MathModelica® System Designer Professional

Electric Circuit

© MathCore Engineering AB

Abstract

This notebook illustrates how a *MathModelica System Designer Professional* model may be developed directly in a *Mathematica* notebook. First a superclass of elements with two electrical pins is modeled. Then an ideal electrical resistor, a capacitor, an inductor and a sine wave voltage source are modeled, inherited the superclass. Before the components are used to connect an entire electrical circuit a ground element is also modeled. Finally the circuit is simulated and several plots from the simulation are presented.

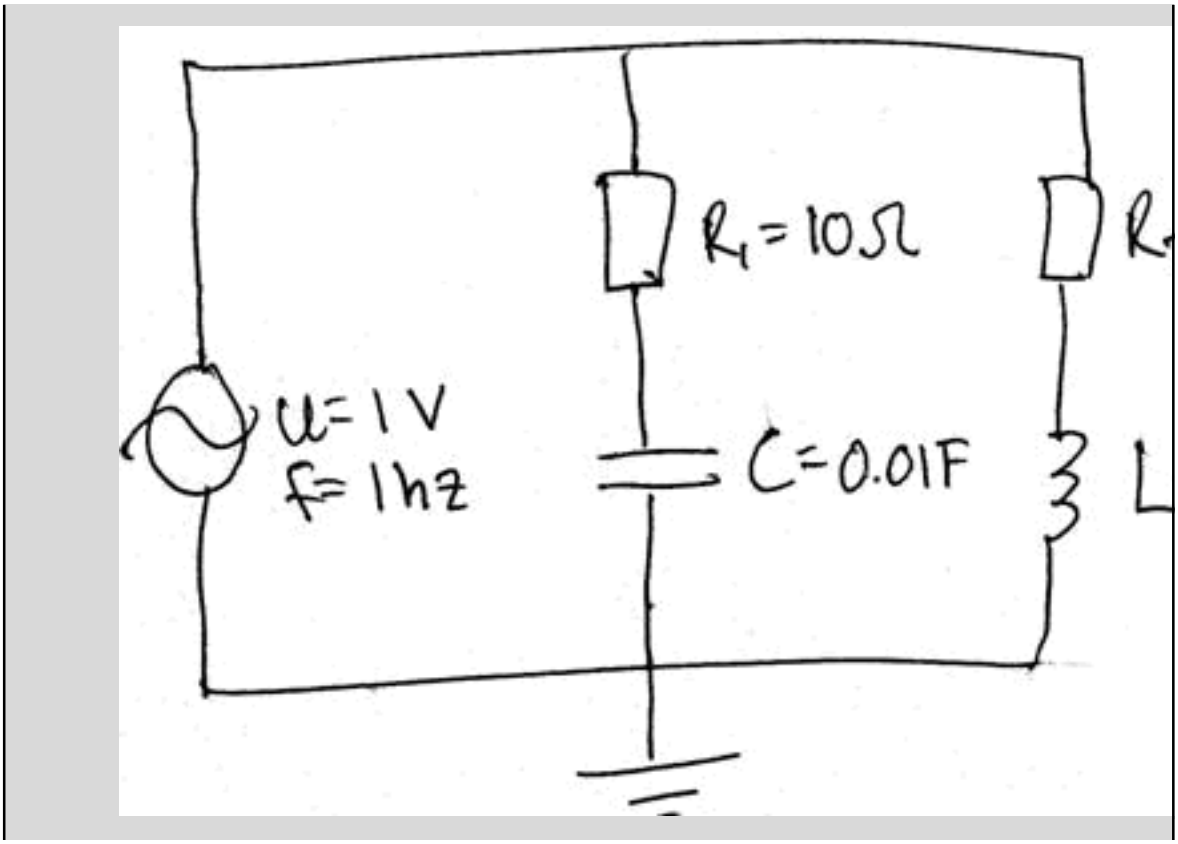
Note that models can also be developed in the model editor using drag and drop. See the Getting Started Examples of the model editor for more information on how to do this.

Initialization

To be able to use *MathModelica System Designer* within *Mathematica* we need to load the *MathModelica* package.

```
Needs["MathModelica`"];
```

3 Electric Circuit



We will develop the above model from scratch to illustrate how Modelica models can be built. The Getting Started document illustrates how this model can be modeled using drag&drop of ready-made components.

3.1 Modeling Electrical Components

First a superclass of elements with two electrical pins is modeled. Then an ideal electrical resistor, a capacitor, an inductor and a sine wave voltage source are modeled, inherited the superclass.

