OpenModelica Annual Workshop

Teaching Modelica to First-Semester Engineering Students

Konstantin Filonenko, Rema Ahmid,
Jan Petersen, Ole Albrektsen,
Henrik Midtiby, Christian Veje
University of Southern Denmark

February 04, 2019

Overview

General information about the 1st semester course

- Idea of the course
- Why to take the course?
- Teaching strategy
- Form of evaluation

Stages of a learning process

- Learn to handle data
- From physical system to mathematical model
- Verify and validate
- Understand
- Apply

Semester plan

Date	Seminar
MPRG. Part 1 (September-October)	
September, 7	*Lesson 1. Matlab scripts, functions and mathematical operations. Application to electrical circuits*
September, 10	*Exercises*
September, 14	*Lesson 2. Linear algebra. Circuit analysis (DC)*
September, 17	*Exercises*
October, 5	*Lesson 3. Import, plotting, fitting, derivative, export*
October, 12	*Lesson 4. ODEs and RCL-curcuits (AC)*
BREAK	
MPRG. Part 2 (November)	
November, 2	*Lesson 5. Repetition. Equivalents and bridges. Parasitic effects*
November, 9	*Lesson 6. Complex impedances and Steady State Analysis*
November, 16	*Lesson 7. Tylor and Fourier series. Fourier analysis and filters*
November, 21	*Lesson 8. Logic and loops. Logical circuits*
November, 23	*Lesson 9. Repetition. Electronic devices.*
November, 30	*Lesson 10. Additional topics. Exam info.*
PROJECT	
MPRG. Examination (January-February)	

MATLAB
The Language of Technical Computing

MATLAB® Grader™

OpenModelica

Focus of the course

1. Simple programming
2. Simple numerical math
3. RLC-circuit modeling

SDU

Idea of the course

1. Data handling in Matlab: **Import/export**, **fitting**, if, for, while
2. Basic math: **Linear matrix equations**, integration, differentiation
3. Generalization: **ODEs** are a linear matrix equation with dynamics
4. Lumped models: **RLC – analogy** with mechanical systems
5. **State space** representation and **verification** of Modelica models

Why to take the course?

1. Little math and code are enough to **simulate** reality

2. **RLC - analogy** gives control over many physical domains

3. Model **verification** is a first step to validation and prediction

4. **Model-based control** can be implemented on equipment to adjust its behavior to our needs

Teaching strategy

1. Ability to solve **differential equations** is a cornerstone of engineering careers

2. ODE solvers can be used without prior understanding of their principles: **first do, then understand**

3. After the solvers are applied to simple problems, their principles are being **gradually understood**

4. Benefits of the **equation-based modelling** are explained in the last stage based on experience

Form of evaluation

1. Student project, with a goal of simulating, analyzing and visualizing analogy between mechanical and electrical oscillators in Matlab and Modelica

2. Code piece, which requires a slight stretch beyond the program. Students are expected to improvise at the exam based on what they have learned in the course.

Learning process

Stage 1

LEARN TO HANDLE DATA

Covered topics

1. function, if, for, while
2. data, including vectors and matrices
3. import/export, fitting, plotting

Matlab Grader

1. **Automatic evaluation**

2. Evaluation speed: < 1 min/student/problem

3. Individual approach combined with self-study

4. Gradual transfer from easy to hard assignments

Example: importing from Modelica

In Modelica:

In Matlab:

exportFromModelica.m

```
%% Import and plot data from modelica
clear, close all
data=dlmread('simple.csv',',',1,0);
t=data(:,1);
v=data(:,2);
i=data(:,3);
subplot(1,2,1)
plot(t, v, 'ko', 'markersize',1.5, 'LineWidth',2)
xlabel('time, s'), ylabel('voltage, V')
subplot(1,2,2)
plot(t, v, 'ko', 'markersize',1.5, 'LineWidth',2)
xlabel('time, s'), ylabel('current, A')
```

Example: explain code at the exam

Code_piece_21.m

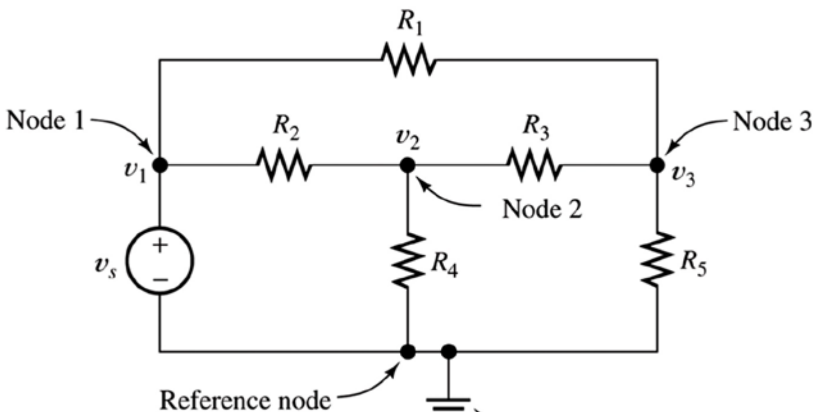
```
clc,clear,close
delete('test.m') %delete the script file called test.m, if it existed
edit('test.m')
fid=fopen('test.m','w');
fprintf(fid,'function [XleY, Xnew] = test(X,Y)\nXleY= X<=Y;\nXnew=X(XleY);\nend');

X =[3,4,5,7]';
Y =[-1,5,6,0]';

[XleY, Xnew]=test(X,Y);
```

