TTK4130 Modeling and Simulation

Lecture 1

TTK4130 Modeling and Simulation

<u>To learn:</u>

- Formulate mathematical models from first principles
- Simulate models using computer

Main purpose: Control system design/testing/validation,
hence *dynamic* modeling

Instructors:
- Lecturer: Leif Erik Andersson
- Teaching assistant: Michael Ernesto Lopez
  - 3 student assistants

Course Information

- All course information is provided on Blackboard. There will be no handout of material
- We will not cover the complete curriculum in our lectures; rather focus on the most important and/or difficult parts

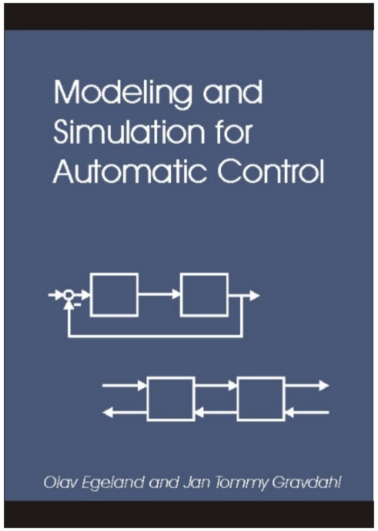
- Lectures:
  - Mondays 8:15-10:00 (KJL1)
  - Thursdays 10:15-12:00 (KJL1)

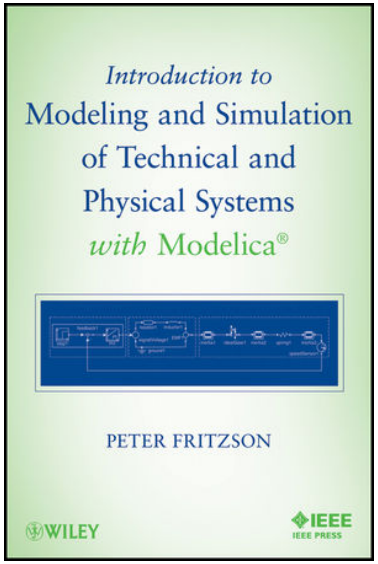
- Exam:
  - May 15th, 2019
  - Examination support code: A (Open book)

- Grading: Exam counts 100%

Syllabus

- Books:
  - "Modeling and Simulation for Automatic Control", 2002, by O.Egeland and J.T. Gravdahl (ISBN 82-92356-00-2) **(E)**
    - Errata in Blackboard
    - Download: https://www.researchgate.net/publication/256492530_Modeling_and_Simulation_for_Automatic_Control
  - "Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica, 2011, by Peter Fritzson **(F)**





Lecture schedule

Week	Date	Theme	Literature
2	07.01	Introduction to Modelica	F: 1, 2
	10.01	More introduction. State-space models, transfer functions. Modeling software, network models.	E: 1.1-1.3, 2.1-2.2 (E:1.4-1.5)
3	14.01	Energy functions, passivity	E: 2.3-2.4
	17.01	More passivity	E: 2.4
4	21.01	Modeling of complex systems. Simulation: Order, test system	F: 3, 4, E: 14.1-14.2
	24.01	Explicit Runge-Kutta methods	E: 14.3-14.4
5	28.01	*Guest lecture – Sebastien Gros (exam relevant)*	-confirmed -
	31.01	Implicit Runge-Kutta methods	E: 14.5
6	04.02	Stability, Padé approximations	E: 14.6
	07.02	Stability, frequency properties, automatic step size adjustment Implementation, BDF and differential-algebraic systems	E: 14.6-14.7 E: 14.8, 14.11, 14.12
7	11.02	Vectors, dyadics, rotation matrices	E: 6.1-6.4
	14.02	Euler angles, angle axis	E: 6.5-6.6
8	18.02	Euler parameters, angular velocities	E: 6.7-6.8
	21.02	Kinematic differential equations	E: 6.9
9	25.02	Kinematics of a rigid body, Newton-Euler equations of motion	E: 6.12-6.13, 7.3
	28.02	Newton-Euler equations of motion, Modelica.Multibody	E: 7.3
10	04.03	Friction	E: 5
	07.03	Electrical motors	E: 3.1-3.4
11	11.03	Lagrange equations of motion	E: 7.7, 8.1-8.2
	14.03	Lagrange equations of motion, recap, examples	E: 7.7, 8.1-8.2
12	18.03	Process modelling and balance laws, I	E: 10.4, 11.1-4 (+ slides)
	21.03	Process modelling and balance laws, II	E: 10.4, 11.1-4 (+ slides)
13	25.03	Hydraulic motors, transmission lines	E: 4.1-4.6
	28.03	Process modelling and balance laws (closure relations)	E: 10.4, 11.1-4
14	01.04	Process modelling and balance laws (differential balance)	E: 10.4, 11.1-4
	04.04	*Guest lecture: Erlend Kristiansen, Comsol Multiphysics*	-confirmed -
15	08.04	No lecture (excursion)	
	11.04	No lecture (excursion)	
16	15.04	No lecture (excursion)	
	18.04	No lecture (Easter)	
17	22.04	No lecture (Easter)	
	25.04	Recap	
18	29.04	Recap	
	02.05	Discussion - possible topics: past Exams	

E: “Modeling and Simulation for Automatic Control” by O. Egeland and J.T. Gravdahl
F: “Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica” by P. Fritzon

Course information, assignments

- A minimum number of exercises (8 of 11) must be approved to enter the final examination
- The deadlines for all assignments are absolute

Exercise Schedule

Assignment	Out	In
1	January 14 at 8:00	January 27 at 20:00
2	January 21 at 8:00	February 3 at 20:00
3	January 28 at 8:00	February 10 at 20:00
4	February 4 at 8:00	February 17 at 20:00
5	February 11 at 8:00	February 24 at 20:00
6	February 18 at 8:00	March 3 at 20:00
7	February 25 at 8:00	March 10 at 20:00
8	March 4 at 8:00	March 17 at 20:00
9	March 11 at 8:00	March 24 at 20:00
10	March 18 at 8:00	March 31 at 20:00
11	March 25 at 8:00	April 7 at 20:00

Exercise class:

Week	Assignment(s) supported in guidance hours
3	1
4	1, 2
5	2, 3
6	3, 4
7	4, 5
8	5, 6
9	6, 7
10	7, 8
11	8, 9
12	9, 10
13	10, 11
14	11

Course information, assignments

- A minimum number of exercises (8 of 11) must be approved to enter the final examination
- The deadlines for all assignments are absolute
  - Schedule:
    - Out: Latest Mondays
    - In: Sundays
  - Four exercise hours (you have to sign up for one!):
    - Mondays 17:15 – 19:00 (EL2)
    - Tuesdays 16:15 – 18:00 (G116, G118, G122)
    - Wednesdays 17:15 – 18:00 (EL2)
    - Thursdays 16:15 – 18:00 (G116, G118, G122)
  - Several exercises require computers
    - Matlab/Simuling & Dymola
    - Computer lab G116, G118, G122 provides computers, EL2 bring your Laptop!