3.1.1 Voltage, Current

The Modelica language allows to define new types by extending already existing types. Here we will define a new type called Voltage, and another called Current. Using these instead of just declaring variables as Reals will make it easier for the user to interpret results.

```
type Voltage = Real(unit="V");
```

```
type Current = Real(unit="A");
```

3.1.2 Pin

Before we begin writing models for the electrical components, we must first identify the appropriate connector for these components. The connector, called Pin, identifies the two quantities associated with a connection point in electrical circuits, namely voltage and current.

```
connector Pin
   Voltage v;
   flow Current i;
end Pin;
```

An important thing to note about this connector is the flow qualifier in front of the current i. The flow qualifier identifies quantities that sum up to zero in a connection point according to kirchhoff's first law. Variables that are declared without this qualifier are set equal at a connection point, i.e. they follow kirchhoffs second law.

3.1.3 TwoPin

Furthermore the electrical components we will define share a few other common properties, namely

1. They have one positive, and one negative pin
2. The voltage level over the component equals the voltage difference between these pins
3. The current going in and out of the component sums to zero

Therefore we define a superclass with this common properties and we also add a help variable for the current in order to make it easier to analyze results.

```
model TwoPin "Superclass of elements with two electrical
pins"
   Pin p, n;
   Voltage v;
   Current i;
equation
   v = p.v-n.v;
   0 = p.i+n.i;
   i = p.i;
end TwoPin;
```

3.1.4 Resistor

A resistor is a TwoPin that obeys Ohm's law

v = R*i

This component can be defined by extending the TwoPin superclass and adding the desired equation.

```
model Resistor "Ideal electrical resitor"
   extends TwoPin;
   parameter Real R(unit="ohm") "Resistance";
equation
```

```
   R*i = v;
end Resistor;
```

3.1.5 Capacitor

A capacitor can be defined in a similar way as we defined the resistor.

```
model Capacitor "Ideal electrical capacitor"
   extends TwoPin;
   parameter Real C(unit="F") "Capacitance";
equation
   der(v) = i/C;
end Capacitor;
```

3.1.6 Inductor

And an inductor can also be defined.

```
model Inductor "Ideal electrical inductor"
   extends TwoPin;
   parameter Real L(unit="H") "Inductance";
equation
   L*der(i) = v;
end Inductor;
```

3.1.7 VsourceAC

In a similar fashion we define a voltage source with default amplitude 220 V and default frequency 50 Hz.

```
model VsourceAC "Sine-wave voltage source"
   extends TwoPin;
   parameter Voltage VA=220 "Amplitude [V]";
   parameter Real f=50 "Frequency [Hz]";
protected
   constant Real PI=Modelica.Constants.pi;
equation
   v = VA*sin(((2*PI)*f)*time);
end VsourceAC;
```

3.1.8 Ground

Finally all electrical circuits need a ground.

```
model Ground "Ground"
   Pin p;
equation
   p.v = 0;
end Ground;
```

3.2 Modeling the Electrical Circuit

Having defined all components we can now model the circuit. This is done by declaring the components and then connecting them using connect statements.

```
model Circuit
   Resistor resistor1(R=10);
   Capacitor capacitor1(C=0.01);
   Resistor resistor2(R=100);
   Inductor inductor1(L=0.1);
   VsourceAC sineVoltage1;
   Ground ground1;
equation
   connect(sineVoltage1.p, resistor1.p);
   connect(resistor1.n, capacitor1.p);
   connect(capacitor1.n, sineVoltage1.n);
   connect(resistor1.p, resistor2.p);
   connect(resistor2.n, inductor1.p);
   connect(inductor1.n, capacitor1.n);
   connect(sineVoltage1.n, ground1.p);
end Circuit;
```

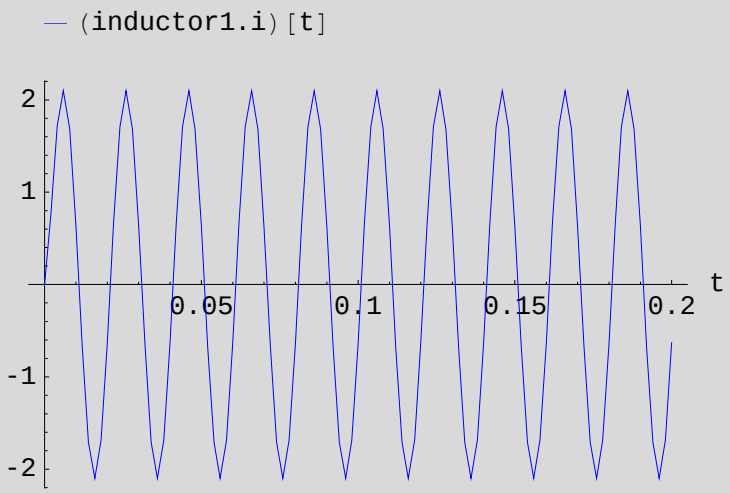
3.3 Simulating the Circuit

First simulate the model with the default initial values and parameter settings in the range $0 \le t \le 1$.

```
Simulate[Circuit, {t, 0, 1}];
```

