WORKSHOP 4.1.2

Using component arrays to discretize models

Contents

Introduction

In this workshop we will look at how we can utilize arrays of components to create segmented models. Modelica is primarily a language for describing ODEs and DAEs and it cannot be used to describe partial differential equations, i.e., equation systems that include gradients or other derivatives with respect to more variables than just time. Segmentation can be a way of representing these types of systems and adding more elements to the segmentation will bring us closer to a continuous solution in the end.

To do this in this workshop we will consider the heat transfer through a cylinder of some material. The system consists of lumped heat capacitances that will transfer heat between themselves via conduction and to the environment through convection. The system can be implemented with components for thermal transfer from MSL. The system could be implemented like below:

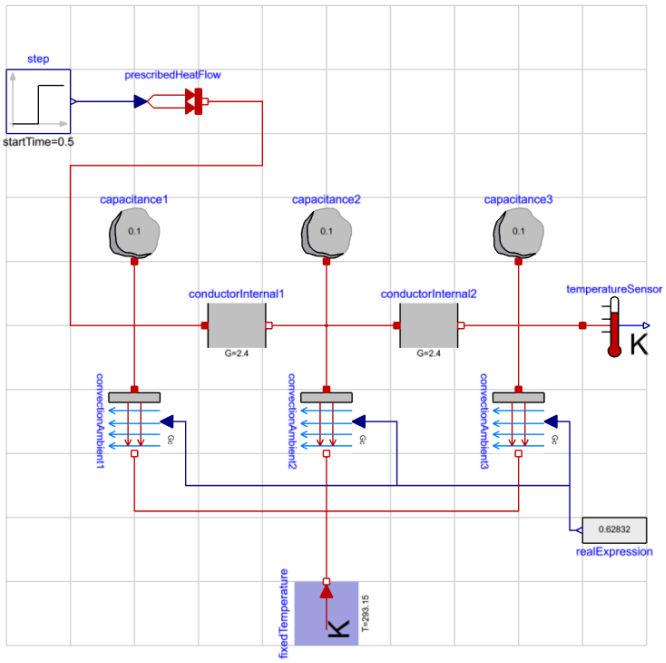

Figure 1 Thermal conductance model outline

With this implementation, the system segmentation consists of three individual segments including a heat capacitance along with thermal conductors and convections that describes the heat flow to the

environment and to the other segments in the bar. Although this representation is perfectly legitimate for this specific segmentation, it would be nice if we could alter the segmentation easily in some way. It is here array of components come in handy.

Creating a segmented component

Start by locating the prepared training material in the training package.

The task is to replace the capacitances, internal heat conductors and ambient heat convections with array of components and connect them together using the code editor. A prepared component has been created with instantiations of the needed components. The task now is to complement the model with code to make it a proper segmented component.

1. Open the prepared model *W1_ComponentArrays.DistributedHeatTransfer*.
2. Start by adding an Integer parameter ***N*** for the number of elements in the segmented component. This can both be done through the properties tab with the + button or by declaring it directly in the code layer. Give it a default value of 3.
3. To convert the components into arrays, we need to access the text layer of the model. This can be accessed by right clicking the model canvas -> edit source.
4. Now we can convert the components into arrays by adding the array notation to the 3 instantiated components: *capacitance*, *conductorInternal*, *convectionAmbient*. This is done by adding a suffix ***[N]*** to the component name.

   HINT: ***N*** denotes the number of elements. How many of the different components should there be of each kind? Look at Figure 1 for clues.
5. We also need to propagate the declared system parameters to the different components. Add the following modifiers to the components:

Component	Parameter	Value
capacitance	C	C/N
capacitance	T	start=T0, fixed=true
conductorInternal	G	G_internal*(N-1)

Table 1 Component parametrization

   HINT: Add the *each* keyword to apply the modifiers to all elements in the arrays at once. Finally make a syntax check and save if it is ok.

