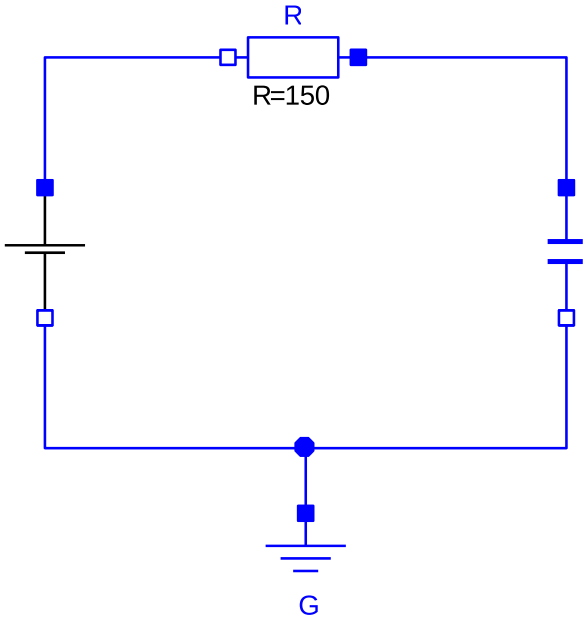
Virtual Physics
Equation-Based Modeling

TUM, November 8, 2022

Modeling in Modelica – Basic Principles

```
model SimpleCircuit "A simple RC circuit"
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C = 0.001 "Capacity";
  parameter SI.Resistance = 100  "Resistance";
  parameter SI.Voltage V0 = 10 "Source Voltage";
  SI.Current i "Current" ; SI.Voltage uC
 "Capacitor Voltage";
initial equation
  uC = 0;
equation
  V0-uC = R*i;
  der(uC)*C = i;
end SimpleCircuit;
```



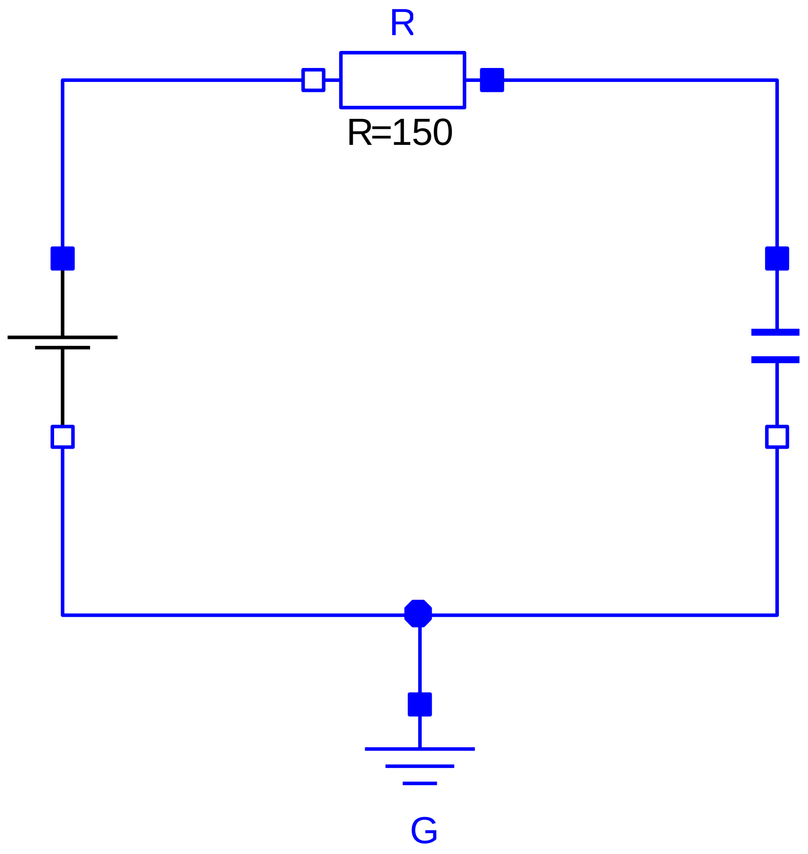
Dr. Dirk Zimmer

German Aerospace Center (DLR), Robotics and Mechatronics Centre

Motivation

In this lecture, the language Modelica is officially introduced.