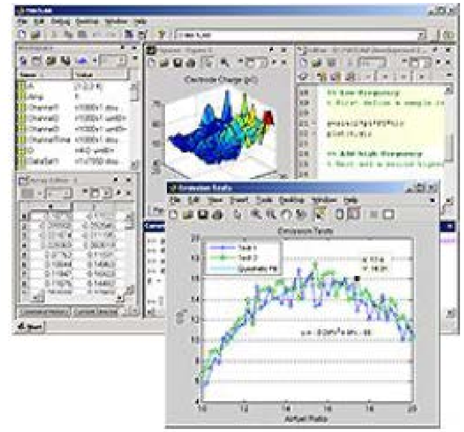
Blackboard – Forum (under “Course work”)

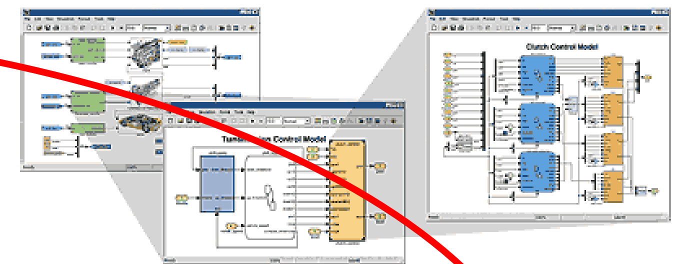
- Use the forum for questions etc.
- Create well-named Thread:
  - for example:
    - Exam 2016
    - Exercises 2 Task 3
    - Lectures 5

- Advantageous:
  - Probably not only you have the questions, so it helps others and we only have to answer the questions once.
  - Also other students can answer, for example to questions about the exercises. Therefore, you might get faster an answer.

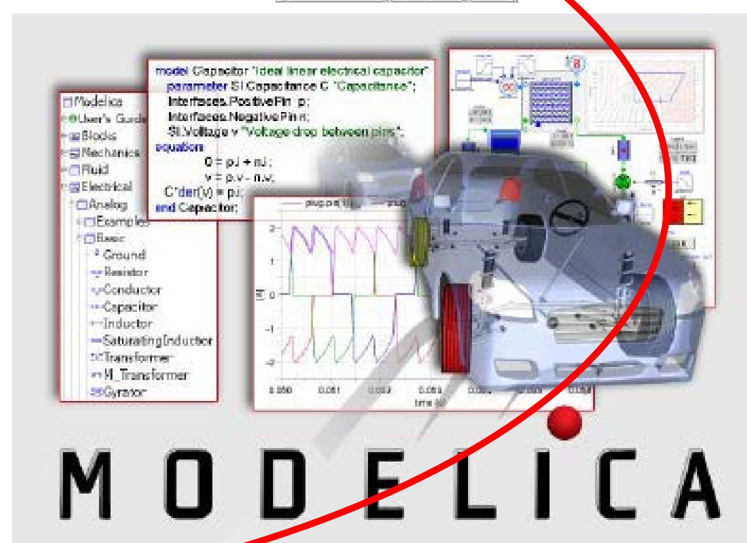
Software

- Matlab/Simulink
  - Computer labs
  - Your own computer
  - Some familiarity with Matlab/Simulink is assumed





- Dymola
  - Based on the Modelica modelling language
  - Installed on computer labs, but you can also use your PCs (Windows and Linux)
  - License:
    - License sever (via VPN), but limited number of licenses (75)
    - Much can be done with demo license



Introduction to Modelica

Slides taken/adapted from Peter Fritzson

Dymola on your own computer

- Dymola download: The Windows and Linux version of Dymola can be downloaded from \\itk-stud01.win.ntnu.no\progdist\Dymola, requires connection to the NTNU VPN

- To be able to simulate models, you must have a C++ compiler installed, e.g. C++ express from http://www.microsoft.com/express/windows (did not work for me) or MinGW-w64 on https://www.sourceforge.net/projects/mingw-w64/
  – Choose simulation tab and choose Simulation – Setup – Compiler
  – Choose custom and add the complete path of the “VC” directory of express or add the path to “gcc” for MinGW-w64
  – Press verify to test

- Dymola license: license server is kyb-lisens01.itk.ntnu.no
  – Copy address in “server name” under Help -- License -- Setup

Modelica for Mac

1. Install Windows on a parallel machine like:
   - Parallels (http://www.parallels.com/eu/products/desktop/)
   - VirtualBox (https://www.virtualbox.org/)

2. Install OpenModelica which has a Mac version
   - https://www.openmodelica.org/download/download-mac

3. There is also the option to download Virtual Machine on the Open Modelica webpage, which explains installation steps etc.
   - https://www.youtube.com/watch?v=11OkQs8VUrU (Youtube video on the installation with VirtualBox + OpenModelica)

4. Install Windows on partition with Boot Camp:
   - You can install Dymola and Visual Studio on the partition.

History

- Development from autumn 1996 through person from industry and academics

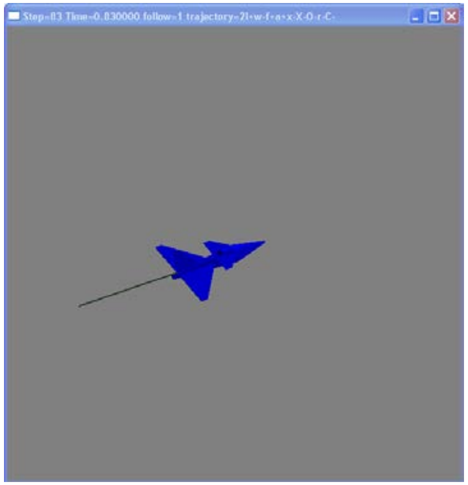
- Modelica 1.0 in September 1997

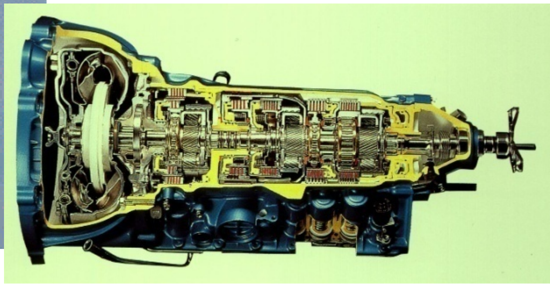
- At the moment: Modelica 3.4 (2017)

What is Modelica?