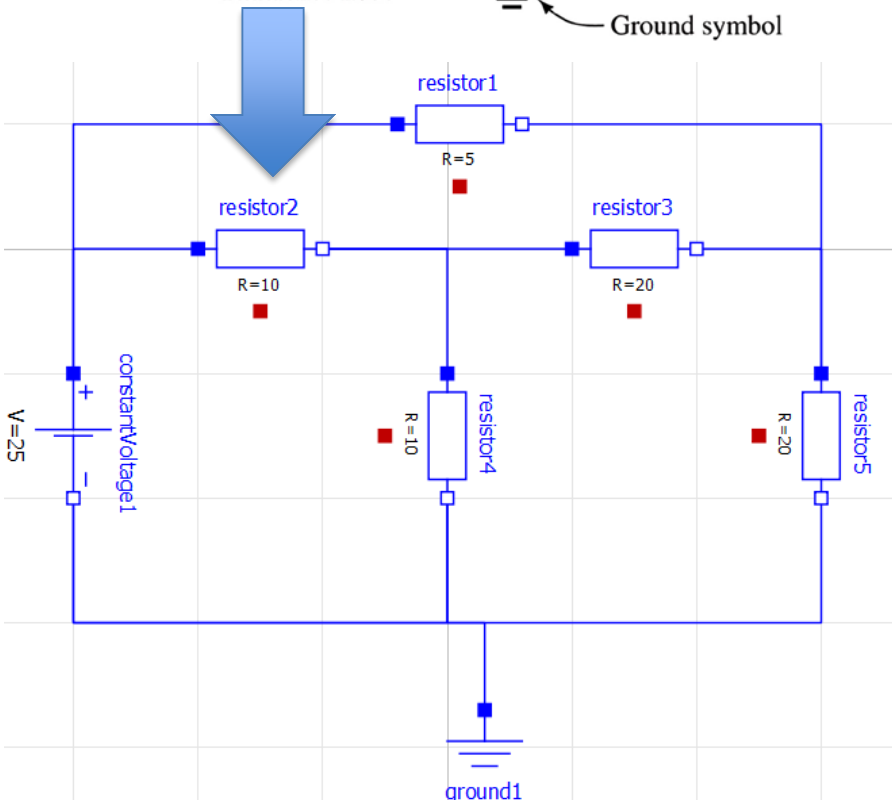
Stage 2

PHYSICAL SYSTEM -> MATHEMATICAL MODEL

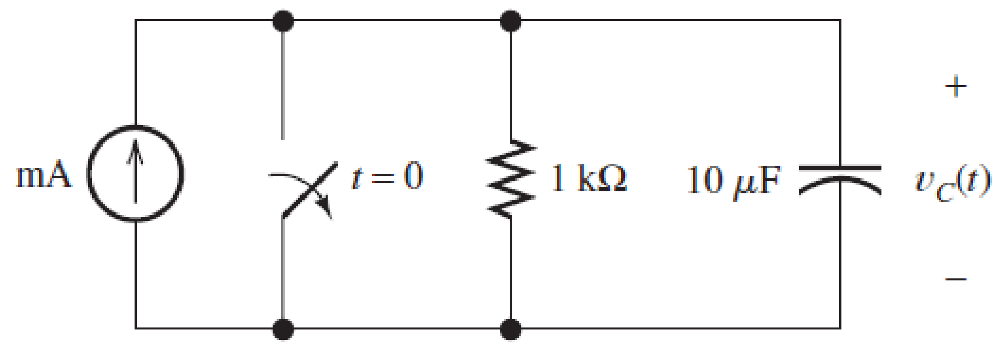
Modelica as a visualization tool

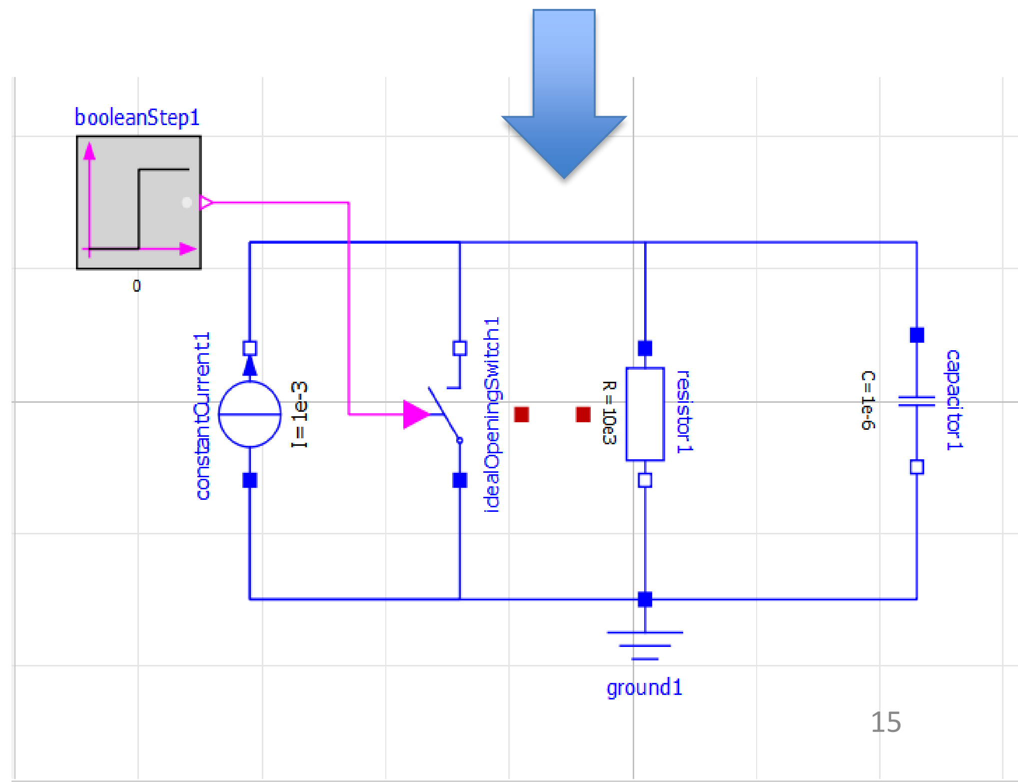
Static system





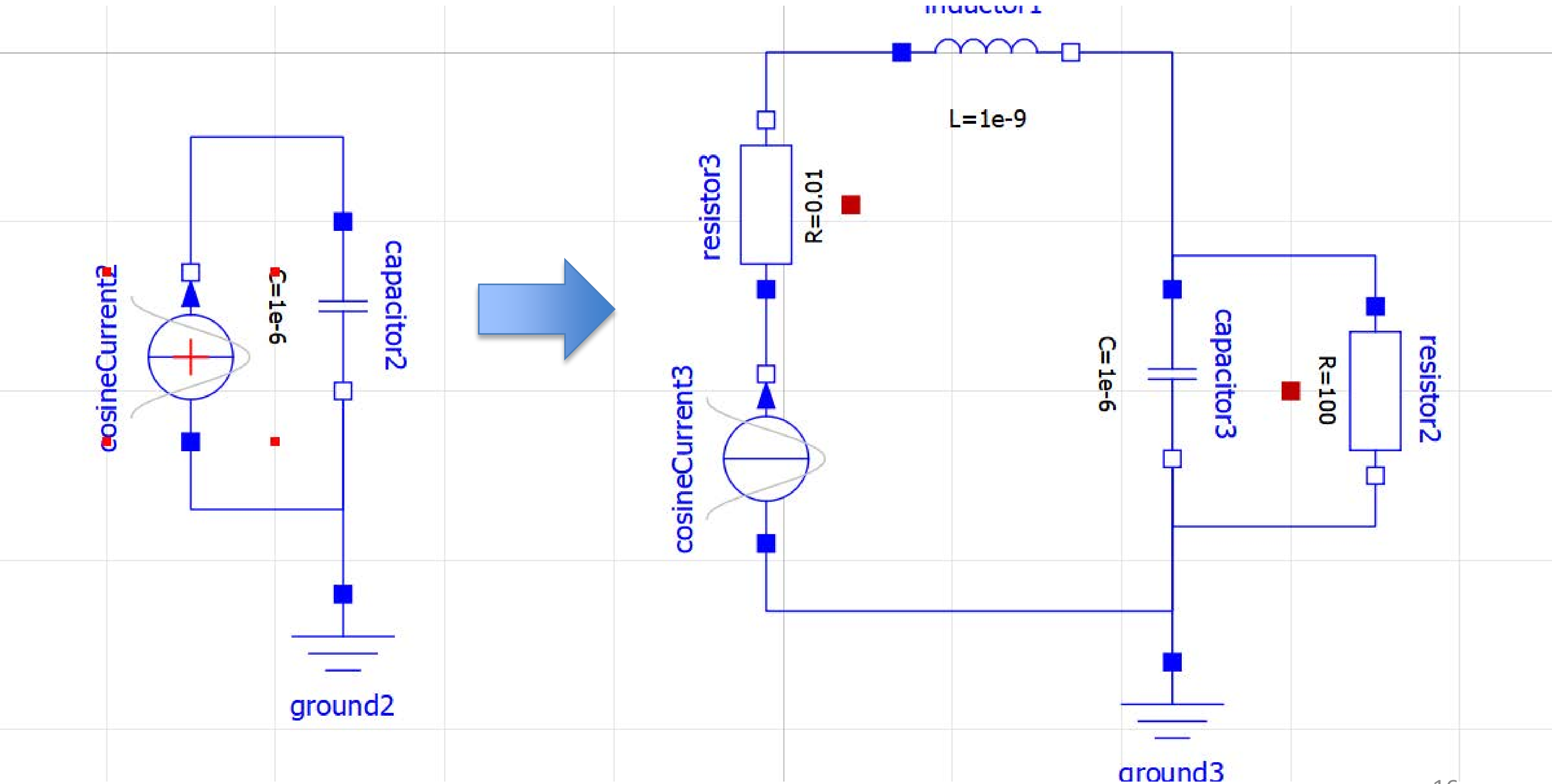
Dynamic system



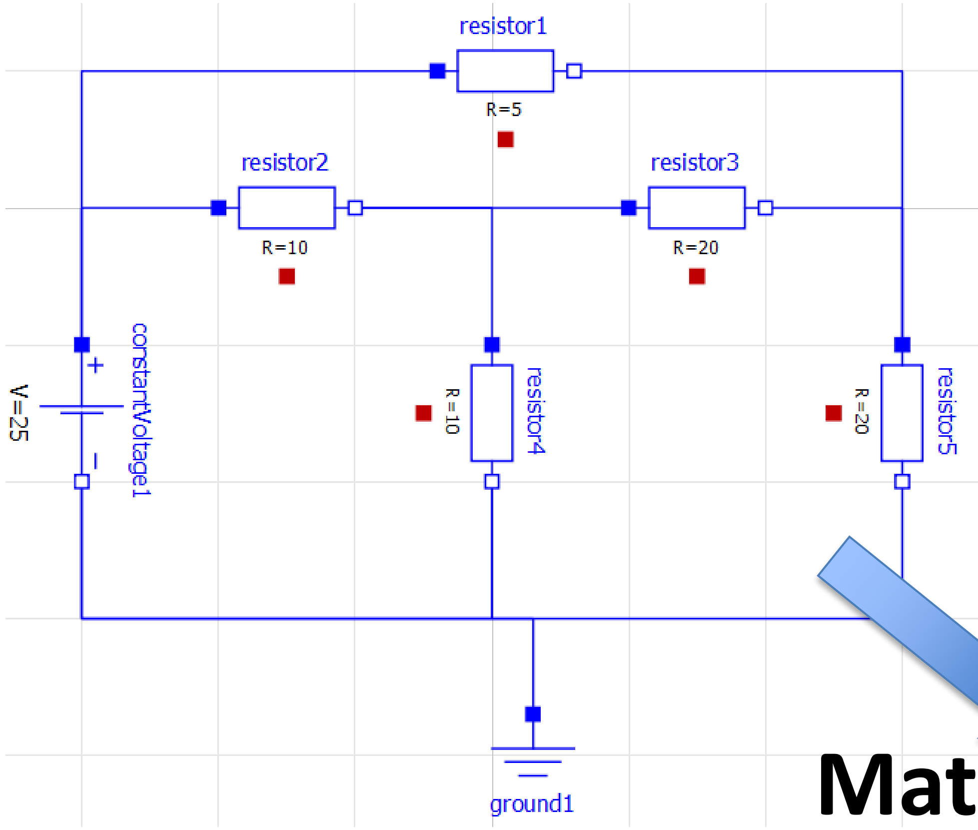


Modelica allows to vary complexity

Ideal capacitor **Real capacitor**



Mathematical model for a static problem

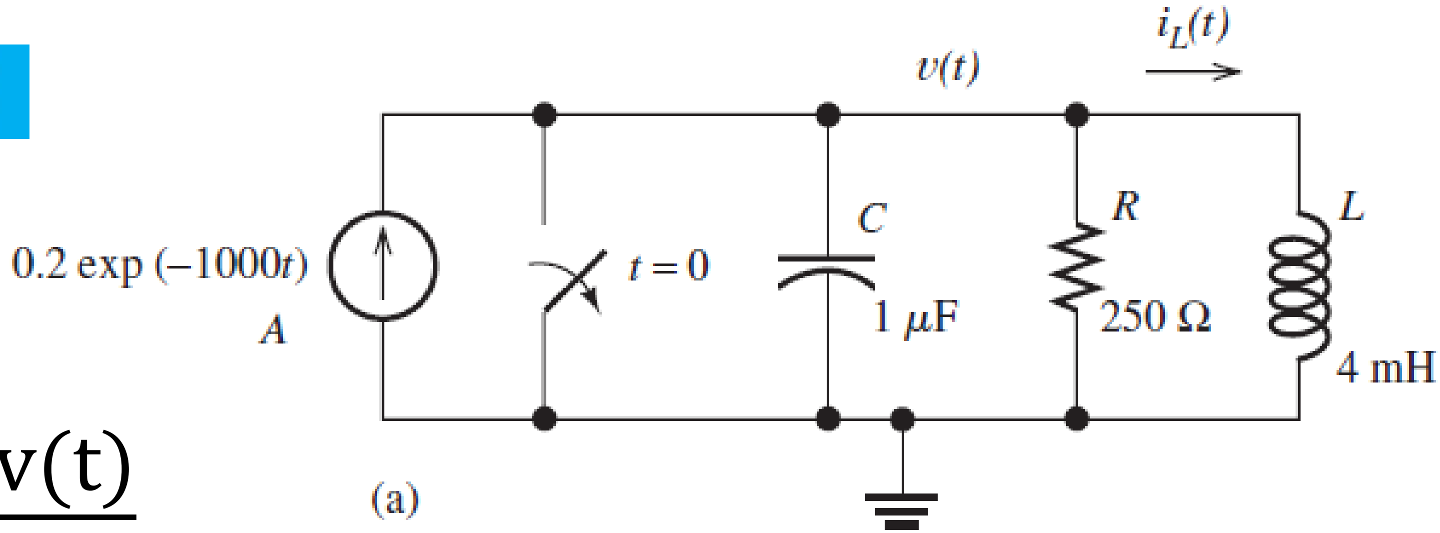


Matrix equation

$$\underbrace{\begin{pmatrix} \frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4} & -\frac{1}{R_3} \\ -\frac{1}{R_3} & \frac{1}{R_1}+\frac{1}{R_3}+\frac{1}{R_5} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} v_2 \\ v_3 \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} \frac{v_s}{R_2} \\ \frac{v_s}{R_1} \end{pmatrix}}_{\mathbf{b}}$$

"Recognizing" math in dynamic systems

