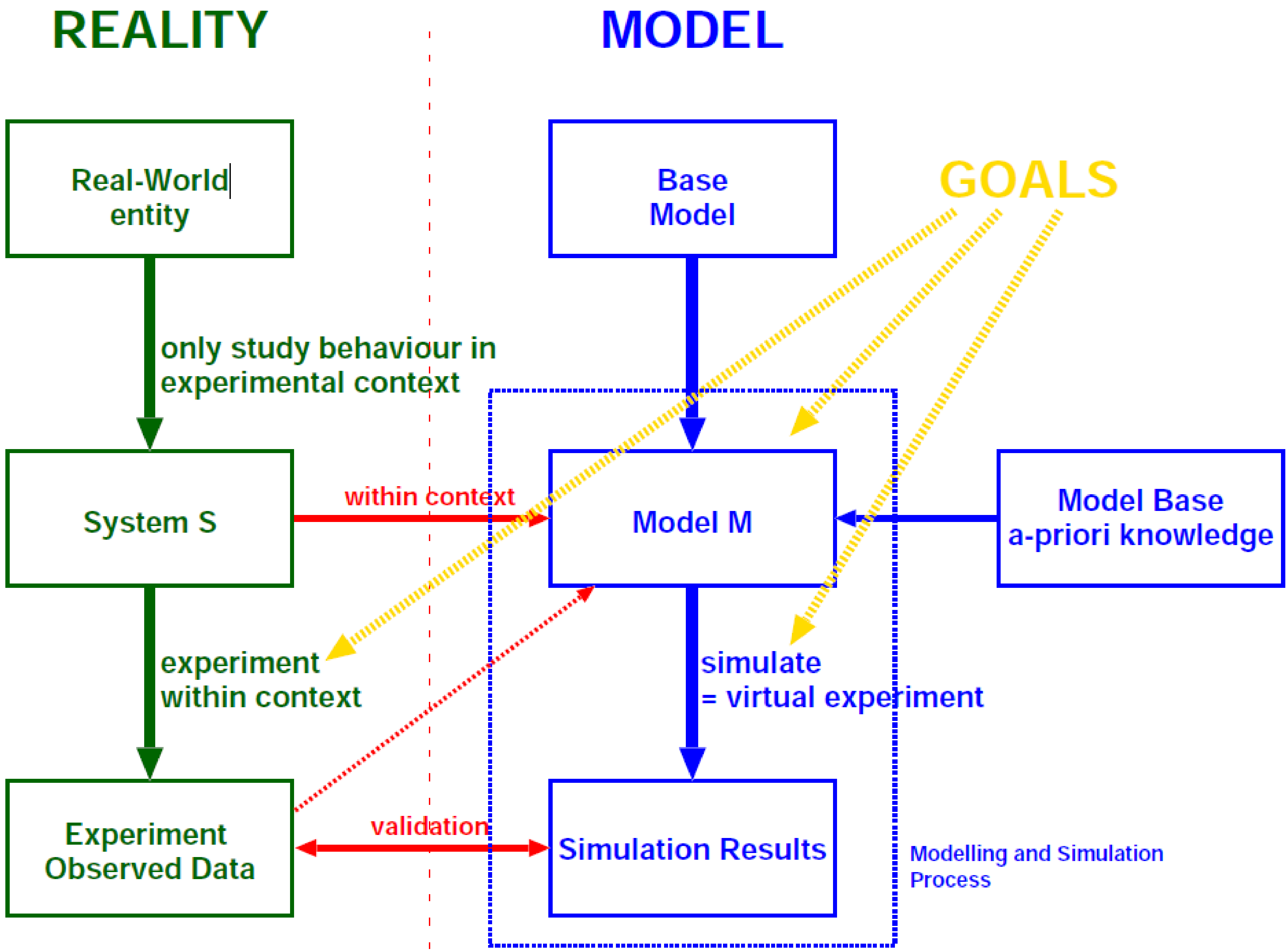
An Introduction to Equation-based Object-Oriented Modelling of Cyber-Physical Systems with (Open)Modelica

Hans Vangheluwe and Rakshit Mittal

Universiteit Antwerpen
Ansymo
Antwerp Systems & Software Modelling
University of Antwerp
Nexor
Cyber-Physical Systems
University of Antwerp
FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK
MPM4CPS
cost



Bernard P. Zeigler. *Multi-faceted Modelling and Discrete-Event Simulation*. Academic Press, 1984.

Mathematical Characterization of Battery Models

Kenneth W. Eure
Langley Research Center Hampton, Virginia

Edward F. Hogge
National Institute of Aerospace, Hampton, Virginia

Figure 22. Octocopter Used for Experimental Flight.

Figure 23. Battery Used for Flight.

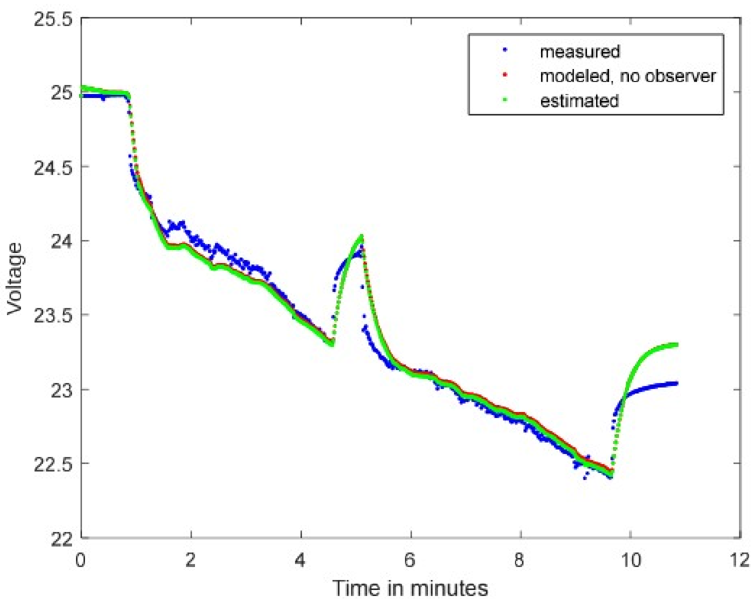


Figure 24. Battery Voltage.

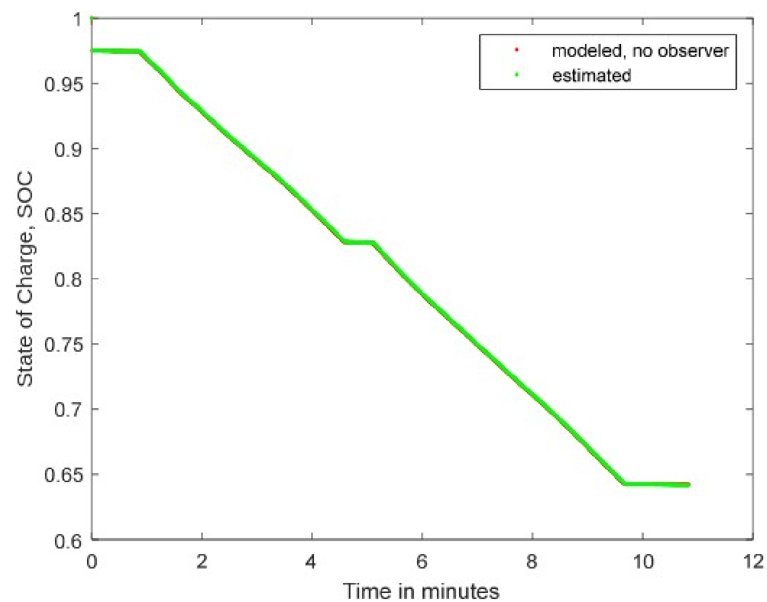
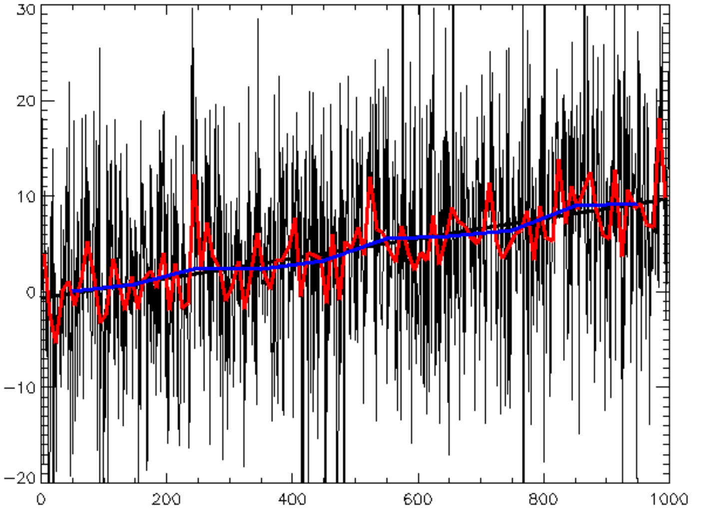


Figure 25 State of Charge.