Now we have instantiated and parameterized the components, the only thing left is to connect them together. To make this fit dynamically to the segmentation size of the component we will need to use for loops and loop over the component arrays to make all the connections.

6. Now it is time to connect the array components together from the source code editor. The general idea how we want to connect the components was outlined in Figure 1 Thermal conductance model outline. In summary we need to:
   a. Connect the internal conductions to the capacitances.
   b. Connect the heat capacitances at the edges to the heat ports.
   c. Connect the ambient convections to the capacitances.
   d. Connect the ambient conductions to the ambient heat port.
   e. Couple the Gc_ambient parameter to the convections.

You are welcome to try to figure out the syntax yourself, otherwise one correct implementation is outlined below:

```
equation

      // Connect internal conductions
      for i in 1:N-1 loop
        connect(conductorInternal[i].port_a,capacitance[i].port);
        connect(conductorInternal[i].port_b,capacitance[i+1].port);
      end for;

      // Connect ambient convections to capacitances and ambient port and connect thermal conductance input
      for i in 1:N loop
        connect(capacitance[i].port, convectionAmbient[i].solid);
        connect(ambient, convectionAmbient[i].fluid);
        convectionAmbient[i].Gc = Gc_ambient;
      end for;

      // Connect port_a and port_b to first and last capacitance
      connect(port_a, capacitance[1].port);
      connect(port_b, capacitance[N].port);
```

Figure 2 Connect statements

Make a syntax check and save if it is ok.

Now the component should be finalized and ready to be used as a component in a system model. Below the source code of the finalized model is outlined (make the additional changes in your model):

```
model DistributedHeatTransfer
  "Segmented heat transfer model of cylinder of some material"
      // Declare connector interface of the component
      .Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a port_a "Heat port"
          annotation (Placement(transformation(extent={{-108,-10},{-88,10}})));
      .Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_b port_b "Heat port"
          annotation (Placement(transformation(extent={{90,-10},{110,10}})));
      .Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a ambient "Heat port to ambient"
          annotation (Placement(transformation(extent={{-10,-108},{10,-88}})));

      // Declare propagated model parameters
      // Segmentation:
      parameter Integer N(min=2)=3 "Number of segments, minimum of 2";
      // Geometry:
      parameter Modelica.SIunits.Length L = 1 "Length of cylinder";
      parameter Modelica.SIunits.Length d = 0.05 "Diameter of cylinder";
      final parameter Modelica.SIunits.Area A_n = d*Modelica.Constants.pi*L/N "Outer area of cylinder element (excluding edges)";
      // Thermal properties:
      parameter .Modelica.SIunits.HeatCapacity C=0.3 "Heat capacity of material";
      parameter Modelica.SIunits.CoefficientOfHeatTransfer h = 12 "Heat transfer coefficient between ambient and material";
      parameter .Modelica.SIunits.ThermalConductance G_internal=1.2 "Constant internal thermal conductance of material";
      final parameter .Modelica.SIunits.ThermalConductance Gc_ambient=A_n*h "Constant thermal conductance of one element to ambient";
      // Initialization:
      parameter .Modelica.SIunits.Temperature T0=293.15 "Start temperature of material";

      // Declare array components
      .Modelica.Thermal.HeatTransfer.Components.HeatCapacitor capacitance[N](each C=C/N, each T(fixed=true, start=T0))
          annotation (Placement(transformation(extent={{-10,32},{10,52}})));
      .Modelica.Thermal.HeatTransfer.Components.ThermalConductor conductorInternal[N-1](each G=G_internal*(N-1))
          annotation (Placement(transformation(extent={{-10,-10},{10,10}})));
      .Modelica.Thermal.HeatTransfer.Components.Convection convectionAmbient[N]
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=270,
        origin={0,-40})));
equation

      // Connect internal conductions
      for i in 1:N-1 loop
        connect(conductorInternal[i].port_a,capacitance[i].port);
        connect(conductorInternal[i].port_b,capacitance[i+1].port);
      end for;

      // Connect ambient convections to capacitances and ambient port and connect thermal conductance input
      for i in 1:N loop
        connect(capacitance[i].port, convectionAmbient[i].solid);
        connect(ambient, convectionAmbient[i].fluid);
        convectionAmbient[i].Gc = Gc_ambient;
      end for;

      // Connect port_a and port_b to first and last capacitance
      connect(port_a, capacitance[1].port);
      connect(port_b, capacitance[N].port);
```

Figure 3 Source code of finalized model

Assemble the system model

In this last part of the workshop, we will mainly add the boundary conditions needed to use the created component in a simulation.