Example 4.7



(a)

resistor: $i_R = \dfrac{v(t)}{R}$

inductor: $i_L = \int_0^t v(t)\, dt \,/L + i_L(0)$

capacitor: $i_C = C\dfrac{dv}{dt}$

source: $i = a \exp\left(-\dfrac{t}{\tau}\right)$ → $f(t) = \dfrac{1}{C}\dfrac{di}{dt} = -\dfrac{a}{C\tau} e^{-\frac{t}{\tau}}$

KCL: $i_R + i_C - i_S = 0$ → $C\dfrac{d^2v}{dt^2} = -\dfrac{1}{R}\dfrac{dv}{dt} - \dfrac{1}{L}v(t) + Cf(t)$

ODE as a matrix equation with dynamics

$$\frac{d^2v}{dt^2} = -\frac{1}{RC}\frac{dv}{dt} - \frac{1}{L}v(t) + f(t)$$

State-Space Model

$$x_1 = v(t)$$
$$x_2 = dv/dt$$

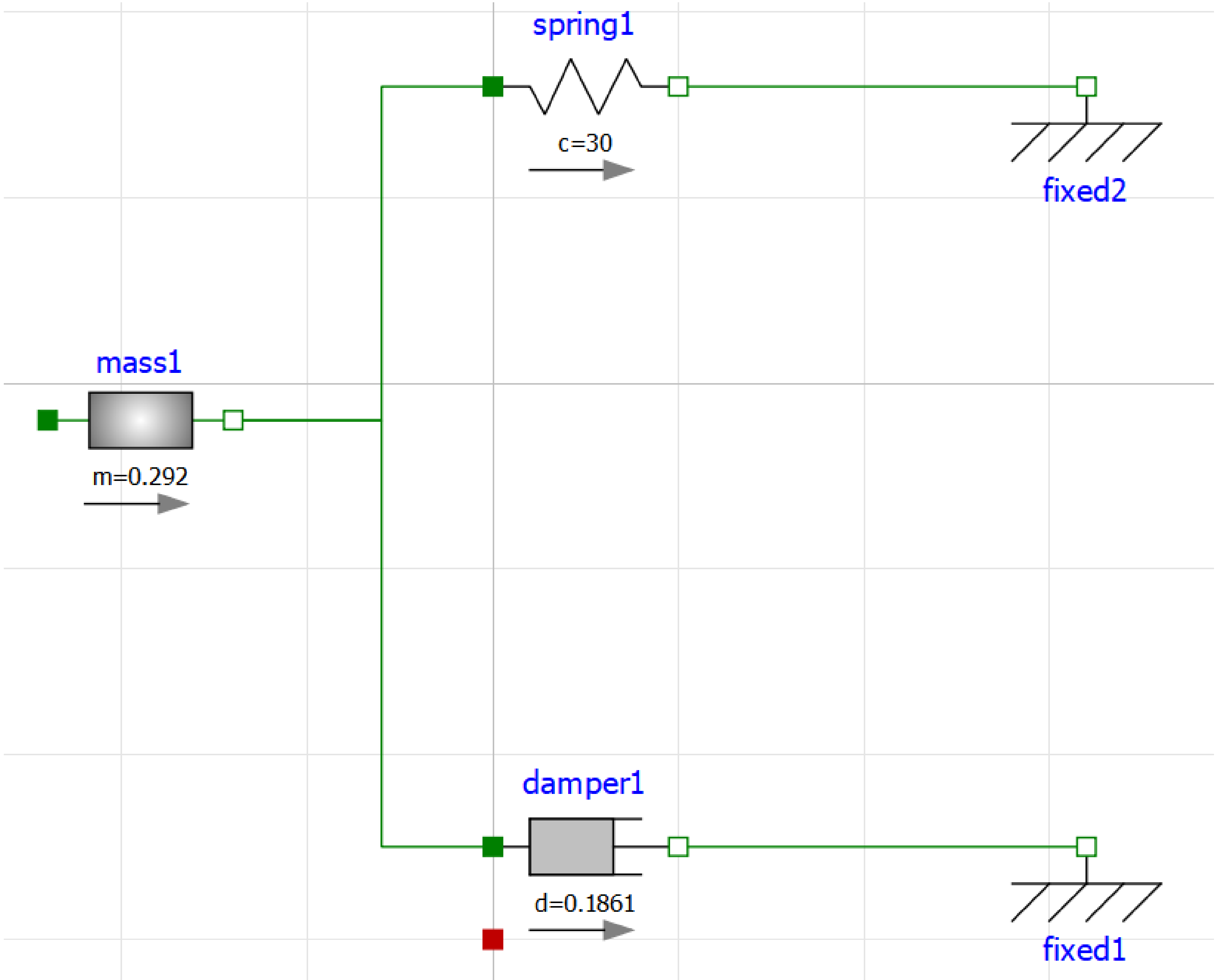
$$\underbrace{\begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix}}_{d\mathbf{x}/dt} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{1}{RC} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{\mathbf{x}} + \underbrace{\begin{pmatrix} 0 \\ f \end{pmatrix}}_{\mathbf{u}}$$

