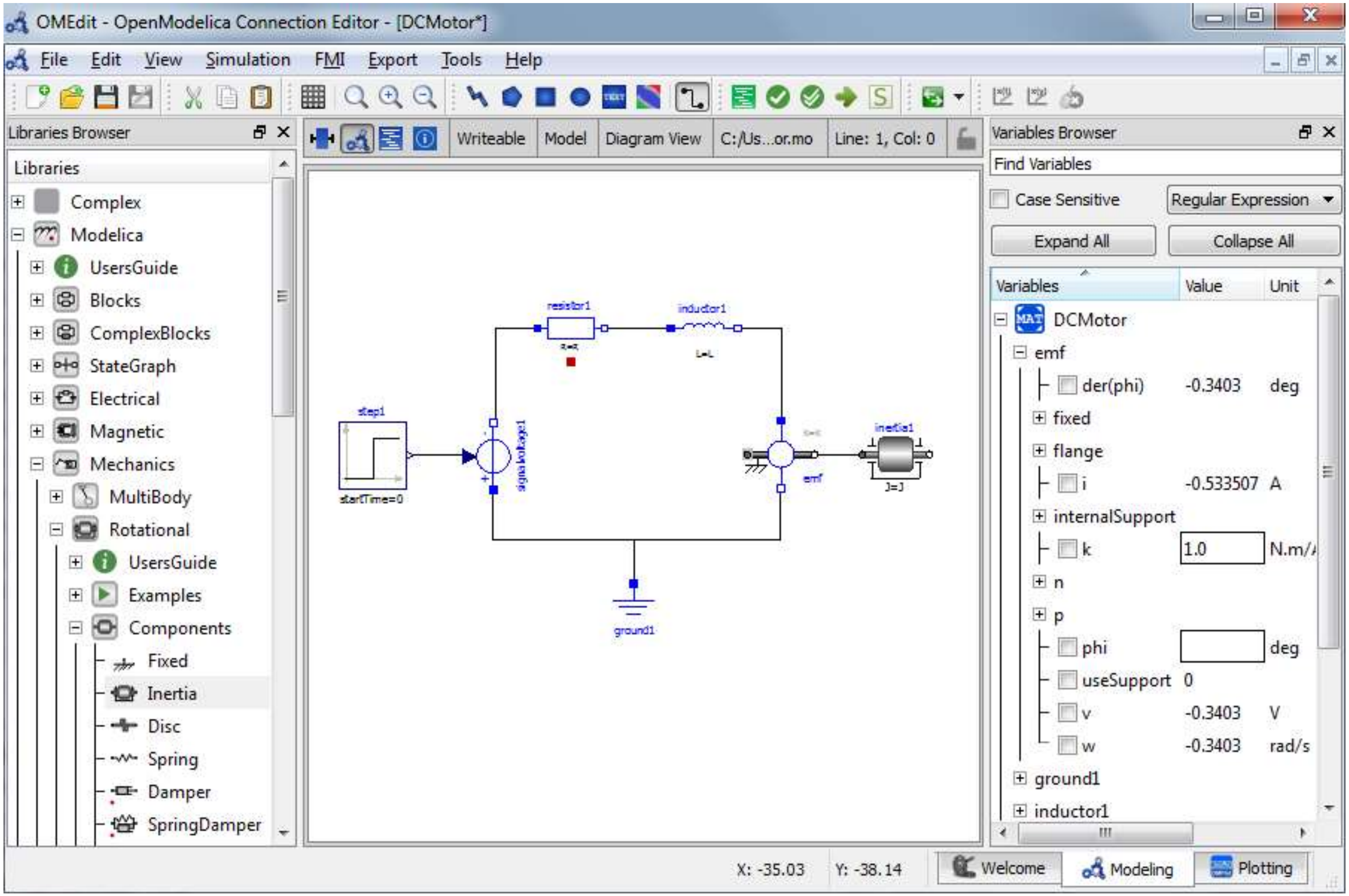
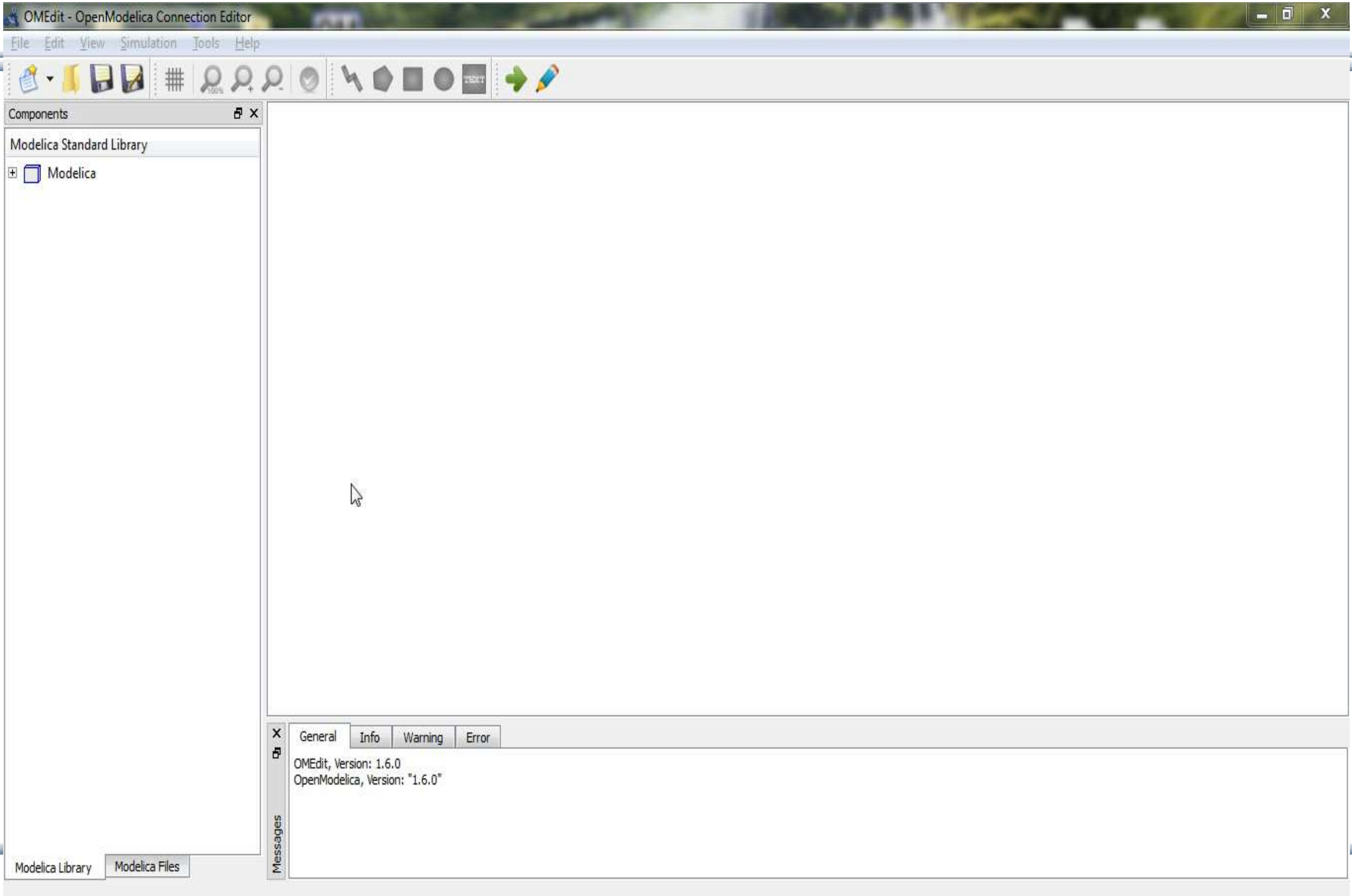
More Graphical Modeling Exercises with Libraries