Let us plot the current in the inductor.

```
PlotSimulation[inductor1.i[t], {t, 0, 0.2}];
```

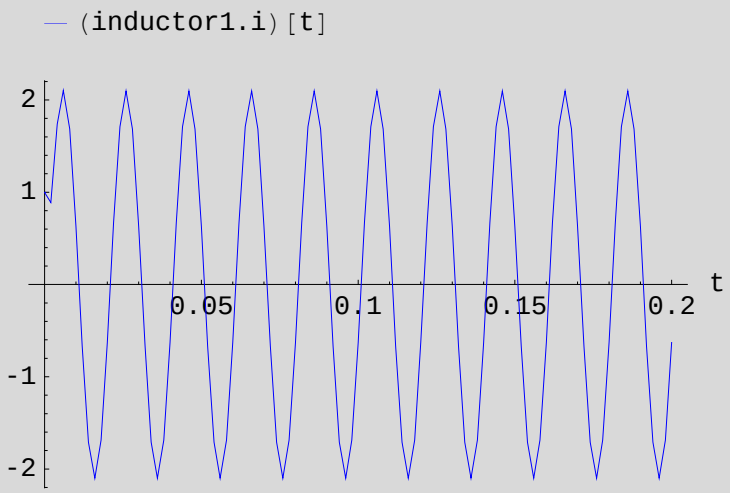


The default initial value for L.i=0. Setting another initial value is done by giving a list of equal statements in for the option InitialValues in Simulate.

```
Simulate[Circuit, {t, 0, 1}, InitialValues -> {inductor1.i == 1}];
```

A plot shows the result.

```
PlotSimulation[inductor1.i[t], {t, 0, 0.2}];
```

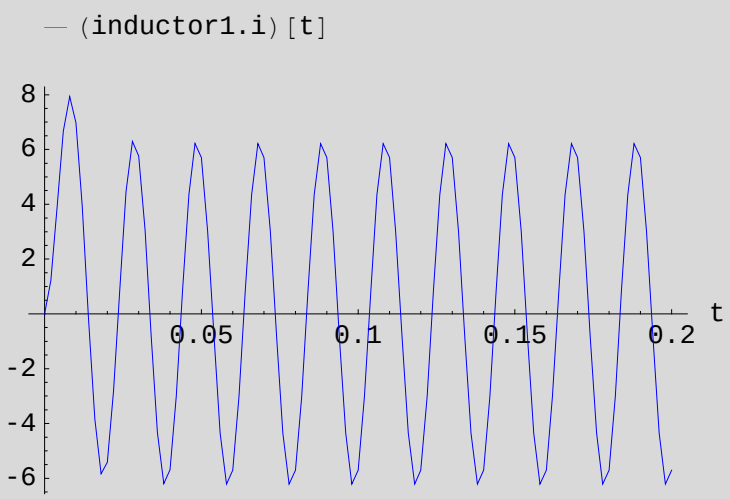


Setting parameter values is done in the same way for the option ParameterValues.

```
Simulate[Circuit, {t, 0, 1}, ParameterValues ->
   {resistor1.R == 15, resistor2.R == 15, inductor1.i == 1}];
```

Another plot shows the new result.

```
PlotSimulation[inductor1.i[t], {t, 0, 0.2}];
```