Initial conditions

$$C\frac{dv}{dt} + \frac{v(t)}{R} + \frac{1}{L}\int_0^t v(t)dt + i_L(0) = i$$

$$x(0) = 0$$
$$x'(0) = a/C$$

Different physics

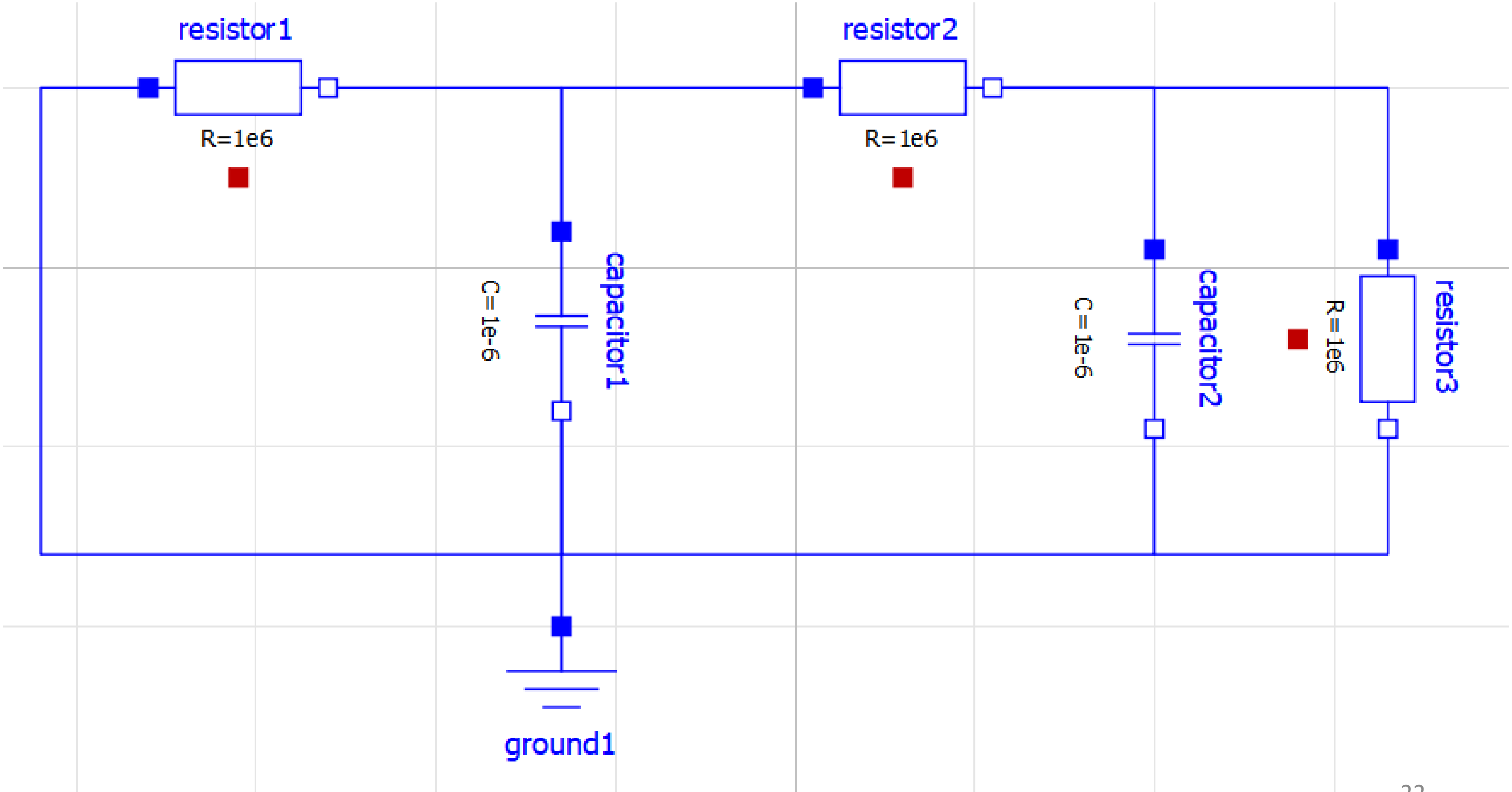


Stage 3

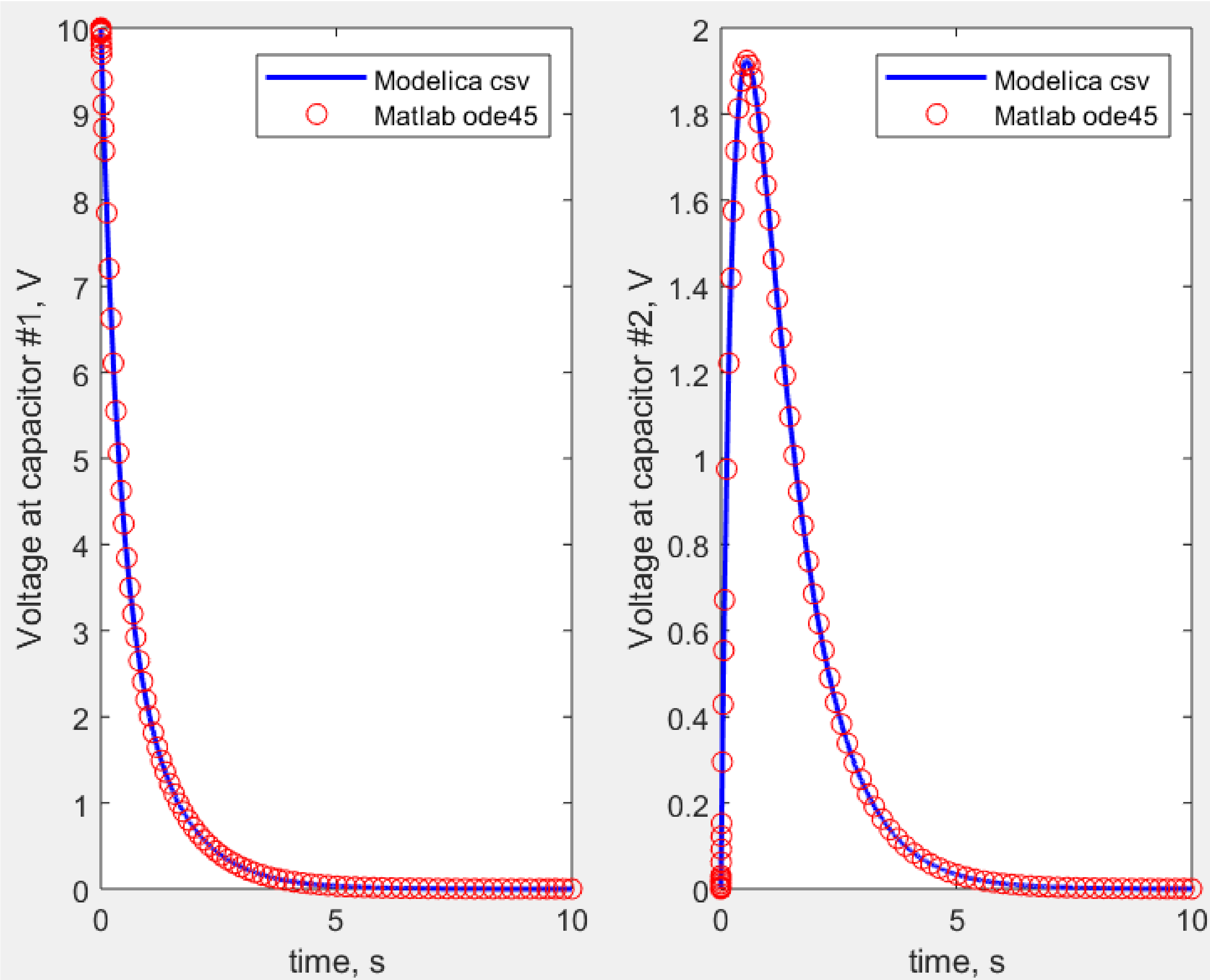
VERIFY AND VALIDATE

Model verification

Mandatory assignment: verify the model of the circuit below



Model verification



Model validation

Student work:

Grey-box estimation

Compare the predictions of mechanical oscillator dynamics with viscosity model $\mathbf{F} = \boldsymbol{bv} + \boldsymbol{kv}^2$ for cases:

1. **k=0**
2. **k≠0**

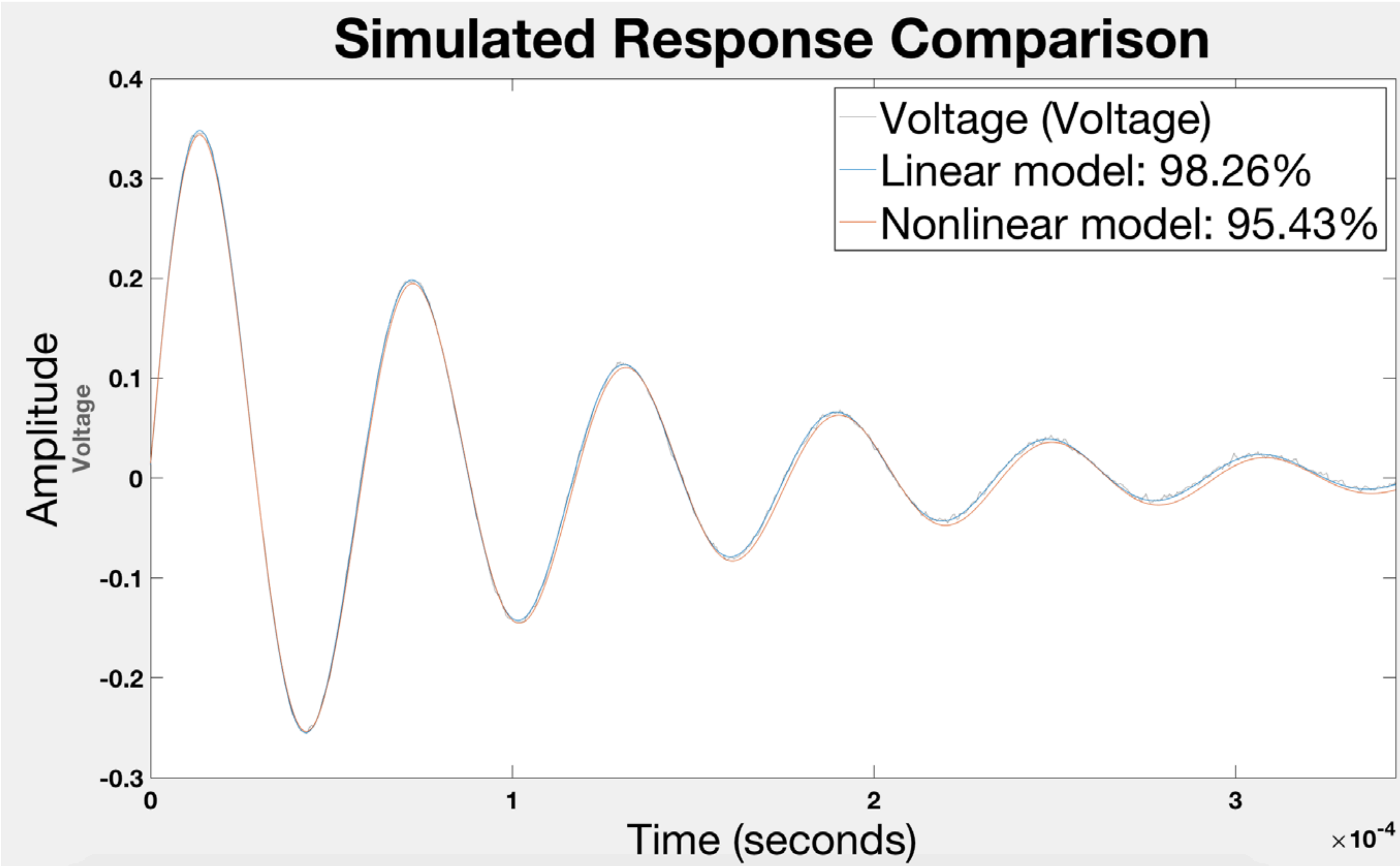
```
function [dx,y] = NonlinearPendulum1(t,x,u,m,w,l,b,varargin)

% Output equation.
y = x(1); % Angular position.

% State equations.
dx = [x(2);                                         ... % Angular position
        -w*x(1)-b*x(2)-l*x(2)^2 + m*u     ... % Angular velocity
       ];
end
```

State-space model

Model validation

Student work:

Grey-box estimation



Conclusion: account of quadratic term does not lead to better accuracy

Stage 4

UNDERSTAND

Why is Modelica “easier” than Matlab?

1. Because of the object-oriented approach

2. Because of equation-based approach

3. Because of annotations

Object-oriented modelling

1. Object in Modelica encapsulates data and code
2. Object represents physical entity, fex **Pin** or **Resistor**
3. Object is an instant of a Class (definition/template)
4. Inheritance => reusable components with clear structure
5. Polymorphism => exchange objects in a big model
6. Variables, parameters, constants

Object-oriented modelling

Pin (node) is explained as an object, which “knows” the common properties (i, v) of two objects to make their interaction possible,

Therefore, the corresponding class is usually implemented in the subpackage called “Interfaces”

TwoPin describes the properties of the electric field in the circuit (potentiality)

OnePort describes general relation between two nodes in the circuit

Resistor, **Capacitor**, **Inductor** give specific form for this relation

Equation-based modelling

1. In Matlab one needs to convert equations of the mathematical model into a sequence of assignment statements

2. In Modelica, equations can be written directly after the keyword equation in the code.

3. When using connect on the two pins of different objects they “become a single node”, which means that the Kirchhoff's Laws are automatically generated for this node