A language for modeling of complex physical systems

- Robotics
- Automotive
- Aircrafts
- Satellites
- Process systems
- Power plants
- Systems biology

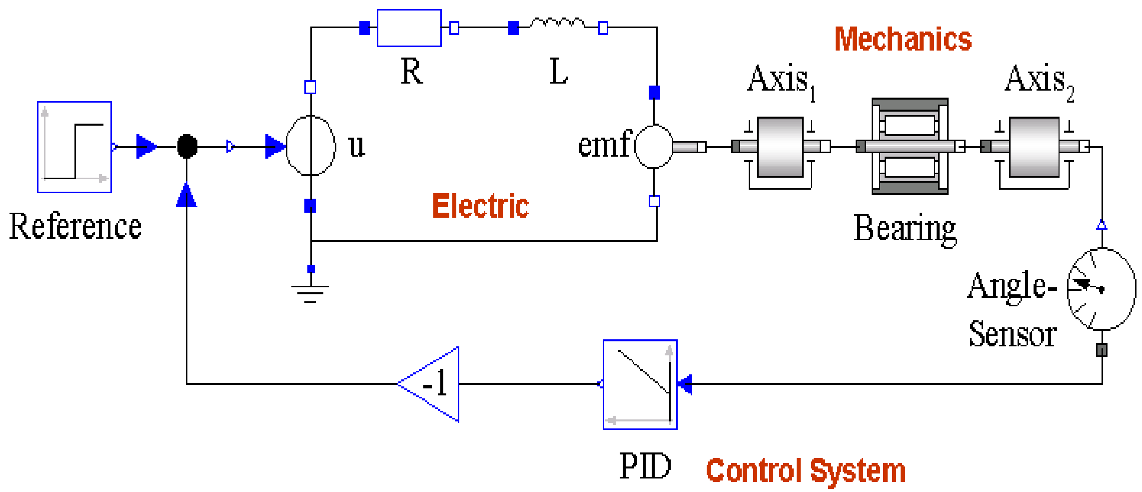




What is Modelica?

What is Modelica?

A language for modeling of complex physical systems



Primary designed for simulation, but there are also other usages of models, e.g. optimization.

What is Modelica?

A language for modeling of complex physical systems

i.e., Modelica is not a tool

Free, open language specification:

Available at: http://www.modelica.org/

There exist several free and commercial tools, for example:

- OpenModelica from OSMC
- Dymola by Dassault systems / Dynasim
- SimulationX by ITI
- MapleSim by MapleSoft
- SystemModeler by Wolfram

Modelica technology

- Modelica is primarily an *equation-based* language

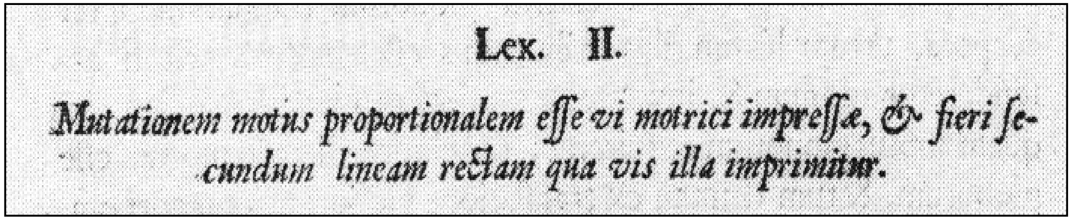
The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557

The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557

Newton still wrote text (Principia, vol. 1, 1686)



Lex. II.

*Mutationem motus proportionalem esse vi motrici impressæ, & fieri se-
cundum lineam rectam qua vis illa imprimitur.*

"The change of motion is proportional to the motive force impressed"

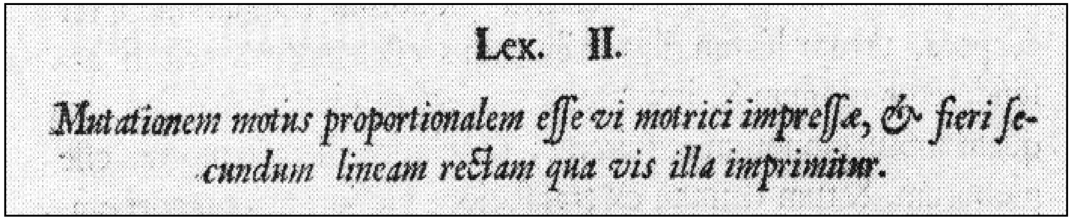
or

$$m \cdot a = \sum F$$

The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557

Newton still wrote text (Principia, vol. 1, 1686)



Lex. II.

Mutationem motus proportionalem esse vi motrici impressæ, & fieri secundum lineam rectam qua vis illa imprimitur.

Programming languages usually do not allow equations!

Do other programming languages not also use equations?

- Difference between *equation* and *assignment*

Do other programming languages not also use equations?

- Difference between *equation* and *assignment*

- Equations do not prescribe a certain data flow direction a execution order

Equation	Assignment
$R * i = v;$	$i := v/R;$ $v := R * i;$ $R := v/i;$

Do other programming languages not also use equations?

- Difference between *equation* and *assignment*

- Equations do not prescribe a certain data flow direction a execution order

Equation	Assignment
$R * i = v;$ Causality unspecified	$i := v/R;$ $v := R * i;$ $R := v/i;$ Output ⟸ Input

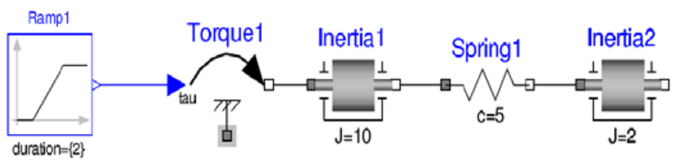
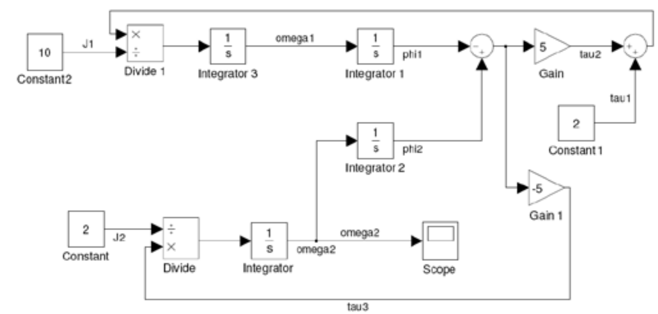
- → Acausal modeling

Advantages of Equations

- More flexible than assignments
- Key to physical modeling capabilities
- Increases reuse potential of Modelica classes/models

Acausal modeling

The order of computations is not decided at modeling time

	Acausal	Causal
Visual Component Level		

Equation Level	A resistor *equation*:
	`R*i = v;`
		Causal possibilities:
		`i := v/R;`
		`v := R*i;`
		`R := v/i;`

Example – Simple differential equation

- Equation (ODE):
  - $\frac{dx}{dt} = ax \; ; \; x(0) = 1;$

Example – Simple differential equation

Equation (ODE):

$$\frac{dx}{dt} = -x, \quad x(0) = 1$$

Name of model

Initial condition

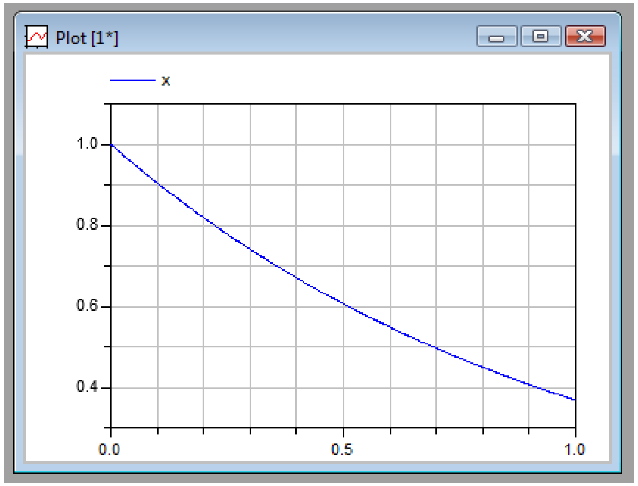
Continuous-time variable

Parameter, constant during simulation

```
model FirstOrder "A simple equation"
  Real x(start=1);
  parameter Real a = -1;
equation
  der(x)= a*x;
end FirstOrder;
```