https://ntrs.nasa.gov/api/citations/20205008059/downloads/NASA-TM-20205008059.pdf

models based on measurements



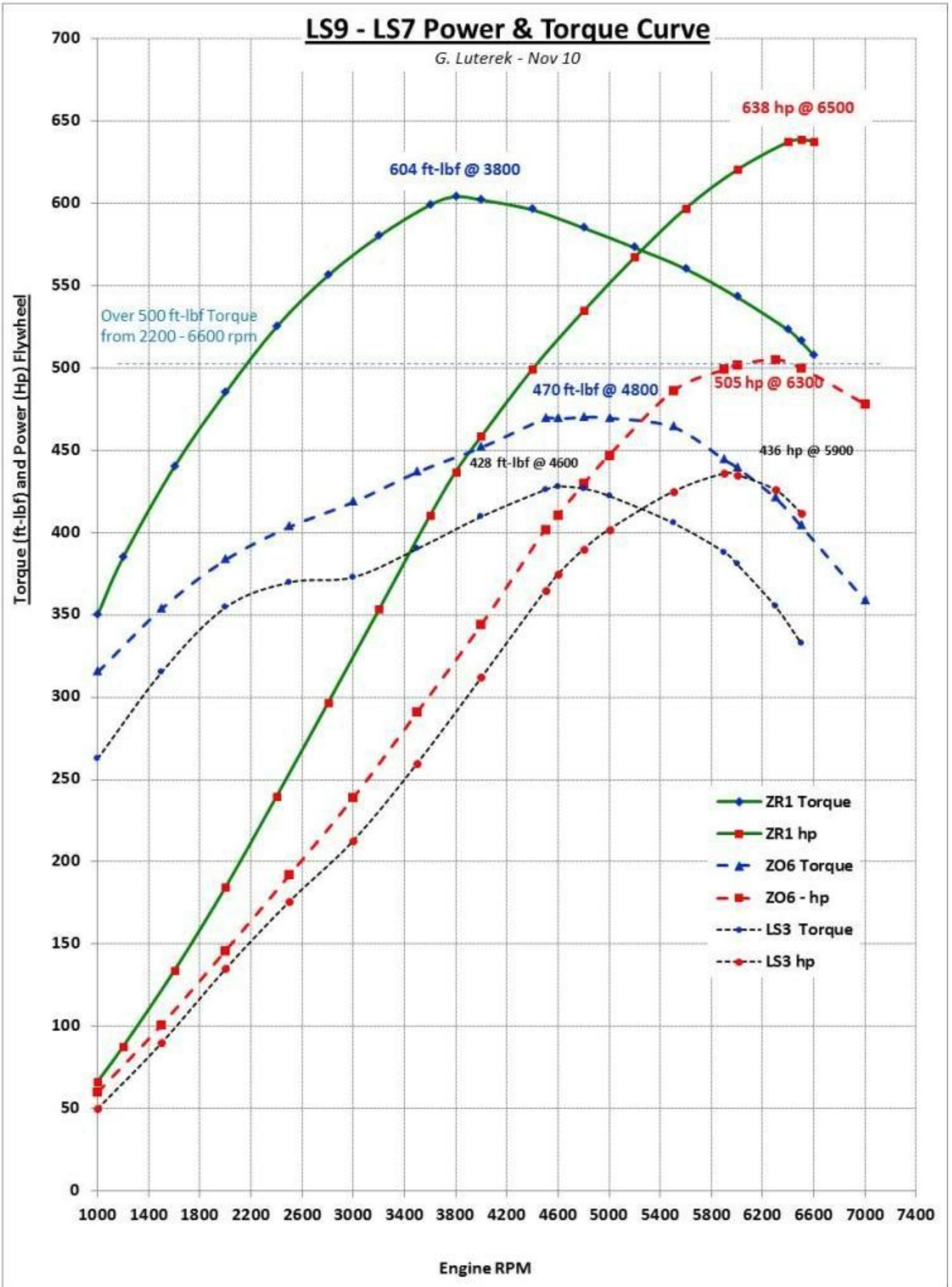
- instance (technology) – specific

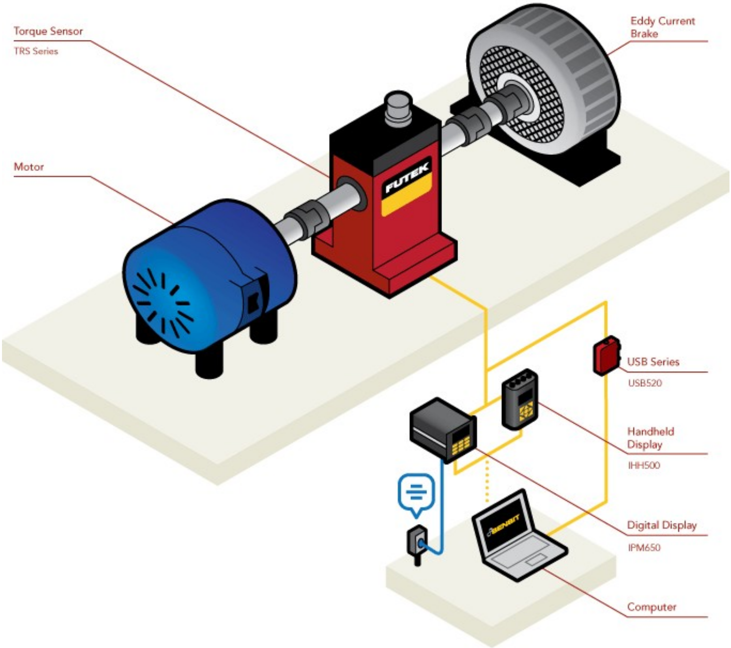
- high (experimentation) cost
- may not even be possible to measure

- allows reproducing data, no extrapolation;
  no insight/explanation

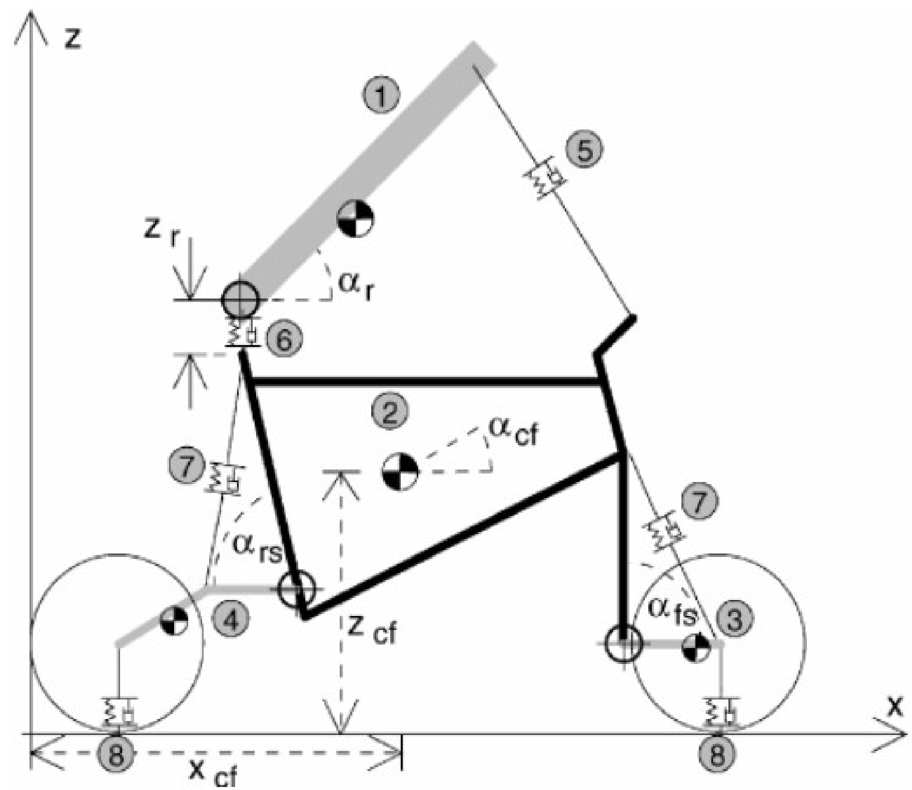
- inductive vs. deductive modelling workflow
  science vs. engineering, usually combination

Torque Curve “model” (measured)





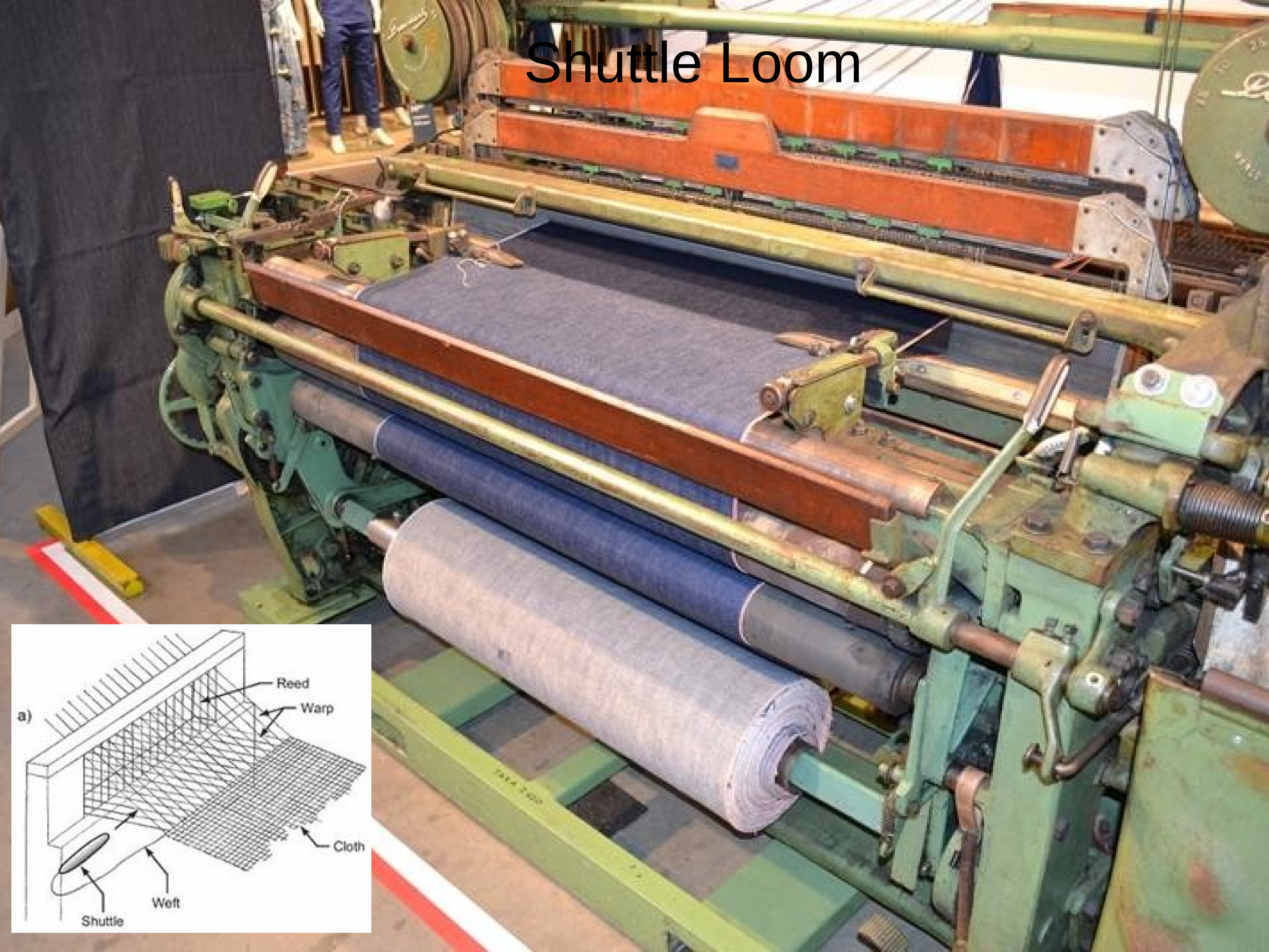
Parameters
(vs. Constants and Variables)