- We will study the modeling of physical systems in Modelica.
- To this end, we examine the modeling of simple electric circuits.
- Let us start with the modeling of the electric circuit from the last lecture



A Simple Example

Robotics and Mechatronics Centre

For this simple circuit, we can still derive all equations by hand, even in a very compact form.

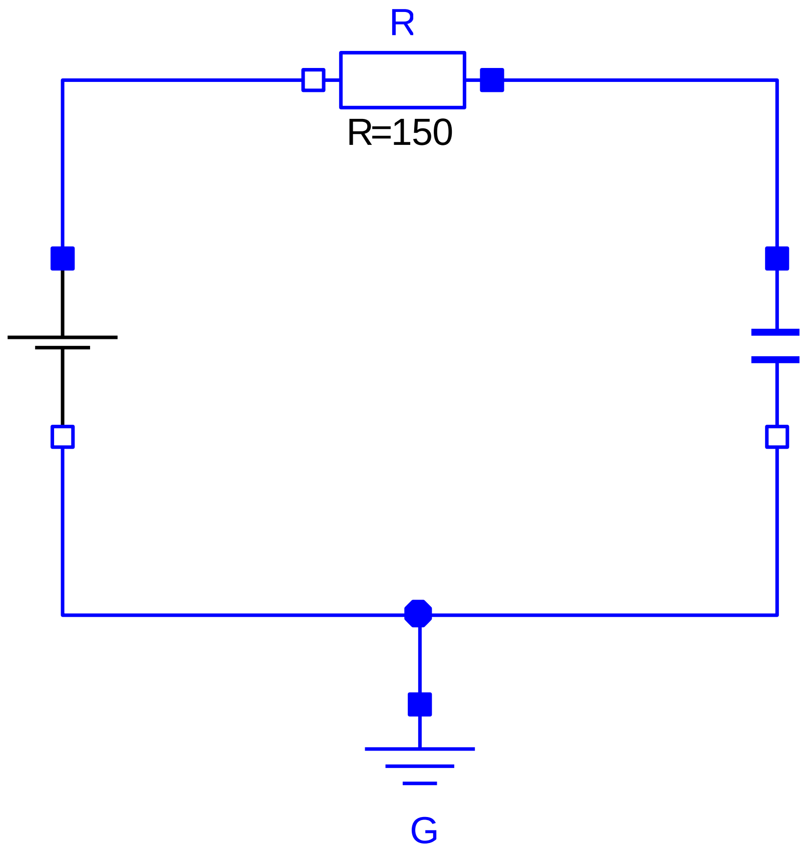
- The current is determined by:

$$10 - u_C = R \cdot i$$

- And the capacitor voltage is state of the system:

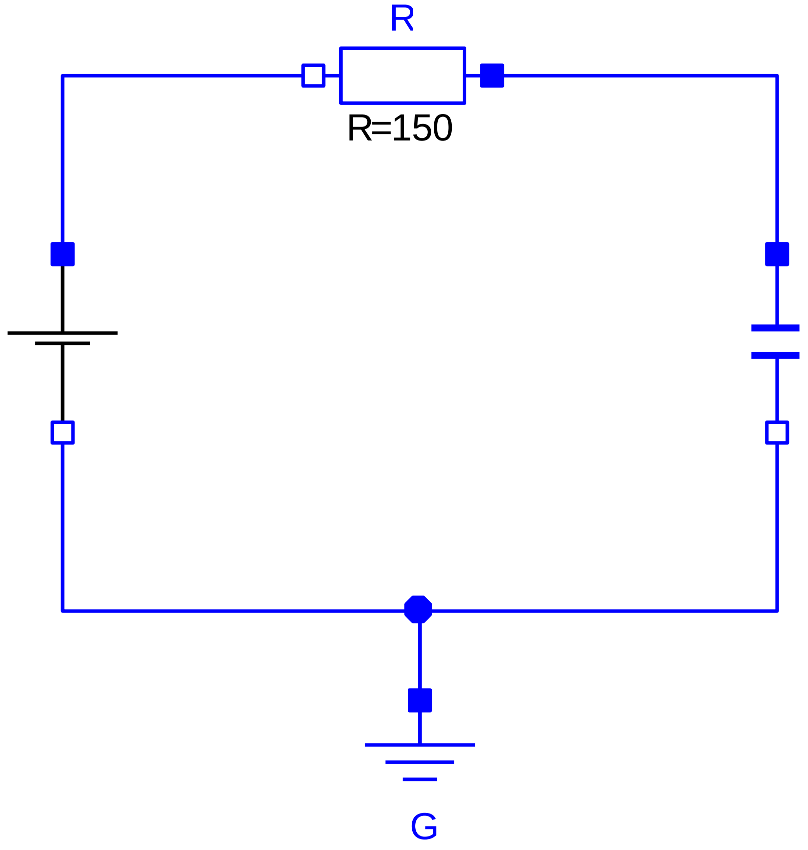
$$du_C/dt \cdot C = i$$

- Let us punch that into the computer by using Modelica



A Simple Example


```
model SimpleCircuit


  parameter Real C;
  parameter Real R;
  parameter Real V0;


  Real i;
  Real uC;



equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```



Model Structure

Robotics and Mechatronics Centre

```
model SimpleCircuit


  parameter Real C;
  parameter Real R;
  parameter Real V0;


  Real i;
  Real uC;



equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- The fundamental Modelica entity is a class, in this case it is a model.
- Then we declare the Model parameters...
- ... and the system variables.
- Finally, there are the model equations.
- This is the general structure of every model.
- The order of equations does not matter. This holds also for the order of declarations.

Model Structure

TUM + DLR
Robotics and Mechatronics Centre

```
model SimpleCircuit
 "A simple RC circuit"


  parameter Real C;
  parameter Real R;
  parameter Real V0;


  Real i;
  Real uC;



equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- Each model starts with the header…
- …and ends with a repetition of its name.
- We can add a model description and indeed we should do so.

Parameters

Robotics and Mechatronics Centre

```
model SimpleCircuit
 ”A simple RC circuit”

  parameter Real C = 0.001
  "Capacity";
  parameter Real R = 100
 "Resistance”;
  parameter Real V0 = 10
  ”Source Voltage”;

  Real i;
  Real uC;


equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- Parameter represent values that are determined before the simulation starts and remain constant during the simulation time.
- Also the parameter declarations can be extended by a description.
- We can also provide default values.

Variable Types

Robotics and Mechatronics Centre

```
model SimpleCircuit
 "A simple RC circuit"

  parameter Real C = 0.001
  "Capacity";
  parameter Real R = 100
  "Resistance";
  parameter Real V0 = 10
  "Source Voltage";

  Real i "Current" ;
  Real uC "Capacitor Voltage";


equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- The most important basic variable types are:
  - Real
  - Integer
  - Boolean
- There are also strings and enumerations.
- A textual description should also be assigned to the variables.

Equation Block

Robotics and Mechatronics Centre

```
model SimpleCircuit
 ”A simple RC circuit”

  parameter Real C = 0.001
  ”Capacity”;
  parameter Real R = 100
  ”Resistance”;
  parameter Real V0 = 10
  ”Source Voltage”;

  Real i ”Current” ;
  Real uC ”Capacitor Voltage”;


equation

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- Equations can be stated in any order.
- The operator der(…) is a built-in operator and represents the time-derivative
- IMPORTANT! The equation is not causal; it is a non-causal relation between variables. It is not an assignment!
- If we want to state a causal assignment we can use

  :=

  instead of

  =

Initial equations

Robotics and Mechatronics Centre

```
model SimpleCircuit
 "A simple RC circuit"

  parameter Real C = 0.001
  "Capacity";
  parameter Real R = 100
  "Resistance";
  parameter Real V0 = 10
  "Source Voltage";

  Real i "Current" ;
  Real uC "Capacitor Voltage";

initial equation
  uC = 0;
equation
  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- We are still missing the initial equations.
- They can be stated in a separate block.
- If there is a insufficient number of initial equations, Dymola will assume zero (or a nominal value) for the remaining undetermined variables.
- Also a warning is displayed.

Units (1)


