Introduction to Modelica Modeling for Building Systems

A Tutorial Developed for MIT Class 4.421: Space-Conditioning Systems for Low-Carbon Buildings

3/29/2021

By David Blum

I. Requirements

1. Dassault Systemes Dymola (v2020), License, and C Compiler

    a. For notes about choosing and setting up the C Compiler, see: https://www.3ds.com/products-services/catia/products/dymola/c-compiler/

2. Modelica Buildings Library (v7.0.0)

    a. For download, see: https://simulationresearch.lbl.gov/modelica/download.html

    b. For notes about setting up the library, see: https://simulationresearch.lbl.gov/modelica/installLibrary.html

II. Additional Resources

This is going to be a brief introduction. There is much more to learn about how to use Modelica and Dymola. I recommend the following resources:

1. Dymola Users Guide
2. Modelica by Example (https://mbe.modelica.university/)
3. Buildings Library Users Guide (https://simulationresearch.lbl.gov/modelica/userGuide/index.html)
4. Buildings Library Tutorials (https://simulationresearch.lbl.gov/modelica/training.html)

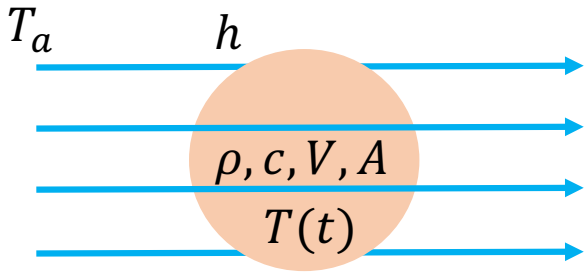
Part 1: Modelica Syntax and Dymola Development Environment

Modelica is a language, like C++ or Python. Therefore, you need to learn its syntax and how it can be used to represent models. Dymola is the development environment that will help us do that, in addition to helping us organize, compile, and simulate our models, as well as explore the results of those simulations.

Therefore, in this part, you will:

1. Write a simple heat transfer model in Modelica
2. Check that the model is valid
3. Simulate the model
4. Explore the results

First, let's define the physical phenomenon we will model. Consider a heat transfer problem where a small rock is being cooled by an air stream, as shown in the diagram. We are interested in simulating the temperature of the rock over time. This can be described by the differential equation and initial condition below:



$$\rho V c_v \frac{dT}{dt} = hA(T_a - T)$$

$$T(t = 0) = T_0$$

Where:

T is the rock temperature [K]

T_0 is the initial rock temperature [K]

T_a is the air temperature [K]

ρ is the rock density [kg/m^3]

V is the rock volume [m^3]

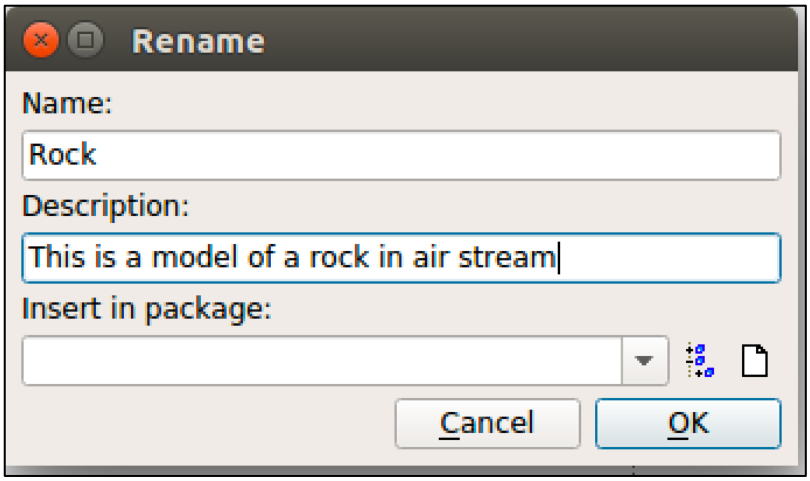
c_v is the rock specific heat capacity [J/kg-K]

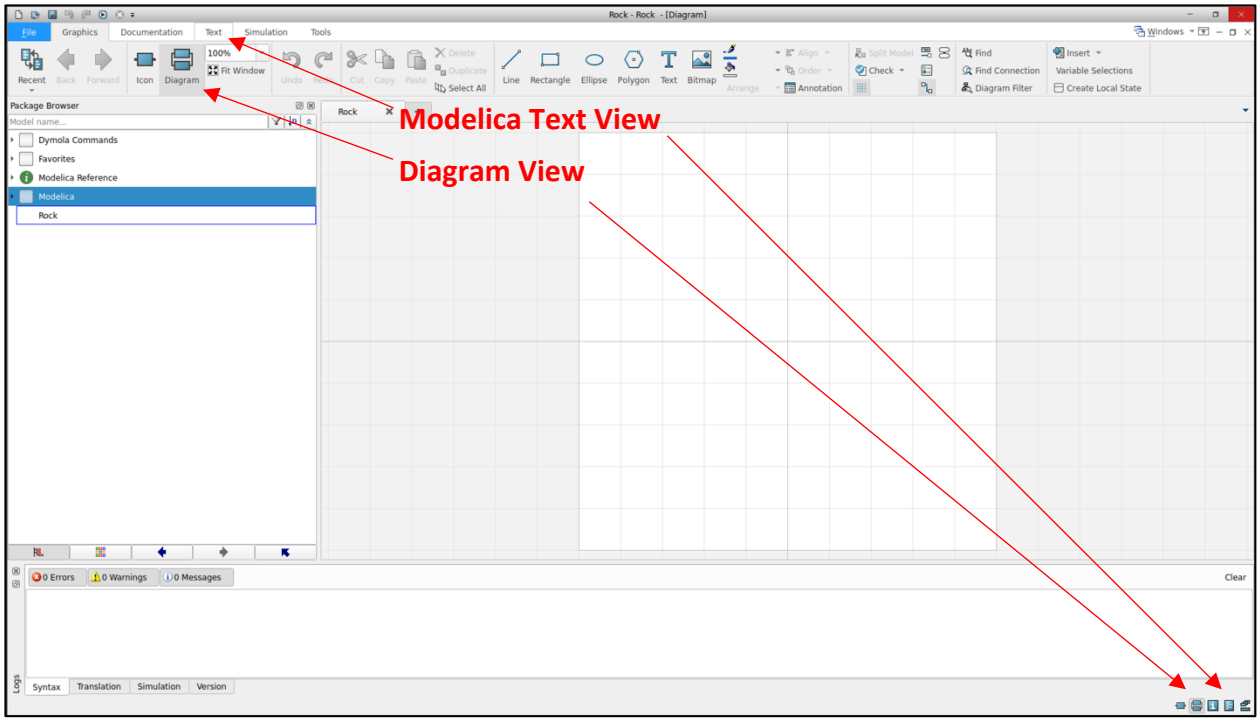
h is the rock-air heat transfer coefficient [W/m^2-K]

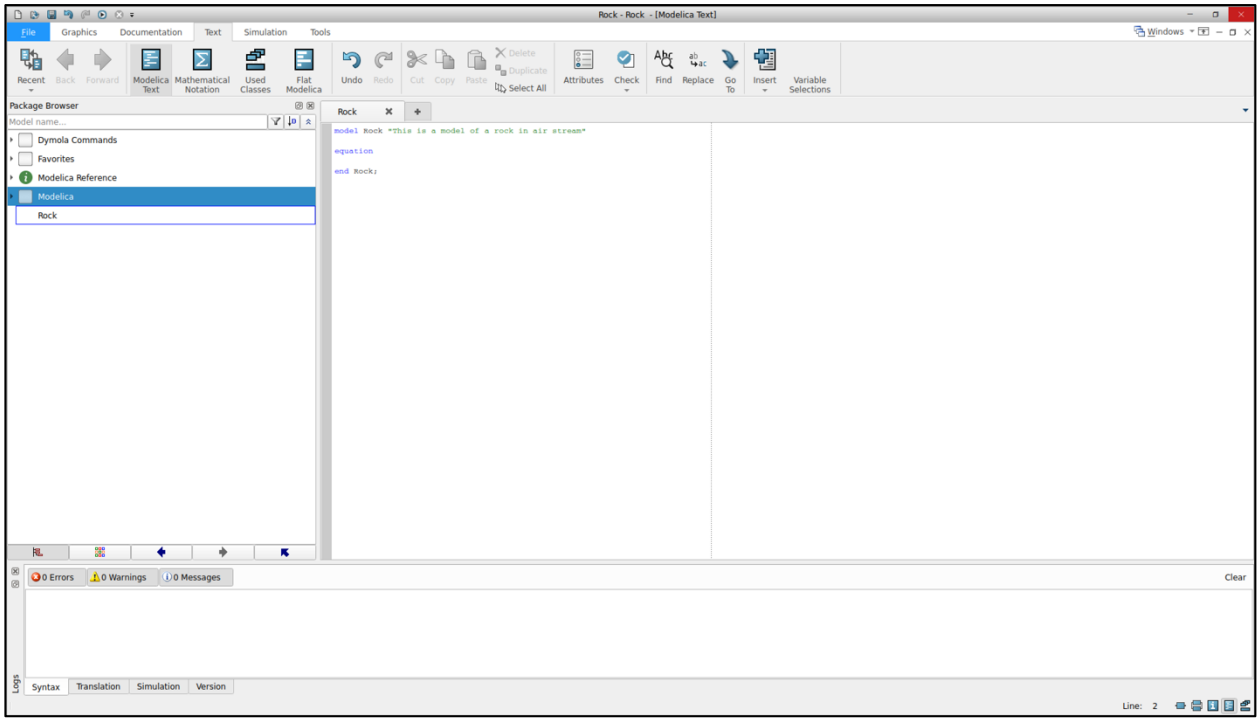
A is the rock surface area [m^2]

1. Open Dymola. Go to File > Save > SaveAs and save the model file as shown in the Figure below (it will be saved as Rock.mo). Then, switch to the "Modelica Text" view by clicking on the icon indicated in the top Figure on the next page. Your screen should look like the bottom Figure on the next page.

By default, Dymola will open to the "Diagram View." This view is used to configure models using graphical blocks that can be pieced together. We will explore this later in the tutorial. For now, we are concentrating on the underlying text-based implementation of models.




Rock - Rock - [Diagram]
Modelica Text View
Diagram View
Package Browser
Dymola Commands
Favorites
Modelica Reference
Modelica
Rock
0 Errors
0 Warnings
0 Messages
Clear
Syntax
Translation
Simulation
Version


Rock - Rock - [Modelica Text]
model Rock "This is a model of a rock in air stream"
equation
end Rock;
Package Browser
Dymola Commands
Favorites
Modelica Reference
Modelica
Rock
0 Errors
0 Warnings
0 Messages
Clear
Syntax
Translation
Simulation
Version

2. Declare all of the variables in the model as shown in the Figure below.

The syntax of defining a basic model is split into two main sections, one that is used to declare all of the variables in the model and one that is used to define the equations of the model. The declaration section is above the word "equation" and the equations section is below the word "equation."

The declaration section is used to give variables names and various characteristics. Here, `parameter` *means the variable is given a value that does not change over time.* `Real` *means that the value of the variable is a real number, as opposed to an Integer or Boolean (True/False). Other attributes can be assigned, such as the* unit *and* start *value. Finally,* `"descriptions"` *of the variables can be given, which are used later to help us document the model.*

Notice that all of our variables are given, except the temperature of the rock. This is our only variable that will change with time and needs to be calculated using an equation (this is also called a "state" variable). Notice also that the volume and area parameters are defined as a function of the radius parameter, assuming that our rock is a sphere.

```
model Rock "This is a model of a rock in air stream"
  parameter Real r(unit="m") = 0.1 "Radius of rock";
  parameter Real rho(unit="kg/m3") = 2230 "Density of rock";
  parameter Real c(unit="J/(kg.K)") = 880 "Specific heat capacity of rock";
  parameter Real h(unit="W/(m2.K)") = 1000 "Heat transfer coefficient";
  parameter Real T_a(unit="K") = 273.15+10 "Temperature of air stream";
  parameter Real T_0(unit="K") = 273.15+20 "Initial temperature of rock";
  parameter Real V(unit="m3") = 4/3*3.14*r^3 "Volume of rock";
  parameter Real A(unit="m2") = 4*3.14*r^2 "Surface area of rock";
  Real T(unit="K", start=T_0) "Temperature of rock";
equation

end Rock;
```

3. Write the equations of the model as shown in the Figure below.

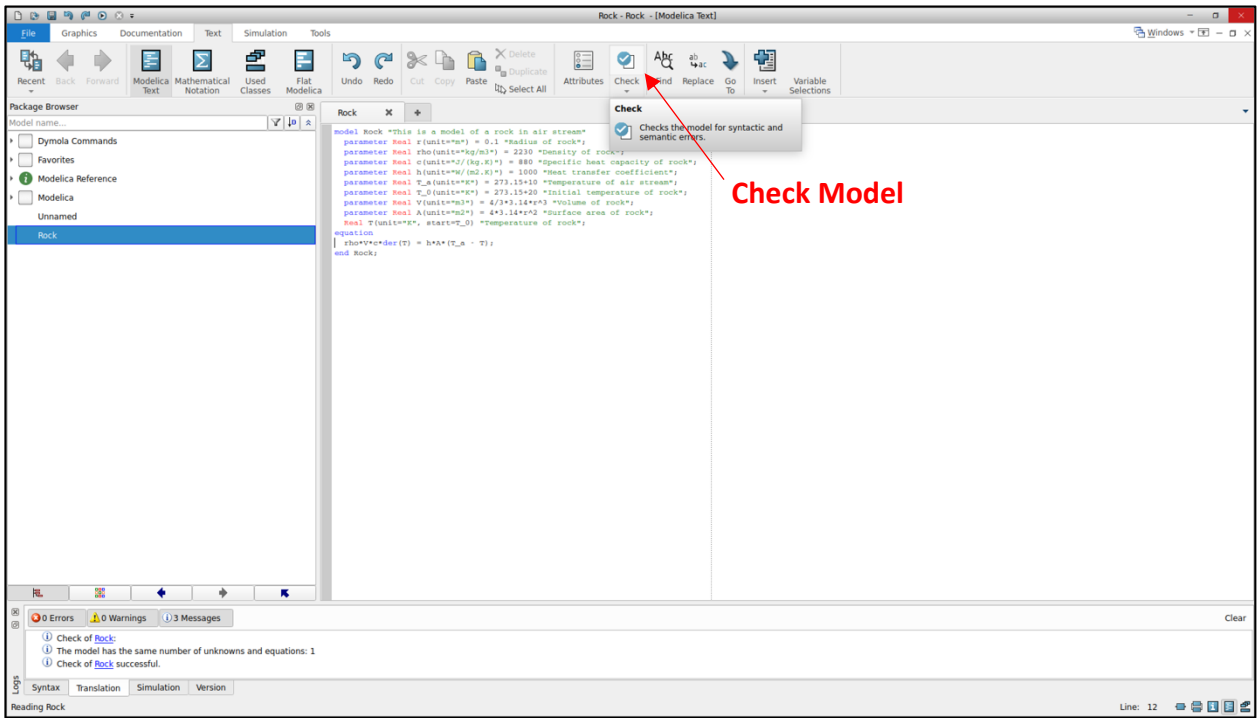
Here, we only have one equation. It is an ordinary differential equation (ODE). Modelica syntax naturally accepts the specification of variable derivatives with respect to time using the `der()` *syntax.*

Notice we can also write equations in an "acausal" manner. That is, it does not matter which variables appear on each side of the equation. In addition, it does not matter the order in which we specify equations, if we had more than one. All of this makes it easier for us to define systems of differential and algebraic equations (DAEs) without having to first solve them analytically and determine the proper order of assigning values to variables. Instead, we define them in Modelica, and let the tool (in this case Dymola) solve them for us.

```
model Rock "This is a model of a rock in air stream"
  parameter Real r(unit="m") = 0.1 "Radius of rock";
  parameter Real rho(unit="kg/m3") = 2230 "Density of rock";
  parameter Real c(unit="J/(kg.K)") = 880 "Specific heat capacity of rock";
  parameter Real h(unit="W/(m2.K)") = 1000 "Heat transfer coefficient";
  parameter Real T_a(unit="K") = 273.15+10 "Temperature of air stream";
  parameter Real T_0(unit="K") = 273.15+20 "Initial temperature of rock";
  parameter Real V(unit="m3") = 4/3*3.14*r^3 "Volume of rock";
  parameter Real A(unit="m2") = 4*3.14*r^2 "Surface area of rock";
  Real T(unit="K", start=T_0) "Temperature of rock";
equation
  rho*V*c*der(T) = h*A*(T_a - T);
end Rock;
```

4. Check that the model is valid by using the "Check" command, indicated by a green check mark at the top middle of the screen. You should see messages at the bottom of your screen as shown in the Figure.

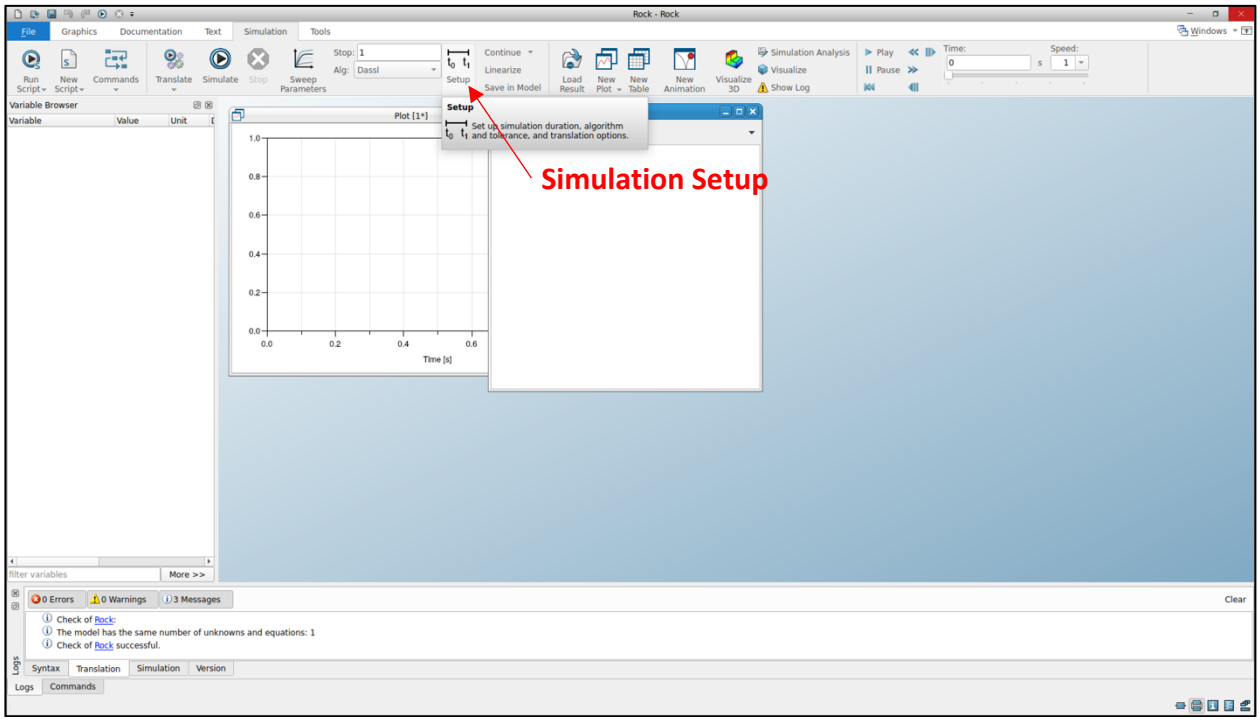
Notice that Dymola has identified that there is 1 unknown variable in our model (T), and that there is 1 equation in the model (our ODE). Since we have the same number of equations and unknowns, we can solve the problem. Dymola also runs a number of other checks on the model which are not explicitly listed here. However, if there were problems with the syntax or problems with how the model is implemented, Dymola would try to detect and report them here, as either Warnings or Errors. Models with Warnings can still be simulated, while models with Errors will not simulate.