1. Create a new model in the workshop package.
2. Add an instance of the created component by dragging it on top of the canvas.
3. Add the boundary conditions and a temperature sensor, i.e. one of each of these components:
    a. Modelica.Thermal.HeatTransfer.Sensors.TemperatureSensor
    b. Modelica.Thermal.HeatTransfer.Sources.FixedTemperature
    c. Modelica.Thermal.HeatTransfer.Sources.PrescribedHeatFlow
    d. Modelica.Blocks.Sources.Step
4. Connect them to the segmented component we created previously in the workshop like this:

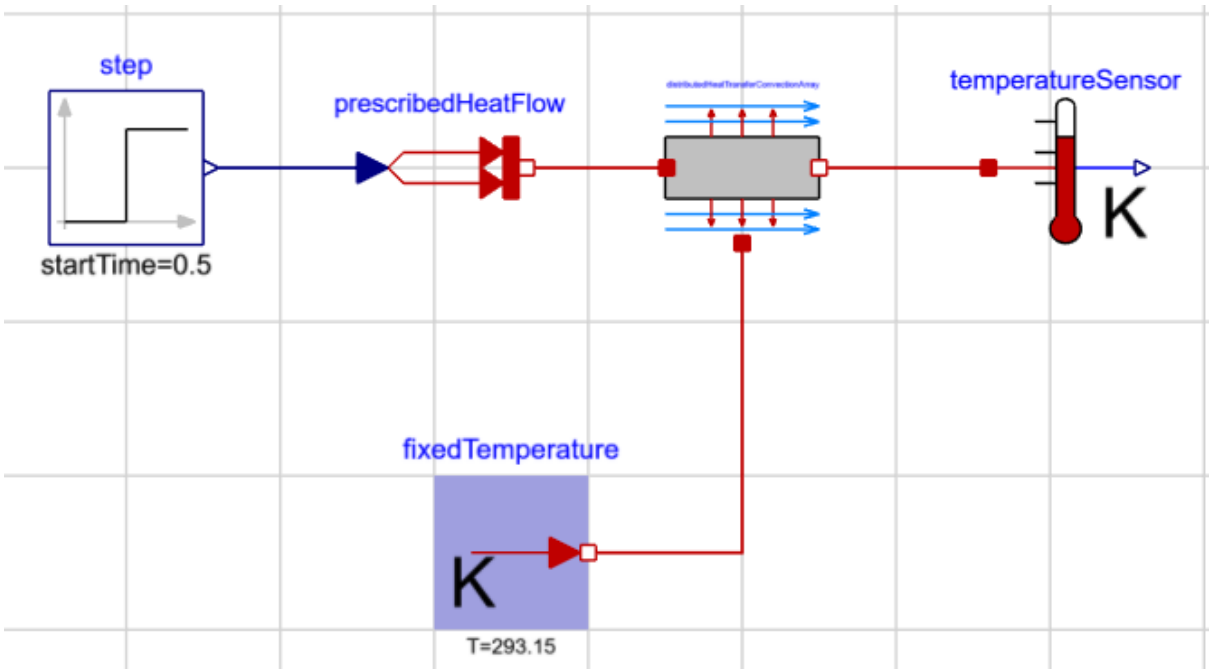

Figure 4 System model outline

5. Parametrize the boundary conditions:

Component	Parameter	Value
step	startTime	0.5
step	height	10
fixedTemperature	T	293.15 K