using OpenModelica

Graphical Modeling - Using Drag and Drop Composition



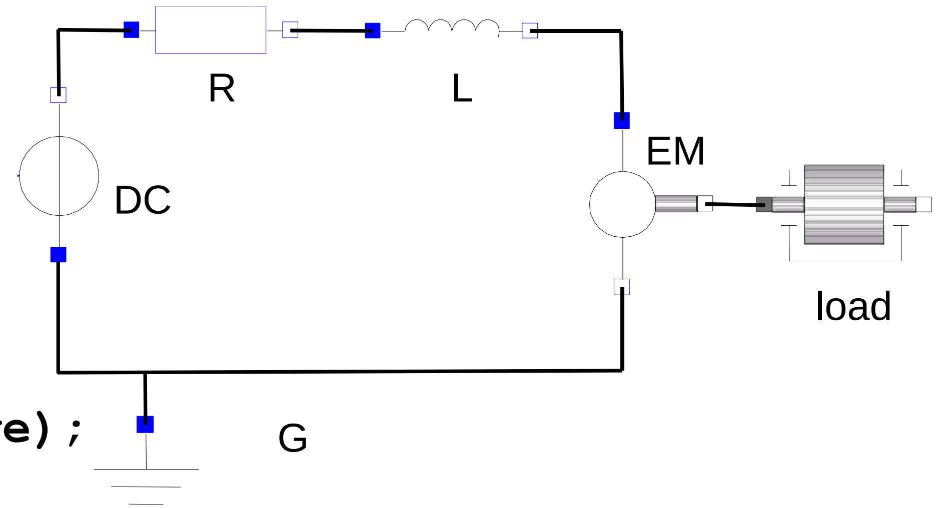
Graphical Modeling Animation – DCMotor



Multi-Domain (Electro-Mechanical) Modelica Model

- A DC motor can be thought of as an electrical circuit which also contains an electromechanical component

```
model DCMotor
   Resistor R(R=100);
   Inductor L(L=100);
   VsourceDC DC(f=10);
   Ground G;
   ElectroMechanicalElement EM(k=10,J=10, b=2);
   Inertia load;
equation
   connect(DC.p,R.n);
   connect(R.p,L.n);
   connect(L.p, EM.n);
   connect(EM.p, DC.n);
   connect(DC.n,G.p);
   connect(EM.flange,load.flange);
end DCMotor
```



Corresponding DCMotor Model Equations

The following equations are automatically derived from the Modelica model:

```
0 == DC.p.i + R.n.i          EM.u == EM.p.v - EM.n.v              R.u == R.p.v - R.n.v
DC.p.v == R.n.v              0 == EM.p.i + EM.n.i                 0 == R.p.i + R.n.i
                             EM.i == EM.p.i                       R.i == R.p.i
0 == R.p.i + L.n.i           EM.u == EM.k * EM.ω                  R.u == R.R * R.i
R.p.v == L.n.v               EM.i == EM.M / EM.k
                             EM.J * EM.ω == EM.M - EM.b * EM.ω    L.u == L.p.v - L.n.v
0 == L.p.i + EM.n.i                                               0 == L.p.i + L.n.i
L.p.v == EM.n.v              DC.u == DC.p.v - DC.n.v              L.i == L.p.i
                             0 == DC.p.i + DC.n.i                 L.u == L.L * L.i'
0 == EM.p.i + DC.n.i         DC.i == DC.p.i
EM.p.v == DC.n.v             DC.u == DC.Amp * Sin[2 π DC.f * t]

0 == DC.n.i + G.p.i
DC.n.v == G.p.v
```

(`load` component not included)

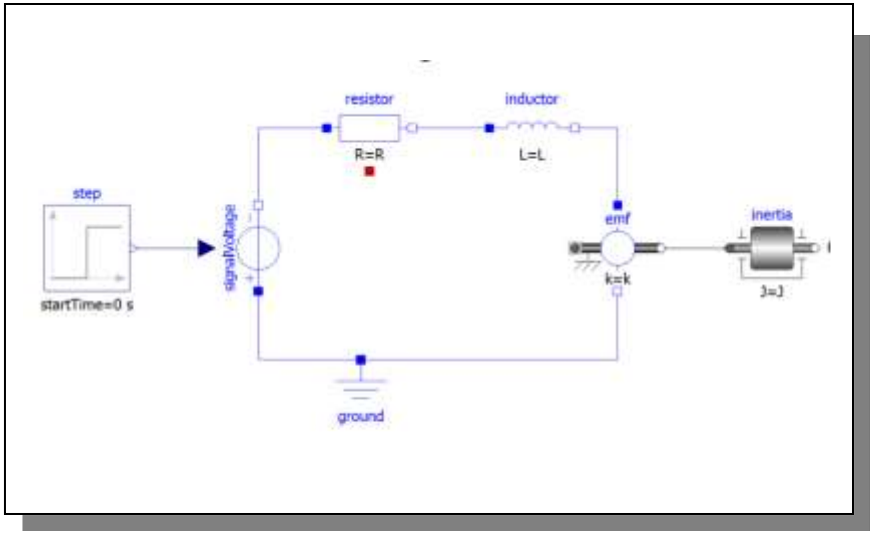
Automatic transformation to ODE or DAE for simulation:

$$\frac{dx}{dt} == f[x, u, t] \qquad g\left[\frac{dx}{dt}, x, u, t\right] == 0$$

Exercise 3.1

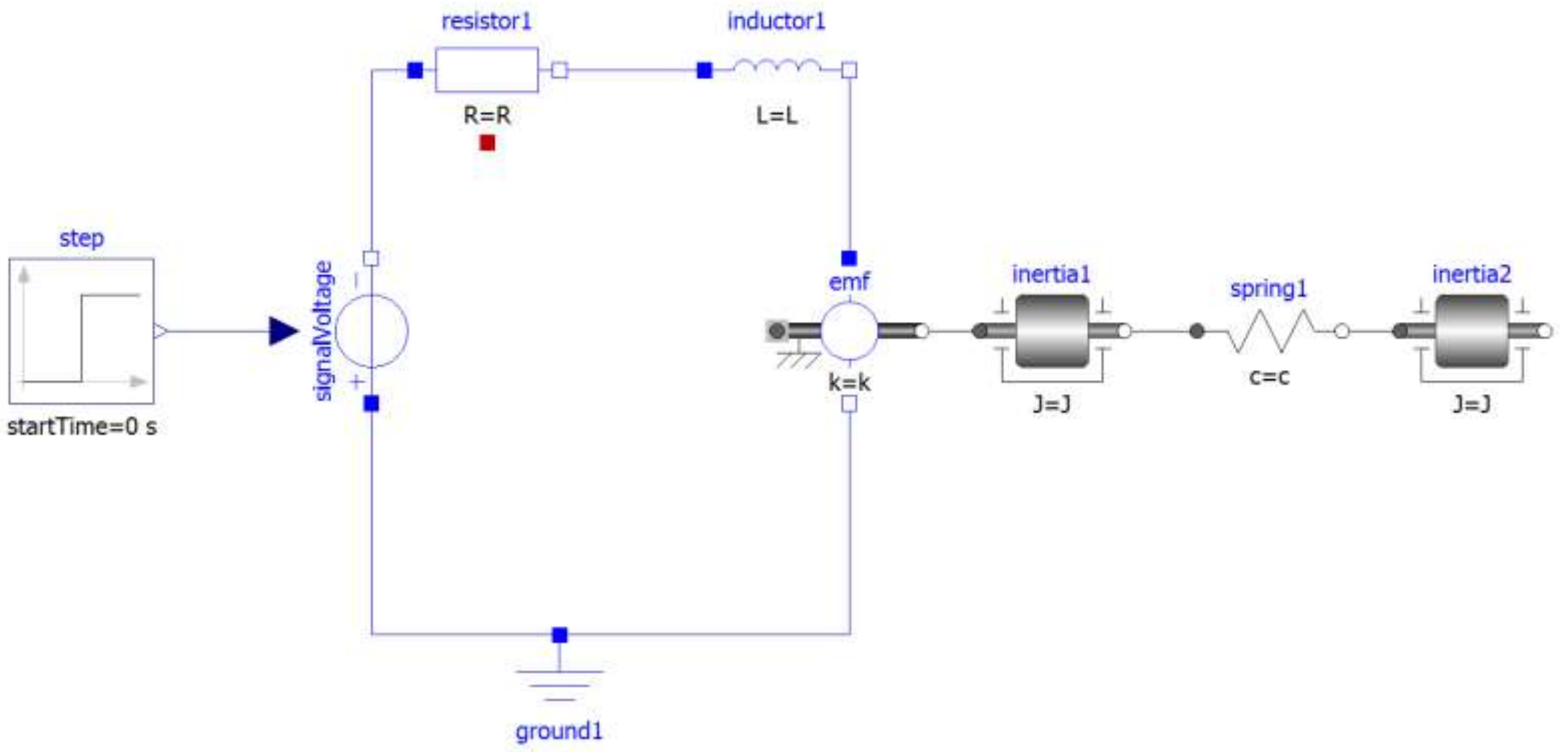
- Draw the `DCMotor` model using the graphic connection editor using models from the following Modelica libraries:
  `Mechanics.Rotational.Components`,
  `Electrical.Analog.Basic`,
  `Electrical.Analog.Sources - signalVoltage`
  `Step` in `Blocks.Sources`

- Simulate it for 15s and plot the variables for the outgoing rotational speed on the inertia axis and the voltage on the voltage source in the same plot.



Exercise 3.2

- Add a torsional spring to the outgoing shaft and another inertia element. Simulate again and see the results. Adjust some parameters to make a rather stiff spring.



Exercise 3.3

- Add a `PI` controller to the system and try to control the rotational speed of the outgoing shaft. Verify the result using a step signal for input. Tune the PI controller by changing its parameters in OMEdit.
- PI controller in `Blocks.Continuous`, `Feedback` in `Math` library, `Step` in `Blocks.Sources`