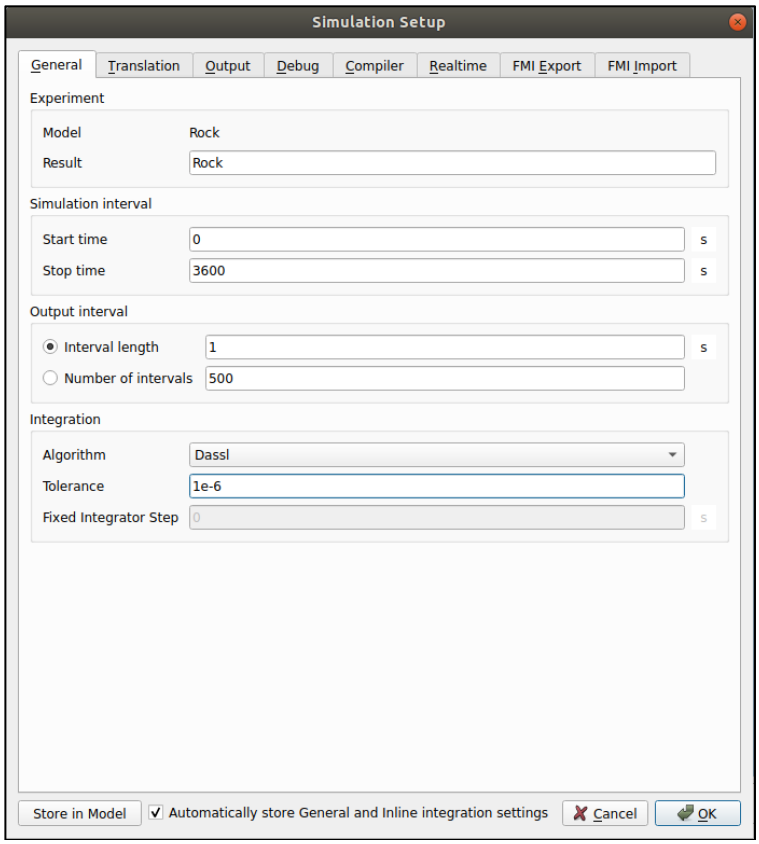


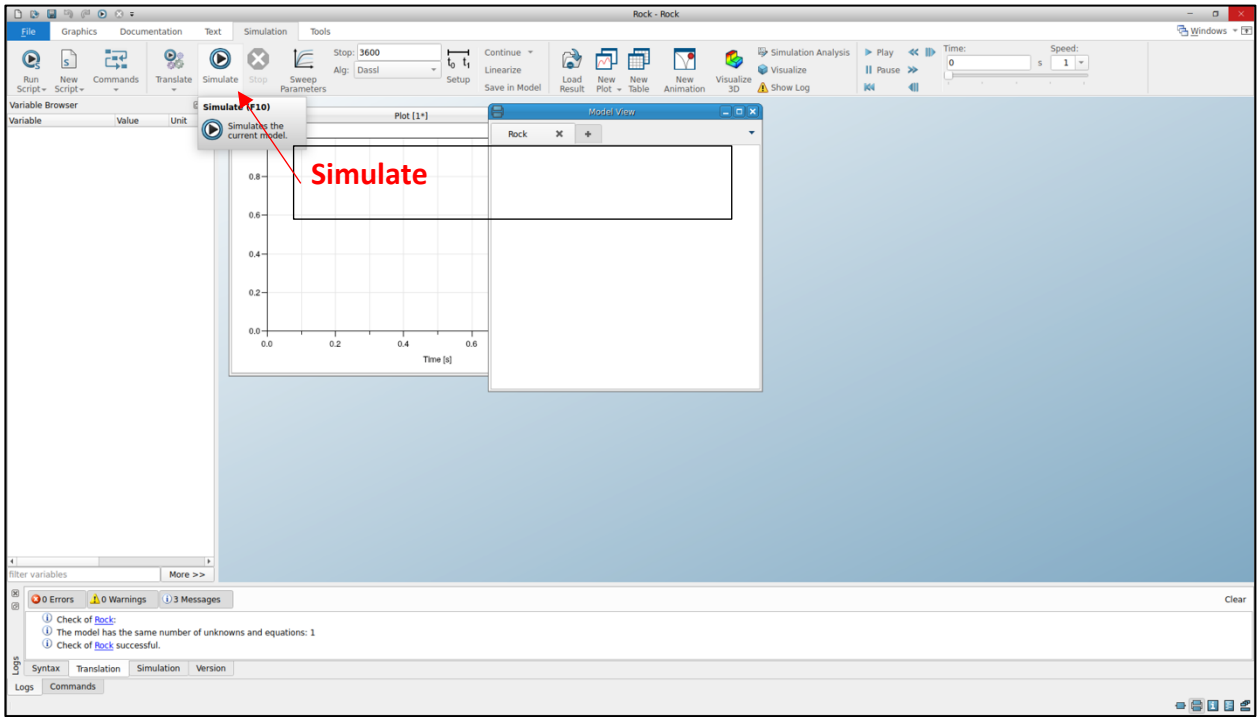
5. Go to the Simulation view by clicking on the "Simulation" tab at the top of the screen. Then, setup the simulation of the model using the "Setup" command at the top middle of the screen. Configure as shown in the Figure. Finally, simulate the model using the "Simulate" command at the top of the screen.

In the simulation setup, we are configuring the simulation to run for one hour of simulation time, have an output interval of one second, and utilize the Dassl solver with a tolerance of 1e-6. Note that the output interval of one second does NOT define our integration timestep. The Dassl solver is a variable time-step solver, meaning the actual integration timestep will be whatever is needed to successfully solve the equations through time, and will vary depending on how quickly the variables are changing.

Clicking the "Simulate" command actually invokes Dymola to perform multiple processes. First, it processes the equations into a format in which the variables can be sequentially solved. It also determines which groups of equations may need to be solved iteratively, called algebraic loops (we do not have that problem in this simple example). Then, it writes the equations into C code and compiles it. Finally, it executes the compiled code, using the specified solver to solve (or "integrate") the model through time.

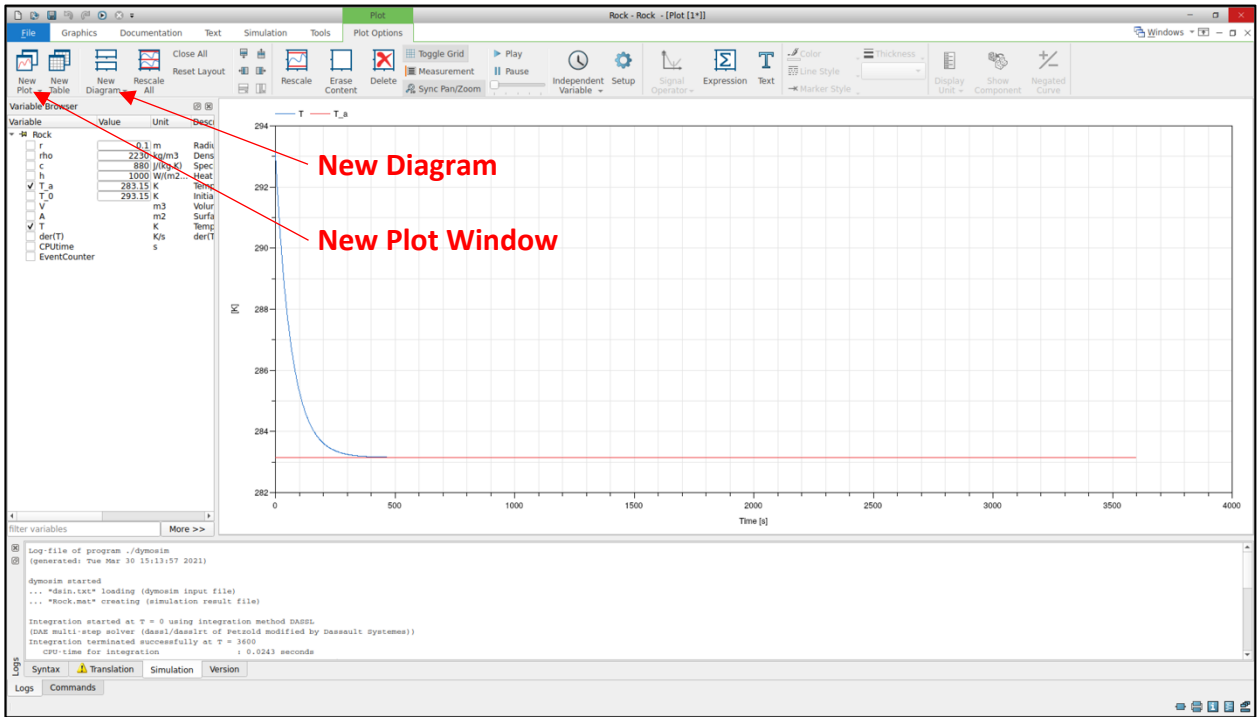



Simulation Setup
General
Translation
Output
Debug
Compiler
Realtime
FMI Export
FMI Import
Experiment
Model
Rock
Result
Rock
Simulation interval
Start time
0
s
Stop time
3600
s
Output interval
Interval length
1
s
Number of intervals
500
Integration
Algorithm
Dassl
Tolerance
1e-6
Fixed Integrator Step
0
s
Store in Model
Automatically store General and Inline integration settings
Cancel
OK


Rock - Rock
Simulate
Simulate (F10)
Simulates the current model.
Plot [1*]
Model View
Rock
Time [s]
0 Errors
0 Warnings
3 Messages
Check of Rock:
The model has the same number of unknowns and equations: 1
Check of Rock successful.
Syntax
Translation
Simulation
Version
Logs
Commands

6. Explore the results by selecting variables to plot in the Variable Browser to the left of the screen. You can create new plot windows and plot diagrams by using the commands at the top of the screen in the Plot Options tab, which appears when you select a plot window. Plot the rock temperature and air temperature over time. How long does it take for the rock to cool down?

Change the values of the parameters for air temperature and initial rock temperature in the Variable Browser within the Value boxes and simulate again. Does the cool-down time change? Now change the values of the parameters for heat transfer coefficient or the radius of the rock. Does the cool-down time change?



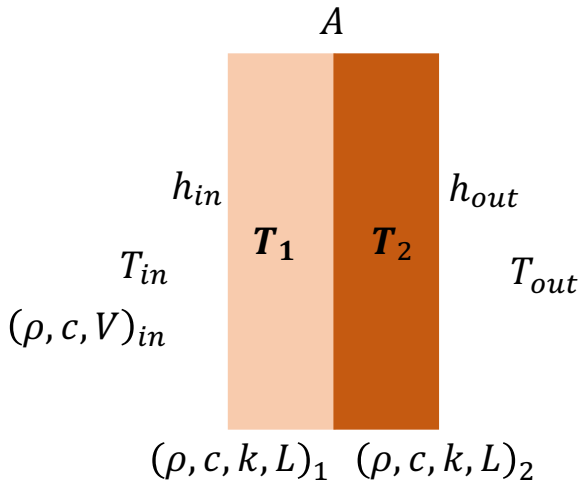
Part 2: Connectors, Components, and Graphic Syntax

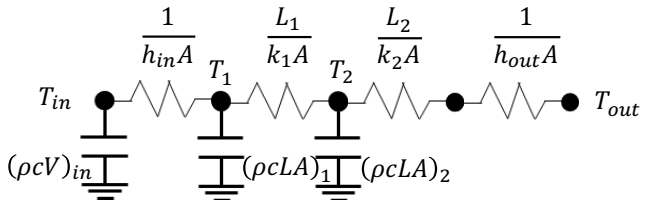
One of the strengths of Modelica is that it is object-oriented. This helps us be able to build up libraries of component models which can be pieced together to form larger, custom system models. There are two additional aspects that help us achieve this capability. The first is the definition of connectors and the second is graphical annotation.

Therefore, in this part, you will:

1. Learn the basic concept of connectors
2. Use the graphical syntax to connect component models into a larger system

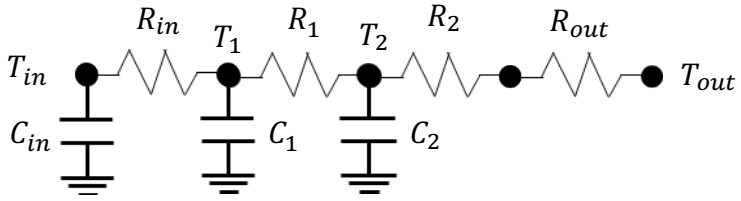
First, let's again give ourselves an example problem. Consider heat flow through a multilayer wall, as shown in the diagram below, and a corresponding resistor-capacitor network model (resistor-capacitor modeling in heat transfer is an analogue to electrical circuit modeling). We are interested in simulating the temperature of the inside air over time given a time-varying outside air temperature.



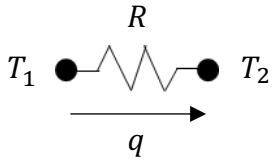


In this problem, we could write down the governing system of equations and implement the model in Modelica as we did in Part 1. However, this would be time-consuming. Instead, let's utilize the connector, component modeling, and graphic syntax capabilities of Modelica to not only solve this problem, but give us the ability to easily solve other, similar, problems.

Before doing this, we need to explore the nature of the resistance-capacitor model a bit. To make things easier, we can simplify the model variables by giving the resistors a resistance, R, and the capacitors a capacitance, C, which can each be calculated from the more specific variables described before.

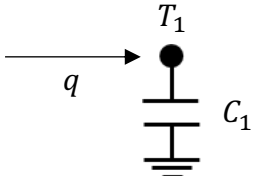


For a resistor, a constitutive equation relates the heat flow through it to the resistance and temperature different across it:



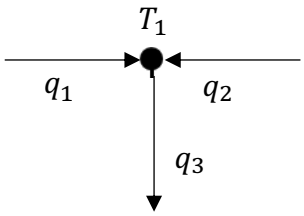
$$q = \frac{T_1 - T_2}{R}$$

For a capacitor, a state equation relates the heat flow into the capacitor to the capacitance and time-rate-of-change of the temperature:



$$C_1 \frac{dT_1}{dt} = q$$

For any node where two or more components (a resistor or capacitor) connect, Kirchoff Current and Voltage Laws (also conservation of energy) say that the net heat flow at the node is zero and the temperature of all components at the node is equal. That is, energy that leaves one component must flow into its neighbors and at the point it connects to its neighbors the temperature must be the same:



$$q_1 + q_2 - q_3 = 0$$

Thinking back to our original problem, we have 19 unknown variables (4 R's, 3 C's, 5 node temperatures, and 7 component heat flows). We can assign all of the R's and C's based on material properties. Furthermore, we can assume the outside air temperature is just an input that we define. Therefore, we have 11 remaining unknown variables; 4 remaining node temperatures and 7 component heat flows. If we build a generic resistor model using the constitutive equation and instantiate 4 of them, that gives us 4 more equations. Similarly, if we build a generic capacitor model using the state equation and instantiate 3 of them, that gives us another 3 equations. Finally, if we could "connect" these components together as shown in the diagram, we could gain 1 more equation per node which says that the sum of heat flows is zero at the node. Then, we would have 4+3+4=11 equations, which will allow us to solve for all 11 unknown variables!

Why did we go through all of this? What it tells us is we can individually build generic component models that just define the relationship between the heat flow through the component and the temperature(s) at the interface(s). Then, we can connect these component models together in virtually any configuration and have enough equations to solve for the whole system!

Now, let's call the heat flow the "flow" variable and the temperature the "potential" variable for a particular component (you can think of heat flowing through the component while the temperature drives the potential for heat flow). Then, we could say more generally that at a connection node: "flow" variables sum to zero and "potential" variables are equal. As it turns out, this pairing of "flow"

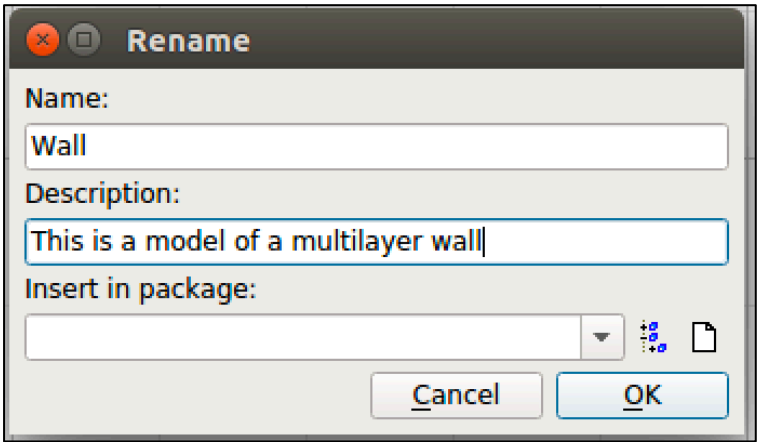
and “potential” variables is analogous in other domains than just heat transfer. In particular, electrical circuits, fluid flow, translational kinematics, and rotational kinematics! This is one of the most powerful concepts that Modelica takes advantage of in order to be able to facilitate the simulation of new systems. If you’re interested in learning more about this concept of modeling that generalizes across physical domains, it is often referred to as Linear Graph Modeling, and a good discussion is presented from the MIT Mechanical Engineering Department:

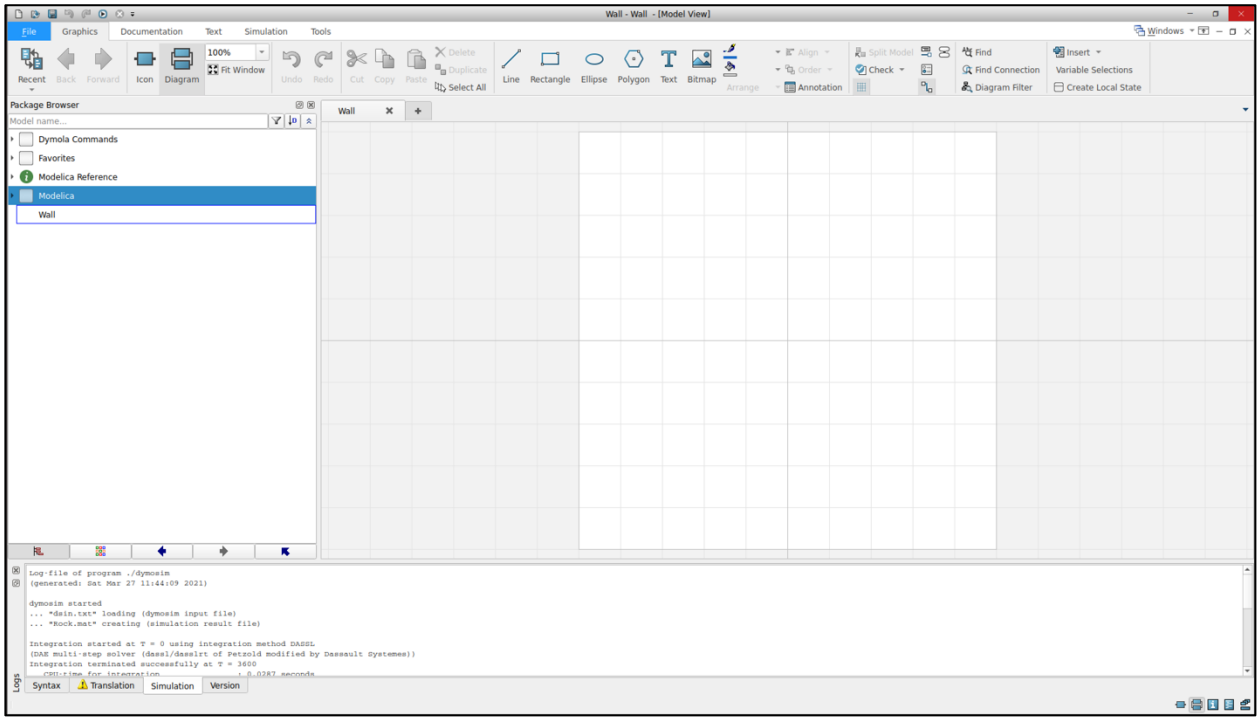
- Part 1: http://web.mit.edu/2.14/www/Handouts/OnePorts.pdf
- Part 2: http://web.mit.edu/2.14/www/Handouts/TwoPorts.pdf

Physical Domain	**Potential (Across)**	**Flow (Through)**
Heat Transfer	Temperature (T)	Heat Flow (q)
Fluid Flow	Pressure (P)	Mass Flow (m)
Electrical	Voltage (V)	Current (i)
Translational	Displacement (s)	Force (F)
Rotational	Angle (θ)	Torque (τ)

Having gone through all of this, let’s build our wall model.

1. Open Dymola. Go to File > Save > SaveAs and save the model file as shown in the Figure below (it will be saved as Wall.mo).