Distributed

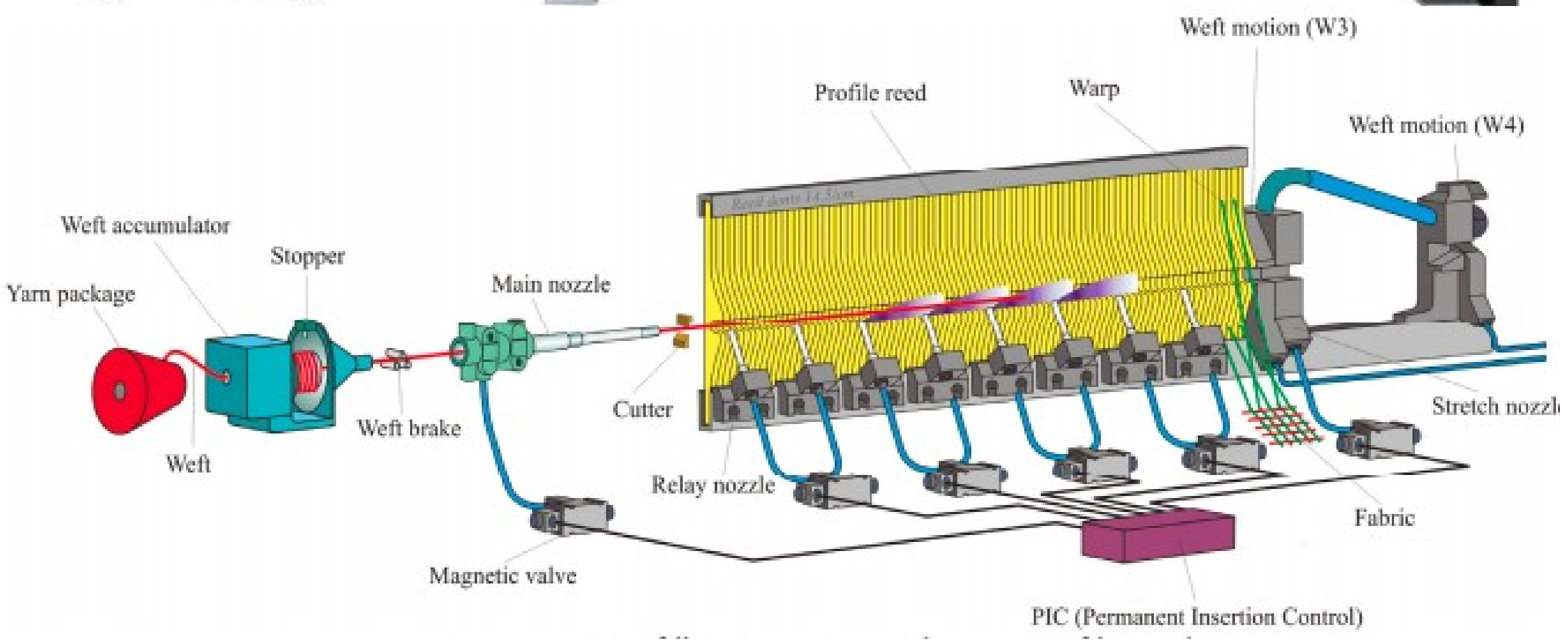
Lumped

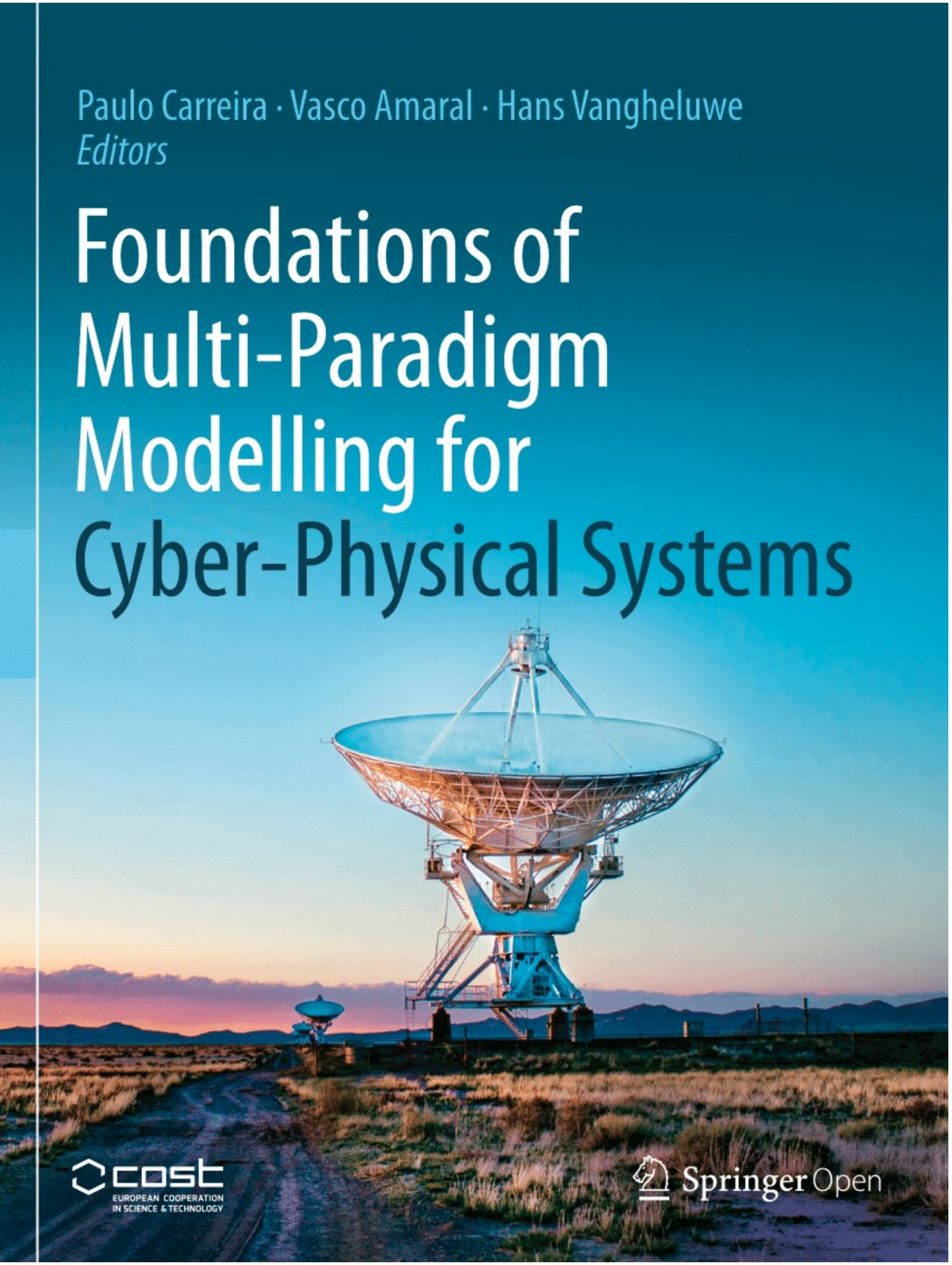
Shuttle Loom



Air Jet Loom

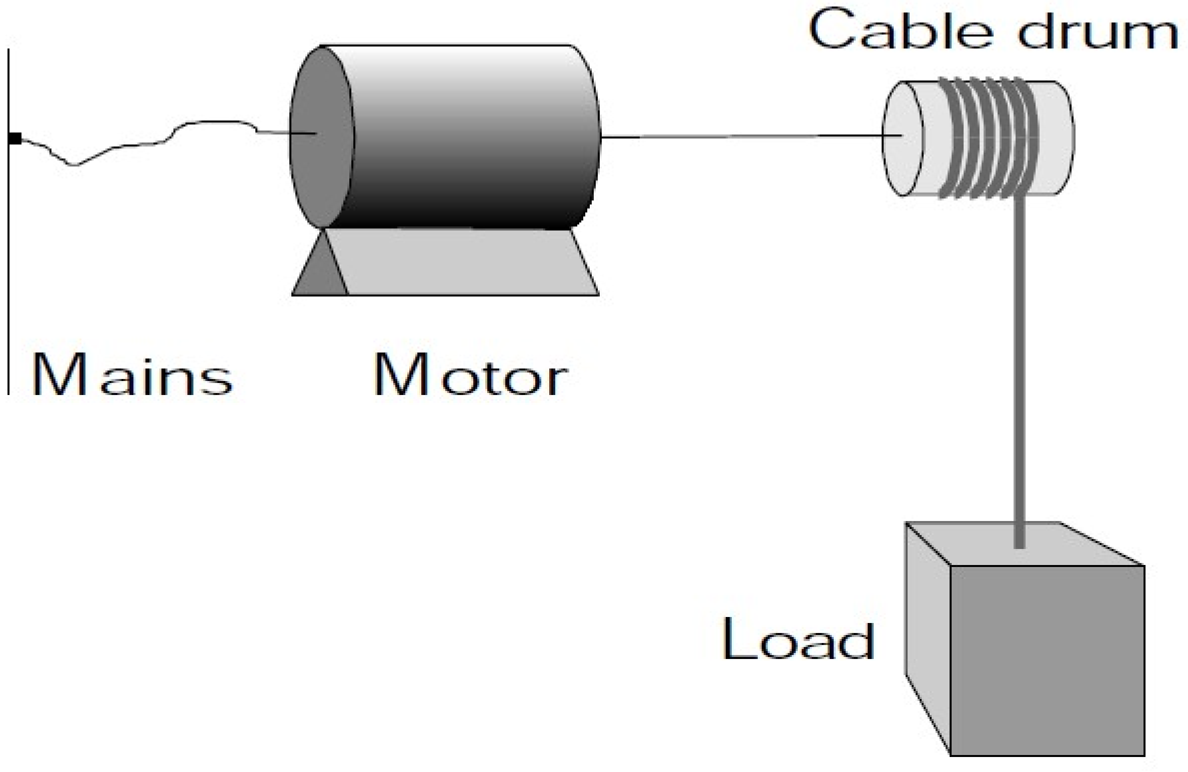

PICANOL
OMNIplus Summum


Weft motion (W3)
Profile reed
Warp
Weft motion (W4)
Weft accumulator
Stopper
Yarn package
Main nozzle
Cutter
Weft brake
Weft
Relay nozzle
Stretch nozzle
Fabric
Magnetic valve
PIC (Permanent Insertion Control)

