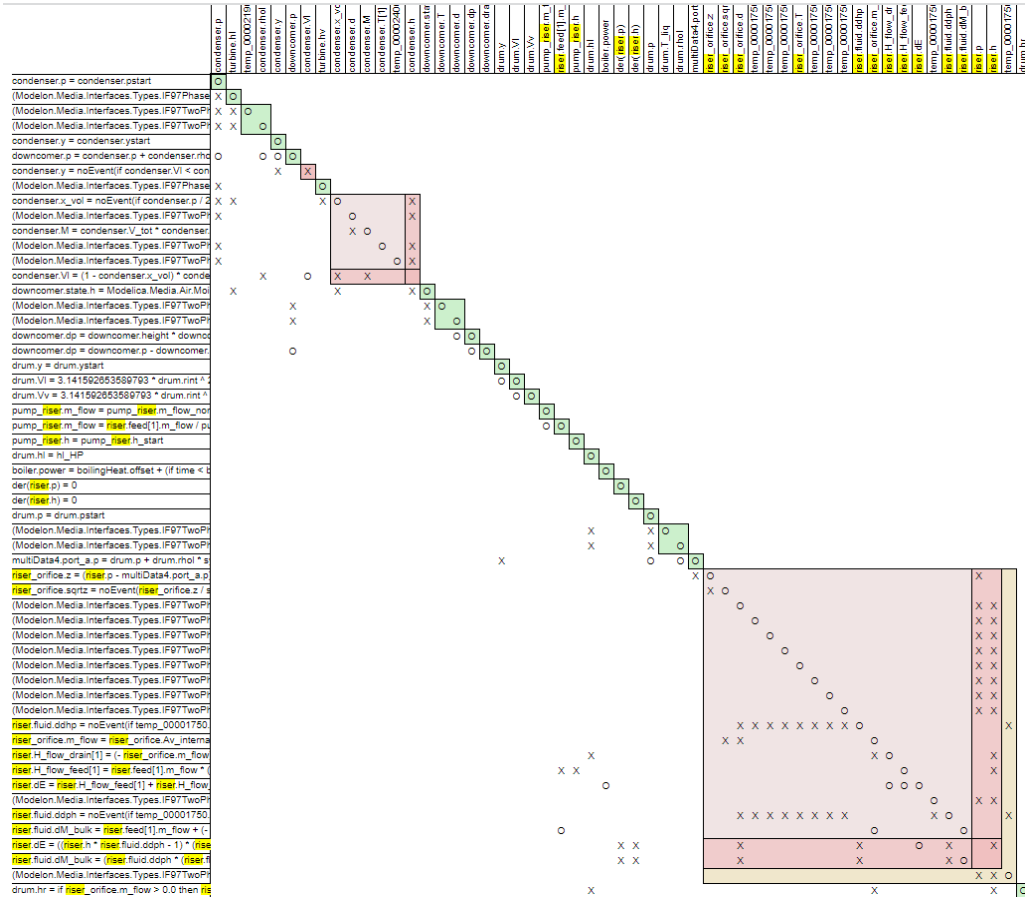
4. Try and run the model.

5. The model fails and throws an error:

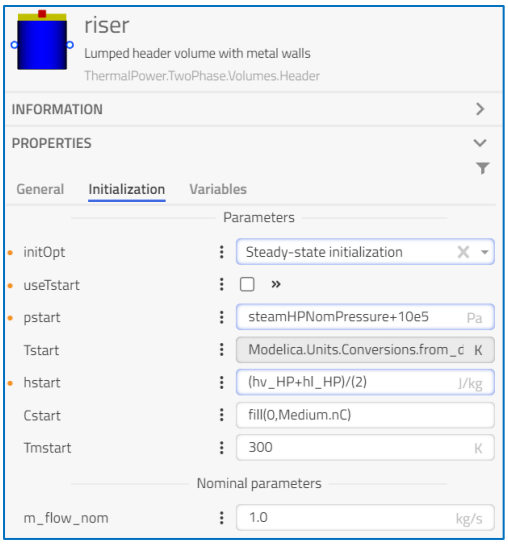


6. (scroll to bottom) Initialization failed, and could not find a solution for init block 3. (non-linear system of equations)
7. Open the “diagnostics”, in the “Initialization equation blocks” identify what iterations variables are part of the block “(init)3”:





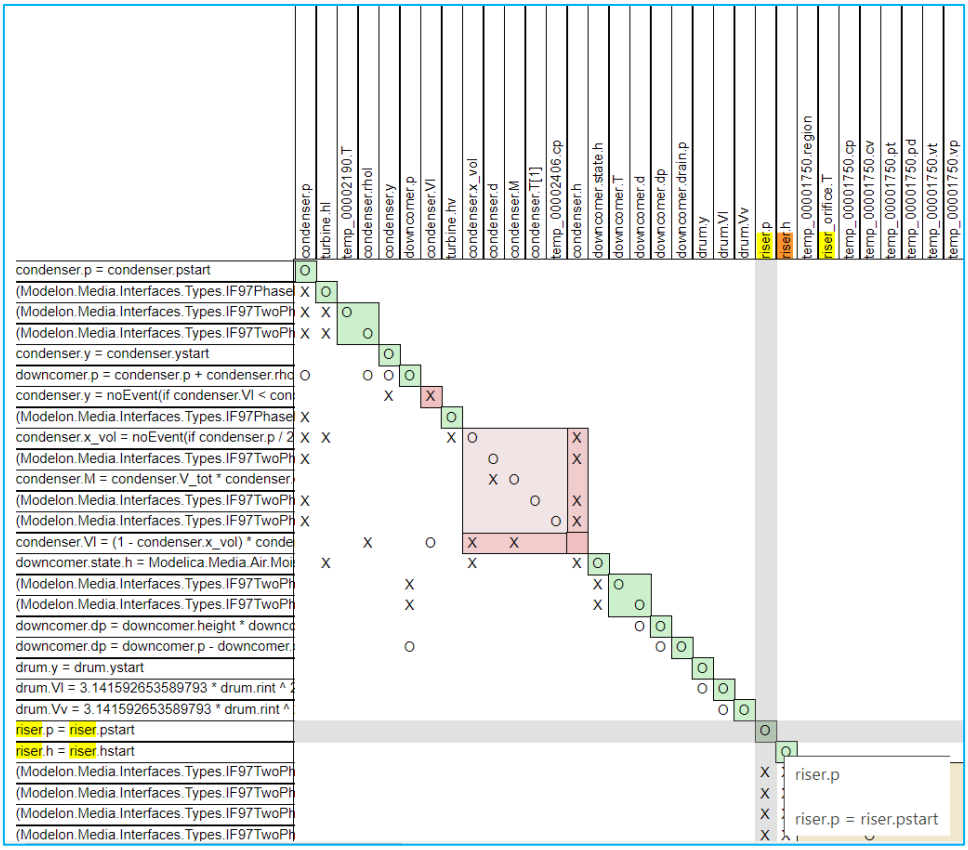
8. Review the Initialization settings in the “riser” component:



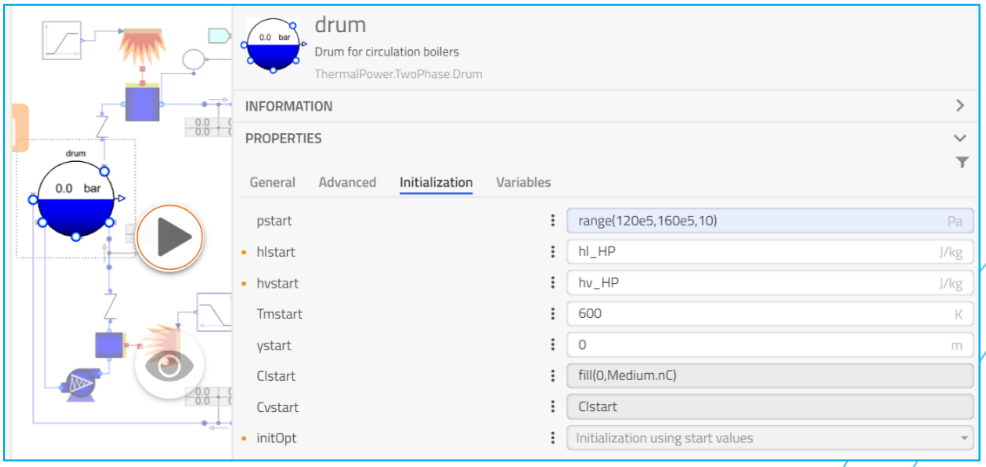
9. Since initOpt “Steady-state” is selected here, a successful initialization requires the existence of a steady state solution for the block. This is often challenging and requires very accurate guess values.
10. Try initOpt “Initialization using start values” instead.
11. Re-run the simulation.
12. The initialization should run successful now – review the result, specifically those of “riser.der(p)” and “riser.p” – see that at t=0s, the pressure derivative differed

significantly from zero, indicating that neibouring initial conditions force a rapid change of pressure in the riser.

13. Review the “diagnostics” again, see that through removing the steady-state initialization option, the non-linear block “(init)3” that did not converge before, is now



14. Change the initial condition in the riser back to “steady-state”. Since the solver was not able to find a solution by iterating inside the block, changing its boundary conditions might enable the solver to find a solution. E.g. use the “range-opeator” to sweep the “pstart” in the neiboring “drum” component to see what initial values allow the system to solve:



This concludes the workshop. Well done!