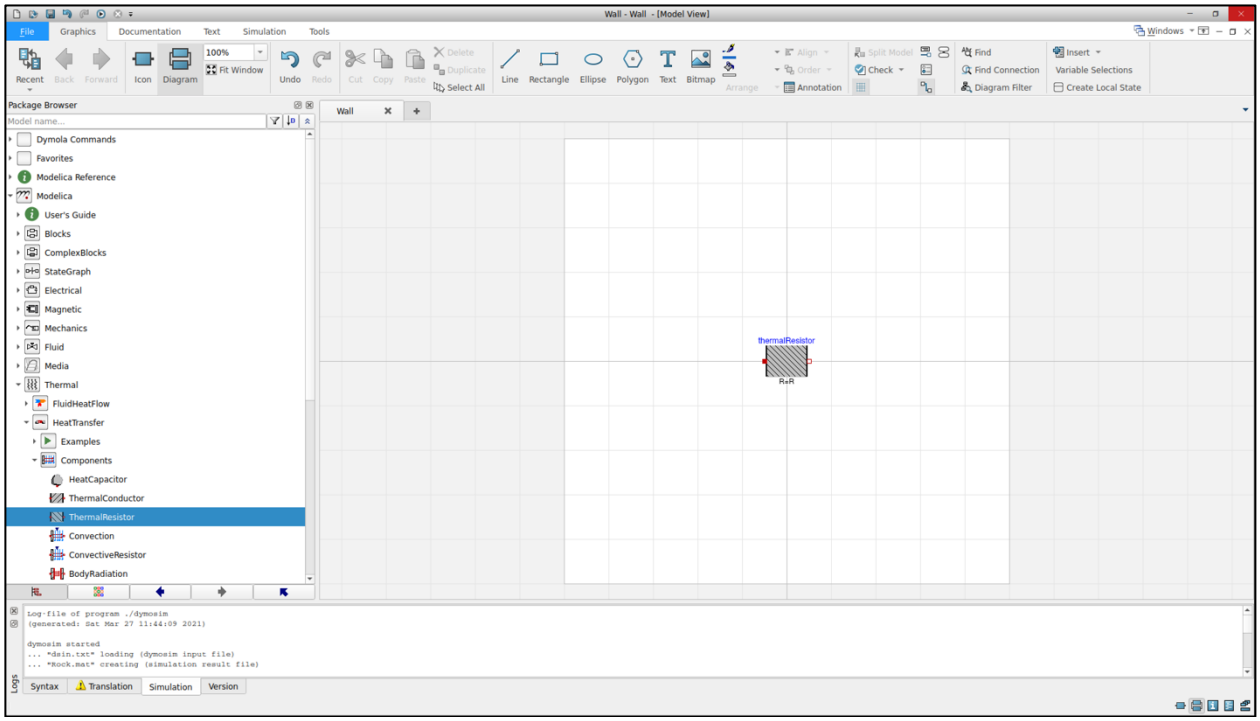


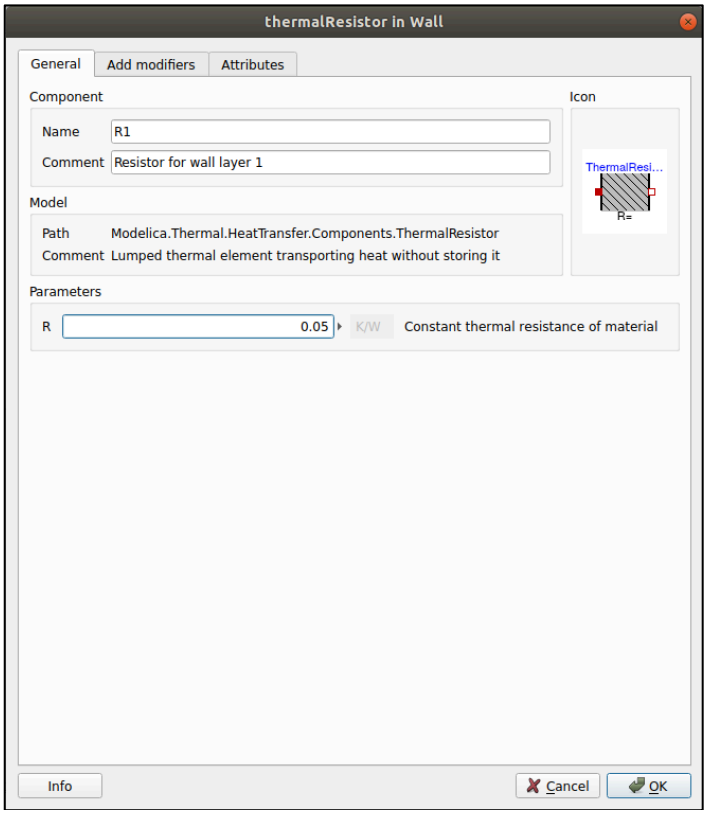


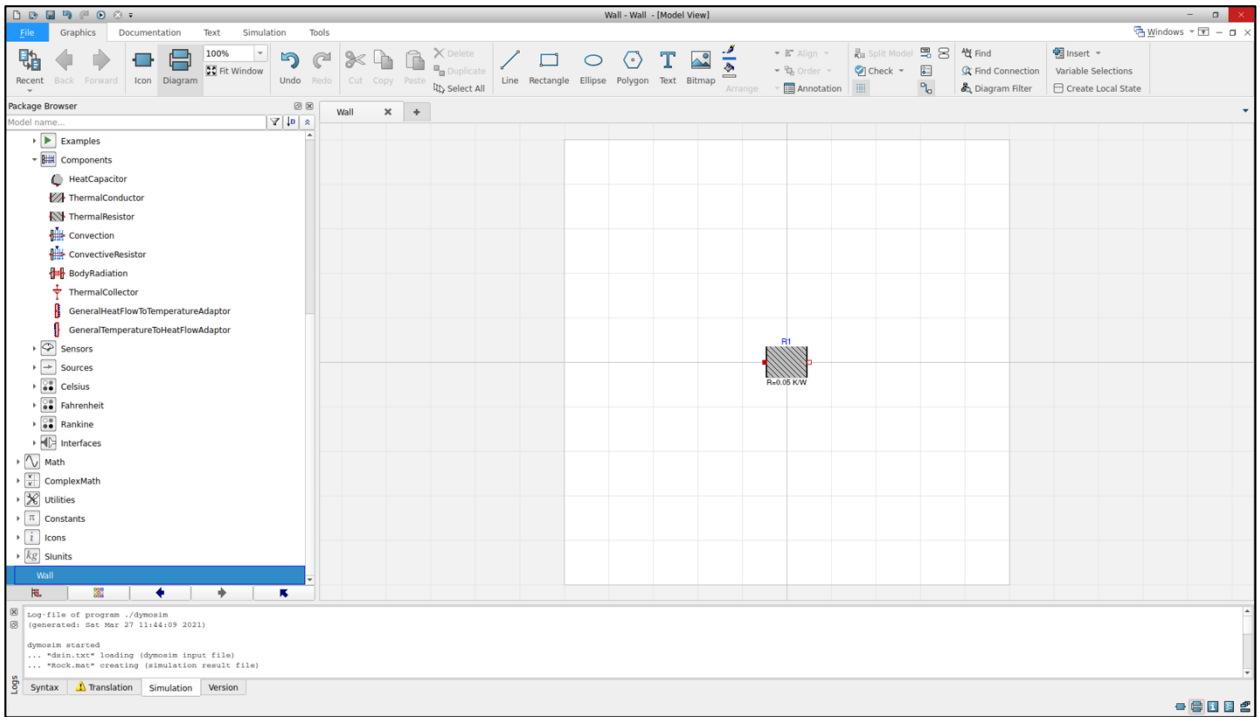
2. Explore the Modelica Standard Library using the Package Browser on the left side of the screen. In particular, navigate to the Thermal Resistor component model at `Modelica.Thermal.HeatTransfer.Components.ThermalResistor`. Then, click and drag it onto the modeling canvas. Then, double-click on the model to configure it with the name "R1" and parameter R=0.05 K/W.

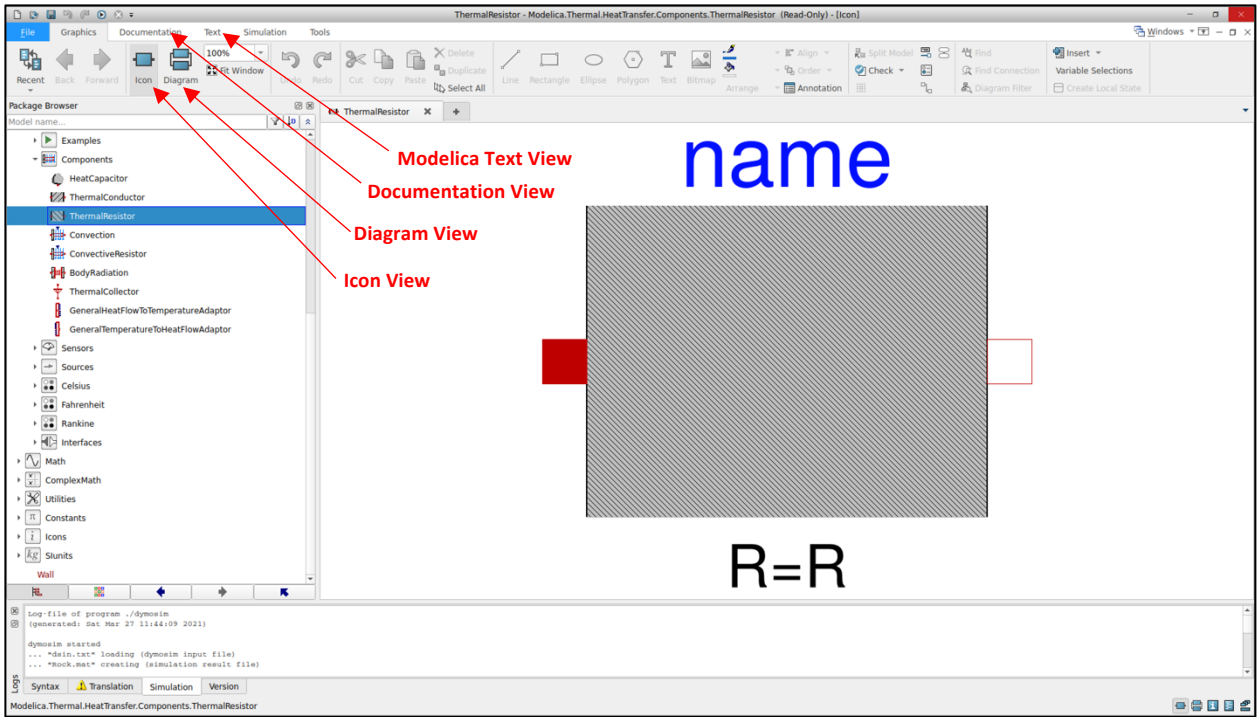
Double-click on the Thermal Resistor model in the Package Browser to open it. Switch to the Modelica Text view (as we used in Part 1) and notice that the graphical component has a corresponding text implementation. In fact, notice that the parameter R is defined just as we did in our analytical model in Part 1, and that Dymola used that information to populate the configuration we just used to assign a value of 0.05 in a graphical context. Try also switching to the "Documentation" view to read any information the model developer may have given about the model, such as modeling approach, major assumptions, and typical usage. Finally, switch to the "Icon" view to see how the graphic content is constructed.

After you've finished exploring, switch back to the "Diagram" view of our Wall model to continue building.




thermalResistor in Wall
General
Add modifiers
Attributes
Component
Name
R1
Comment
Resistor for wall layer 1
Icon
ThermalResi...
R=
Model
Path
Modelica.Thermal.HeatTransfer.Components.ThermalResistor
Comment
Lumped thermal element transporting heat without storing it
Parameters
R
0.05
K/W
Constant thermal resistance of material
Info
Cancel
OK

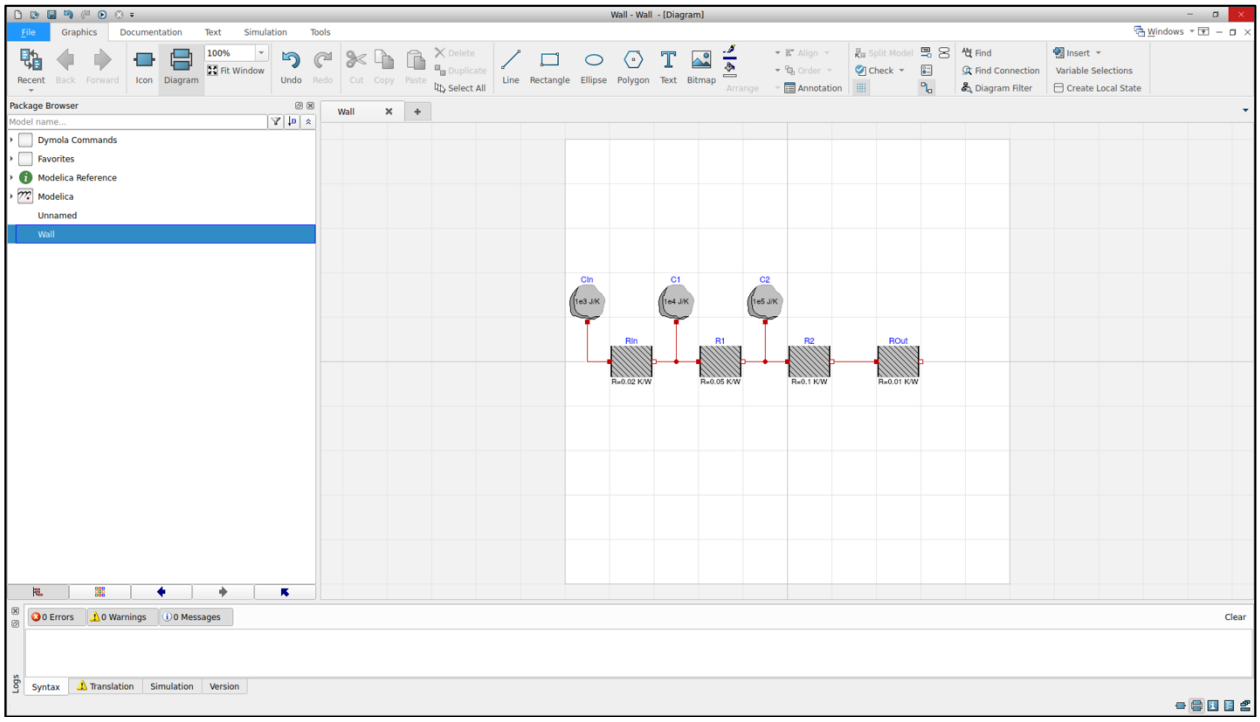

Wall - Wall - [Model View]
File
Graphics
Documentation
Text
Simulation
Tools
Package Browser
Examples
Components
HeatCapacitor
ThermalConductor
ThermalResistor
Convection
ConvectiveResistor
BodyRadiation
ThermalCollector
GeneralHeatFlowToTemperatureAdaptor
GeneralTemperatureToHeatFlowAdaptor
Sensors
Sources
Celsius
Fahrenheit
Rankine
Interfaces
Math
ComplexMath
Utilities
Constants
Icons
SIunits
Wall
R1
R=0.05 K/W
Log-file of program ./dymosim
(generated: Sat Mar 27 11:44:09 2021)
dymosim started
... "dsin.txt" loading (dymosim input file)
... "Rock.mat" creating (simulation result file)
Syntax
Translation
Simulation
Version


ThermalResistor - Modelica.Thermal.HeatTransfer.Components.ThermalResistor (Read-Only) - [Icon]
File
Graphics
Documentation
Text
Simulation
Tools
Recent
Back
Forward
Icon
Diagram
100%
Fit Window
Undo
Redo
Cut
Copy
Paste
Delete
Duplicate
Select All
Line
Rectangle
Ellipse
Polygon
Text
Bitmap
Arrange
Align
Order
Annotation
Split Model
Check
Find
Find Connection
Diagram Filter
Insert
Variable Selections
Create Local State
Package Browser
Model name...
Examples
Components
HeatCapacitor
ThermalConductor
ThermalResistor
Convection
ConvectiveResistor
BodyRadiation
ThermalCollector
GeneralHeatFlowToTemperatureAdaptor
GeneralTemperatureToHeatFlowAdaptor
Sensors
Sources
Celsius
Fahrenheit
Rankine
Interfaces
Math
ComplexMath
Utilities
Constants
Icons
SIunits
Wall
ThermalResistor
Modelica Text View
Documentation View
Diagram View
Icon View
name
R=R
Log-file of program ./dymosim
(generated: Sat Mar 27 11:44:09 2021)
dymosim started
... "dsin.txt" loading (dymosim input file)
... "Rock.mat" creating (simulation result file)
Logs
Syntax
Translation
Simulation
Version
Modelica.Thermal.HeatTransfer.Components.ThermalResistor

3. Add a second resistor to the Wall model by clicking and dragging a second instance of `Modelica.Thermal.HeatTransfer.Components.ThermalResistor` to the right of R1. Then, configure it to have the name "R2" and parameter R=0.1 K/W. Then, click the right heat port of R1 and hold while dragging your mouse to the left heat port of R2. Then, let go. This "connects" the two components together at that heat port. Continue this process of dragging, dropping, configuring, and connecting components until all resistors and capacitors are added according to the specifications in the table below. For capacitors, use `Modelica.Thermal.HeatTransfer.Components.HeatCapacitor`.

Component Name	Parameter Value
R1	R=0.05 K/W
R2	R=0.1 K/W
ROut	R=0.01 K/W
RIn	R=0.02 K/W
C1	C=1e4 J/K
C2	C=1e5 J/K
CIn	C=1e3 J/K

Once the component models are implemented, switch to the Modelica Text view and explore the corresponding text syntax of the wall model. Notice the declaration of component models, instead of individual variables, and the uses of the "connect" statements in the equation section.

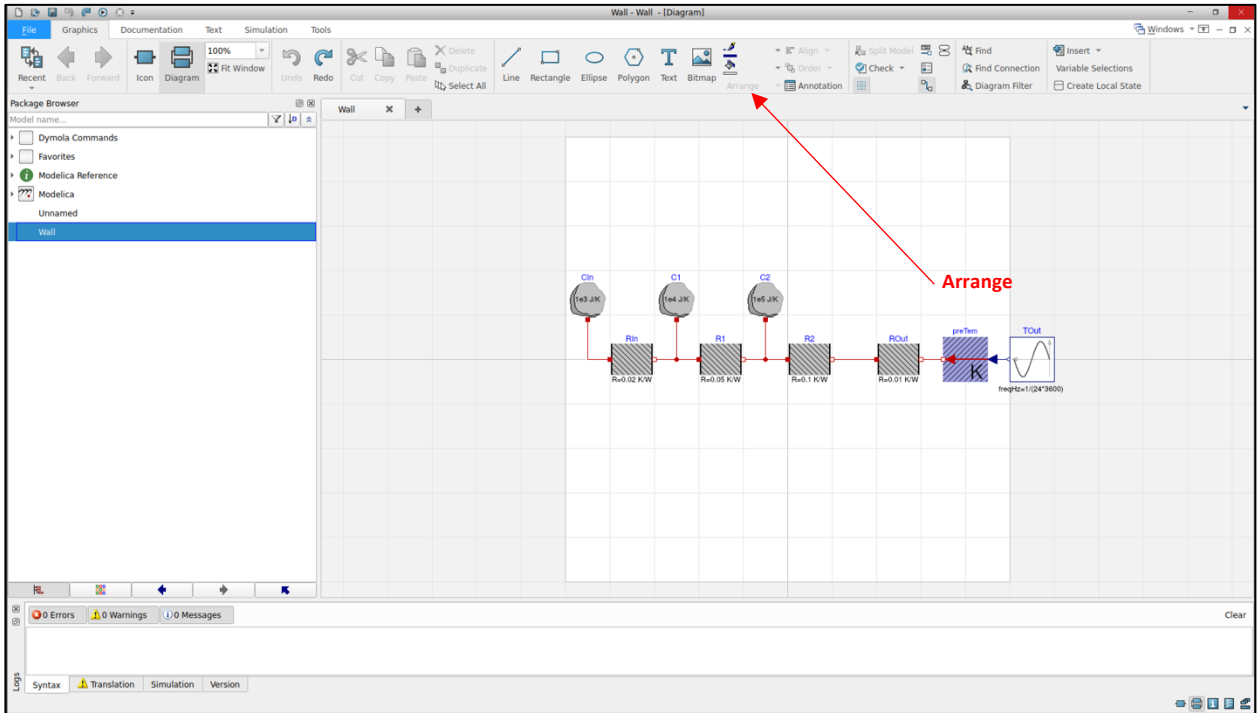


```
model Wall "This is a model of a multilayer wall"

  Modelica.Thermal.HeatTransfer.Components.ThermalResistor R1(R=0.05)
    "Resistor for wall layer 1"
    ;
  Modelica.Thermal.HeatTransfer.Components.ThermalResistor R2(R=0.1)
    "Resistor for wall layer 2"
    ;
  Modelica.Thermal.HeatTransfer.Components.ThermalResistor ROut(R=0.01)
    "Resistor for outside convection"
    ;
  Modelica.Thermal.HeatTransfer.Components.ThermalResistor RIn(R=0.02)
    "Resistor for inside convection"
    ;
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor C1(C=1e4)
    "Capacitor for wall layer 1"
    ;
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor C2(C=1e5)
    "Capacitor for wall layer 2"
    ;
  Modelica.Thermal.HeatTransfer.Components.HeatCapacitor CIn(C=1e3)
    "Capacitor for inside air volume"
    ;
equation
  connect(R1.port_b, R2.port_a)
    ;
  connect(R2.port_b, ROut.port_a)
    ;
  connect(RIn.port_b, R1.port_a)
    ;
  connect(C1.port, R1.port_a)
    ;
  connect(C2.port, R2.port_a)
    ;
  connect(C2.port, R1.port_b)
    ;
  connect(RIn.port_a, CIn.port)
    ;
  
end Wall;
```

4. Specify the outside air temperature by using `Modelica.Blocks.Sources.Sine` and `Modelica.Thermal.HeatTransfer.Sources.PrescribedTemperature`. For the sine block, use the parameterization shown on the next page. Flip the direction of the blocks by selecting on them and using Arrange > Flip Horizontal at the top of the screen (a keyboard shortcut is to press the letter "h" while selected).

Note that the sine block has an interface made of an "output" instead of a "heat port" as the heat transfer models do. This block outputs a Real signal. In Modelica, we can incorporate the generation, mathematical manipulation, and usage of Real signals in our system models. The connection of signals from an output of one component to the input of another is done similarly to connecting the heat ports from before. Here, the prescribed temperature component model uses a Real signal as an input to specify the value of the temperature at its heat port, which allows us to define the outside temperature node for ROut. In this case, according to a sinusoidal signal defined by the sine block. Explore some of the other types of Real signal generation blocks, as well as Boolean signals and available logic blocks. Note that signals are only for the passage of numeric or Boolean values from one component to another and do not represent anything physical like heat ports.