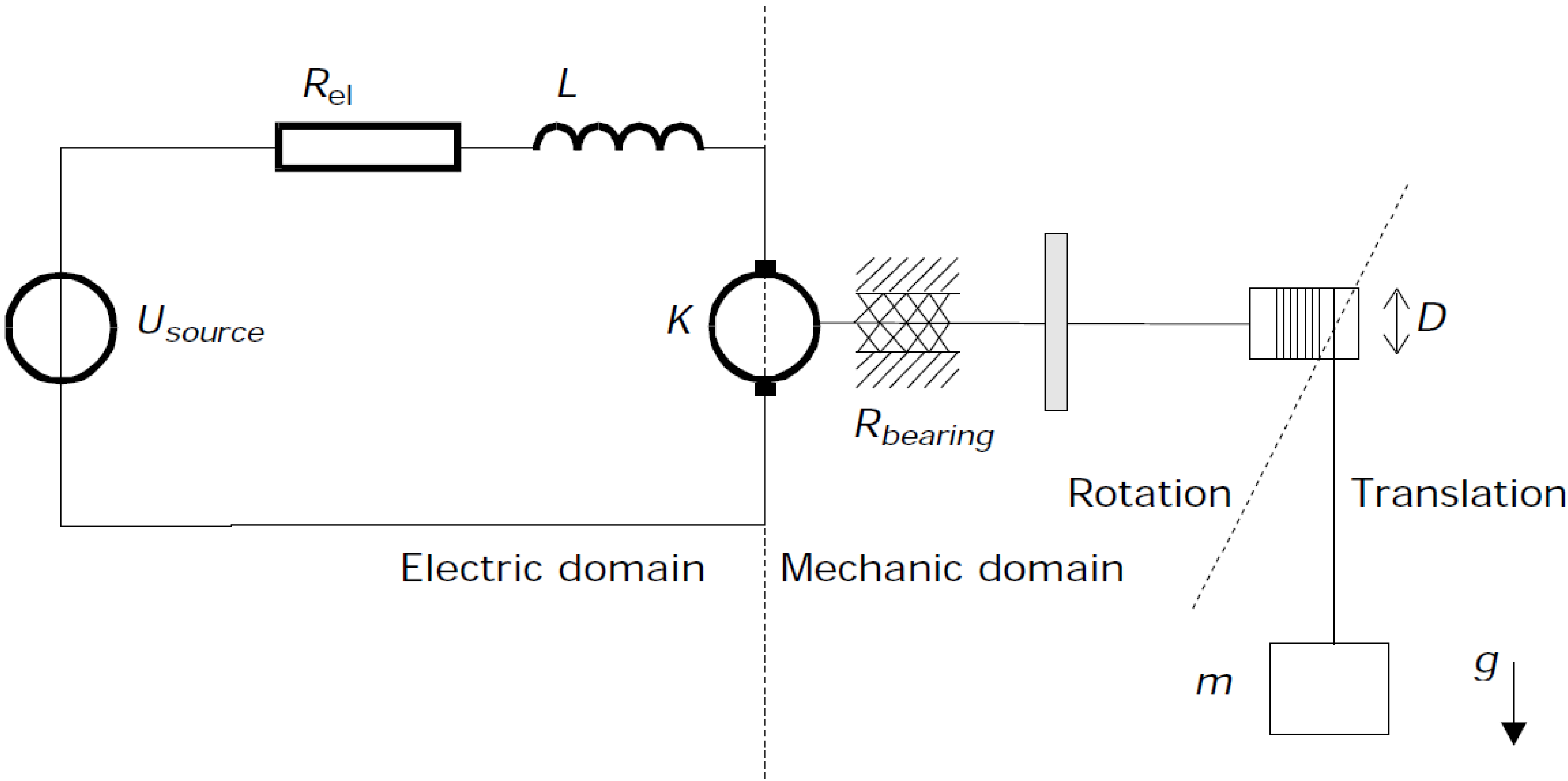


Broenink J.F. (2020) Bond Graphs: A Unifying Framework for Modelling of Physical Systems.
In: Carreira P., Amaral V., Vangheluwe H. (eds) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Springer, Cham.
https://doi.org/10.1007/978-3-030-43946-0_2

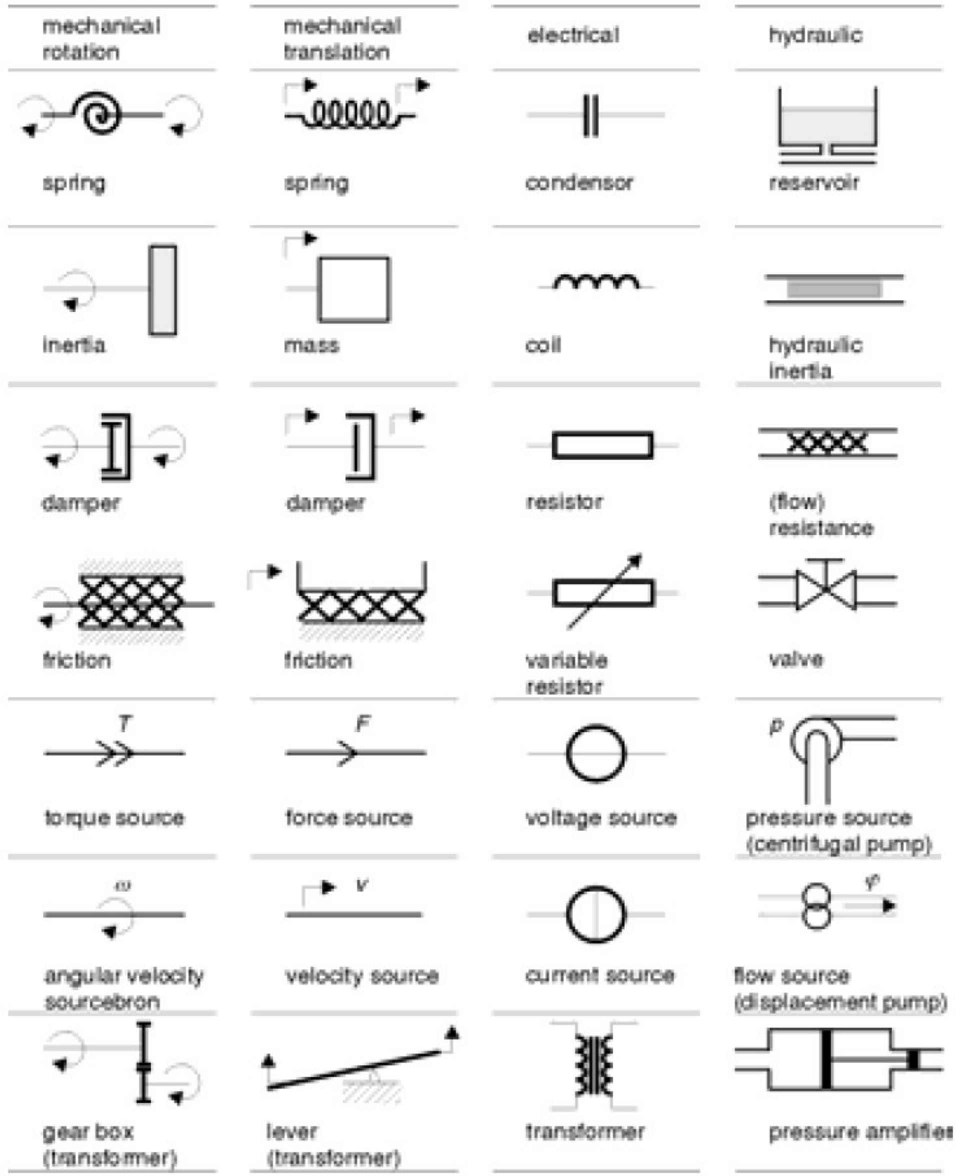
model using domain notation



Idealized Physical Model (IPM)
1D aka “lumped parameter”