Differential equation

Simulation in Dymola:



Modelica Variables and Constants

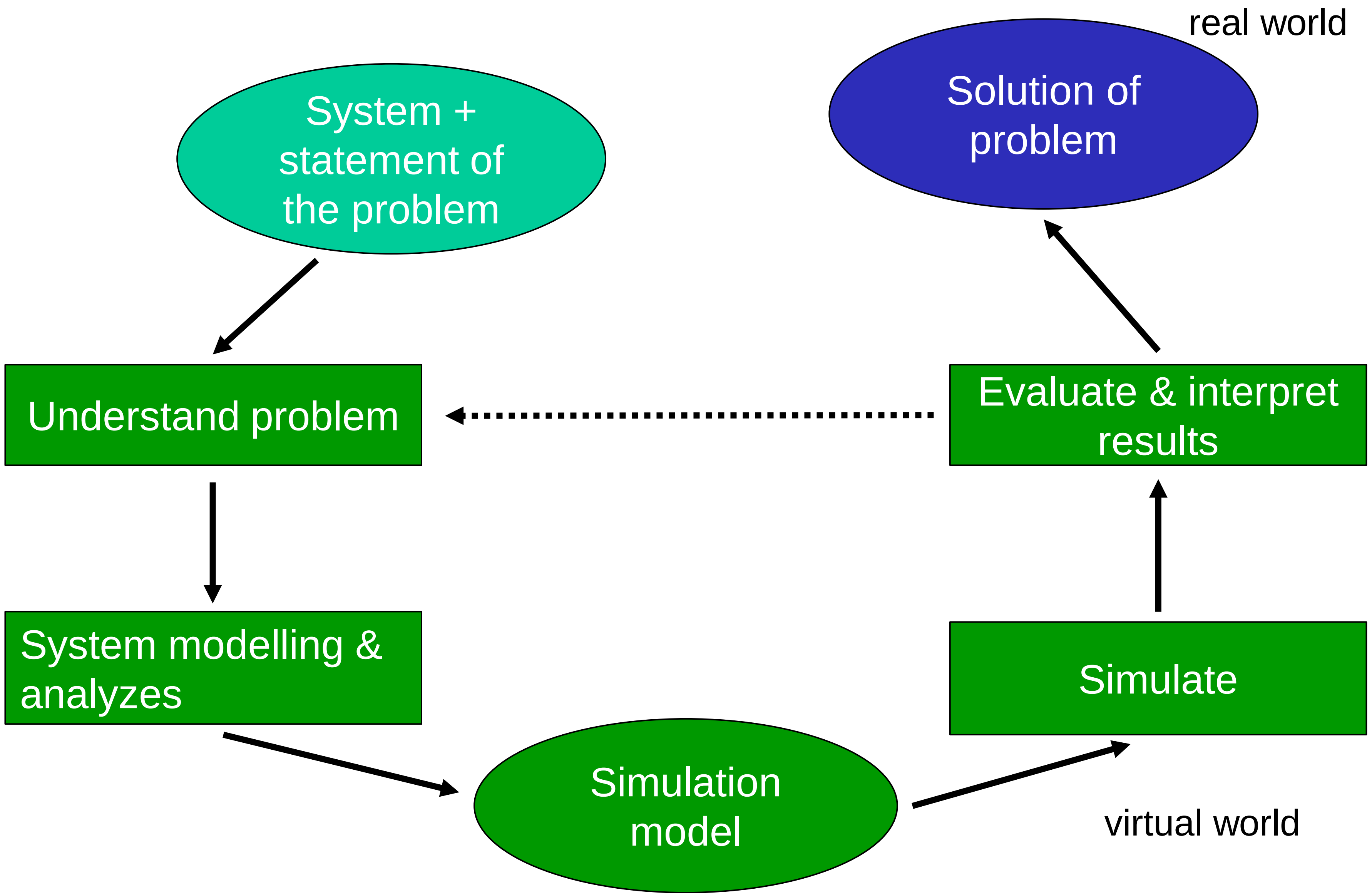
- Built-in primitive data types

Boolean	**true** or **false**
Integer	Integer value, e.g. **42** or **–3**
Real	Floating point value, e.g. **2.4e-6**
String	String, e.g. **“Hello world”**
Enumeration	Enumeration literal e.g. **ShirtSize.Medium**

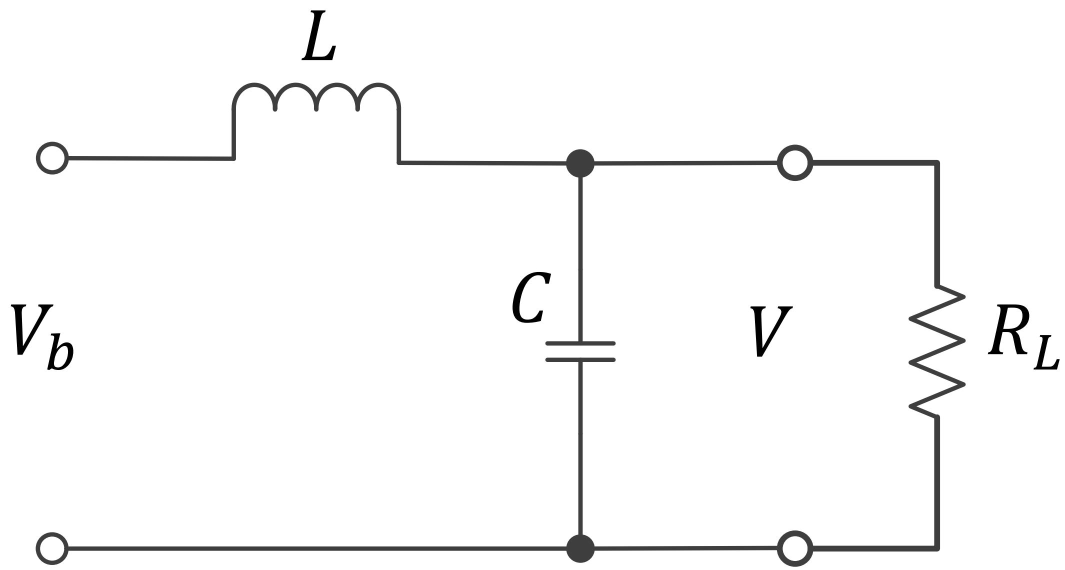
- Parameters are constant during simulation
- Two types of constants in Modelica
  - `constant`
  - `parameter`