TOut in Wall

General | Add modifiers | Attributes

Component

Name: TOut

Comment: Outside air temperature

Icon

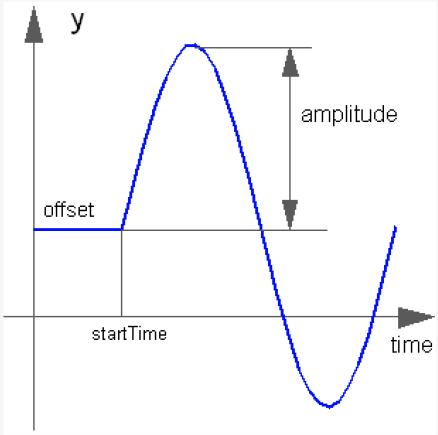
Model

Path: Modelica.Blocks.Sources.Sine

Comment: Generate sine signal

Parameters

amplitude	10		Amplitude of sine wave
freqHz	1/(24*3600)	Hz	Frequency of sine wave
phase	0	°	Phase of sine wave
offset	273.15 + 20		Offset of output signal y
startTime	0	s	Output y = offset for time < startTime

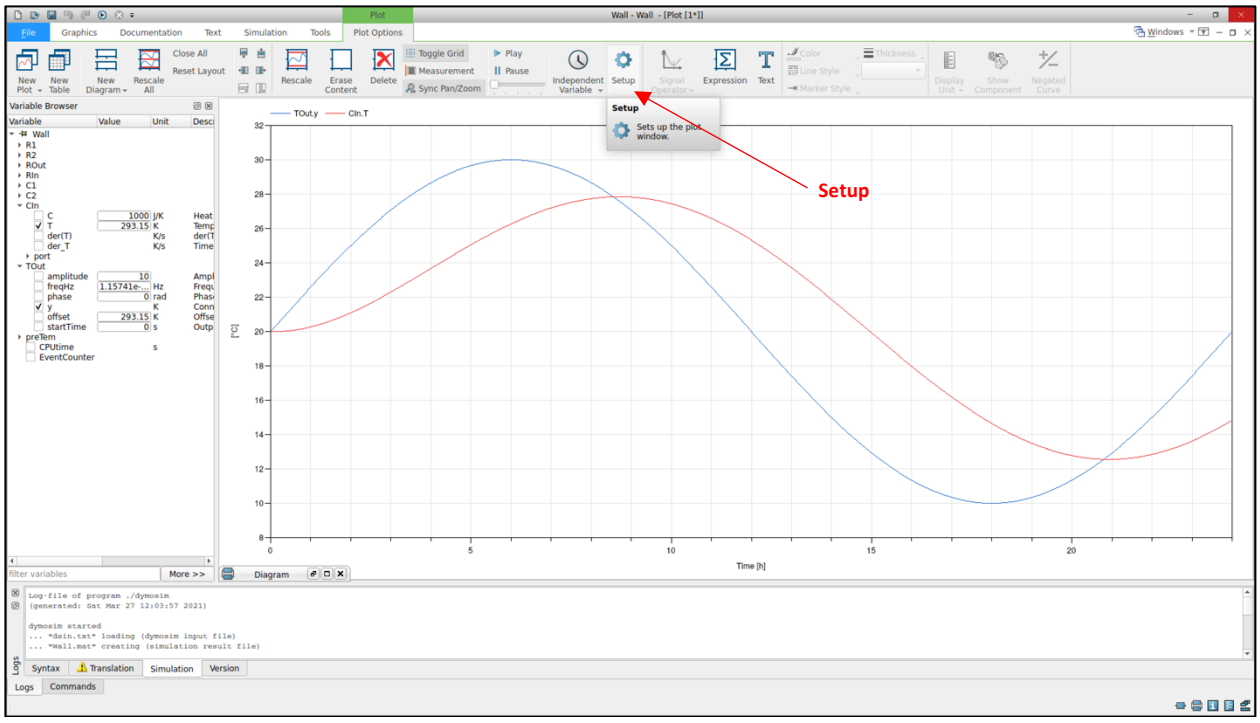


Info | Cancel | OK

5. Simulate the model for 1 day (86400 seconds) and an output interval of 60 seconds with the Dassl solver and tolerance of 1e-6. Plot the outside temperature and the inside temperature as shown in the Figure below.

Change the x-axis to hours by right-clicking on the word "Time" > Time Unit > h. Change the y-axis to ℃ by clicking Setup… > Display Unit > degC. Export the results in the plot to a .csv file by right-clicking the word "Wall" in the Variable Browser > Export Result > Only Plot Window. Then, for "Files of type" select "Comma Separated Values", give the file a name, and save it. Try opening it in excel or using it in a Python script for post-processing.

Try changing the values of the R and C parameters and explore how it changes the indoor air temperature. Plot the temperatures of C1 and C2 to understand how the temperature is changing through the wall over time. Add more layers to the wall, or try adding more sources of heat to the outside, like the sun, or inside, like lights! For this, you will find the model `Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow` *helpful.*



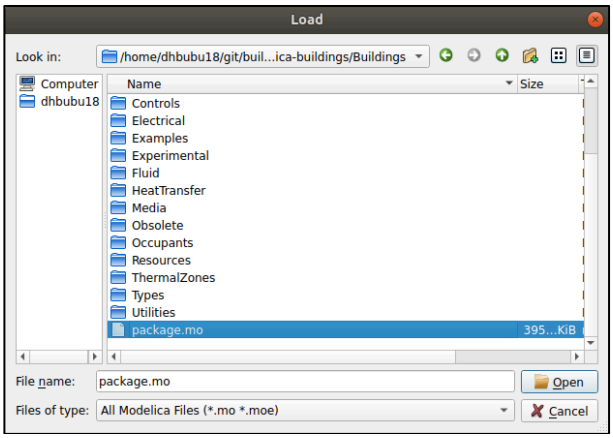
Part III: System Modeling

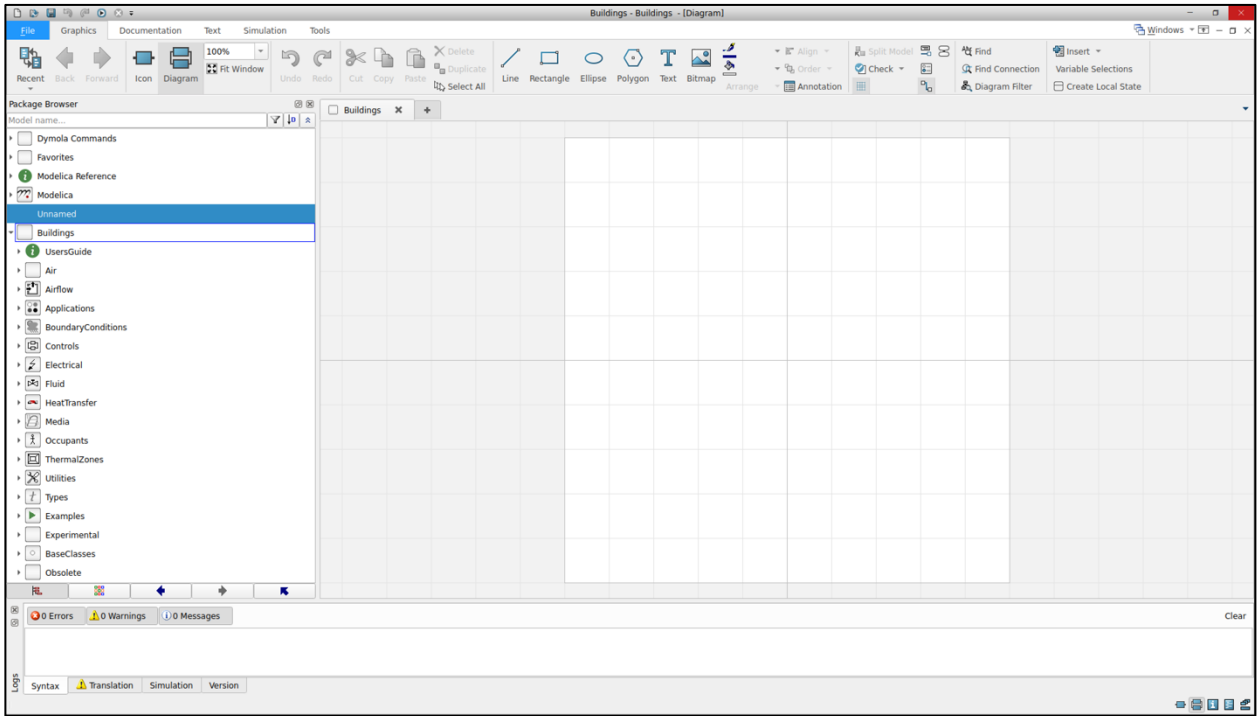
Now that you've learned the basics of Modelica syntax, the Dymola environment, and connecting component models to form larger models, let's build and analyze a simple system model that contains a room and HVAC system for cooling. In doing so, we can explore the performance of a number of design and control scenarios.

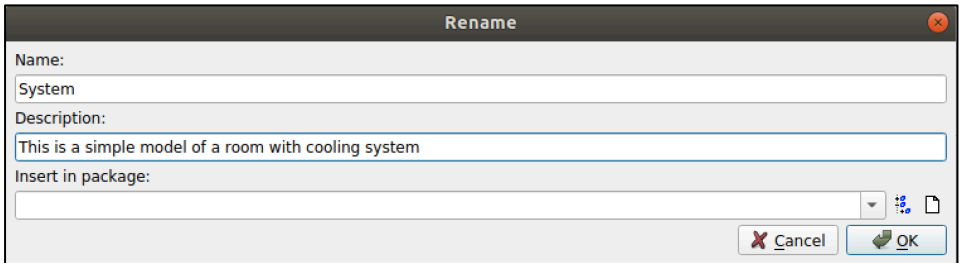
Therefore, in this part, you will:

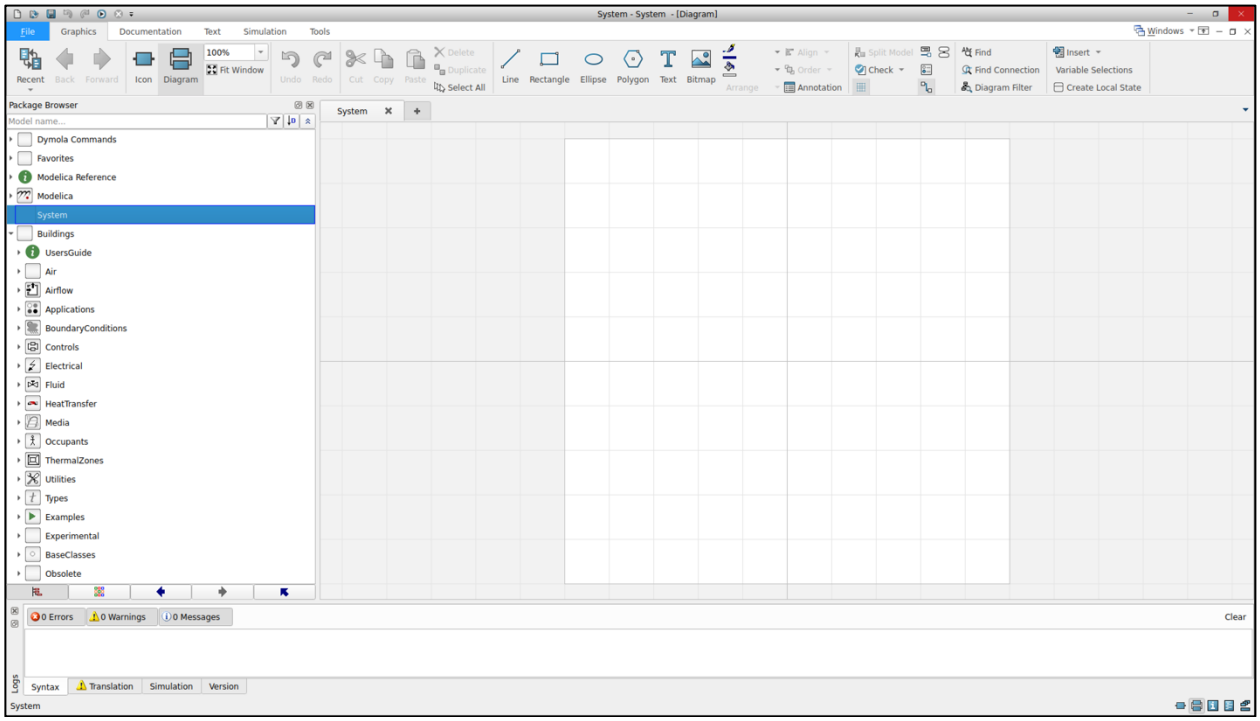
1. Implement a system model that includes a room, fan, and weather data.
2. Explore the impact of increased thermal mass on the inside air temperature with and without fan operation.
3. Add a simple vapor-compression cooling coil with thermostat control to maintain the room air temperature according to a setpoint.
4. Adjust the room air temperature setpoint to simulate a demand-response event and compare the cooling power profile to the case without the demand-response event.

1. Open Dymola. Load the Modelica Buildings Library by File > Open > Load and browsing to the "package.mo" file within the Buildings Library. Once loaded, you should see Buildings available in your Package Browser. Then, go to File > Save > SaveAs and save the model file as "System.mo."



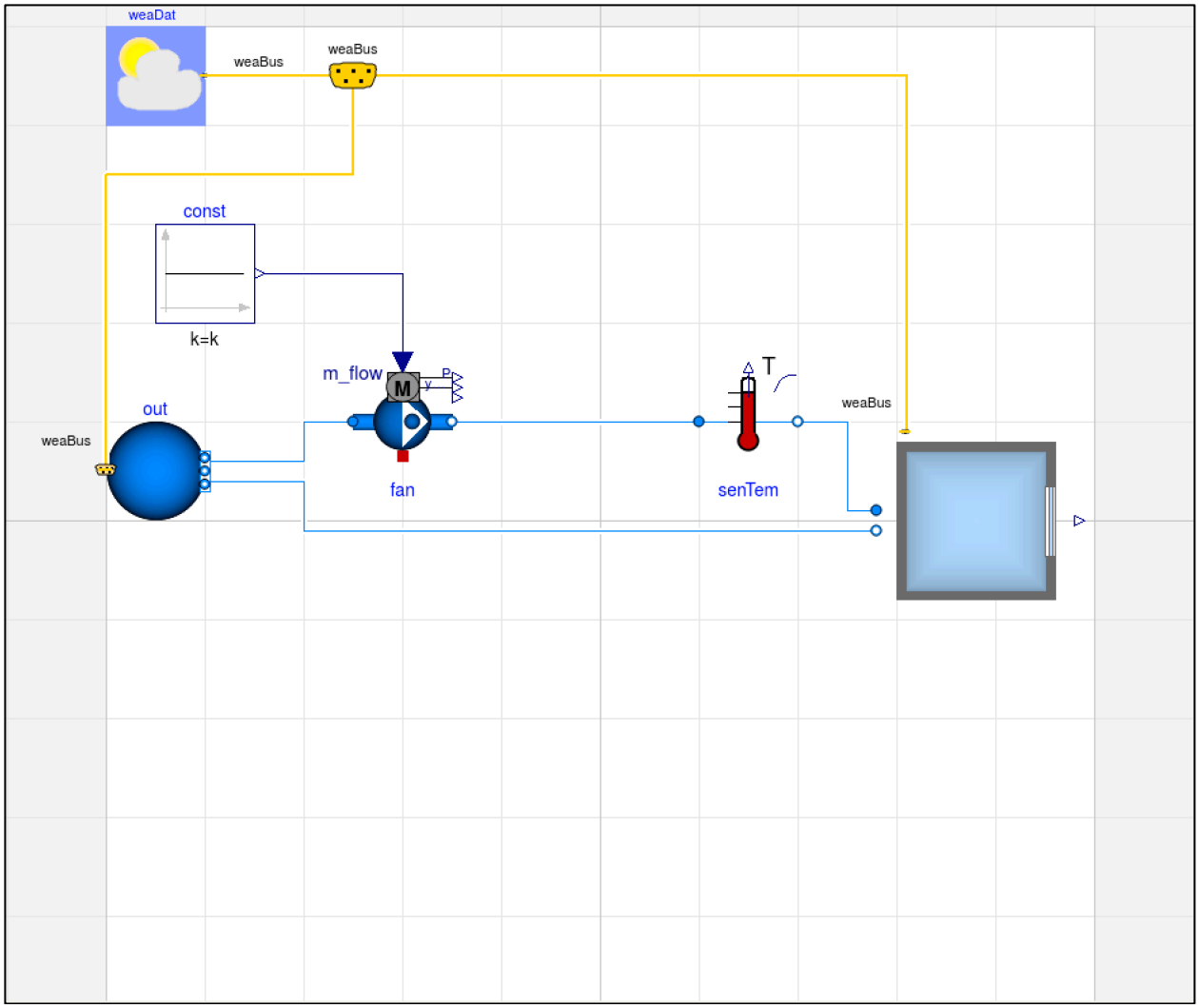


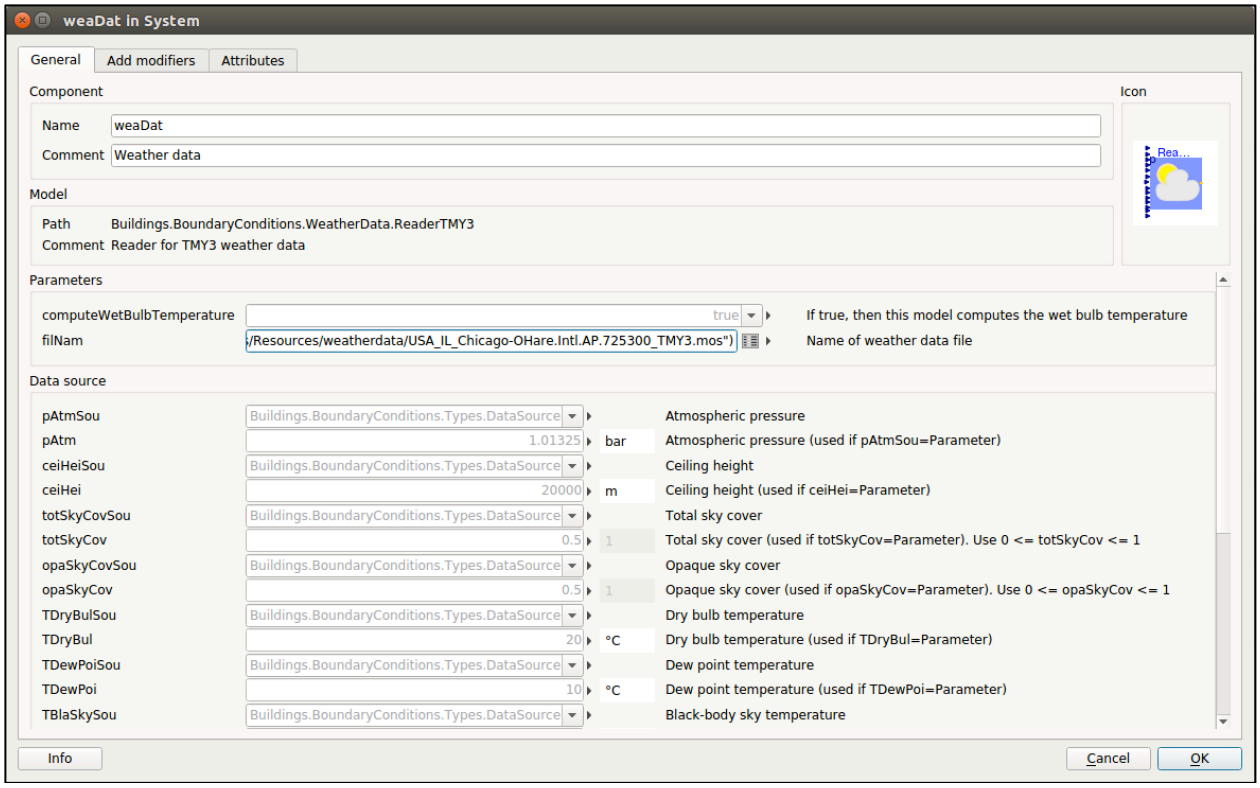

Rename
Name:
System
Description:
This is a simple model of a room with cooling system
Insert in package:
Cancel
OK


System - System - [Diagram]
File
Graphics
Documentation
Text
Simulation
Tools
Windows
Recent
Back
Forward
Icon
Diagram
100%
Fit Window
Undo
Redo
Cut
Copy
Paste
Delete
Duplicate
Select All
Line
Rectangle
Ellipse
Polygon
Text
Bitmap
Arrange
Align
Order
Annotation
Split Model
Check
Find
Find Connection
Diagram Filter
Insert
Variable Selections
Create Local State
Package Browser
Model name...
Dymola Commands
Favorites
Modelica Reference
Modelica
System
Buildings
UsersGuide
Air
Airflow
Applications
BoundaryConditions
Controls
Electrical
Fluid
HeatTransfer
Media
Occupants
ThermalZones
Utilities
Types
Examples
Experimental
BaseClasses
Obsolete
System
0 Errors
0 Warnings
0 Messages
Clear
Logs
Syntax
Translation
Simulation
Version
System

2. Use the following component models to build the system as shown in the diagram below. Screenshots are shown for how to parameterize the models. Note that the parameter values to edit are displayed in black text. Those in grey text are defaults that are already configured and should be left as they are.