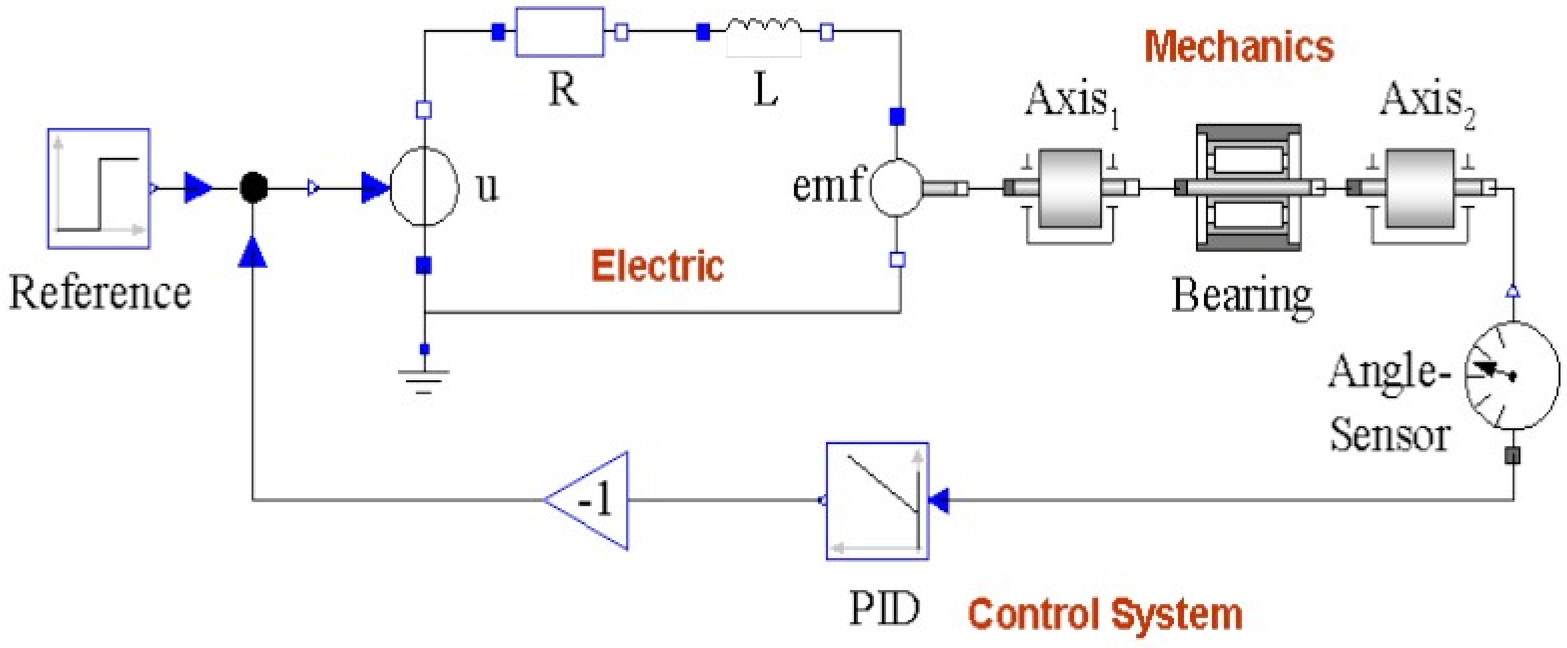
	f *flow*	E *effort*	$q = \int f\,dt$ *generalized displacement*	$p = \int e\,dt$ *generalized momentum*
Electromagnetic	i current	U voltage	$q = \int i dt$ charge	$\lambda = \int u dt$ magnetic flux linkage
mechanical translation	V velocity	F force	$x = \int v dt$ displacement	$p = \int F dt$ momentum
mechanical rotation	ω angular velocity	T torque	$\theta = \int \omega dt$ angular displacement	$b = \int T dt$ angular momentum
hydraulic/ pneumatic	φ volume flow	P pressure	$V = \int \varphi dt$ volume	$\Gamma = \int p dt$ momentum of a flow tube
Thermal	T temperature	F_S entropy flow	$S = \int f_S dt$ entropy	
Chemical	μ chemical potential	F_N molar flow	$N = \int f_N dt$ number of moles	


mechanical rotation
mechanical translation
electrical
hydraulic
spring
spring
condensor
reservoir
inertia
mass
coil
hydraulic inertia
damper
damper
resistor
(flow) resistance
friction
friction
variable resistor
valve
T
F
p
torque source
force source
voltage source
pressure source (centrifugal pump)
ω
v
φ
angular velocity sourcebron
velocity source
current source
flow source (displacement pump)
gear box (transformer)
lever (transformer)
transformer
pressure amplifie

Multi-Domain Modeling

MODELICA

http://www.modelica.org



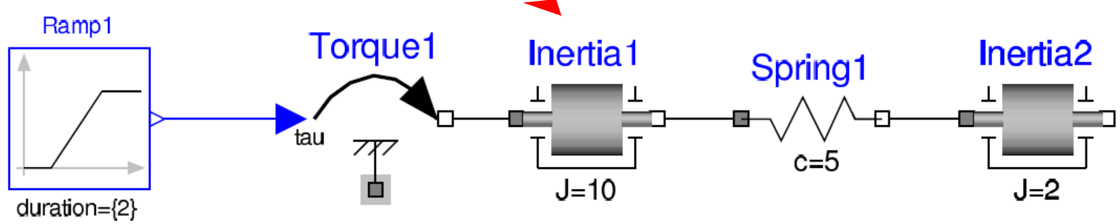
this slide from Peter Fritzson's Modelica tutorial

Multi-Domain Modeling

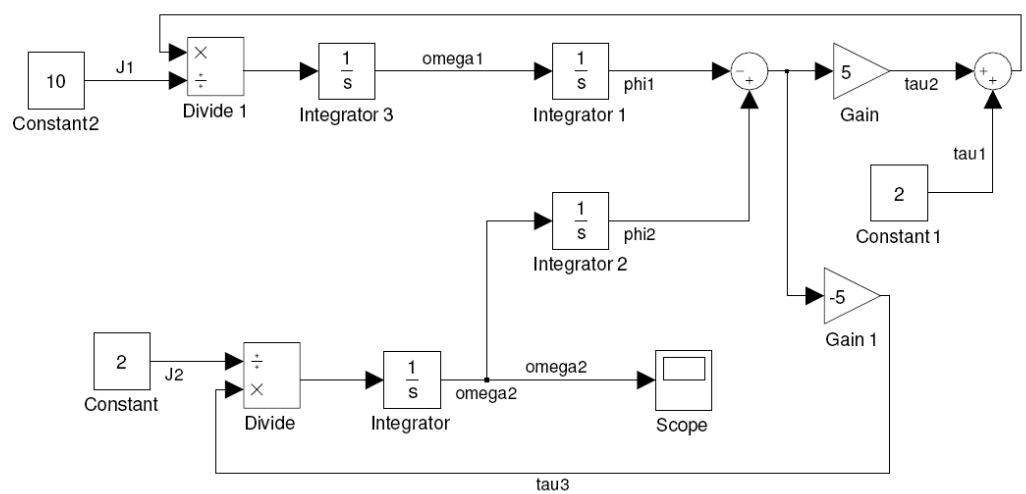
Visual Acausal Hierarchical Component Modeling

Keeps the physical structure

Acausal model (Modelica)



Causal block-based model (Simulink)



this slide from Peter Fritzson's Modelica tutorial

Equation-Based Object-Oriented Modeling Languages and Tools

home | EOOLT 2017

News

EOOLT 2017
The EOOLT workshop took successfully place in Munich, Germany on December 1. Proceedings are now available on ACM Digital Library

Modelica Scalable Test Suite
A new suite of scalable test models can be found here.

Welcome to the EOOLT community!

This site is intended to be a meeting point for researchers and practitioners working in the area of equation-based object-oriented modeling languages and tools. The site's main purpose is to host the workshop pages for the EOOLT workshop series. Below you can find links to the current and past events, together with links to the open access workshop proceedings.

This site is maintained by David Broman. If you have any questions or comments, please send an email.

EOOLT 2017, December 1, Munich, Germany
8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools

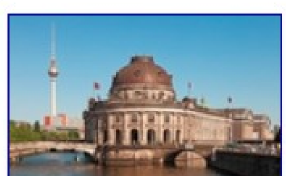
EOOLT 2017 Proceedings (ACM Digital Library)

Workshop site

EOOLT 2016, April 18, Milano, Italy
7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools

EOOLT 2016 Proceedings (ACM Digital Library)

Workshop site (archived)