Annotations

1. Visualization of math and code is always hard work

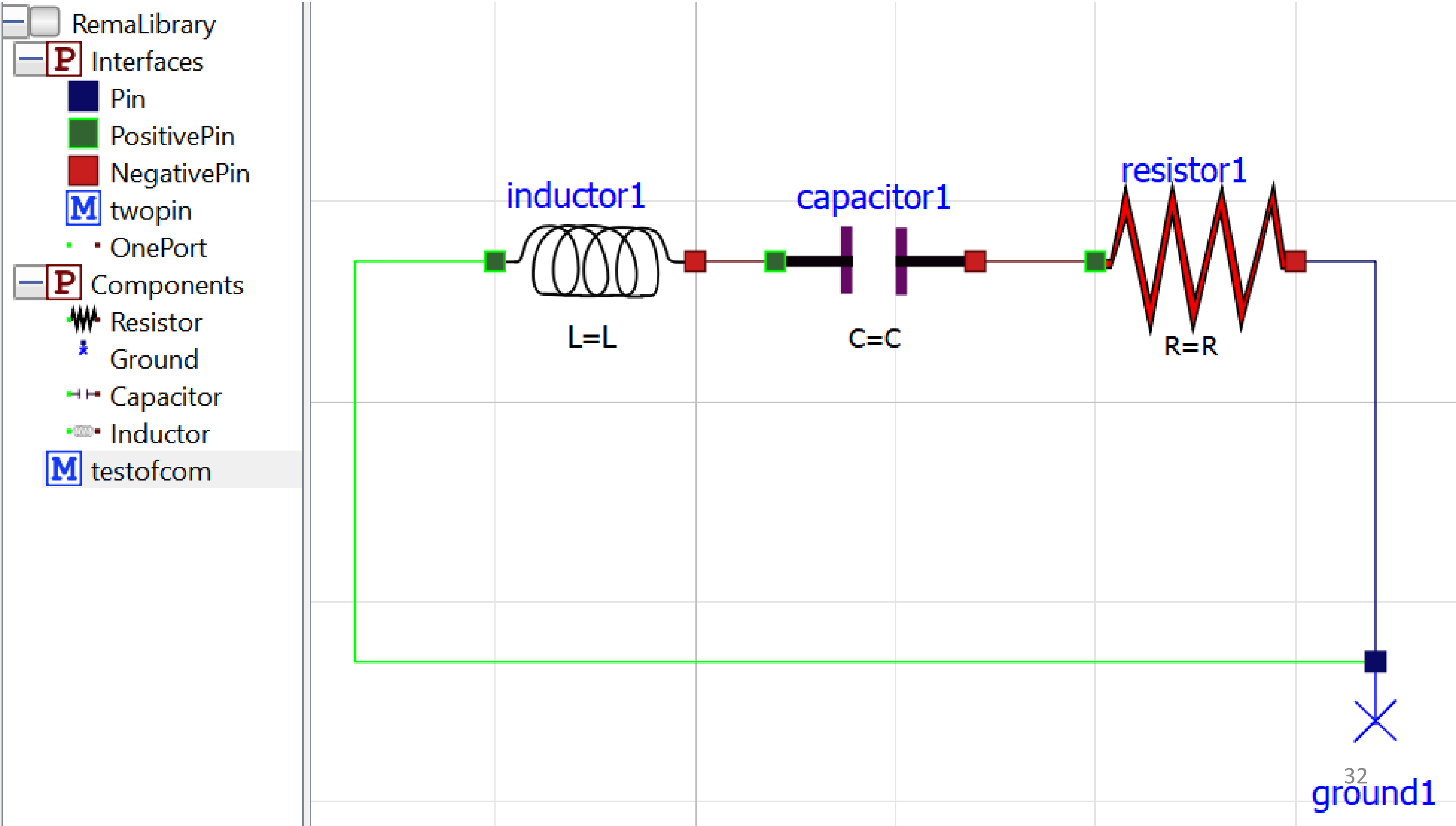
2. Annotations allow us to “draw” the physical meaning of objects