Model Path	**Description/Usage**
Buildings.BoundaryConditions.WeatherData.ReaderTMY3	Loads weather data from a weather file based on EPW.
Buildings.BoundaryConditions.WeatherData.Bus	Allows for the use of weather data throughout the model.
Buildings.Air.Systems.SingleZone.VAV.Examples.BaseClasses.Room	Single zone room model with 4 exterior walls, a roof, and a floor. The south wall has a large window. The model implements a detailed heat balance to calculate the inside air temperature, including conduction through exterior walls/roof/floor from the outside (using a 1-D finite difference approximation), solar radiation on exterior surfaces and windows, interior radiation exchange, surface convection based on temperature difference and surface orientation, internal gains and schedules, and infiltration.
Buildings.Fluid.Movers.FlowControlled_m_flow	Fan model with mass flowrate as control input signal.
Buildings.Fluid.Sensors.TemperatureTwoPort	Temperature sensor for supply air.
Buildings.Fluid.Sources.Outside	Ideal boundary from where air can flow into the fan and out of the room. Weather data can be used to specify the conditions of the boundary air through time.
Modelica.Blocks.Sources.Constants	Generates a constant signal as output. Used here as a control signal for the fan.

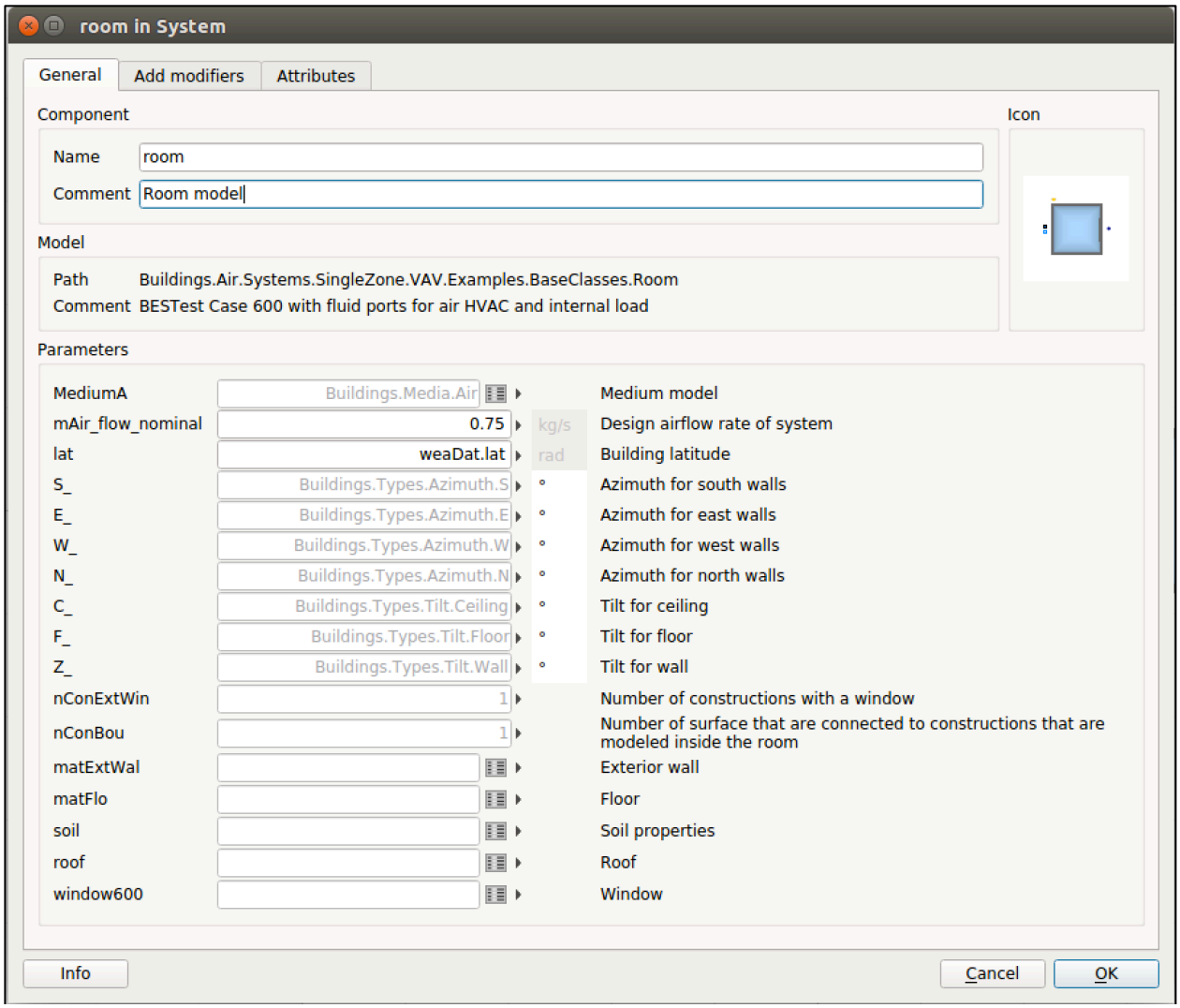

weaDat
weaBus
weaBus
const
k=k
m_flow
M
P
y
out
weaBus
fan
T
senTem
weaBus

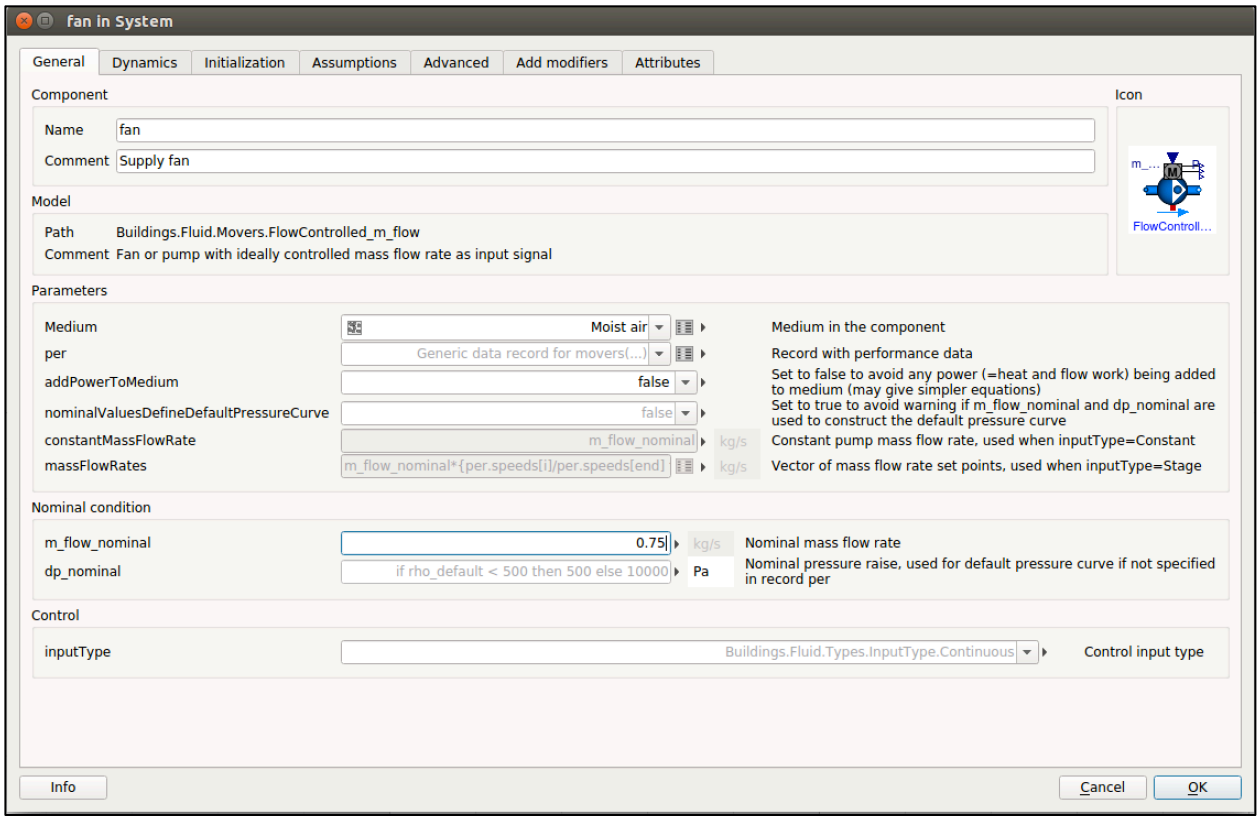


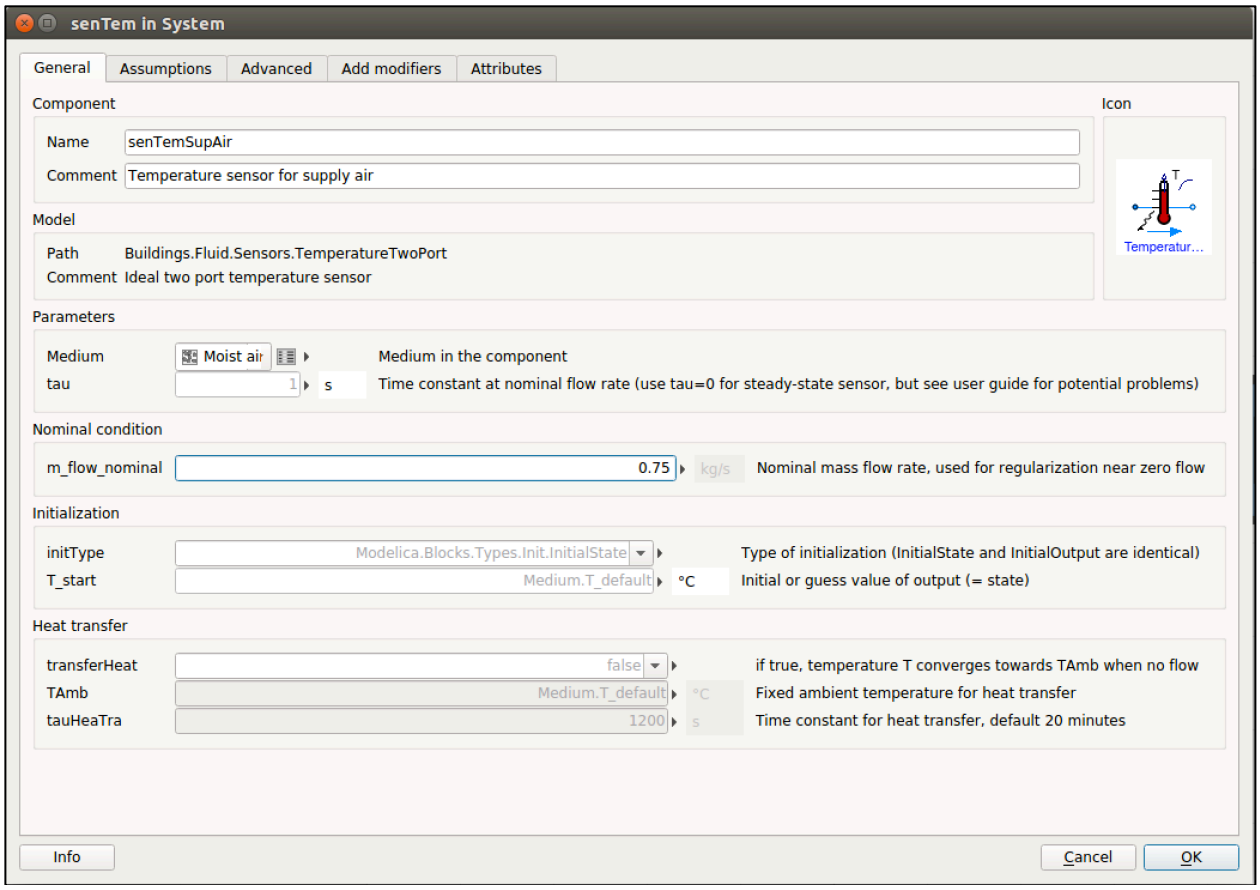
For the "filNam" parameter, specify the full path to the weather file within the quotations. There are example weather files within the Buildings library. Here is an example on my system of how the parameter should look:

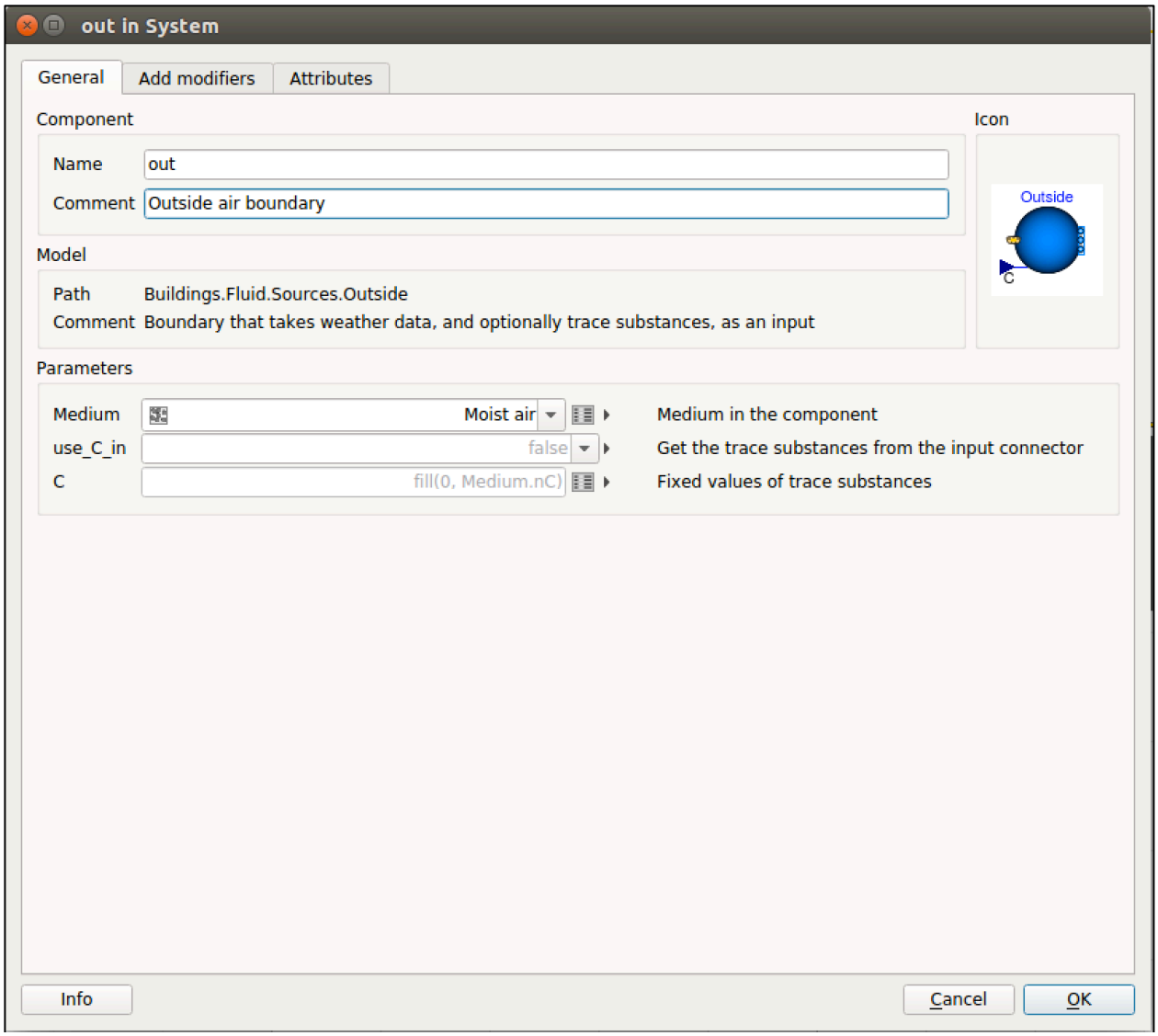
```
ModelicaServices.ExternalReferences.loadResource(
  "/home/dhbubu/git/buildings/modelica-buildings/Buildings/Resources/weatherdata/USA_IL_Chicago-OHare.Intl.AP.725300_TMY3.mos")
```

Notice that the weather file is a .mos file and not a .epw file. This is because the data format needs to be in a "Modelica table" format to be read into the model. The Buildings library provides a program to convert .epw files into .mos files. The program is located at Buildings/Resources/bin/ConvertWeatherData.jar and instructions can be found in the Documentation section "Adding new weather data" of the model Buildings.BoundaryConditions.WeatherData.ReaderTMY3.


room in System
General
Add modifiers
Attributes
Component
Name
room
Comment
Room model
Icon
Model
Path
Buildings.Air.Systems.SingleZone.VAV.Examples.BaseClasses.Room
Comment
BESTest Case 600 with fluid ports for air HVAC and internal load
Parameters
MediumA
Buildings.Media.Air
Medium model
mAir_flow_nominal
0.75
kg/s
Design airflow rate of system
lat
weaDat.lat
rad
Building latitude
S_
Buildings.Types.Azimuth.S
°
Azimuth for south walls
E_
Buildings.Types.Azimuth.E
°
Azimuth for east walls
W_
Buildings.Types.Azimuth.W
°
Azimuth for west walls
N_
Buildings.Types.Azimuth.N
°
Azimuth for north walls
C_
Buildings.Types.Tilt.Ceiling
°
Tilt for ceiling
F_
Buildings.Types.Tilt.Floor
°
Tilt for floor
Z_
Buildings.Types.Tilt.Wall
°
Tilt for wall
nConExtWin
1
Number of constructions with a window
nConBou
1
Number of surface that are connected to constructions that are modeled inside the room
matExtWal
Exterior wall
matFlo
Floor
soil
Soil properties
roof
Roof
window600
Window
Info
Cancel
OK


fan in System
General
Dynamics
Initialization
Assumptions
Advanced
Add modifiers
Attributes
Component
Name
fan
Comment
Supply fan
Icon
FlowControll...
Model
Path
Buildings.Fluid.Movers.FlowControlled_m_flow
Comment
Fan or pump with ideally controlled mass flow rate as input signal
Parameters
Medium
Moist air
Medium in the component
per
Generic data record for movers(...)
Record with performance data
addPowerToMedium
false
Set to false to avoid any power (=heat and flow work) being added to medium (may give simpler equations)
Set to true to avoid warning if m_flow_nominal and dp_nominal are used to construct the default pressure curve
nominalValuesDefineDefaultPressureCurve
false
constantMassFlowRate
m_flow_nominal
kg/s
Constant pump mass flow rate, used when inputType=Constant
massFlowRates
m_flow_nominal*{per.speeds[i]/per.speeds[end]}
kg/s
Vector of mass flow rate set points, used when inputType=Stage
Nominal condition
m_flow_nominal
0.75
kg/s
Nominal mass flow rate
dp_nominal
if rho_default < 500 then 500 else 10000
Pa
Nominal pressure raise, used for default pressure curve if not specified in record per
Control
inputType
Buildings.Fluid.Types.InputType.Continuous
Control input type
Info
Cancel
OK


senTem in System
General
Assumptions
Advanced
Add modifiers
Attributes
Component
Name
senTemSupAir
Comment
Temperature sensor for supply air
Icon
Temperatur…
Model
Path
Buildings.Fluid.Sensors.TemperatureTwoPort
Comment
Ideal two port temperature sensor
Parameters
Medium
Moist air
Medium in the component
tau
1
s
Time constant at nominal flow rate (use tau=0 for steady-state sensor, but see user guide for potential problems)
Nominal condition
m_flow_nominal
0.75
kg/s
Nominal mass flow rate, used for regularization near zero flow
Initialization
initType
Modelica.Blocks.Types.Init.InitialState
Type of initialization (InitialState and InitialOutput are identical)
T_start
Medium.T_default
°C
Initial or guess value of output (= state)
Heat transfer
transferHeat
false
if true, temperature T converges towards TAmb when no flow
TAmb
Medium.T_default
°C
Fixed ambient temperature for heat transfer
tauHeaTra
1200
s
Time constant for heat transfer, default 20 minutes
Info
Cancel
OK


out in System
General
Add modifiers
Attributes
Component
Name
out
Comment
Outside air boundary
Icon
Outside
C
Model
Path
Buildings.Fluid.Sources.Outside
Comment
Boundary that takes weather data, and optionally trace substances, as an input
Parameters
Medium
Moist air
Medium in the component
use_C_in
false
Get the trace substances from the input connector
C
fill(0, Medium.nC)
Fixed values of trace substances
Info
Cancel
OK