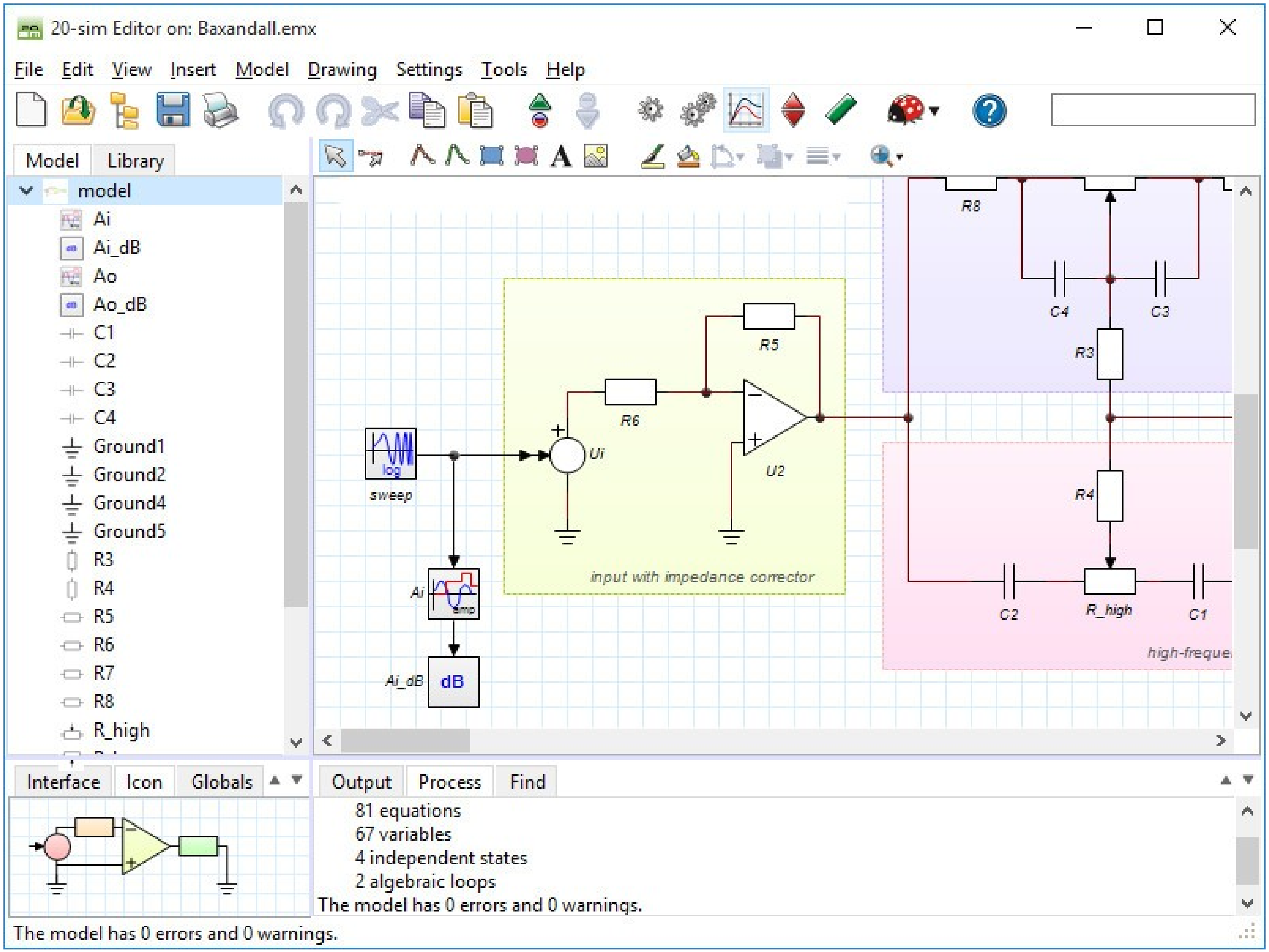


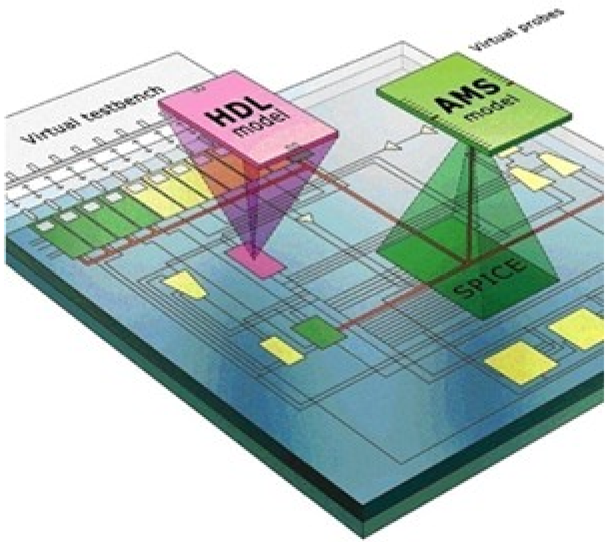
EOOLT 2014, Berlin, Germany
6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools

EOOLT 2014 Proceedings (ACM Digital Library)

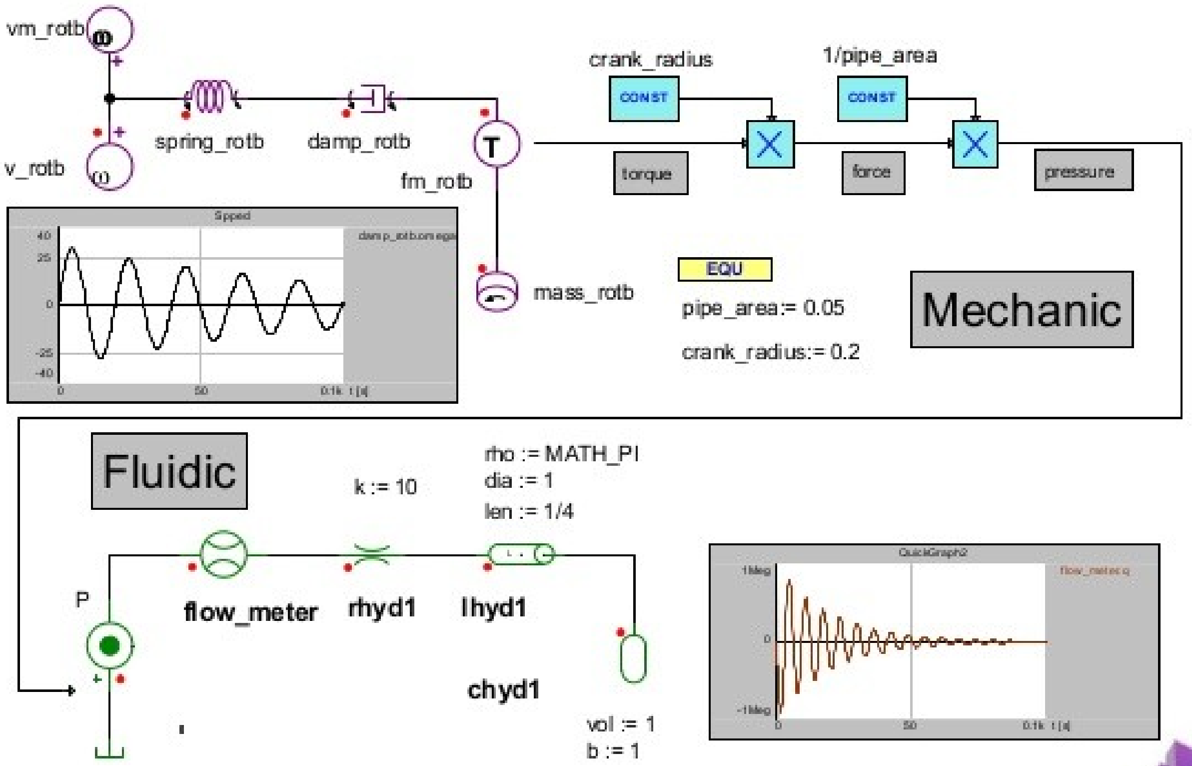
Workshop site (archived)

20-SIM

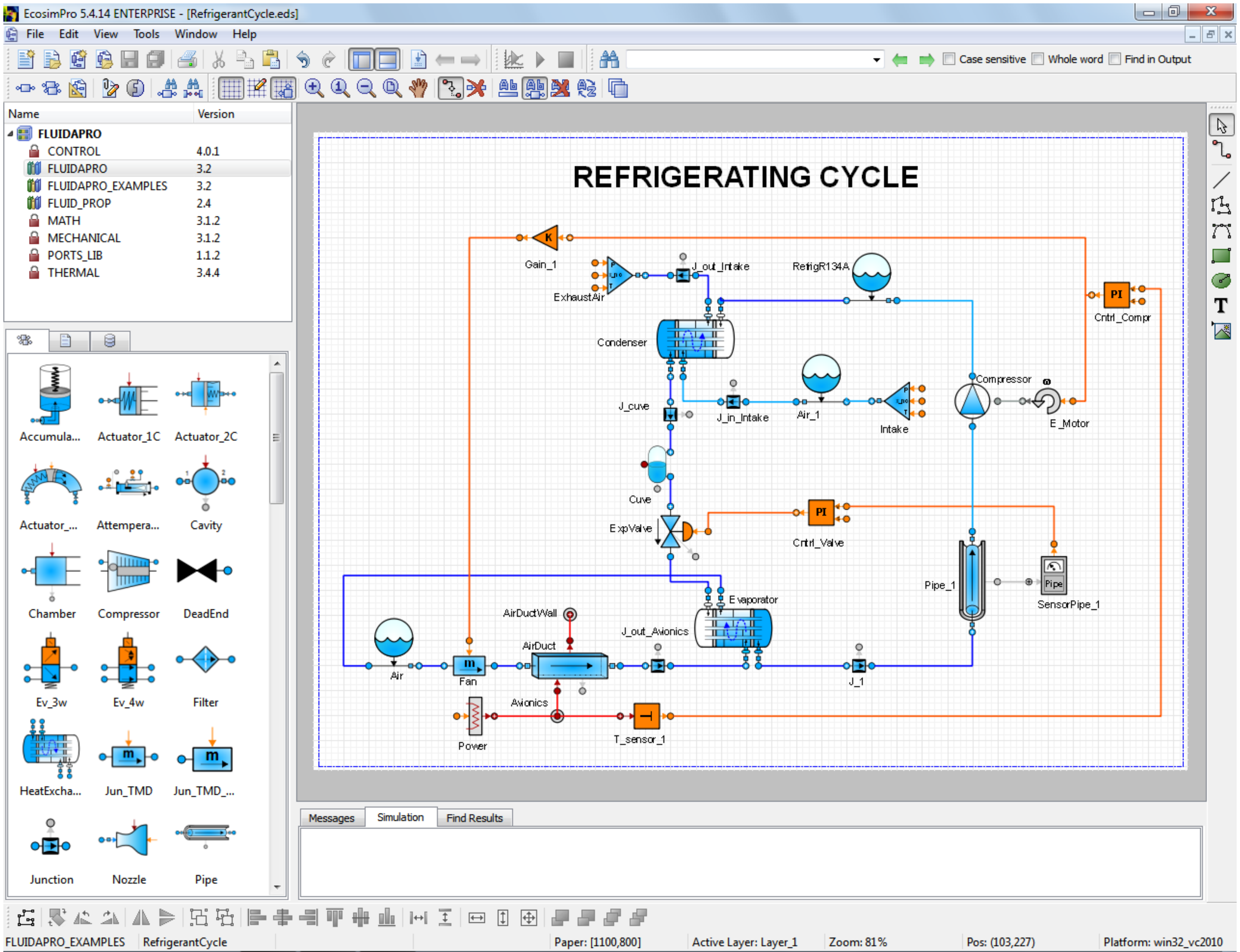

20-sim Editor on: Baxandall.emx
File Edit View Insert Model Drawing Settings Tools Help
Model Library
model
Ai
Ai_dB
Ao
Ao_dB
C1
C2
C3
C4
Ground1
Ground2
Ground4
Ground5
R3
R4
R5
R6
R7
R8
R_high
R8
C4
C3
R3
R5
R6
Ui
U2
sweep
log
Ai
Ai_dB
dB
input with impedance corrector
R4
C2
R_high
C1
Interface Icon Globals
Output Process Find
81 equations
67 variables
4 independent states
2 algebraic loops
The model has 0 errors and 0 warnings.
The model has 0 errors and 0 warnings.