3. They help us explain parameters of the model

Assignment: implement the simplest electric package

Implementation

Student work: Library implementation

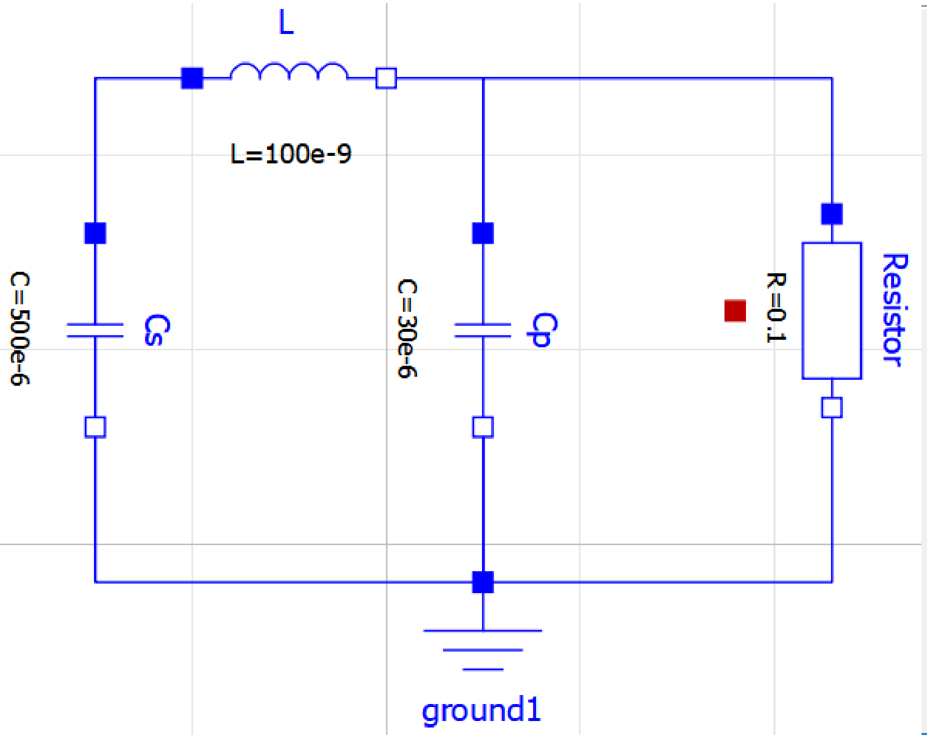


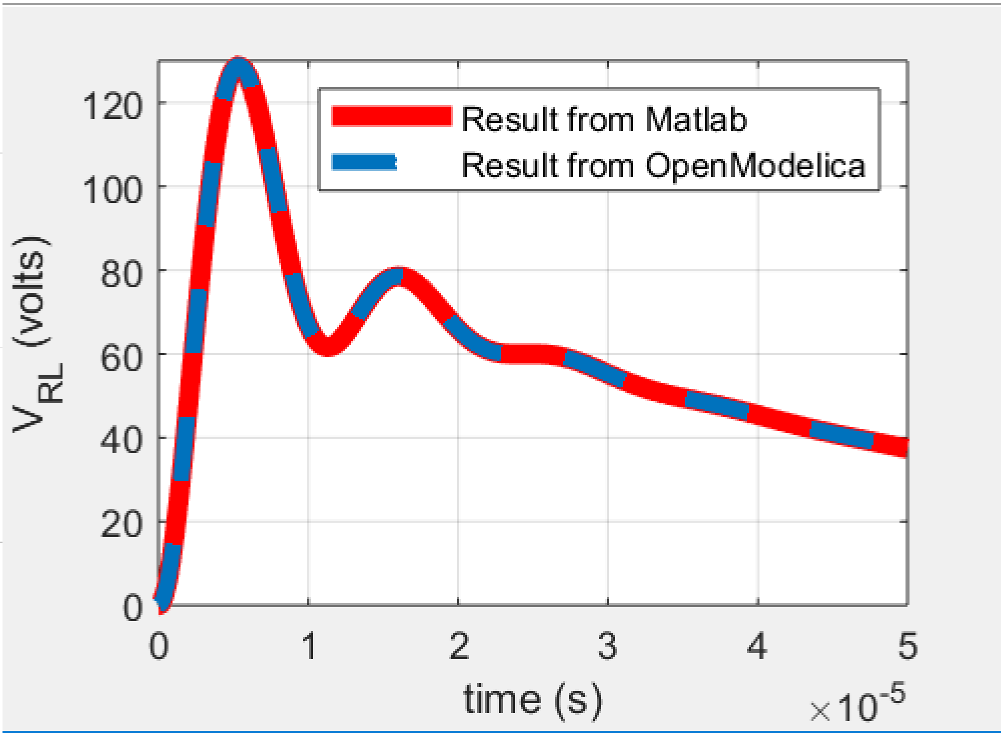
Stage 5

APPLY

Complex physics

Student work:
3^{rd} order circuit

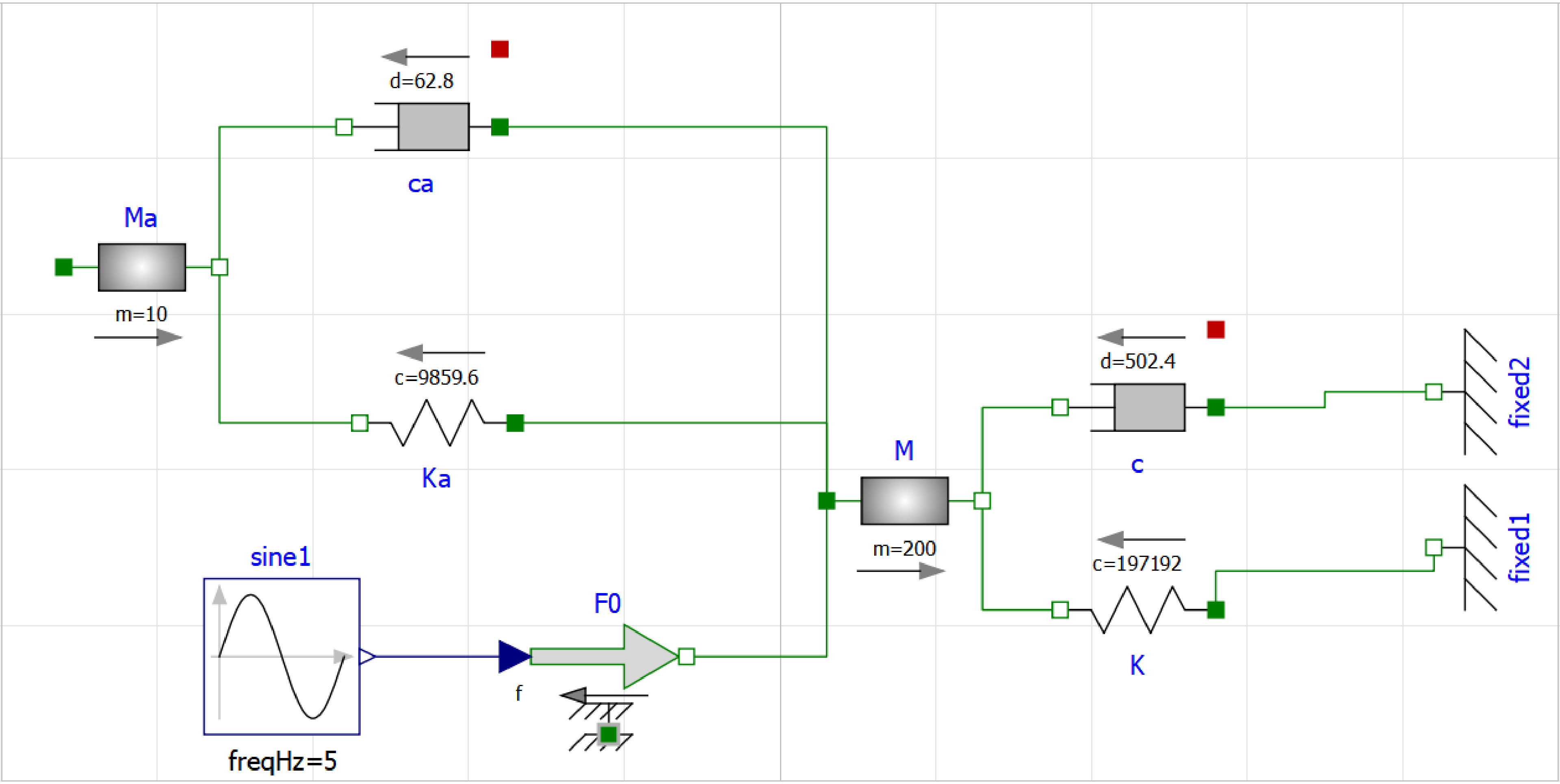




Complex physics

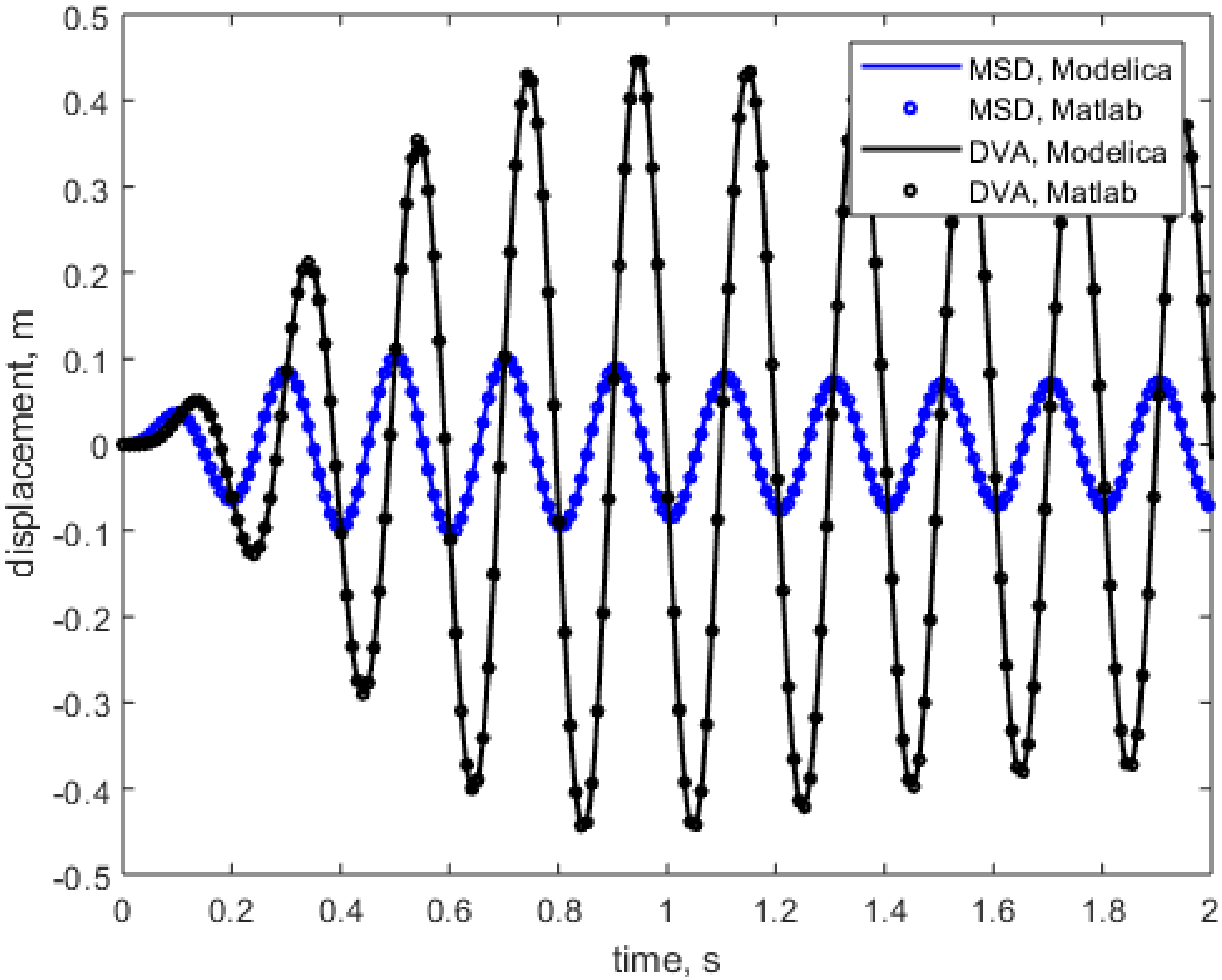
Student work:

Dynamic vibration absorber



Complex physics

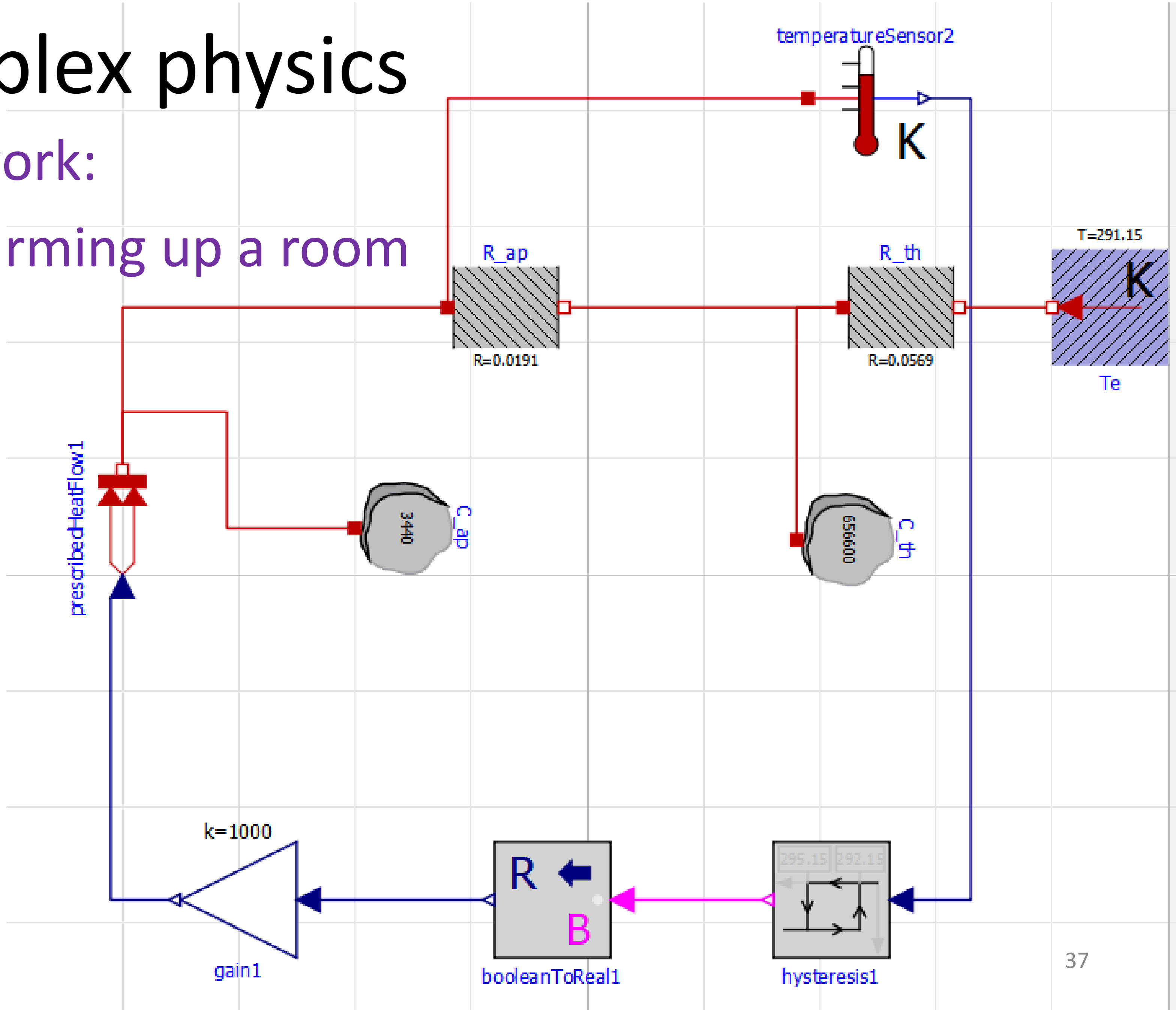
Student work: Dynamic vibration absorber



Complex physics

Student work:

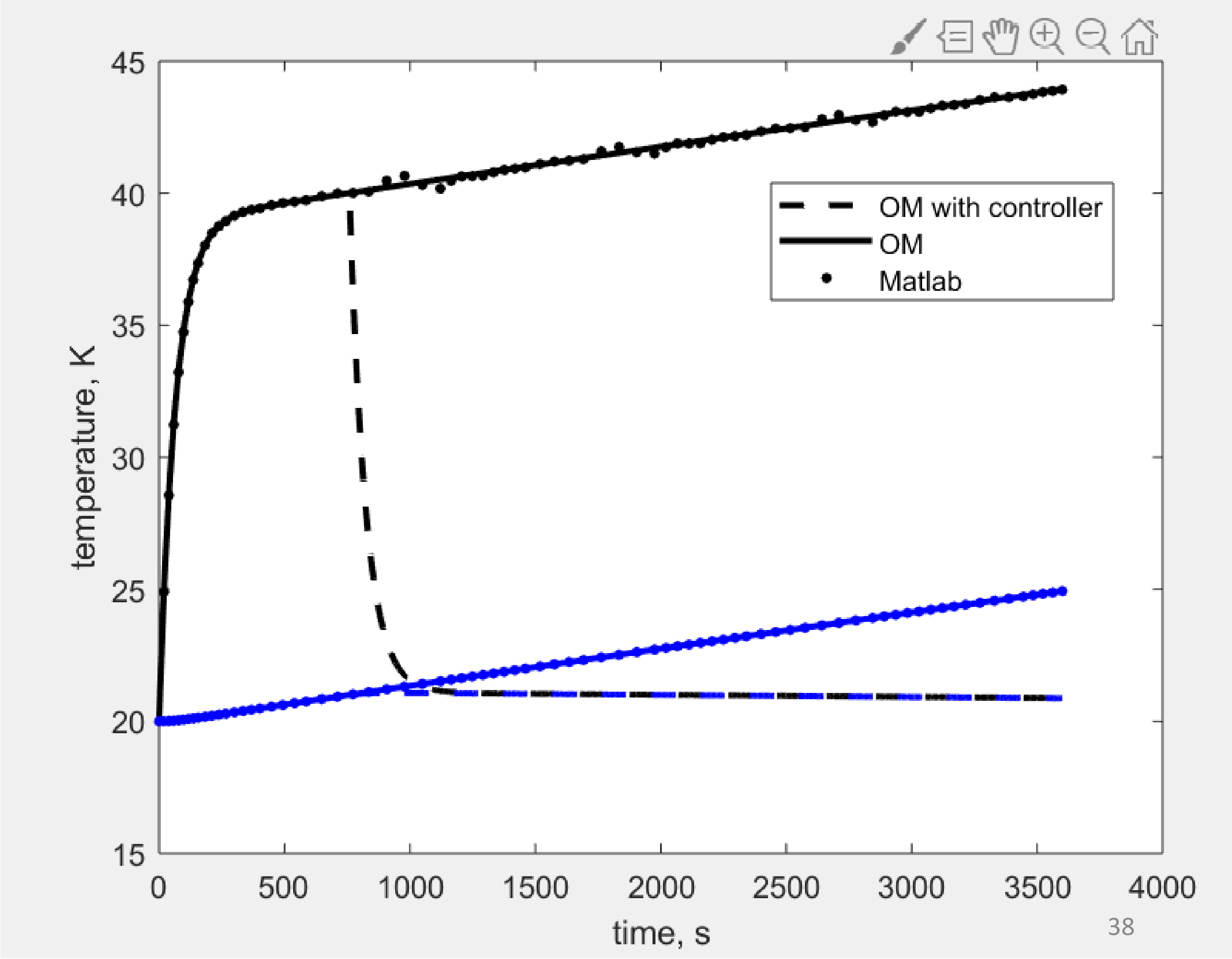
Heater warming up a room



Complex physics

Student work:

Heater warming up a room



Outcome

1. The course is hard, but doable for students
2. Many of those, who did not like the complexity during the semester, changed their minds after the course
3. Most students have shown a good grasp of material at the exam
4. All students have shown understanding of the interplay between physics, mathematics and programming

Closure

THANK YOU!