const in System

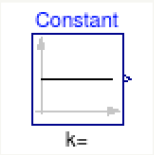
General | Add modifiers | Attributes

Component

Name: con

Comment: Control signal for fan

Icon



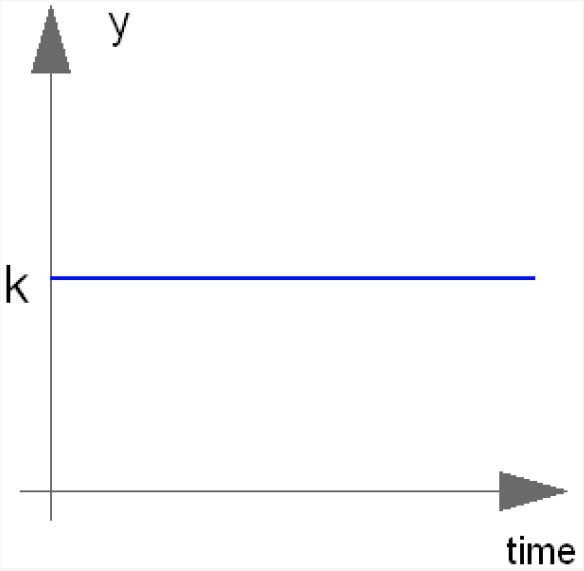
Model

Path: Modelica.Blocks.Sources.Constant

Comment: Generate constant signal of type Real

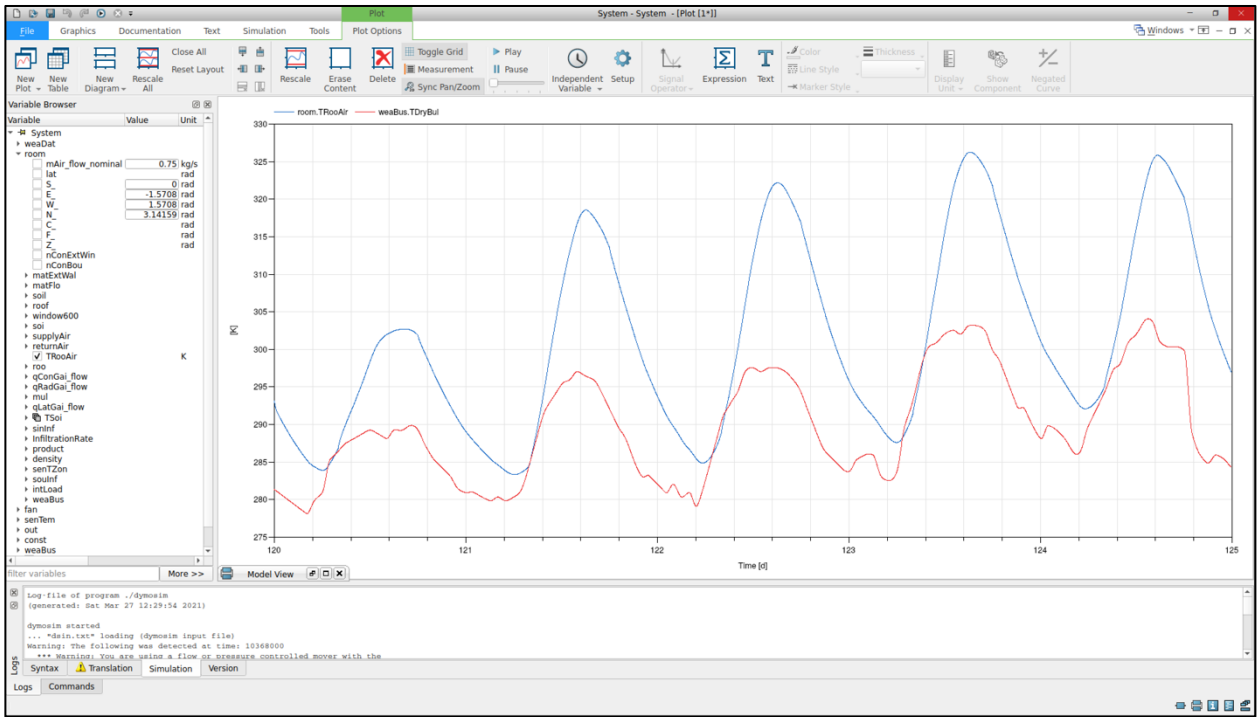
Parameters

k: 0 — Constant output value



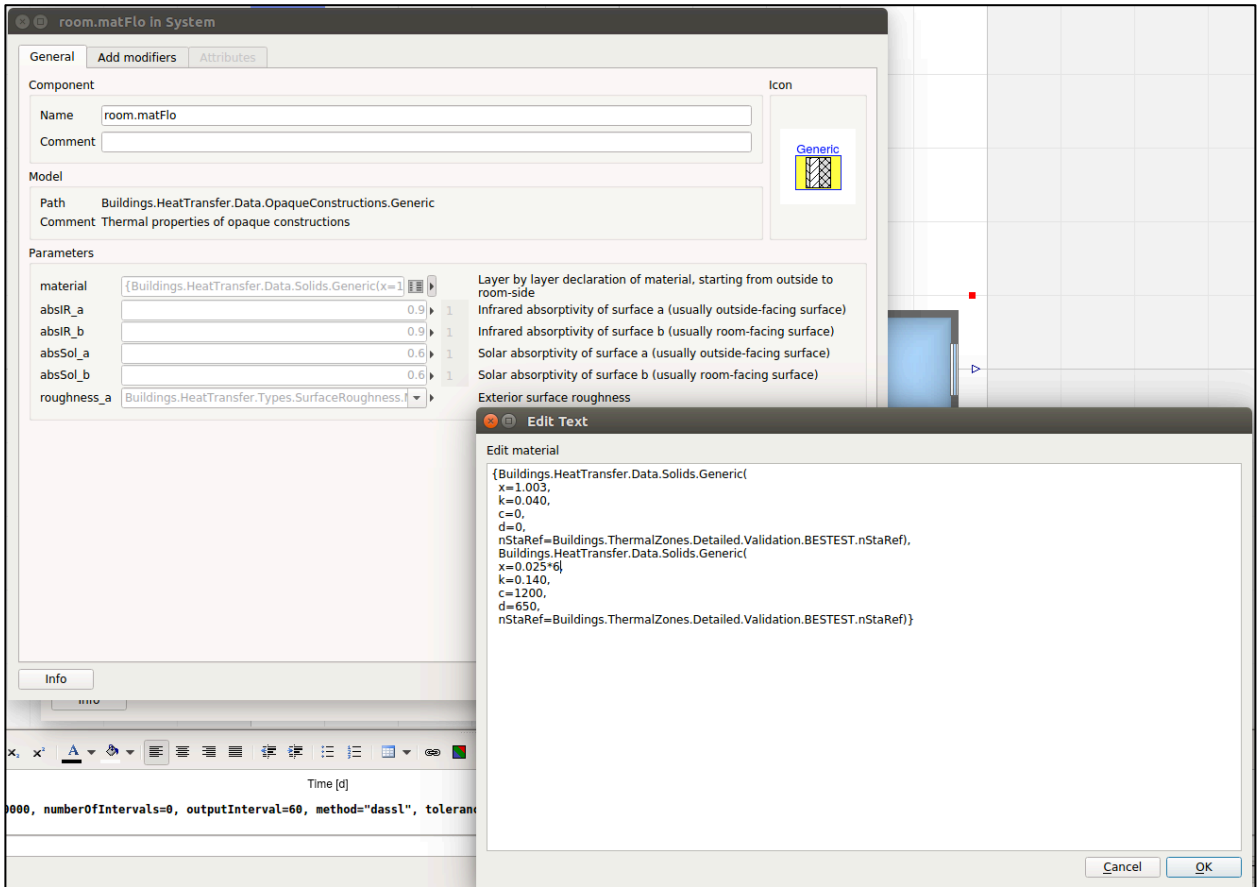
Info | Cancel | OK

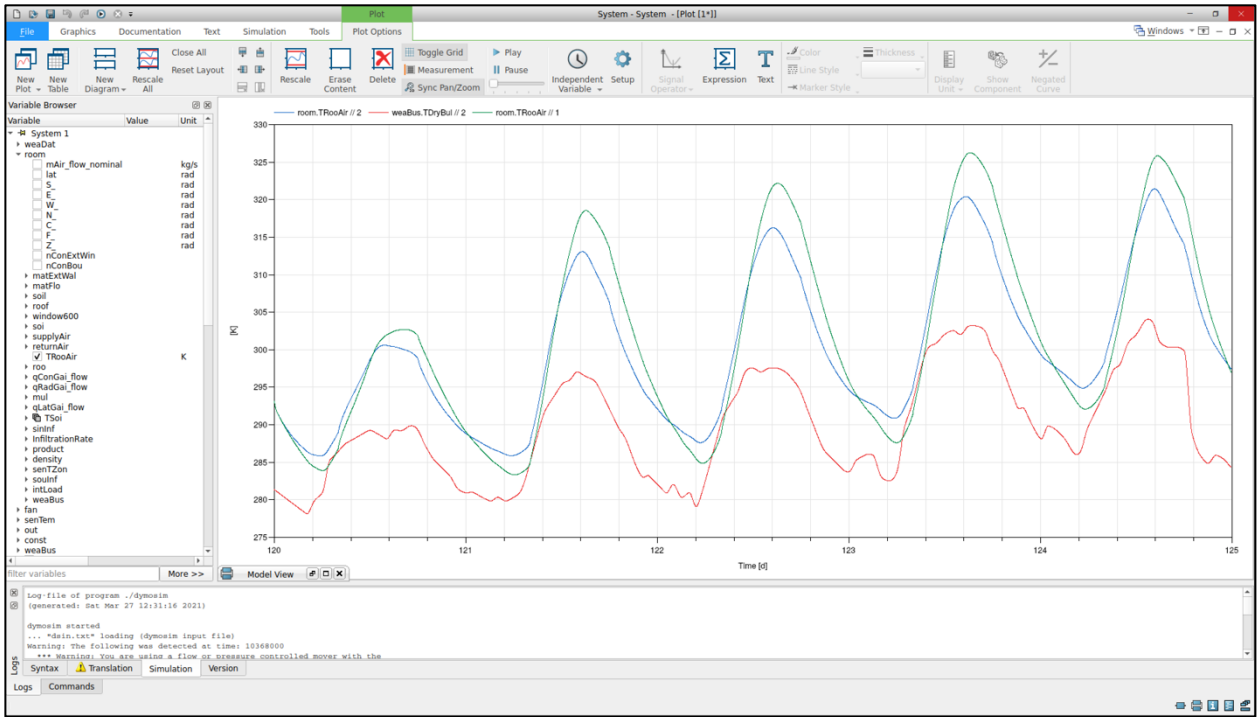
3. Simulate the model from day 120 to 125 with a 60 second output interval using the Dassl solver with a tolerance of 1e-6. Plot the outside air temperature and inside air temperature as shown in the plot below.



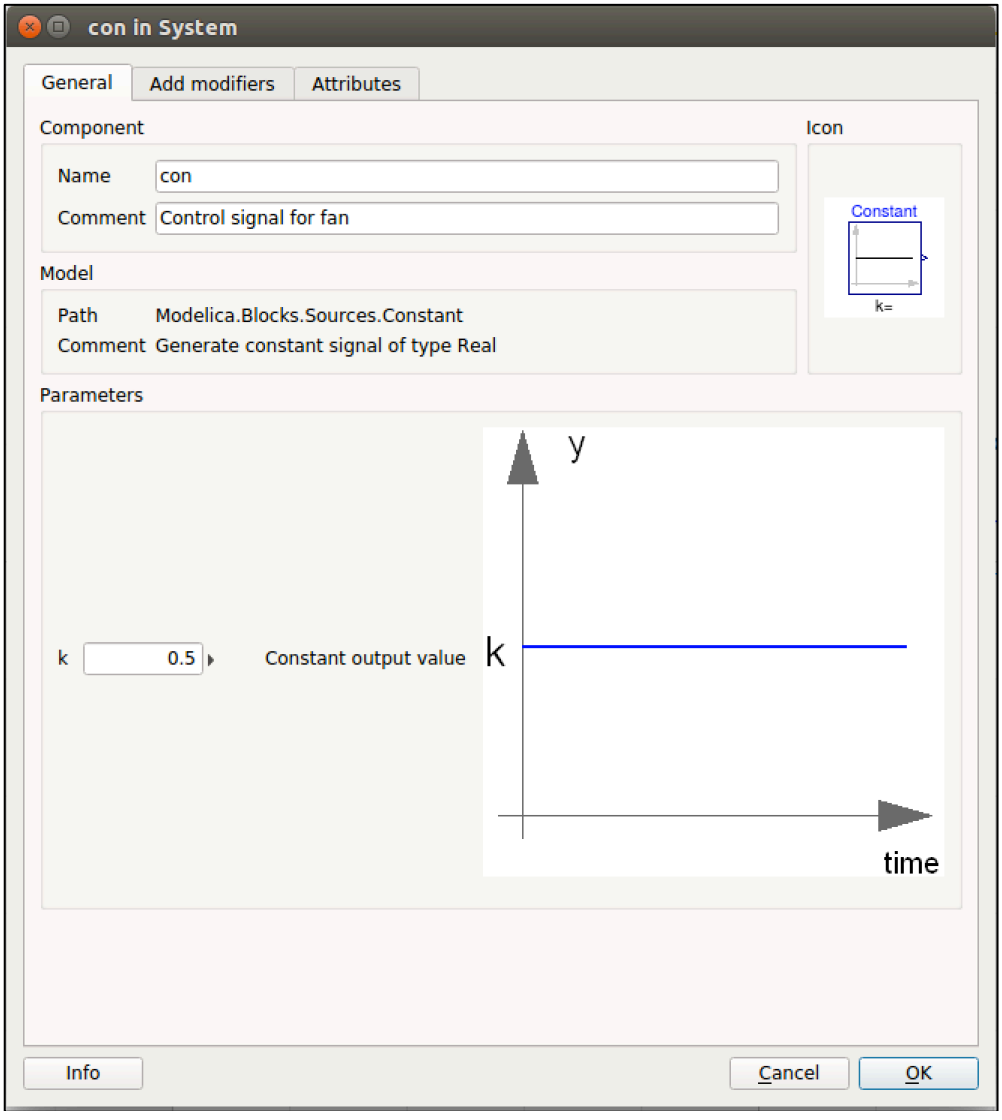
4. Increase the thermal mass in the room by increasing the concrete floor slab thickness from 1” to 6”. Do this by opening the “room” configuration (double-click on the room model). Then, click on the grey box to the right of the parameter called “matFlo”, which sets the material layers for the floor construction. Then, click on the small black arrow next to the “material” parameter and choose Edit Text. Then, change the line that reads “x=0.025” to “x=0.025*6”. This is the concrete slab floor thickness in meters. Click OK on all of the open windows. Now, simulate the model again. The results in the plot will update. Compare with the previous simulation by expanding the previous result in the Variable Browser and plotting the room air temperature variable.

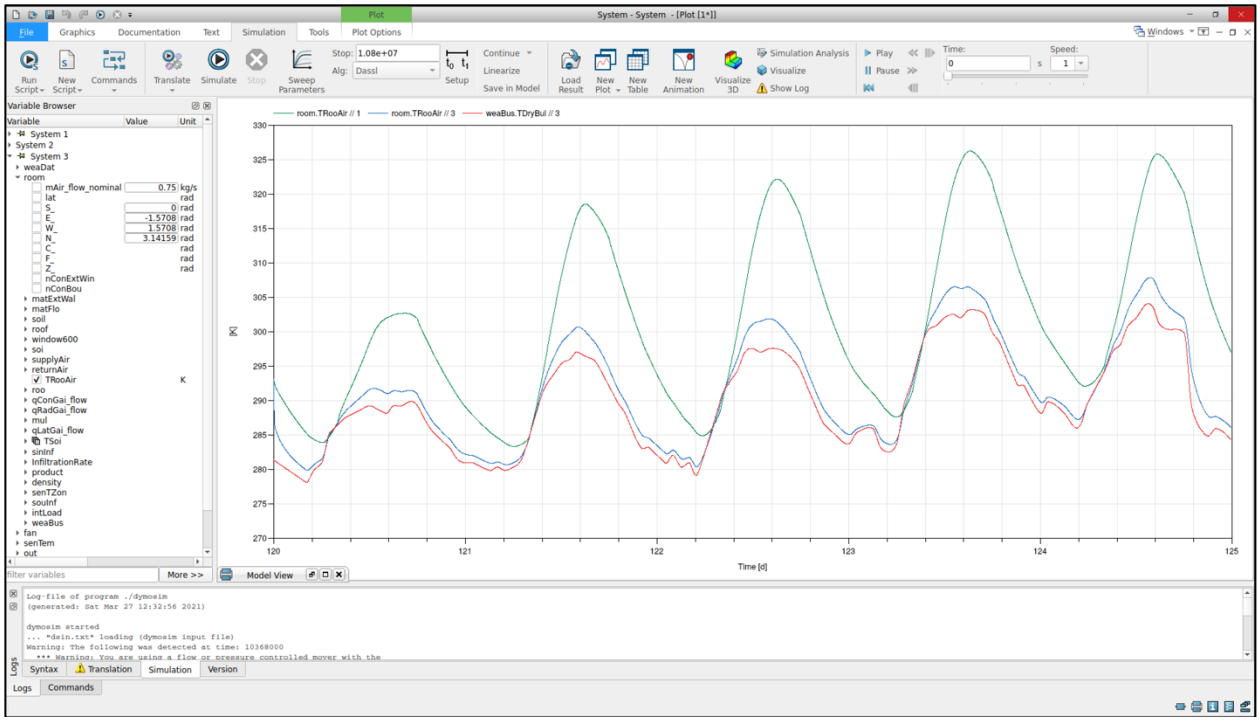
What do you notice about the air temperature with higher thermal mass compared to the temperature with lower thermal mass?




System - System - [Plot [1*]]
File
Graphics
Documentation
Text
Simulation
Tools
Plot
Plot Options
Windows
New Plot
New Table
New Diagram
Rescale All
Close All
Reset Layout
Rescale
Erase Content
Delete
Toggle Grid
Measurement
Sync Pan/Zoom
Play
Pause
Independent Variable
Setup
Signal Operator
Expression
Text
Color
Line Style
Marker Style
Thickness
Display Unit
Show Component
Negated Curve
Variable Browser
Variable
Value
Unit
System 1
weaDat
room
mAir_flow_nominal kg/s
lat rad
S_ rad
E_ rad
W_ rad
N_ rad
C_ rad
F_ rad
Z_ rad
nConExtWin
nConBou
matExtWal
matFlo
soil
roof
window600
soi
supplyAir
returnAir
TRooAir K
roo
qConGai_flow
qRadGai_flow
mul
qLatGai_flow
TSoi
sinInf
InfiltrationRate
product
density
senTZon
souInf
intLoad
weaBus
fan
senTem
out
const
weaBus
filter variables
More >>
room.TRooAir // 2
weaBus.TDryBul // 2
room.TRooAir // 1
330
325
320
315
310
305
300
295
290
285
280
275
[K]
120
121
122
123
124
125
Time [d]
Model View
Log-file of program ./dymosim
(generated: Sat Mar 27 12:31:16 2021)
dymosim started
... "dsin.txt" loading (dymosim input file)
Warning: The following was detected at time: 10368000
*** Warning: You are using a flow or pressure controlled mover with the
Logs
Syntax
Translation
Simulation
Version
Logs
Commands

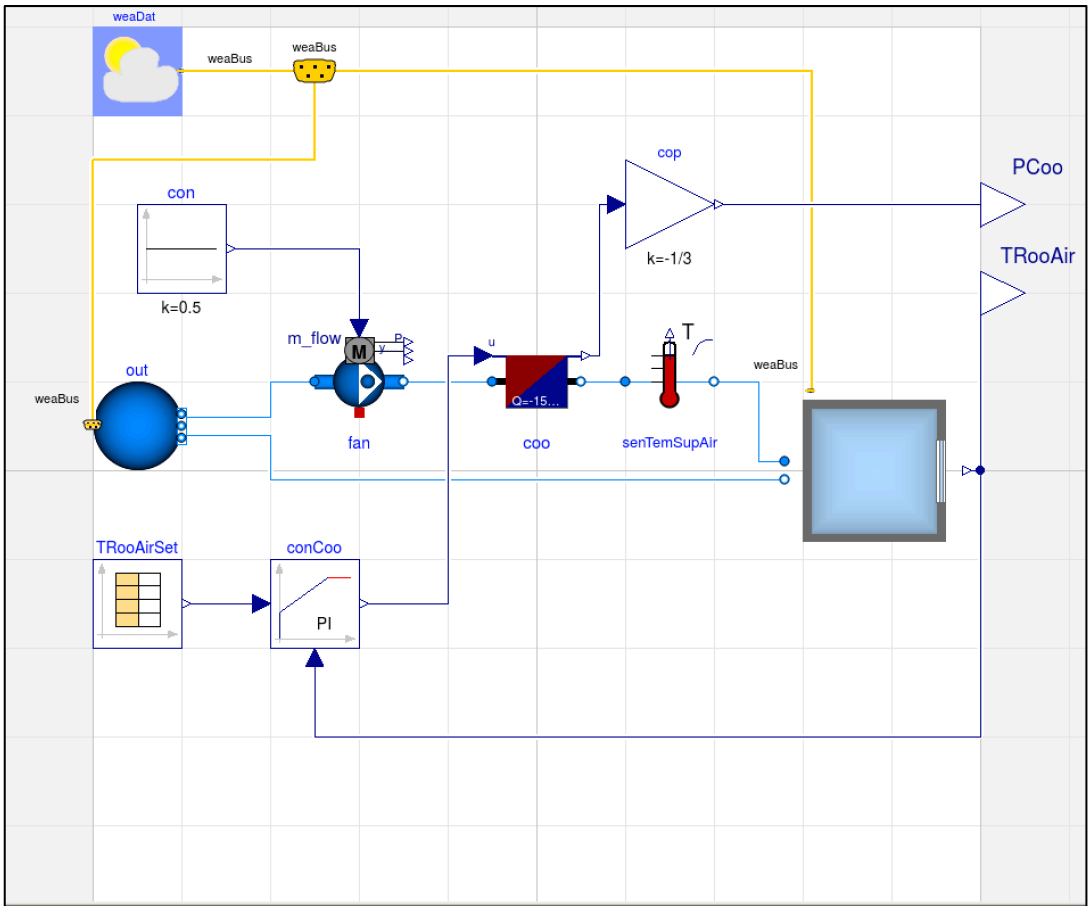
5. Turn on the fan by changing the changing the “k” parameter of the model “con” to 0.5. This will specify that the fan constantly blows 0.5 kg/s of air into the room. Since it looks like our outside air temperature is always less than our inside air temperature, this will help us cool down the room. Simulate the model and look at the updated results.