VHDL-AMS Multi Domain Design

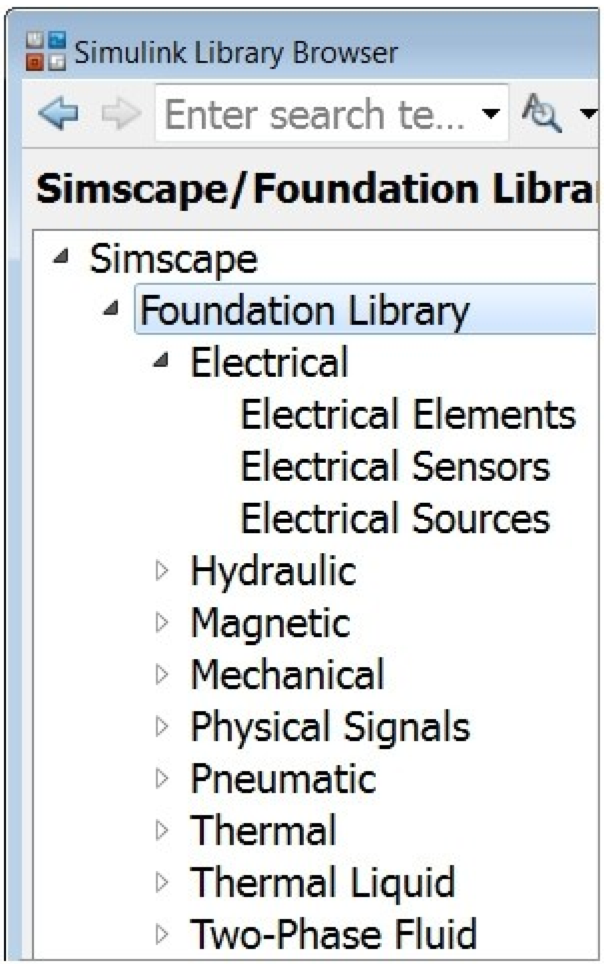


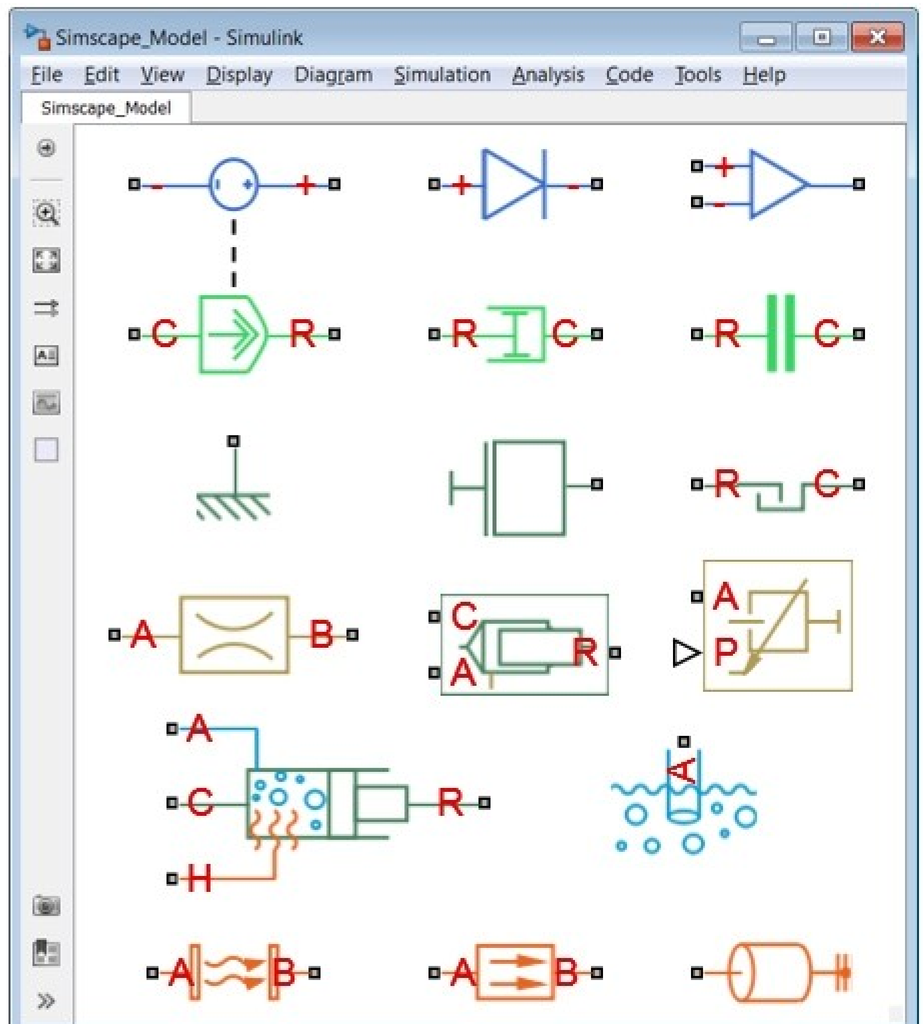
EcosimPro
Modelling and Simulation Software

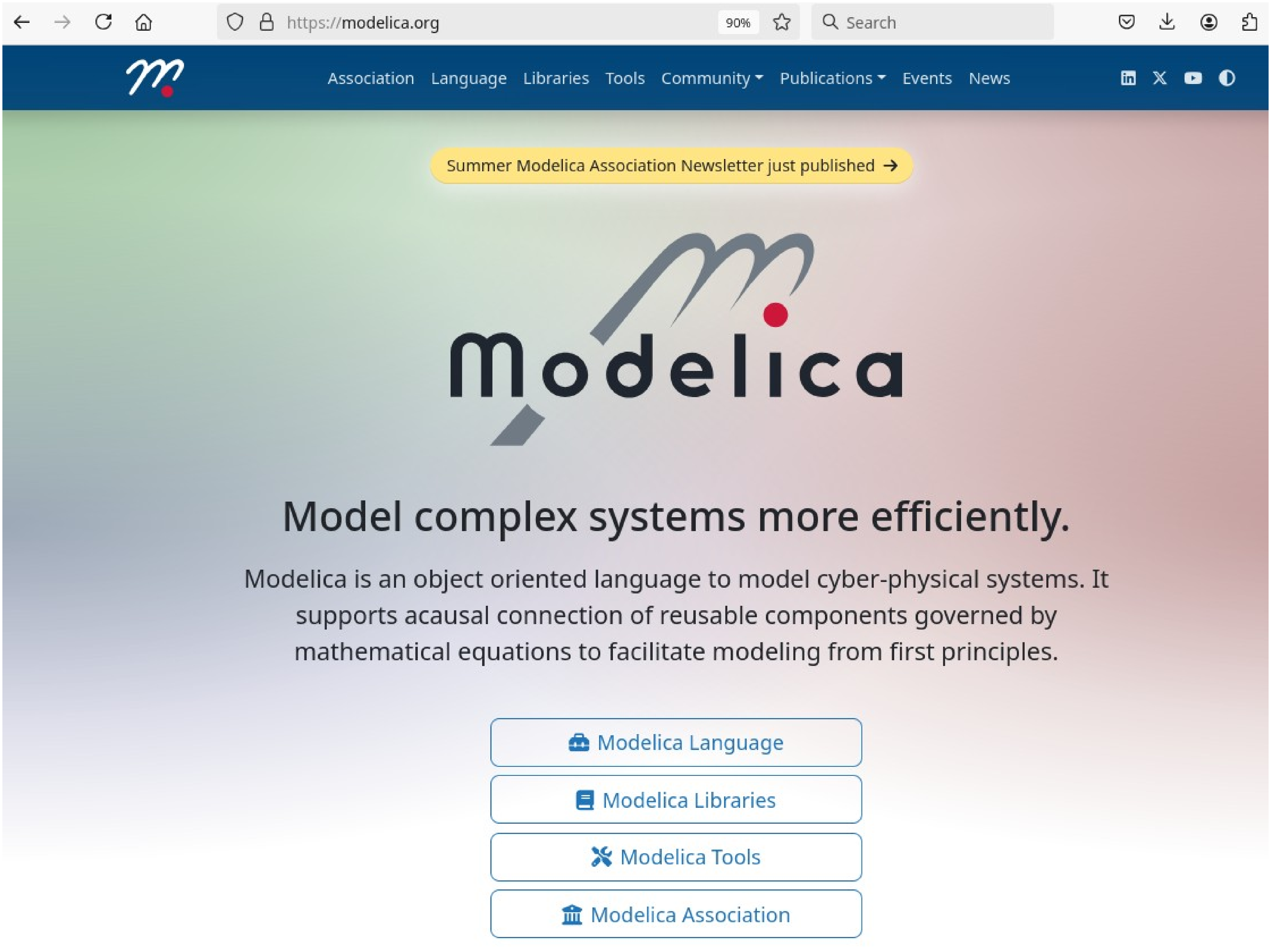


Simscape

Steven Xu






https://modelica.org
90%
Search
Association
Language
Libraries
Tools
Community
Publications
Events
News
Summer Modelica Association Newsletter just published
Modelica
Model complex systems more efficiently.
Modelica is an object oriented language to model cyber-physical systems. It supports acausal connection of reusable components governed by mathematical equations to facilitate modeling from first principles.
Modelica Language
Modelica Libraries
Modelica Tools
Modelica Association