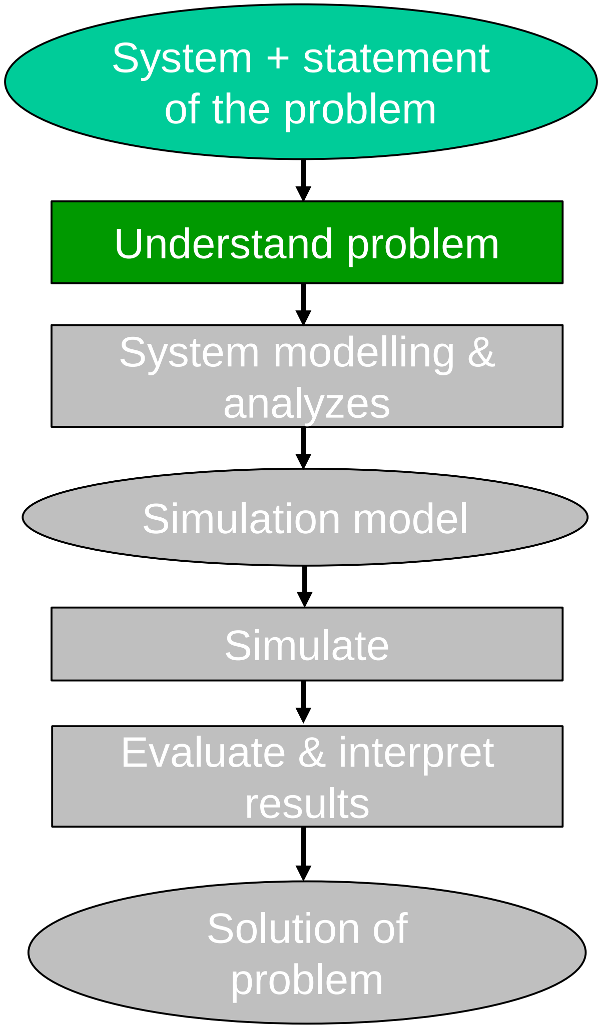
```
constant  Real    PI=3.141592653589793;
constant  String  redcolor = "red";
constant  Integer one = 1;
parameter Real    mass = 22.5;
```