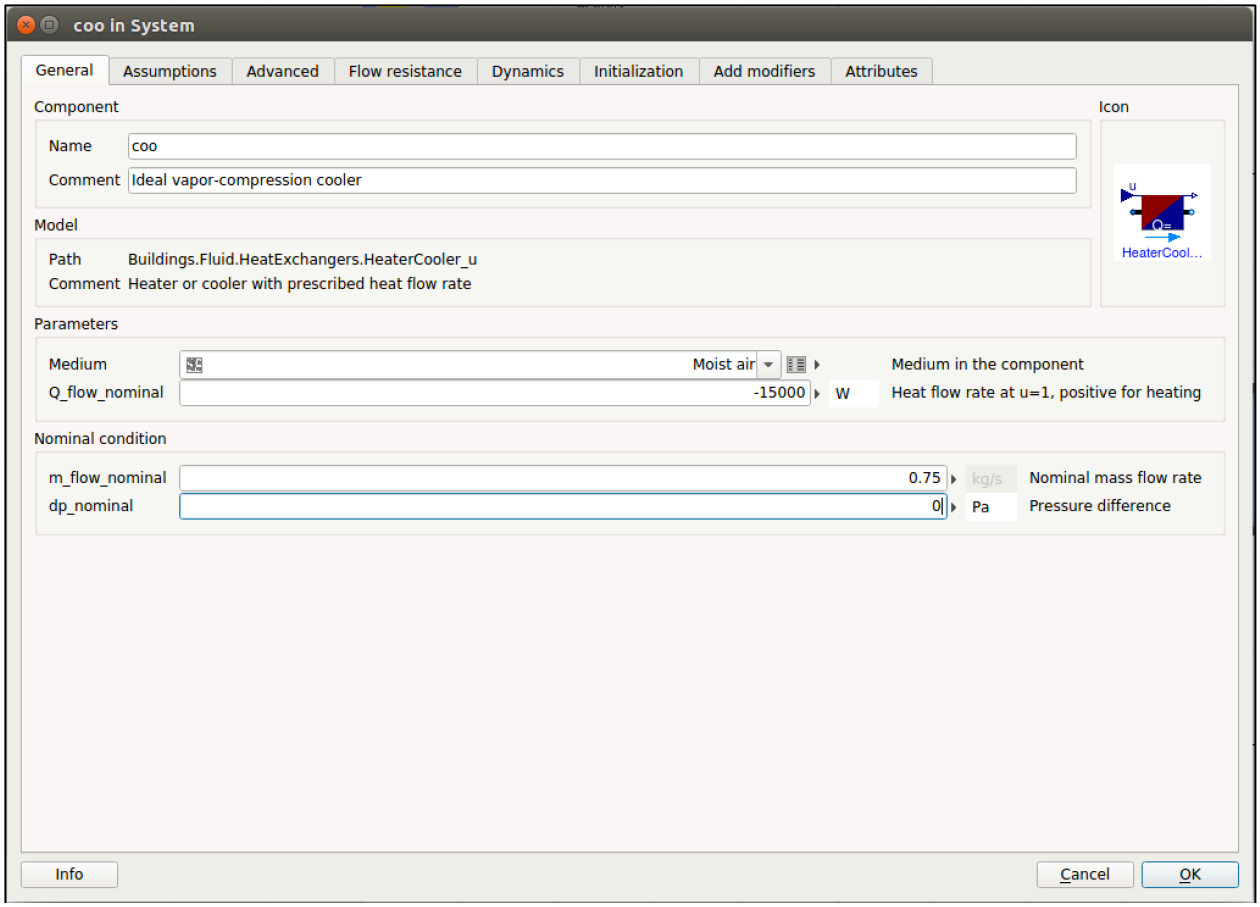


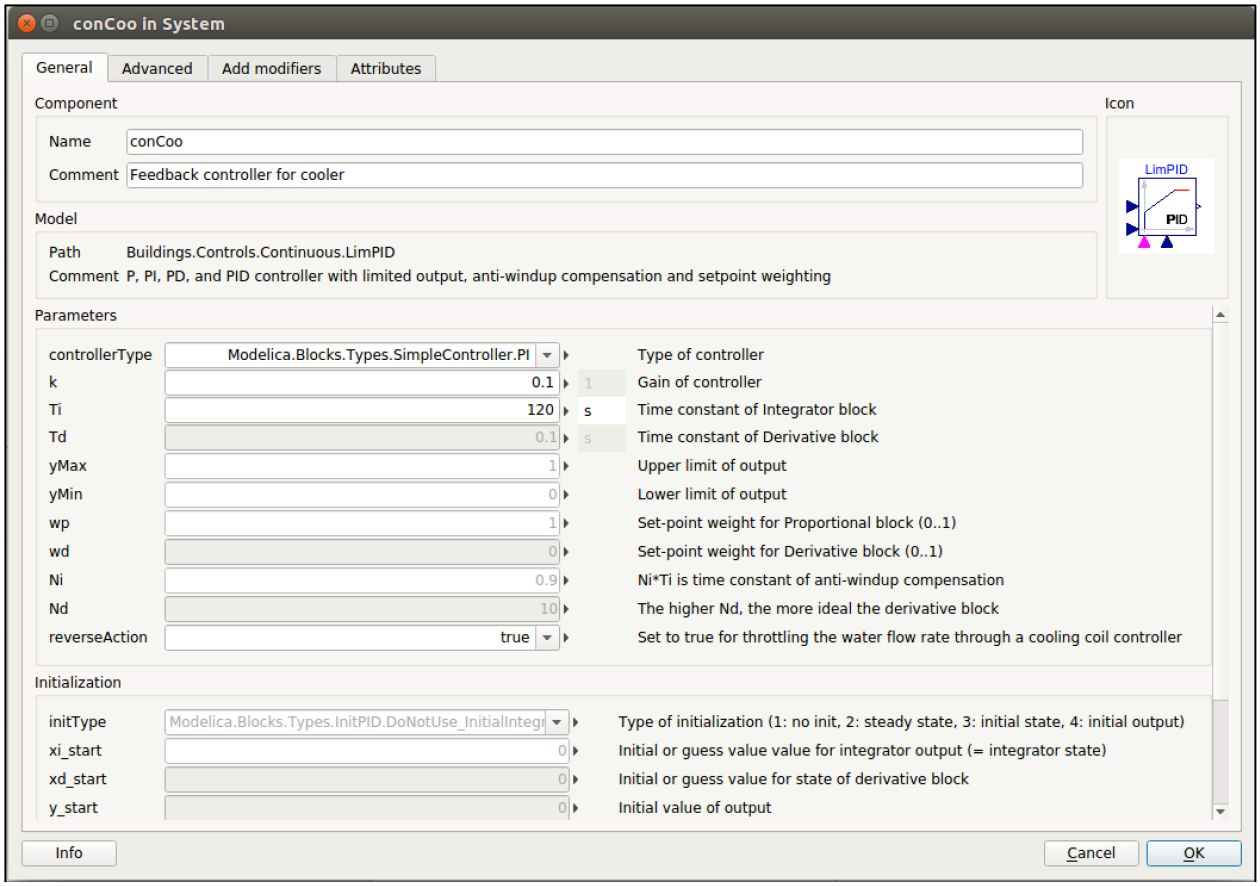

System - System - [Plot [1*]]
File
Graphics
Documentation
Text
Simulation
Tools
Plot Options
Run Script
New Script
Commands
Translate
Simulate
Stop
Sweep Parameters
Stop: 1.08e+07
Alg: Dassl
Setup
Continue
Linearize
Save in Model
Load Result
New Plot
New Table
New Animation
Visualize 3D
Simulation Analysis
Visualize
Show Log
Play
Pause
Time:
Speed:
Variable Browser
Variable
Value
Unit
System 1
System 2
System 3
weaDat
room
mAir_flow_nominal 0.75 kg/s
lat rad
S_ 0 rad
E_ -1.5708 rad
W_ 1.5708 rad
N_ 3.14159 rad
C_ rad
F_ rad
Z_ rad
nConExtWin
nConBou
matExtWal
matFlo
soil
roof
window600
soi
supplyAir
returnAir
TRooAir K
roo
qConGai_flow
qRadGai_flow
mul
qLatGai_flow
TSoi
sinInf
InfiltrationRate
product
density
senTZon
souInf
intLoad
weaBus
fan
senTem
out
Filter variables
More >>
room.TRooAir // 1
room.TRooAir // 3
weaBus.TDryBul // 3
330
325
320
315
310
305
300
295
290
285
280
275
270
K
120
121
122
123
124
125
Time [d]
Model View
Log-file of program ./dymosim
(generated: Sat Mar 27 12:32:56 2021)
dymosim started
... "dsin.txt" loading (dymosim input file)
Warning: The following was detected at time: 10368000
*** Warning: You are using a flow or pressure controlled mover with the
Logs
Syntax
Translation
Simulation
Version
Logs
Commands

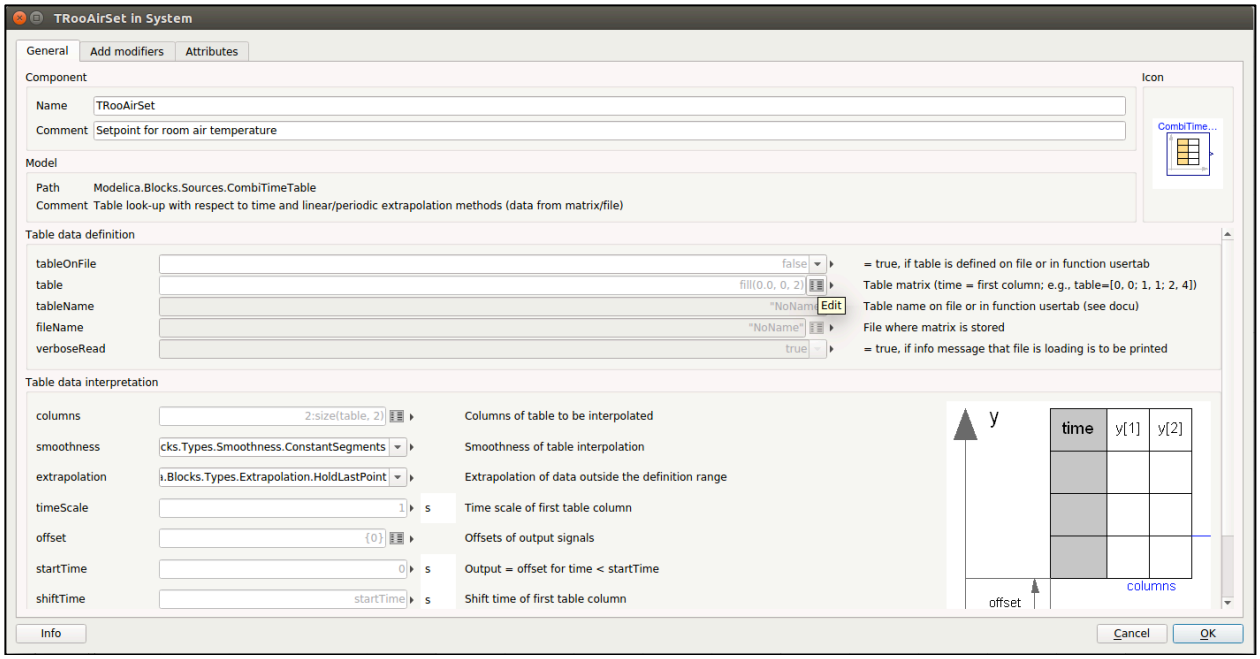
6. The peak inside air temperature still reaches over 305 K, which is about 32 C and 89 F! Add a simple vapor-compression cooling model and controls by adding the following component models as shown in the diagram and screenshots below.

Model Path	Description/Usage
Buildings.Fluid.HeatExchangers.HeaterCooler_u	An ideal heater or cooler with a control input signal that specifies the fraction of maximum heating or cooling power to apply to the passing fluid.
Buildings.Controls.Continuous.LimPID	A PID feedback controller that is used to control the cooling power based on room air temperature setpoint and room air temperature measurement.
Modelica.Blocks.Sources.CombiTimeTable	The output signal is specified using a table of values that can be defined at particular simulation times. We will use this to specify the room air temperature setpoint. When connecting to the component "conCoo", use index [1] in the resulting prompt.
Modelica.Blocks.Math.Gain	Multiplies the input signal by a constant value and outputs the result. We will use this to convert the thermal power calculated by the cooler model to electrical power by a constant assumed COP.
Modelica.Blocks.Interfaces.RealOutput	Allows the input signal to serve as an output for the whole model. This makes it easier to view results, and also allows the model to be connected to other models.

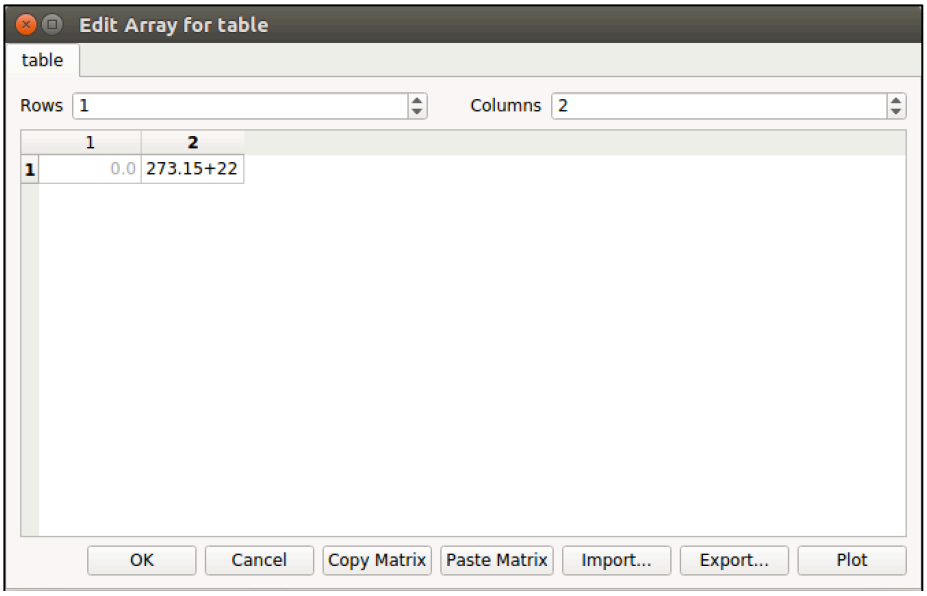



coo in System
General
Assumptions
Advanced
Flow resistance
Dynamics
Initialization
Add modifiers
Attributes
Component
Name
coo
Comment
Ideal vapor-compression cooler
Icon
HeaterCool...
Model
Path
Buildings.Fluid.HeatExchangers.HeaterCooler_u
Comment
Heater or cooler with prescribed heat flow rate
Parameters
Medium
Moist air
Medium in the component
Q_flow_nominal
-15000
W
Heat flow rate at u=1, positive for heating
Nominal condition
m_flow_nominal
0.75
kg/s
Nominal mass flow rate
dp_nominal
0
Pa
Pressure difference
Info
Cancel
OK

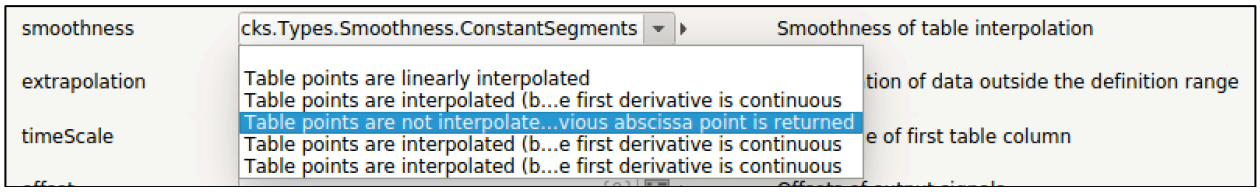

conCoo in System
General
Advanced
Add modifiers
Attributes
Component
Name
conCoo
Comment
Feedback controller for cooler
Icon
LimPID
PID
Model
Path
Buildings.Controls.Continuous.LimPID
Comment
P, PI, PD, and PID controller with limited output, anti-windup compensation and setpoint weighting
Parameters
controllerType
Modelica.Blocks.Types.SimpleController.PI
Type of controller
k
0.1
1
Gain of controller
Ti
120
s
Time constant of Integrator block
Td
0.1
s
Time constant of Derivative block
yMax
1
Upper limit of output
yMin
0
Lower limit of output
wp
1
Set-point weight for Proportional block (0..1)
wd
0
Set-point weight for Derivative block (0..1)
Ni
0.9
Ni*Ti is time constant of anti-windup compensation
Nd
10
The higher Nd, the more ideal the derivative block
reverseAction
true
Set to true for throttling the water flow rate through a cooling coil controller
Initialization
initType
Modelica.Blocks.Types.InitPID.DoNotUse_InitialIntegr
Type of initialization (1: no init, 2: steady state, 3: initial state, 4: initial output)
xi_start
0
Initial or guess value value for integrator output (= integrator state)
xd_start
0
Initial or guess value for state of derivative block
y_start
0
Initial value of output
Info
Cancel
OK

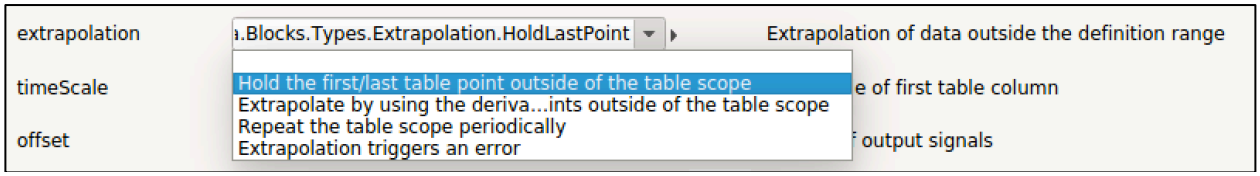


Click on the grey box to the right of the parameter "table" to edit the table. We will just define the output value (22 C) at simulation time 0, and let it be constant for the entire simulation.