```
model SimpleCircuit
 "A simple RC circuit"

  parameter Real C(unit="F") =
  0.001 "Capacity";
  parameter Real R(unit="Ohm")
   = 100  "Resistance";
  parameter Real V0(unit="V")
   = 10 "Source Voltage";

  Real i(unit="A") "Current" ;
  Real uC(unit="V")
  "Capacitor Voltage";
initial equation
  uC = 0;
equation
  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- We can assign units to the variables and parameters.
- These units are than matched in the equations.
- In case of a unit mismatch, a warning is messaged.
- Modeling shall always be performed using the SI units.

Units (2)

Robotics and Mechatronics Centre

```
model SimpleCircuit
 ”A simple RC circuit”
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C =
  0.001 ”Capacity”;
  parameter SI.Resistance R =
   = 100  ”Resistance”;
  parameter SI.Voltage V0
   = 10 ”Source Voltage”;

  SI.Current i ”Current” ;
  SI.Voltage uC
 ”Capacitor Voltage”;
initial equation
  uC = 0;
equation
  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- It is in general more convenient to use the predefined set of types of the Modelica.SIunits package.
- The package can be imported by using the import statement.
- Usually, the import is only done once for each package of models. Not in every model separately as here.

Summary

Robotics and Mechatronics Centre

```
model SimpleCircuit
 "A simple RC circuit"
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C =
  0.001 "Capacity";
  parameter SI.Resistance R =
   = 100  "Resistance";
  parameter SI.Voltage V0
   = 10 "Source Voltage";

  SI.Current i "Current" ;
  SI.Voltage uC
 "Capacitor Voltage";
initial equation
  uC = 0;
equation
  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```

- Voila, our first complete Modelica model.
- Modeling in such a flat way is only possible of small systems.
- In order to model larger system we want to decompose our systems into components.
- We can do this by creating a separate model for each component.

The Ground Model


```
model Ground
 "Ground Element"

  SI.Current i;
  SI.Voltage v;

equation
  v=0;

end Ground;
```

- As we have learned last lecture, the ground is represented by a pair of variables
  - The voltage potential v
  - The current i
- Please note that the model is incomplete. Two variables but just one equation.
- The missing equations will be added when we connect the component within the circuit.

The Resistor

Robotics and Mechatronics Centre

```
model Resistor
 "Resistor Model"

  parameter SI.Resistance R;

  SI.Current i1;
  SI.Voltage v1;

  SI.Current i2;
  SI.Voltage v2;

protected
  SI.Current i;
  SI.Voltage u;

equation
  u = v2-v1;
  i1 + i2 = 0;
  i = i2;
  u = R*i;
end Resistor;
```

- The resistor has two pins, and consequently, it requires two pairs of variables.
- Internal variables can be hidden in a protected section. These variables are not accessible from outside.
- Again the model is incomplete

  6 variables and 4 equations.

The Capacitor

Robotics and Mechatronics Centre

```
model Capacitor
 "Ideal Capacitor Model"

  parameter SI.Capacitance C;

  SI.Current i1;
  SI.Voltage v1;

  SI.Current i2;
  SI.Voltage v2;

protected
  SI.Current i;
  SI.Voltage u;

equation
  u = v2-v1;
  i1 + i2 = 0;
  i = i2;
  der(u)*C = i;
end Capacitor;
```

- The capacitor looks almost the same.