Modelling and Simulation Process



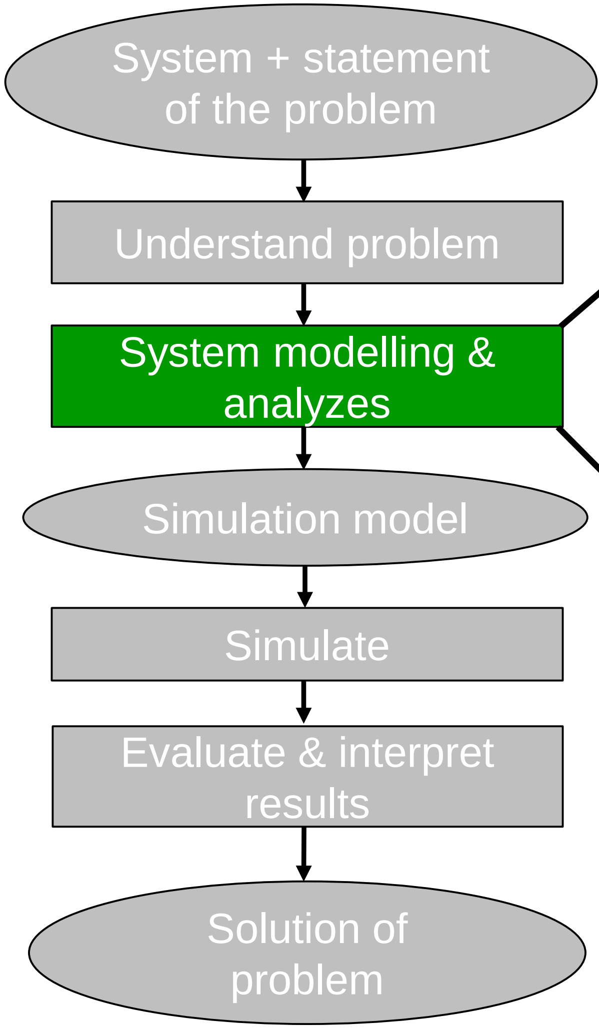
2^{nd} Example – Low-Pass RLC Filter

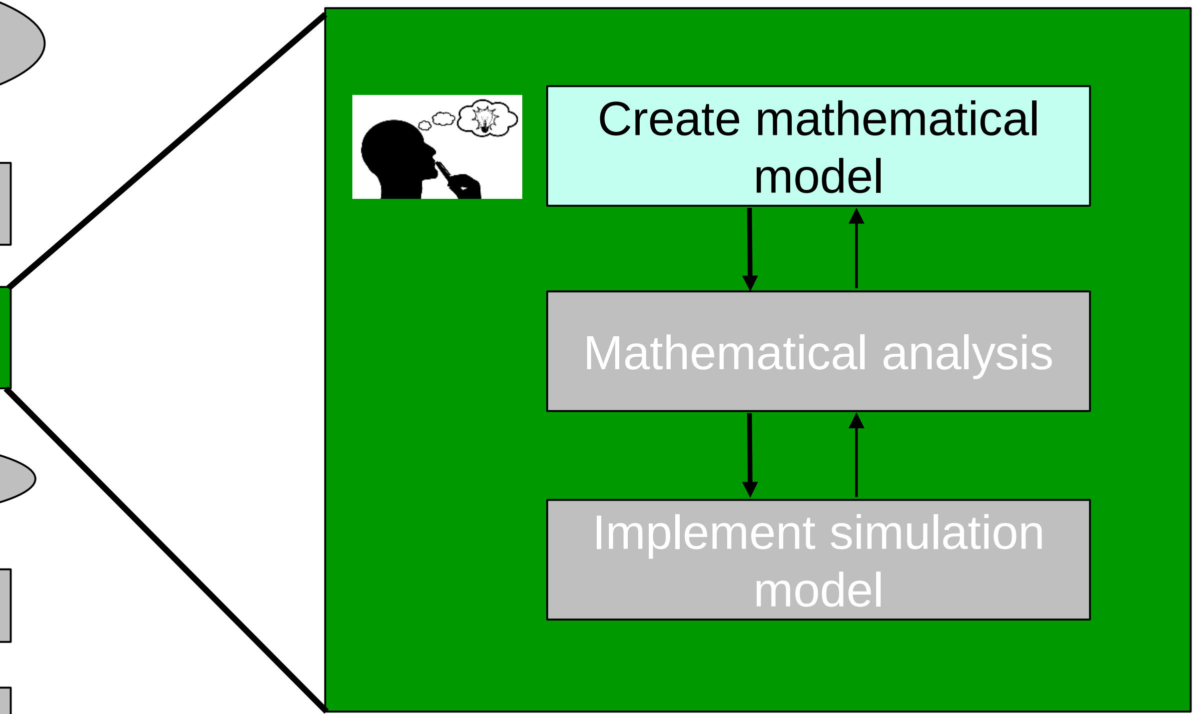


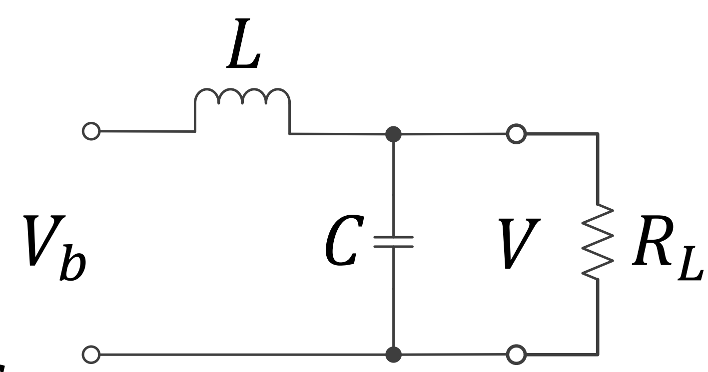


Solve for: V, i_L, i_R and i_C

Create mathematical model



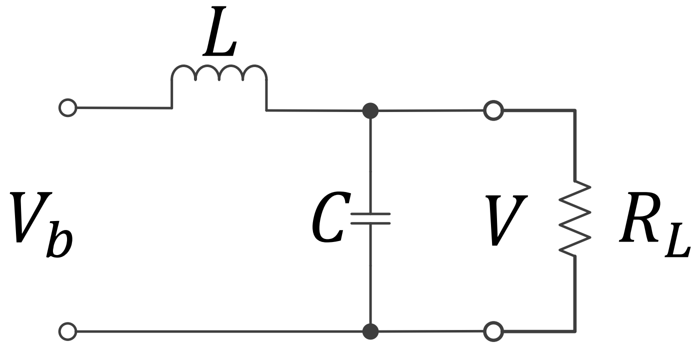




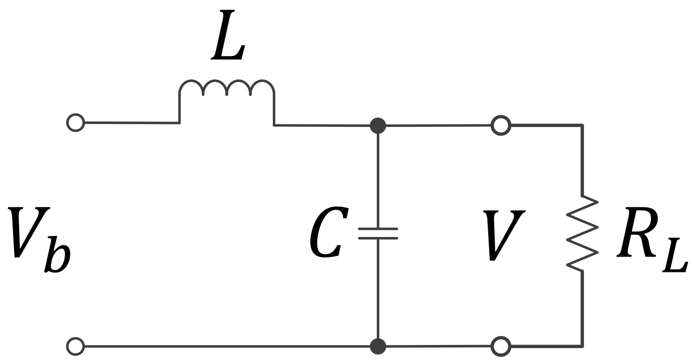
Solve for: V, i_L, i_R and i_C

Equations:

- Type equation here.



2nd Example – Low-Pass RLC Filter

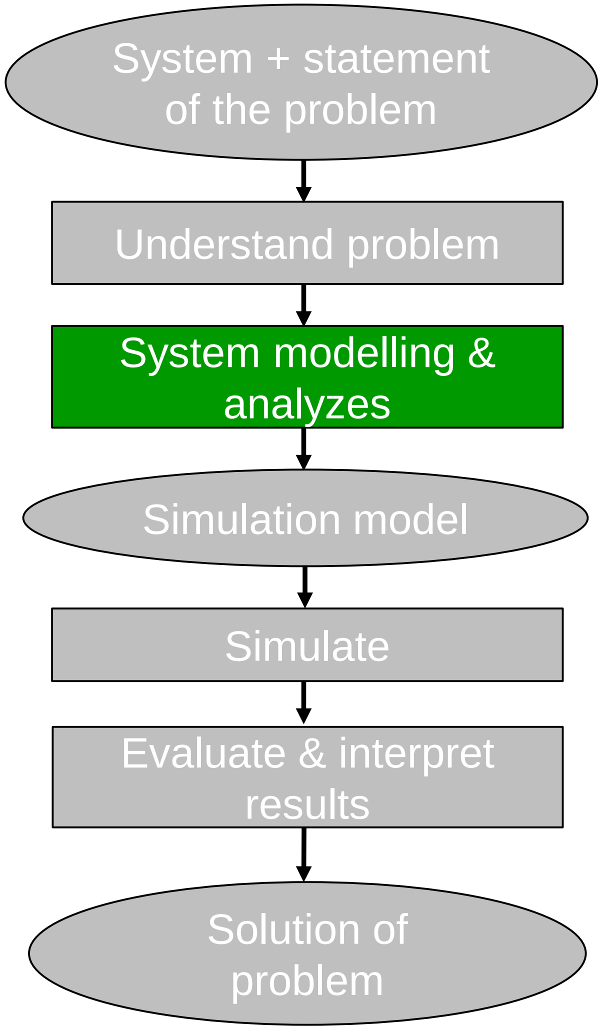


$$V = i_R R_L$$

$$C\frac{dV}{dt} = i_C$$

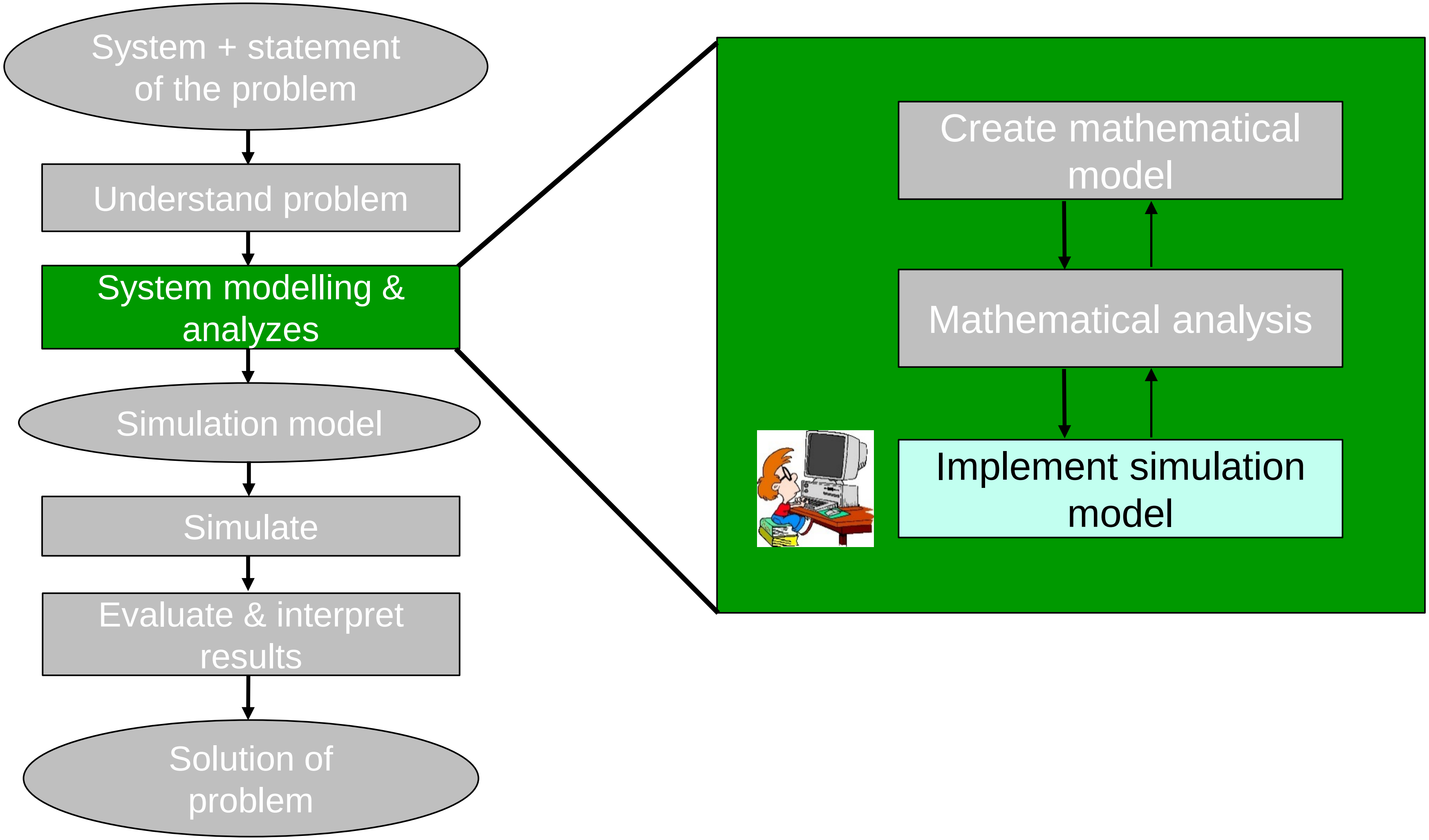
$$L\frac{di_L}{dt} = (\mathrm{V_b} - \mathrm{V})$$