Two additional parameters will specify the output to be constant segments between time intervals (instead of interpolating), and to hold the last value specified for the rest of the simulation time (instead of another extrapolation scheme).





cop in System

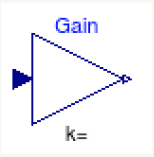
General | Add modifiers | Attributes

Component

Name: cop

Comment: Implements COP of cooling system

Icon



Model

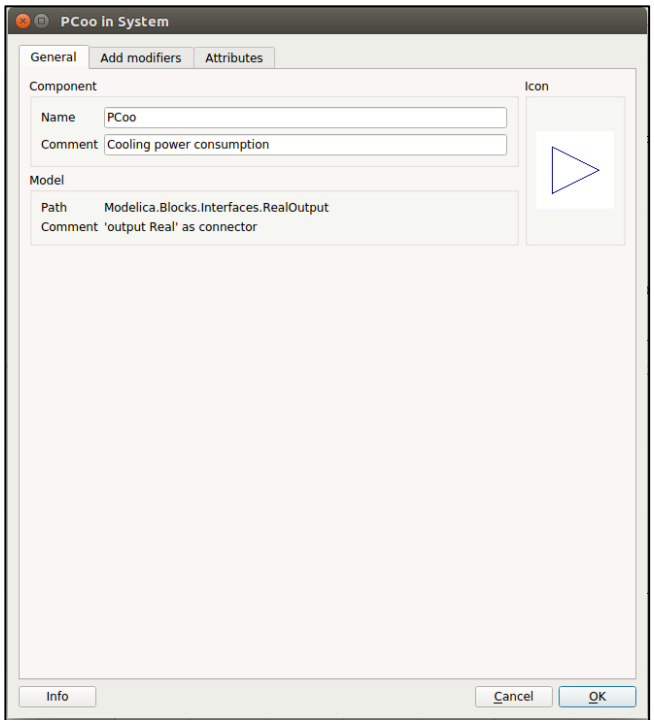
Path: Modelica.Blocks.Math.Gain

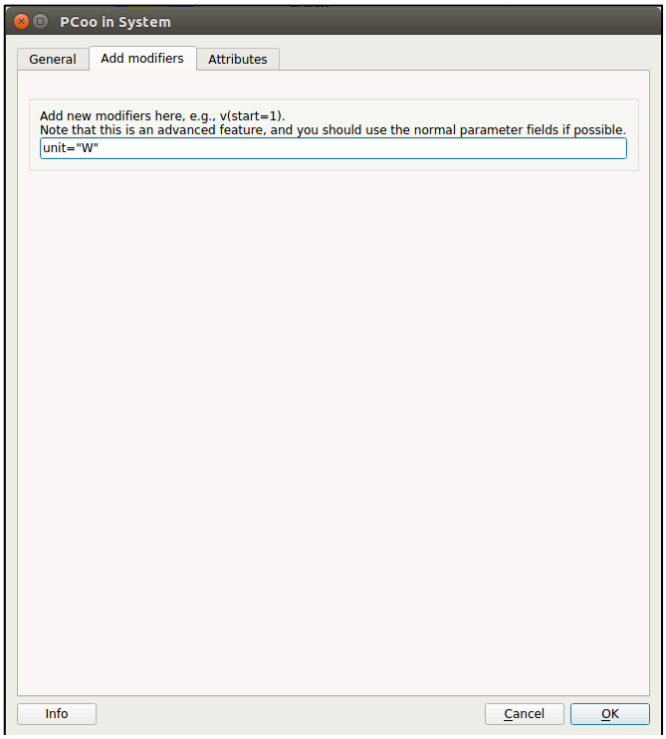
Comment: Output the product of a gain value with the input signal

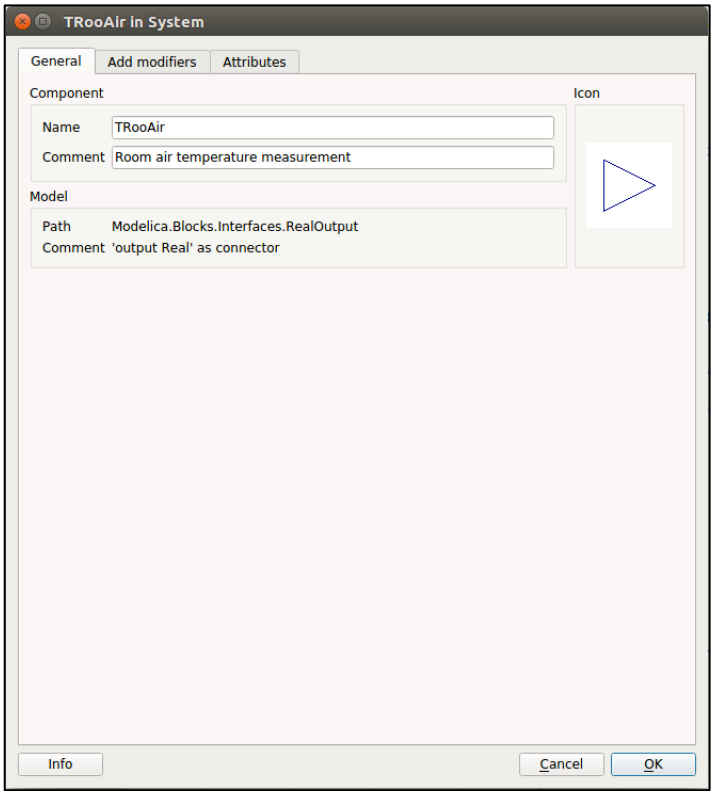
Parameters

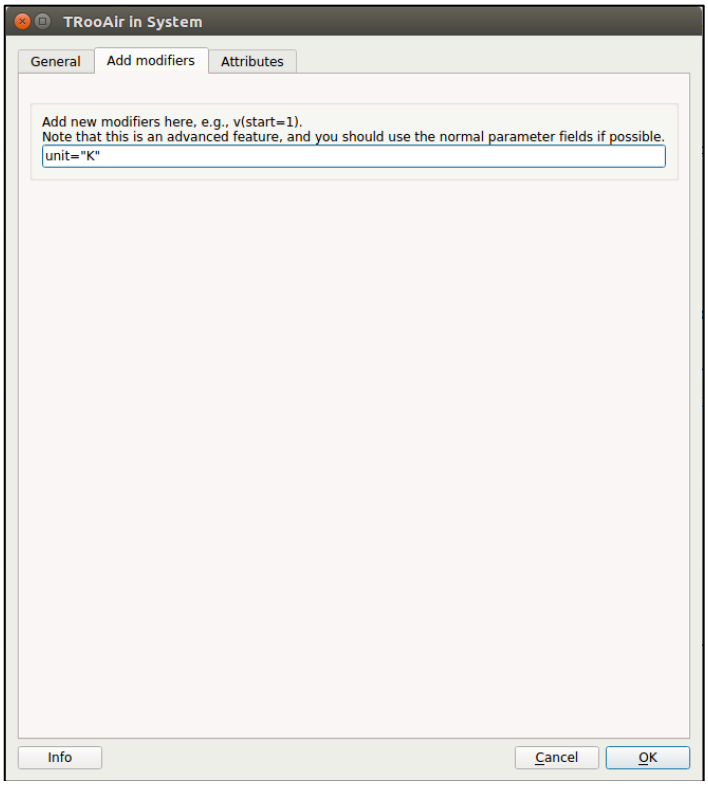
k	-1/3	1	Gain value multiplied with input signal

Info | Cancel | OK

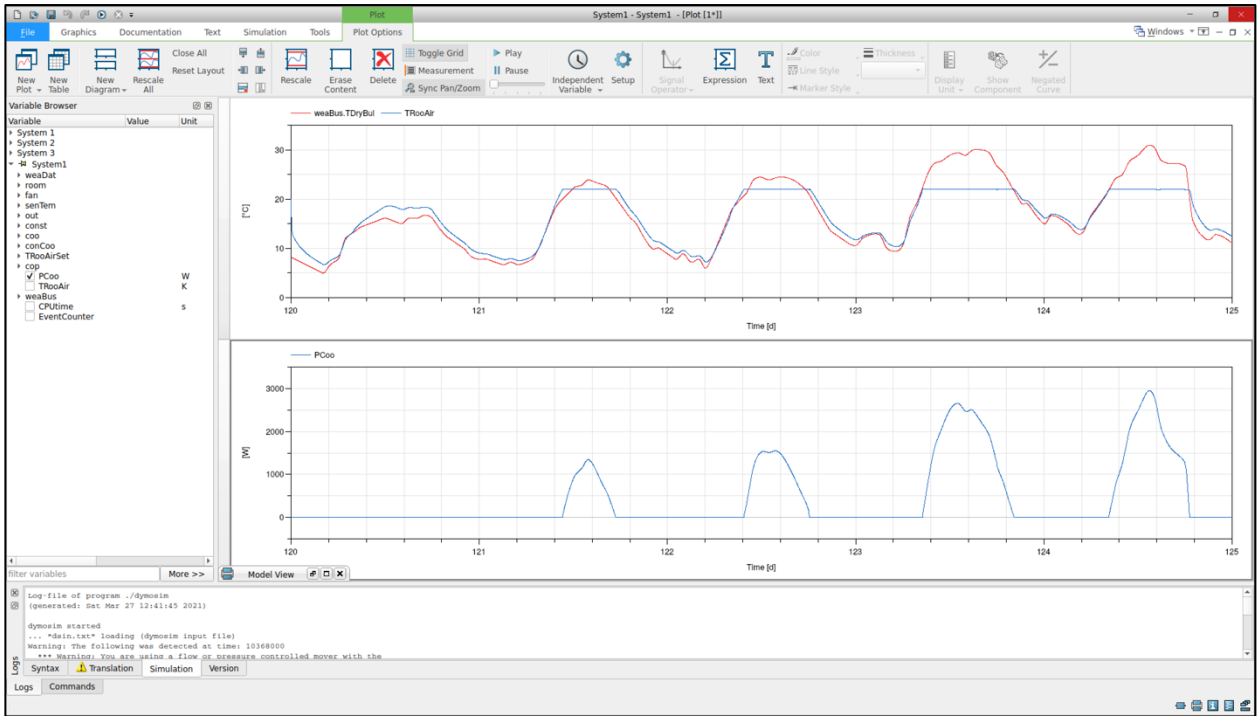

PCoo in System
General
Add modifiers
Attributes
Component
Name
PCoo
Comment
Cooling power consumption
Icon
Model
Path
Modelica.Blocks.Interfaces.RealOutput
Comment
'output Real' as connector
Info
Cancel
OK

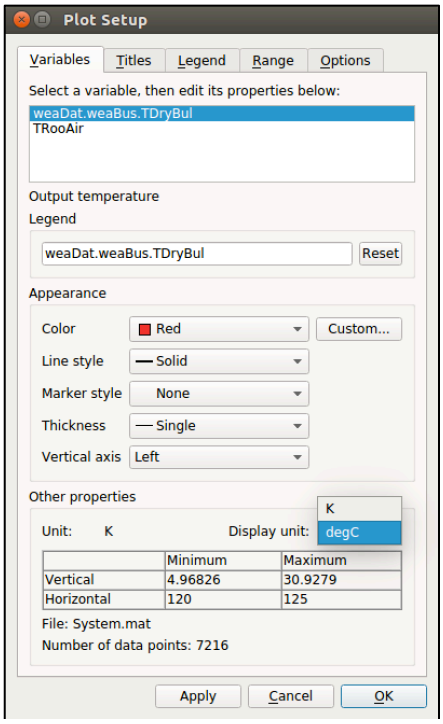

PCoo in System
General
Add modifiers
Attributes
Add new modifiers here, e.g., v(start=1).
Note that this is an advanced feature, and you should use the normal parameter fields if possible.
unit="W"
Info
Cancel
OK


TRooAir in System
General
Add modifiers
Attributes
Component
Name
TRooAir
Comment
Room air temperature measurement
Icon
Model
Path
Modelica.Blocks.Interfaces.RealOutput
Comment
'output Real' as connector
Info
Cancel
OK

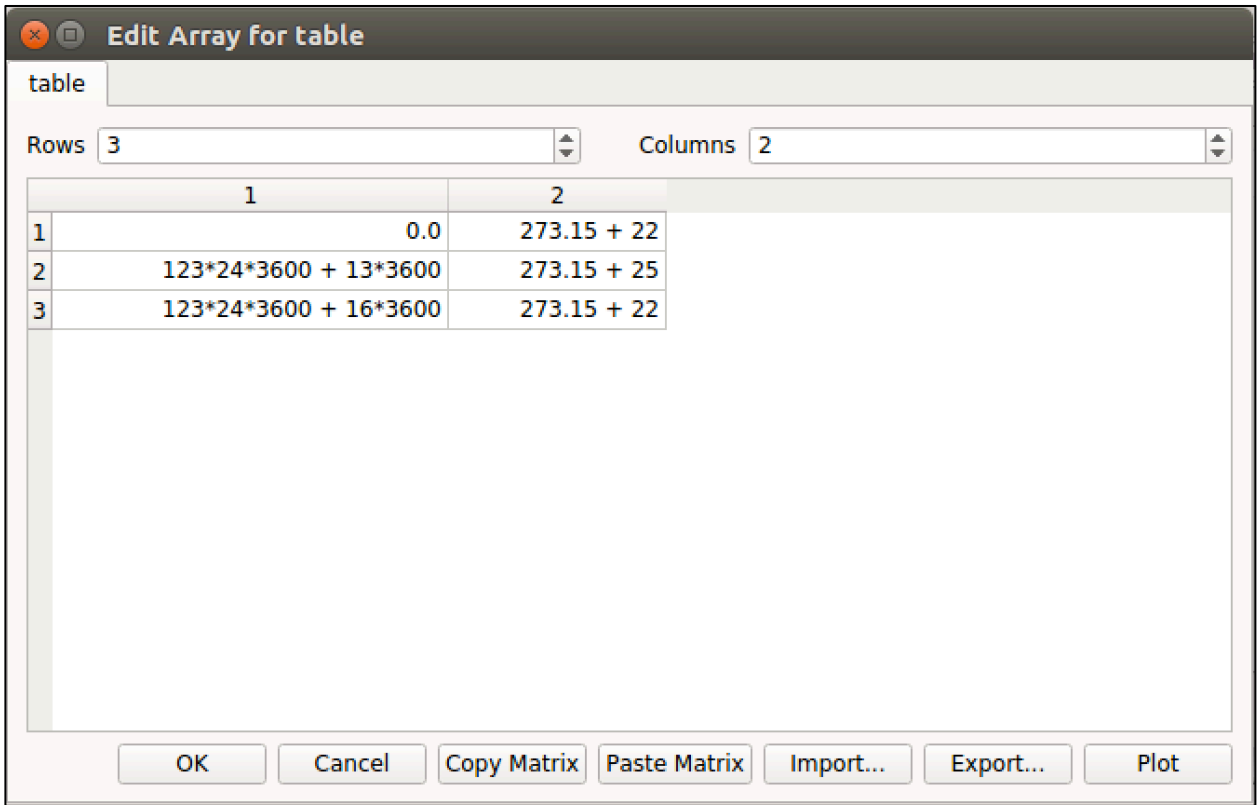

TRooAir in System
General
Add modifiers
Attributes
Add new modifiers here, e.g., v(start=1).
Note that this is an advanced feature, and you should use the normal parameter fields if possible.
unit="K"
Info
Cancel
OK

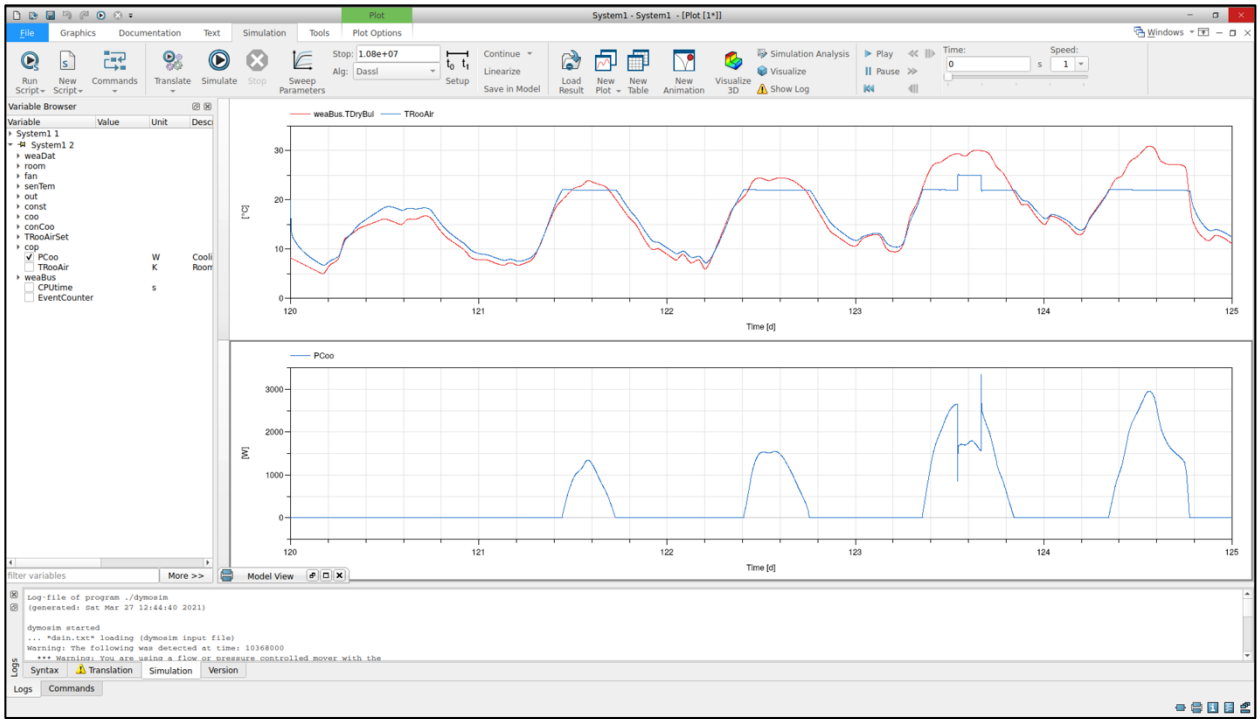
7. Simulate the model and plot the room air temperature, outside air temperature, and cooling power. Use the "New Diagram" command icon to split the plot window between temperature and power measurements. Then, open the "Setup" window, and set the display unit to deg C.



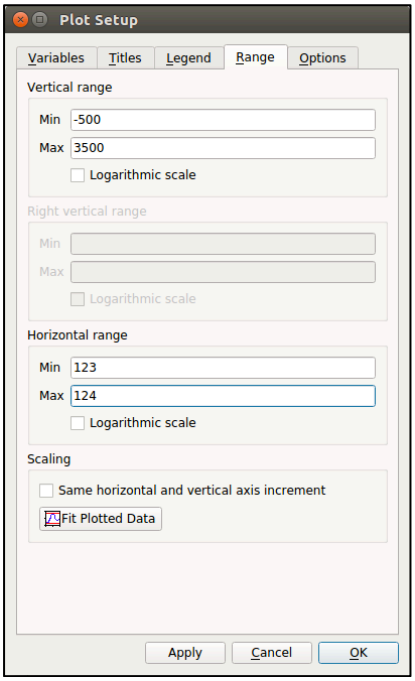


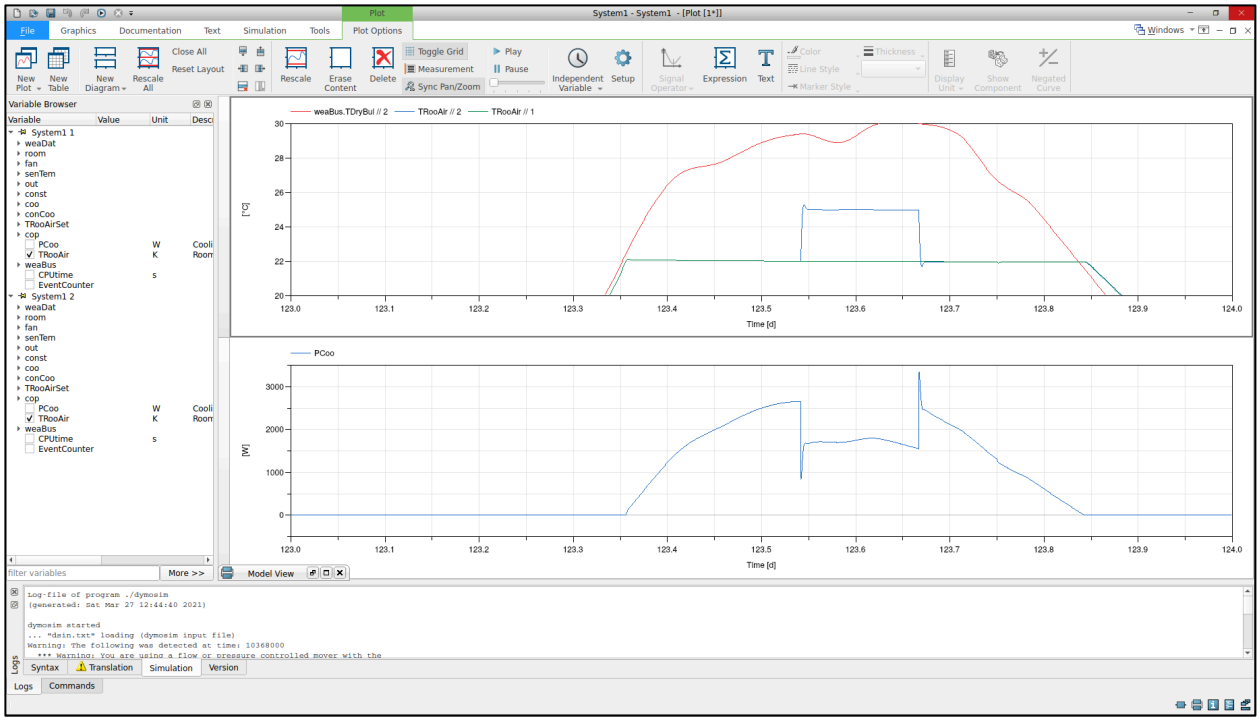
8. Let's implement a demand-response event on the 4th day between 1pm and 4pm. To respond, we want to increase the zone temperature setpoint by 3 C. Edit the setpoint table to achieve this. Then, simulate the model again. Compare the power consumption profile to that without the demand response event. What do you notice?





Zoom in on the demand response day by using plot Setup and modifying the horizontal range. You must do this for both subplots. You can also modify the vertical range for the temperature plot, compare the demand response room air temperature measurement to the previous, and further edit the plot properties as you wish.




System1 - System1 - [Plot [1*]]
Plot
File
Graphics
Documentation
Text
Simulation
Tools
Plot Options
Windows
New Plot
New Table
New Diagram
Rescale All
Close All
Reset Layout
Rescale
Erase Content
Delete
Toggle Grid
Measurement
Sync Pan/Zoom
Play
Pause
Independent Variable
Setup
Signal Operator
Expression
Text
Color
Line Style
Marker Style
Thickness
Display Unit
Show Component
Negated Curve
Variable Browser
Variable
Value
Unit
Desc
System1 1
weaDat
room
fan
senTem
out
const
coo
conCoo
TRooAirSet
cop
PCoo
TRooAir
weaBus
CPUtime
EventCounter
System1 2
filter variables
More >>
weaBus.TDryBul // 2
TRooAir // 2
TRooAir // 1
[°C]
Time [d]
PCoo
[W]
Model View
Log-file of program ./dymosim
(generated: Sat Mar 27 12:44:40 2021)
dymosim started
... "dsin.txt" loading (dymosim input file)
Warning: The following was detected at time: 10368000
*** Warning: You are using a flow or pressure controlled mover with the
Logs
Syntax
Translation
Simulation
Version
Logs
Commands