3.4 Comparing results

As mentioned the example above can be modeled with simple drag&drop using the model editor. Actually the model is available in the Getting Started package. We can print it's definition by using the GetDefinition command.

```
GetDefinition[GettingStarted.ComponentBased.ElectricCircuit]
```

```
model ElectricCircuit
  annotation(Documentation(info="<html>

<head>
<title>Simple circuit example</title>

</head>

<body lang=EN link=blue vlink=purple>

<P><h1>MathModelica® System Designer</h1></p>
```

```
<P><i><h2>Simple Circuit – Component based approach</h2></i></p>
<P><h4>© 2006 MathCore Engineering AB</h4></p>

<p><hr>
<a href="Modelica://GettingStarted">Getting Started</a> <br>
<a href="Modelica://GettingStarted.ComponentBased">Home</a> |
<a href="Modelica://GettingStarted.
  ComponentBased.BlockCircuit">Block Circuit</a> |
<strong>Electric Circuit</strong> |
<a href="Modelica://GettingStarted.
  ComponentBased.ElectricCircuit2">Electric Circuit 2</a>
<hr></p>

<h3>Introduction</h3>

<P>In this part of the example we will show how to
  develop a component based model for the circuit below.</p>

<p><img width=495 height=298 src="ComponentBased/image001.jpg"></p>

<h3>Model</h3>
<p>Naturally, to implement a component based
model of the system above is only
  about drag-and-drop, connecting the components
and setting parameters. All needed components
  can be found in the following two
electrical libraries.</p>

<p>Modelica.Electrical.Analog.Basic:</p>

<p><img width=485 height=174 src="ComponentBased/image012.jpg"></p>

<p> </p>

<p>Modelica.Electrical.Analog.Sources:</p>

<p><img width=489 height=326 src="ComponentBased/image013.jpg"></p>

<p>Connecting this components and setting parameters results
  in a model that looks very much like the drawing we begun
with:</p>

<p><img width=323 height=338 src="ComponentBased/image014.jpg"></p>

<p>Now we can simulate and plot the resulting current
  through the signal voltage, and as expected it looks just
like the result plotted from the block model.</p>
```

```
<p><img width=595 height=210 src="ComponentBased/image015.jpg"></p>

<p><hr>
<a href="Modelica://GettingStarted">Getting Started</a> <br>
<a href="Modelica://GettingStarted.ComponentBased">Home</a> |
<a href="Modelica://GettingStarted.
  ComponentBased.BlockCircuit">Block Circuit</a> |
<strong>Electric Circuit</strong> |
<a href="Modelica://GettingStarted.
  ComponentBased.ElectricCircuit2">Electric Circuit 2</a>
<hr></p>


</body>

</html>
"),Icon(coordinateSystem(extent={{-100,-100},{100,100}}),graphics=
  {Text(visible=true, fillPattern=FillPattern.Solid, extent={{-100,
  -150},{100,-110}}, textString="%name"),Rectangle(visible=true,
  fillColor={85,170,0}, fillPattern=FillPattern.Solid, extent=
  {{-100,-100},{100,100}}),Rectangle(visible=true, fillColor={170,
  85,0}, fillPattern=FillPattern.Solid, extent={{2.5,16.49},{17.5,
  50}}),Rectangle(visible=true, fillColor={170,85,0}, fillPattern=
  FillPattern.Solid, extent={{2.5,-33.51},{17.5,0}}),Rectangle(
  visible=true, fillColor={170,85,0}, fillPattern=FillPattern.
  Solid, extent={{42.5,16.49},{57.5,50}}),Rectangle(visible=true,
  fillColor={170,85,0}, fillPattern=FillPattern.Solid, extent=
  {{42.5,-33.51},{57.5,0}}),Ellipse(visible=true, fillColor={170,
  0,0}, fillPattern=FillPattern.Solid, extent={{-70,-6.45375},
  {-40,23.5462}}),Line(visible=true, points={{-70,-60},{-40,
  -60}}, thickness=1.0),Line(visible=true, points={{-62.5114,
  -65},{-47.1016,-65}}, thickness=1.0),Line(visible=true, points=
  {{-58.1502,-70},{-50,-70}}, thickness=1.0),Line(visible=true,
  points={{-55,25},{-55,72.6877},{50,72.6877},{50,50}}),Line(
  visible=true, points={{10,50},{10,72.1062}}),Line(visible=true,
  points={{10,0},{10,15}}),Line(visible=true, points={{50,0},{50,
  16.5728}}),Line(visible=true, points={{50,-33.4364},{50,-45},{-55,
  -45},{-55,-7.26877}}),Line(visible=true, points={{10,-33.1456},
  {10,-45}}),Line(visible=true, points={{-55,-45},{-55,-60}})}),
  Diagram(coordinateSystem(extent={{-148.5,-105},{148.5,105}})));
  Modelica.Electrical.Analog.Basic.Ground ground1
  annotation(Placement(visible=true,transformation(
  x=-68.4609,y=-14.8828,scale=0.075)));
  Modelica.Electrical.Analog.Basic.Resistor resistor1(R=
  10) annotation(Placement(visible=true,transformation(
  x=-48.2656,y=52.5,scale=0.075,rotation=270)));
  Modelica.Electrical.Analog.Basic.Resistor resistor2(R=
  100) annotation(Placement(visible=true,transformation(
```

```
  x=0,y=54.3255,scale=0.075,rotation=270)));
  Modelica.Electrical.Analog.Basic.Inductor inductor1(L=
  0.1) annotation(Placement(visible=true,transformation(
  x=0,y=18.0729,scale=0.075,rotation=270)));
  Modelica.Electrical.Analog.Sources.SineVoltage
  sineVoltage1 annotation(Placement(visible=true,
  transformation(x=-90,y=37.5,scale=0.075,rotation=270)));
  Modelica.Electrical.Analog.Basic.Capacitor capacitor1(C=
  0.01) annotation(Placement(visible=true,transformation(
  x=-48.3018,y=16.5383,scale=0.075,rotation=270)));

equation
  connect(resistor1.n,capacitor1.p) annotation(Line(visible=
  true,points={{-48.3018,45.2253},{-48.3018,24.3047}}));
  connect(capacitor1.n,ground1.p) annotation(Line(visible=true,
  points={{-48.3018,9.22965},{-48.3018,-7.5},{-68.6071,-7.38372}}));
  connect(sineVoltage1.n,ground1.p) annotation(Line(visible=true,
  points={{-90.1429,30.1502},{-90,-7.5},{-68.6071,-7.38372}}));
  connect(sineVoltage1.p,resistor1.p) annotation(
  Line(visible=true,points={{-90.1429,45.2253},
  {-90,82.5},{-48.3018,82.5},{-48.3018,59.9927}}));
  connect(inductor1.n,ground1.p) annotation(Line(visible=
  true,points={{0,10.7679},{0,-7.5},{-68.6071,-7.38372}}));
  connect(resistor2.n,inductor1.p) annotation(Line(
  visible=true,points={{0,47.0712},{0,25.843}}));
  connect(resistor2.p,resistor1.p) annotation(Line(visible=true,
  points={{0,61.8464},{0,82.5},{-48.2865,82.5},{-48.2865,60.1927}}));
end ElectricCircuit;
```