$$i_L = i_R + i_C$$



Example taken from: http://book.xogeny.com/

2nd Example – Low-Pass RLC Filter



2nd Example – Code

Name of model

Import of SIunits package

New type definition

Parameter, constant during simulation

Continuous-time variable

Algebraic equation

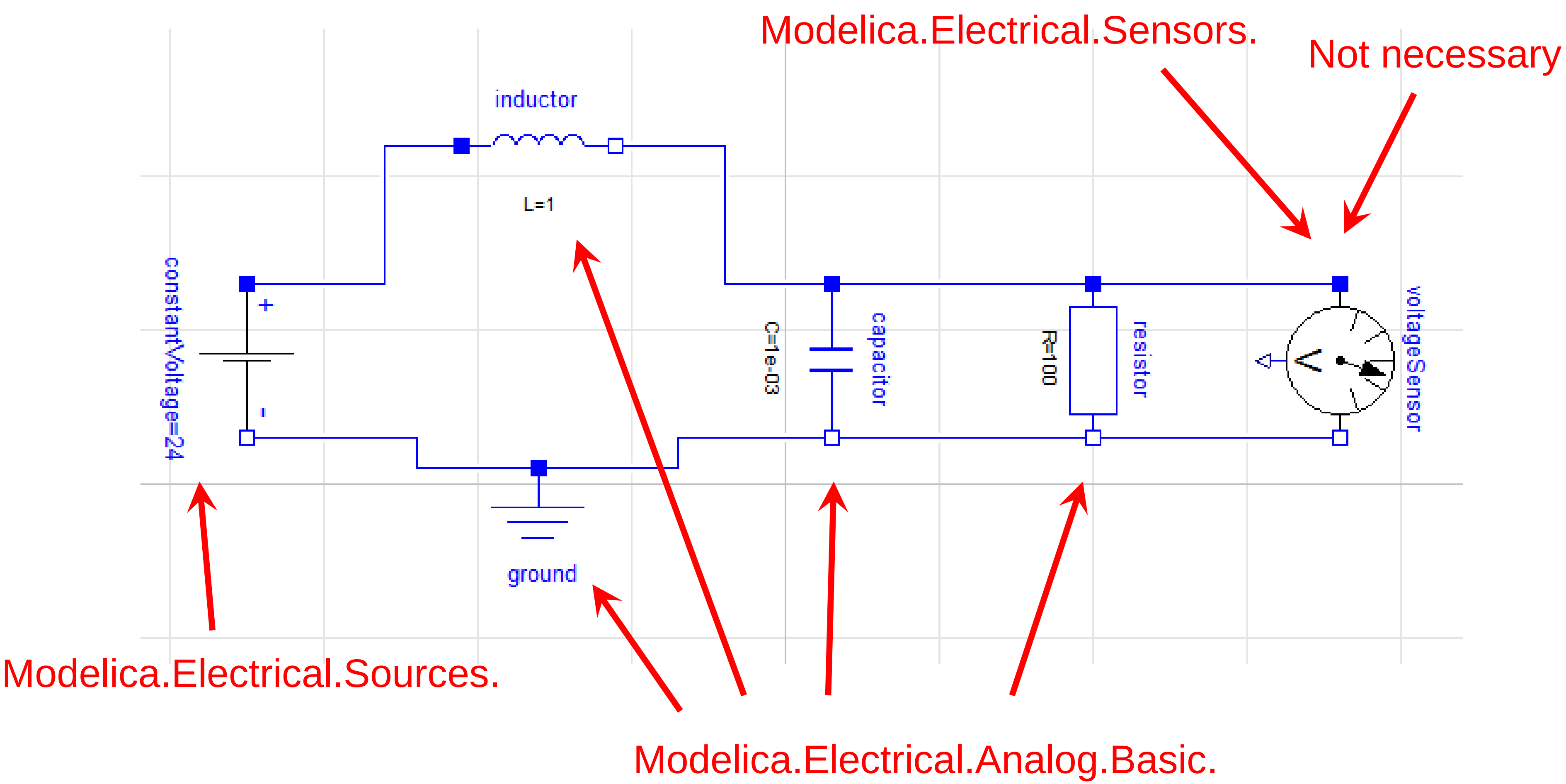
Differential equation

```
model RLC5 "A resistor-inductor-capacitor circuit model"

  import SI = Modelica.SIunits "imports SIunits from Modelica package";
  // import Modelica.SIunits.* "other way to import package. Here top-level";

  a
  type Voltage = Real(unit="V");
  type Current = Real(unit="A");

  parameter Voltage Vb=24 "Battery voltage";
  parameter SI.Inductance L = 1;
  parameter SI.Resistance R = 100;
  parameter SI.Capacitance C = 1e-03;

  Voltage V;
  Current i_L;
  Current i_R;
  Current i_C;

equation
  V = i_R*R;
  C*der(V) = i_C;
  L*der(i_L) = (Vb-V);
  i_L = i_R+i_C;

end RLC5;
```

Modelica technology

- Modelica includes *graphical* editing for application model design based on predefined components

2nd Example – Graphical solution with the help of Dymola standard packages



Connectors and Connector Classes

Connectors are instances of *connector classes*

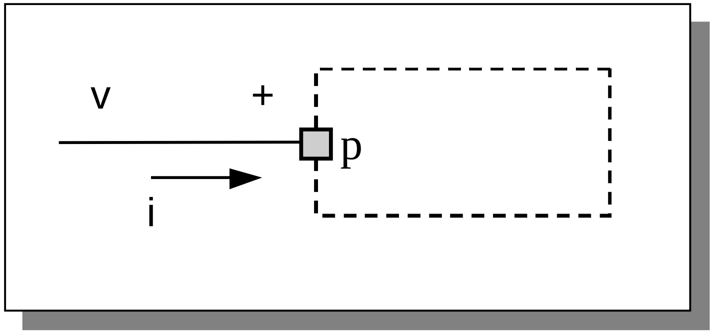
electrical connector

connector class

keyword `flow` indicates that currents of connected pins sum to zero.