The Voltage Source

Robotics and Mechatronics Centre

```
model ConstantVoltage
 "A Source of Constant Voltage"

  parameter SI.Voltage V0 = 100;

  SI.Current i1;
  SI.Voltage v1;

  SI.Current i2;
  SI.Voltage v2;

protected
  SI.Current i;
  SI.Voltage u;

equation
  u = v2-v1;
  i1 + i2 = 0;
  i = i2;
  u = V0;
end ConstantVoltage;
```

- item

Creating a Package


```
package Electrics
 "Basic Electric Elements"

  import SI = Modelica.SIunits;

  model Ground
    …
  end Ground;

  model Resistor
    …
  end Resistor;

  model Capacitor
    …
  end Capacitor;

  …

end Electrics;
```

- All these models can be collected in a Modelica package
- A package can contain arbitrary classes, also sub-packages.
- The look-up of class-names within a package is first done locally within a class and then further up the hierarchy.
- Hence, the import statement is valid for all models in the package.
- Identifiers of instances (variables or components) are only looked up locally.

Using the Package


```
model SimpleCircuit
 "A simple RC Circuit"

 Electrics.Ground G;
 Electrics.ConstantVoltage S
 (V0=10);
 Electrics.Resistor R(R=100);
 Electrics.Capacitor C(C=0.001);

equation
  …

end SimpleCircuit;
```

- Now we can use these models in order to compose our circuit.
- To this end, we simply declare the models like simple variables.
- We can set the parameters of the sub-model in the modifier.
- We are still missing $3 \cdot 2 + 1 = 7$ equations…

Using the Package

Robotics and Mechatronics Centre

```
model SimpleCircuit
 "A simple RC Circuit"

 Electrics.Ground G;
 Electrics.ConstantVoltage S
 (V0=10);
 Electrics.Resistor R(R=100);
 Electrics.Capacitor C(C=0.001);

equation
 S.v2 = R.v1
 S.i2 + R.i1 = 0;

 R.v2 = C.v1
 R.i2 + C.i1 = 0;

 C.v2 = S.v1
 C.v2 = G.v
 C.i2 + S.i1 + G.i = 0;

end SimpleCircuit;
```

- The missing equations are the connection equations of the nodes.
- We still have to enter these equations manually.
- This is highly inconvenient and error-prone.
- For this reason, Modelica enables the definition of connectors.

Connectors


```
connector Pin

  SI.Voltage v "Potential at the pin";
  flow SI.Current i "Current flowing into the pin";

end Pin;
```

- This is the definiton of the corresponding connector.
- It consists in a set of variables.
- These variables can be declared to be...
  - potential variables: `SI.Voltage v`
  - flow variables: `flow SI.Current i`

Connector equations

Robotics and Mechatronics Centre

```
connector Pin

  SI.Voltage v "Potential at the pin";
  flow SI.Current i "Current flowing into the pin";

end Pin;
```

- We can link two ore more pins by using the connect statement.

```
connect(pin1, pin2)   }   pin1.v = pin2.v
connect(pin1, pin3)   }   pin1.v = pin3.v
                          pin1.i + pin2.i +pin3.i = 0
```

- The equations are generated in dependence on the declaration
- Connections form a graph that represents a wood and that is component relevant and structure irrelevant.

Modeling with Connectors

TUM + DLR Robotics and Mechatronics Centre

```
model Resistor
 "Resistor Model"

  parameter SI.Resistance R;

  Pin n;
  Pin p;

  SI.Current i;
  SI.Voltage u;


equation

  u = p.v - n.v;
  n.i + p.i = 0;
  i = p.i;
  u = R*i;

end Resistor;
```

- Let us integrate the pin connector in our Resistor model.
- To this end, we declare two pins
- We can access the connector variables like any other variables.
- Likewise, the procedure is done for all other components

Balanced Models

Robotics and Mechatronics Centre

```
model Resistor
 "Resistor Model"

  parameter SI.Resistance R;

  Pin n;
  Pin p;

  SI.Current i;
  SI.Voltage u;


equation

  u = p.v - n.v;
  n.i + p.i = 0;
  i = p.i;
  u = R*i;

end Resistor;
```

- Using connectors has an immediate advantage for our models.
- In any physical model, there will be exactly one connecting equation for each pair of connector variables.
- Hence, we can immediately check if our system is structurally regular.
- 2 variables + 4 connector variables = 6
- 2 connector equations and 4 local equations = 6
- Bingo!