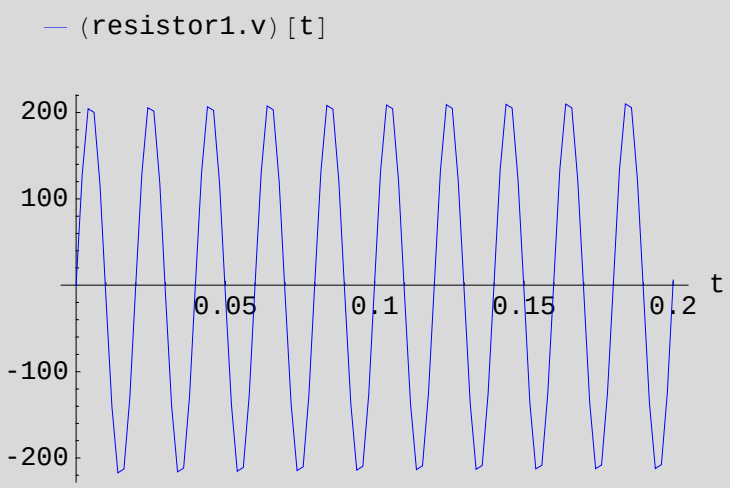
As seen the model also contains some html code used for documentation. The actual model is seen in the last lines and is similar to the model that we have created. The difference being that standard Modelica components are used instead of the components defined by us, and also annotations are added. These annotations contain the graphical information used and created by the model editor.

We can compare the results between the two models by storing simulation results in separate variables. Note that the default values for the voltage source differs between the models, and therefore we modify these in the simulation command, adding modifications for the parameter values.

```
nbCircuit = Simulate[Circuit, {t, 0, 1}];
meCircuit =
  Simulate[GettingStarted.ComponentBased.ElectricCircuit, {t, 0, 1},
   ParameterValues → {sineVoltage1.V == 220, sineVoltage1.freqHz == 50}];
```

The plots below show that the results are identical just as expected.

```
PlotSimulation[resistor1.v[t],
  {t, 0, 0.2}, SimulationResult → nbCircuit];
```




```
PlotSimulation[resistor1.v[t],
  {t, 0, 0.2}, SimulationResult → meCircuit];
```