an instance p of class `Pin`

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

Pin p;
```



The `flow` prefix

Two kinds of variables in connectors:

- *Non-flow variables* ***potential*** or energy level
- *Flow variables* represent some kind of **flow**

The `flow` prefix

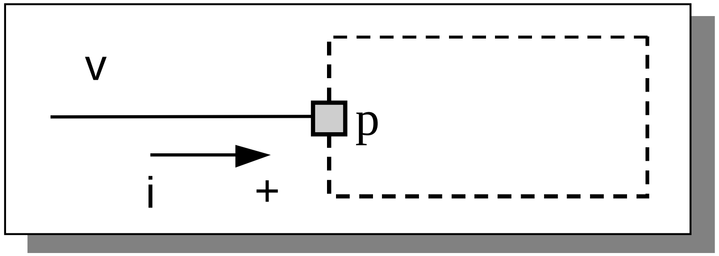
Coupling

- *Equality coupling*, for non-`flow` variables
  - In electrics: $v_1 = v_2 = \ ... = v_n$ (Kirchhoff's 2nd law)

- *Sum-to-zero coupling*, for `flow` variables
  - In electrics: $i_1 + i_2 + \ ... + i_n = 0$ (Kirchhoff's 1st law)

The value of a `flow` variable is *positive* when the current or the flow is *into* the component

positive flow direction:



Connectors and Connector Classes

Connectors are instances of *connector classes*

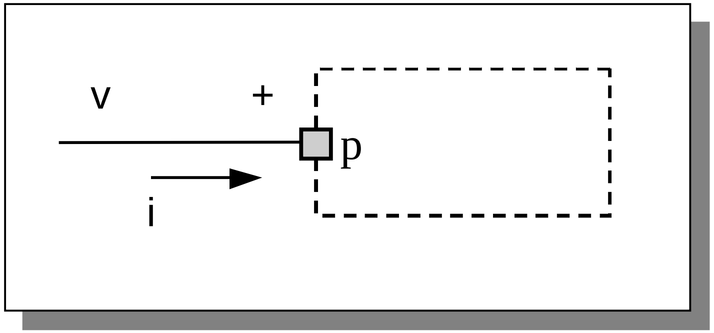
electrical connector

connector class

keyword `flow` indicates that currents of connected pins sum to zero.

an instance p of class `Pin`

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

Pin p;
```



What does the product of the electric pair of connector variables represent?

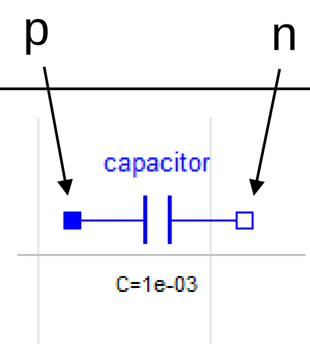
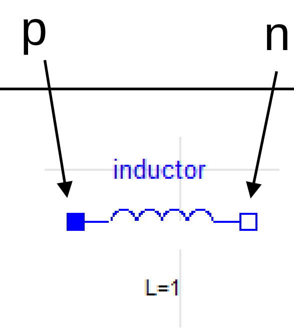
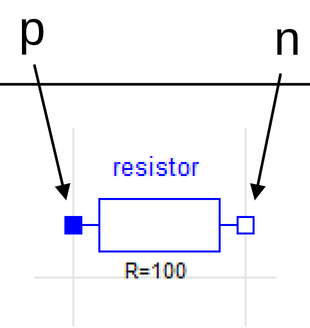
Homework (recommended)

- Install Modelica on your Laptop
- Implement the Low-Pass RLC filter in the texteditior of Modelica
- Implement the Low-Pass RLC filter in the graphical interface of Modelica.

- Try: Implement the capacitor, inductor and resistor using «extend» (we will discuss this in the next lecture – see next slides for help)

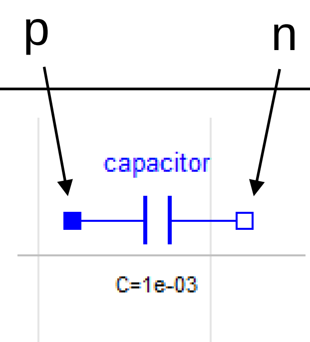
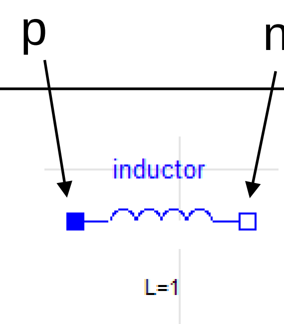
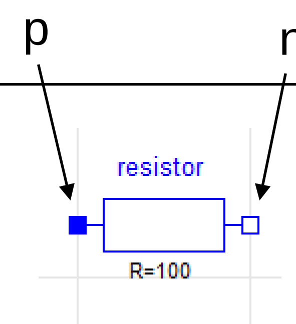
Use of Inheritance & Connectors

- Connector-Name (Pin): p, n

			
Equations:	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $i = C * der(v)$	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $v = L * der(i)$	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $v = R * i$

Use of Inheritance & Connectors

- Connector-Name (Pin): p,n

			
Equations:	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $i = C * der(v)$	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $v = L * der(i)$	$0 = p.i + n.i$ $v = p.v - n.v$ $i = p.i$ $v = R * i$

→ Only <u>one</u> equation different

Reuse same components

1) Create connector:

```
connector Pin
  Modelica.SIunits.Voltage v;//identical at connection
  flow Modelica.SIunits.Current i;//sum-to-0 at connection
end Pin;
```

Reuse same components

1) Create connector:

```
connector Pin
  Modelica.SIunits.Voltage v;//identical at connection
  flow Modelica.SIunits.Current i;//sum-to-0 at connection
end Pin;
```

2) Create “blueprint” model class TwoPin:

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

Reuse same components

1) Create connector:

```
connector Pin
  Modelica.SIunits.Voltage v;//identical at connection
  flow Modelica.SIunits.Current i;//sum-to-0 at connection
end Pin;
```

2) Create "blueprint" model class TwoPin:

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

partial because:
- Problem is structurally singular
- 6 variables & only 5 equations

Reuse same components with extends

3) Create model with previous components

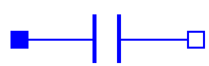
```
model Resistor "Ideal electrical resistor"
  extends TwoPin;
  parameter Modelica.SIunits.Resistance R;

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

```
model Capacitor "Ideal electrical capacitor"
  extends TwoPin;
  parameter Modelica.SIunits.Capacitance C;

equation
  C*der(v) = i;
end Capacitor;
```

Use of Modelica connectors

```
connector Pin
  Voltage      v;   // identical at connection
  flow Current i;   //  sums to zero at connection
end Pin;
```

```
partial model TwoPin
  Pin p, n;   Voltage v; Current i;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end TwoPin;
```



```
model Capacitor
  extends TwoPin;
  parameter Capacitance C;
equation
  C*der(v) = i;
end Capacitor;
```