Table 2 Boundary condition parameterization

Now we are ready to simulate! Simulate the model for 1.5s and inspect the results, the temperature in the sensor should look something like this:

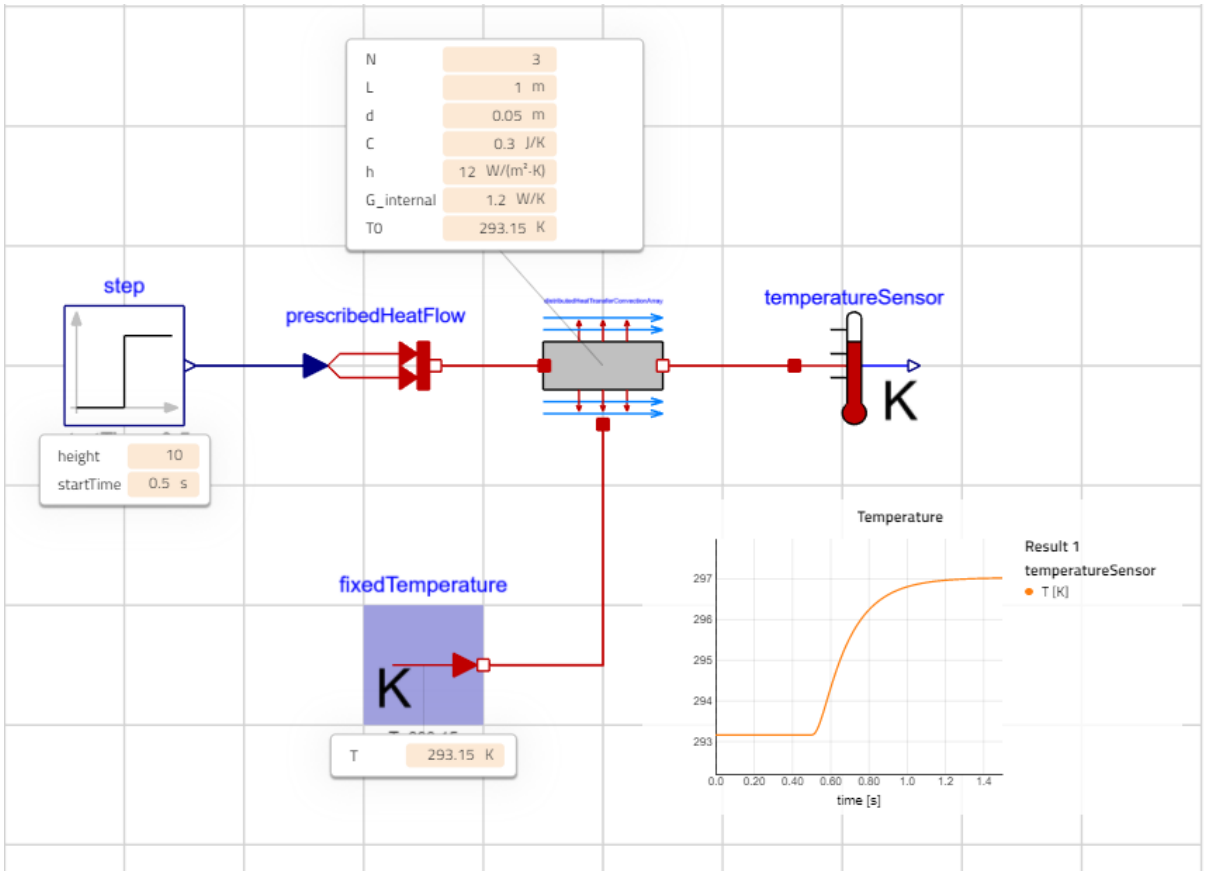

Figure 5 Result

Extra:

- Test what happens as N becomes bigger.
- How does the segmentation size affect the model size? Activate the compiler option "generate_html_diagnostics" and inspect the diagnostics.