Connecting Components

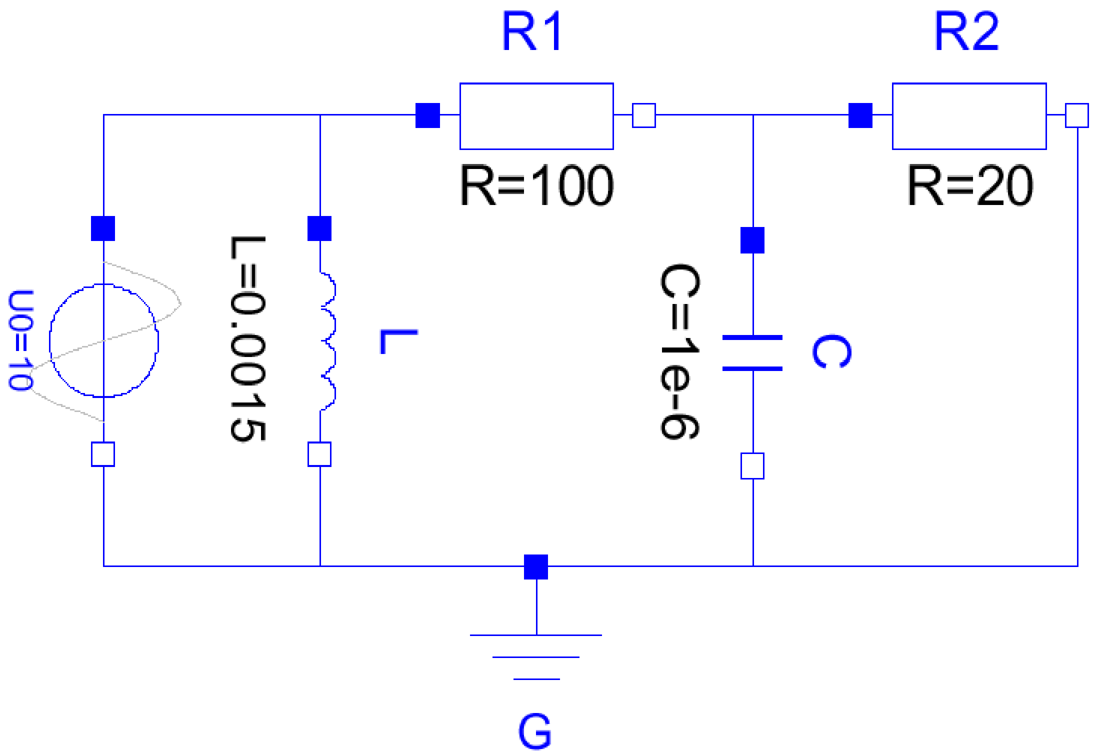
Robotics and Mechatronics Centre

```
model SimpleCircuit
 ”A simple RC Circuit”

 Electrics.Ground G;
 Electrics.ConstantVoltage S
 (V0=10);
 Electrics.Resistor R(R=100);
 Electrics.Capacitor C(C=0.001);

equation

 connect(G.p,S.n);

 connect(S.p,R.n);

 connect(R.p,C.n);

 connect(C.p,G.p);

end SimpleCircuit;
```

- Back to our simple RC Circuit.
- Now we can compose the circuit by using the connect statement.
- This looks much better than before.

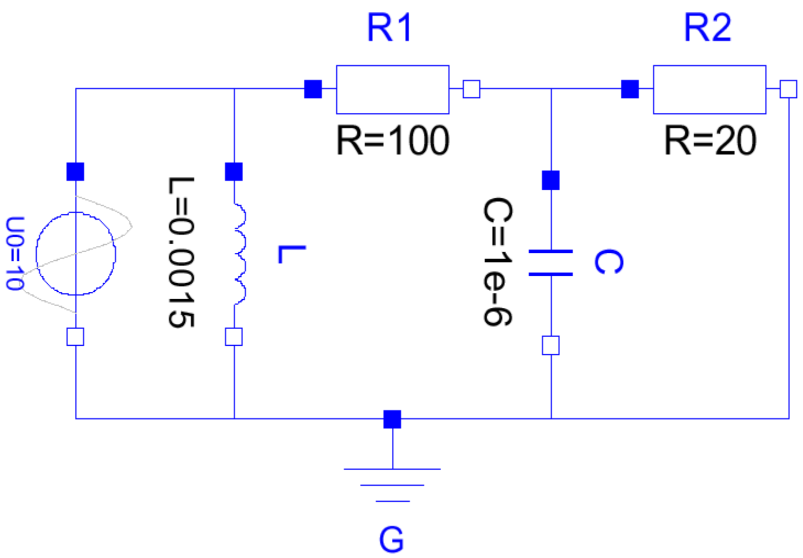
Second Example

- Now, we can compose the model of a (slightly) larger electric circuit almost without having to think (politician proof).



Second Example


```
model Circuit
  Resistor R1(R=100);
  Resistor R2(R=20);
  Capacitor C(C=1e-6);
  Inductor L(L=0.0015;
  SineVSource S(Ampl=15, Freq=50);
  Ground G;

equation
  connect(G.p,S.n)
  connect(G.p,L.n)
  connect(G.p,R2.n)
  connect(G.p,C.n)

  connect(S.p,R1.p)
  connect(S.p,L.p)

  connect(R1.n,R2.p)
  connect(R1.n,C.p)
end Pin;
```



Second Example

Robotics and Mechatronics Centre

```
model Circuit
  Resistor R1(R=100);
  Resistor R2(R=20);
  Capacitor C(C=1e-6);
  Inductor L(L=0.0015;
  SineVSource S(Ampl=15, Freq=50);
  Ground G;

equation
  connect(G.p,S.n)
  connect(G.p,L.n)
  connect(G.p,R2.n)
  connect(G.p,C.n)

  connect(S.p,R1.p)
  connect(S.p,L.p)

  connect(R1.n,R2.p)
  connect(R1.n,C.p)
end Circuit;
```

$5 \cdot 4 + 1 \cdot 1 = 21$

component equ.

$5 \cdot 6 + 1 \cdot 2 = 32$

unknowns

32 equations

32 unknowns

8 potential equations

3 flow equations

Inheritance

Robotics and Mechatronics Centre

```
partial model OnePort

  SI.Voltage u;
  SI.Current i;

  Pin p;
  Pin n;

equation
  u = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```

- We already noticed that the resistor, capacitor, and voltage source share most of their equations.
- We can share this common part by declaring an abstract base model.
- The base model can serve as template for many concrete models.
- It is denoted as partial, since there are equations missing and the abstract base model should not be instantiated.

Inheritance

TUM + DLR
Robotics and Mechatronics Centre

```
partial model OnePort

  SI.Voltage u;
  SI.Current i;

  Pin p;
  Pin n;

equation
  u = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```

```
model Capacitor
  extends OnePort;
  parameter
   SI.Capacitance C=1;
equation
    der(u)*C = i;
end Capacitor;

model Resistor
  extends OnePort;
  parameter
   SI.Resistance R=1;
equation
    u = R*i;
end Resistor;
```

Inheritance

- New models can be generated out of the partial base model by the keyword **`extends`**.
- Then the missing parameters and equations are added.
- The keyword **`extends`** can be applied in a very generic way.
- Multiple inheritance is possible as well.

```
model Capacitor
  extends OnePort;
  parameter
   SI.Capacitance C=1;
equation
    der(u)*C = i;
end Capacitor;

model Resistor
  extends OnePort;
  parameter
   SI.Resistance R=1;
equation
    u = R*i;
end Resistor;
```

Conclusions

Let us conclude by a few general remarks

- Modelica provides means to express differential-algebraic equation systems in a convenient way.
- Modelica enables to organize the knowledge in a hierarchical form.
- Modelica is a declarative modeling language. It is not a programming language.
- The declarative style enables the modeler to focus on ***what*** he wants to model rather than to think about ***how*** to achieve a computational realization.
- In this way, the models become also more self-contained. They represent meaningful semantic entities by themselves even without being simulated.

Outlook

- In this lecture we looked at the textual modeling.
- But most of the modeling in Modelica/Dymola is graphical.

The End