Dokumentutgivare
Lund Institute of Technology
Handläggare
Karl Johan Åström
Författare
Hilding Elmqvist

Dokumentnamn
REPORT
Utgivningsdatum
May 1978

Dokumentbeteckning
LUTFD2/(TFRT-1015)/1-226/(1978)
Ärendebeteckning

Dymola

Dokumenttitel och undertitel

A Structured Model Language for Large Continuous Systems

Referat (sammandrag)

A model language, called DYMOLA, for continuous dynamical systems is proposed. Large models are conveniently described hierarchically using a submodel concept. The ordinary differential equations and algebraic equations need not be converted to assignment statements. There is a concept, cut, which corresponds to connection mechanisms of complex types, and there are facilities to describe the connection structure of a system. A model can be manipulated for different purposes such as simulation and static calculations. The model equations are sorted and they are converted to assignment statements using formula manipulation. A translator for the model language is also included.

Referat skrivet av
Author

Förslag till ytterligare nyckelord
nonlinear systems, compiler, permutations, graph theory

Klassifikationssystem och -klass(er)

Indextermer (ange källa)
Mathematical models, Simulation languages, Computerized simulation, Nonlinear systems, Ordinary differential equations, Compilers. (Thesaurus of Engineering and Scientific Terms, Eng. Joint Council,USA)

Omfång
226 pages

Övriga bibliografiska uppgifter

Språk
English

Sekretessuppgifter

ISSN

ISBN

Dokumentet kan erhållas från
Department of Automatic Control
Lund Institute of Technology
P O Box 725, S-220 07 Lund 7, Sweden

Mottagarens uppgifter

Pris

SIS-DB 1

DOKUMENTDATABLAD enligt SIS 62 10 12

Blankett LU 11:25 1976–07

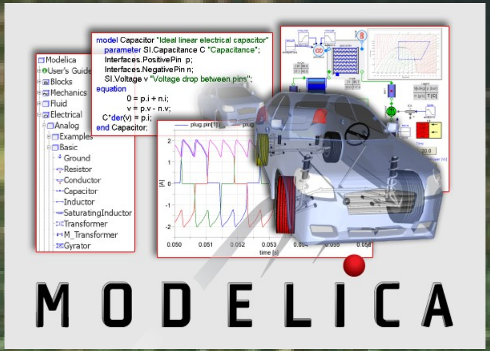
esprit

Simulation in Europe

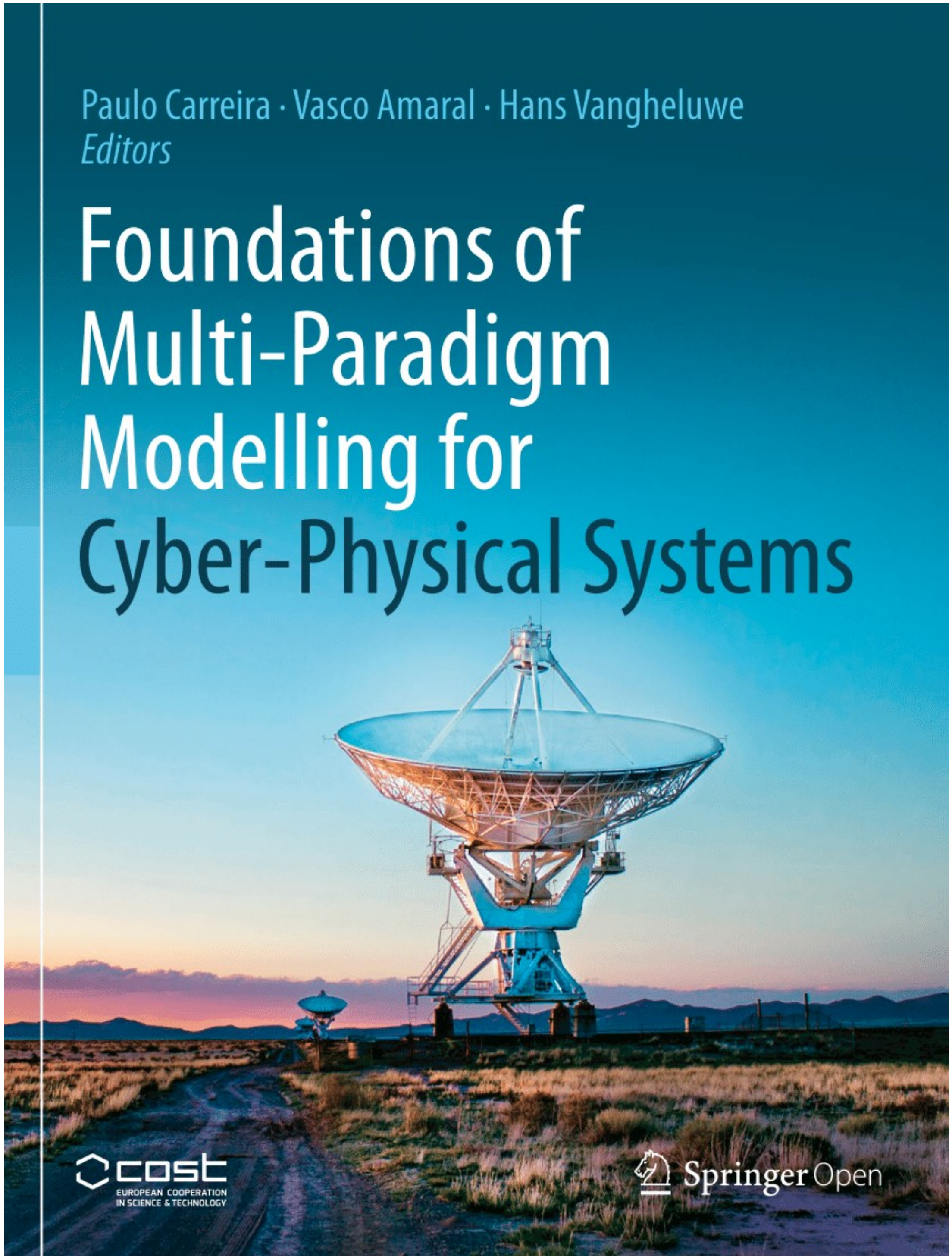
SiE

ESPRIT Basic Research Working Group 8467
Simulation for the Future: New Concepts, Tools and Applications

Keywords:
simulation technologies, multi-paradigm modelling, solvers, standards, interoperability, industrial deployment, demonstrators, user-simulator interfaces



OpenModelica



Fritzson P. (2020) Modelica: Equation-Based, Object-Oriented Modelling of Physical Systems.
In: Carreira P., Amaral V., Vangheluwe H. (eds) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Springer, Cham.
https://doi.org/10.1007/978-3-030-43946-0_3

https://openmodelica.org 110% Search

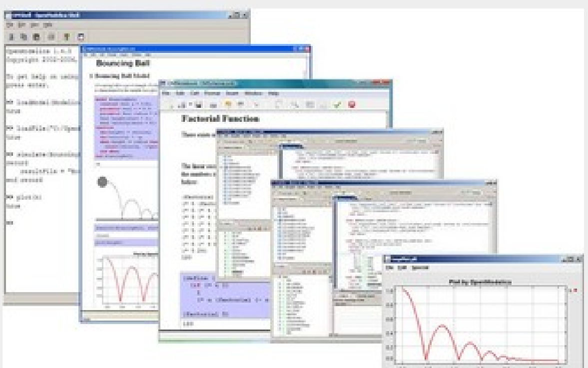
OpenModelica

Home Download Users Developers Events Research

Introduction

OPENMODELICA is an open-source Modelica-based[1] modeling and simulation environment intended for industrial and academic usage. Its long-term development is supported by a non-profit organization – the Open Source Modelica Consortium (OSMC). An overview journal paper is available and slides about Modelica and OpenModelica.

The goal with the OpenModelica effort is to create a comprehensive Open Source Modelica modeling, compilation and simulation environment based on free software distributed in binary and source code form for research, teaching, and industrial usage. We invite researchers and students, or any interested developer to participate in the project and cooperate around OpenModelica, tools, and applications.



Join the OpenModelicaInterest mailing list to get information about new releases.

Help us: get the latest source code or nightly-build and report bugs.

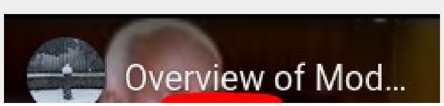
To learn about Modelica, read a book or a tutorial about Modelica.

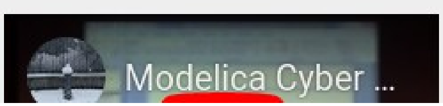
Interactive step-by-step beginners Modelica on-line spoken tutorials
Interactive OMWebbook with examples of Modelica textual modeling and textbook companions with application OpenModelica exercises. A Jupyter notebook Modelica mode, available in OpenModelica.

To get advice how to make existing Modelica libraries work in OpenModelica, see Porting.

For systems engineering with requirement traceability and verification, see ModelicaML.

OpenModelica provides library coverage reports of open-source Modelica libraries showing which libraries work well with OpenModelica and how the support improved over time.

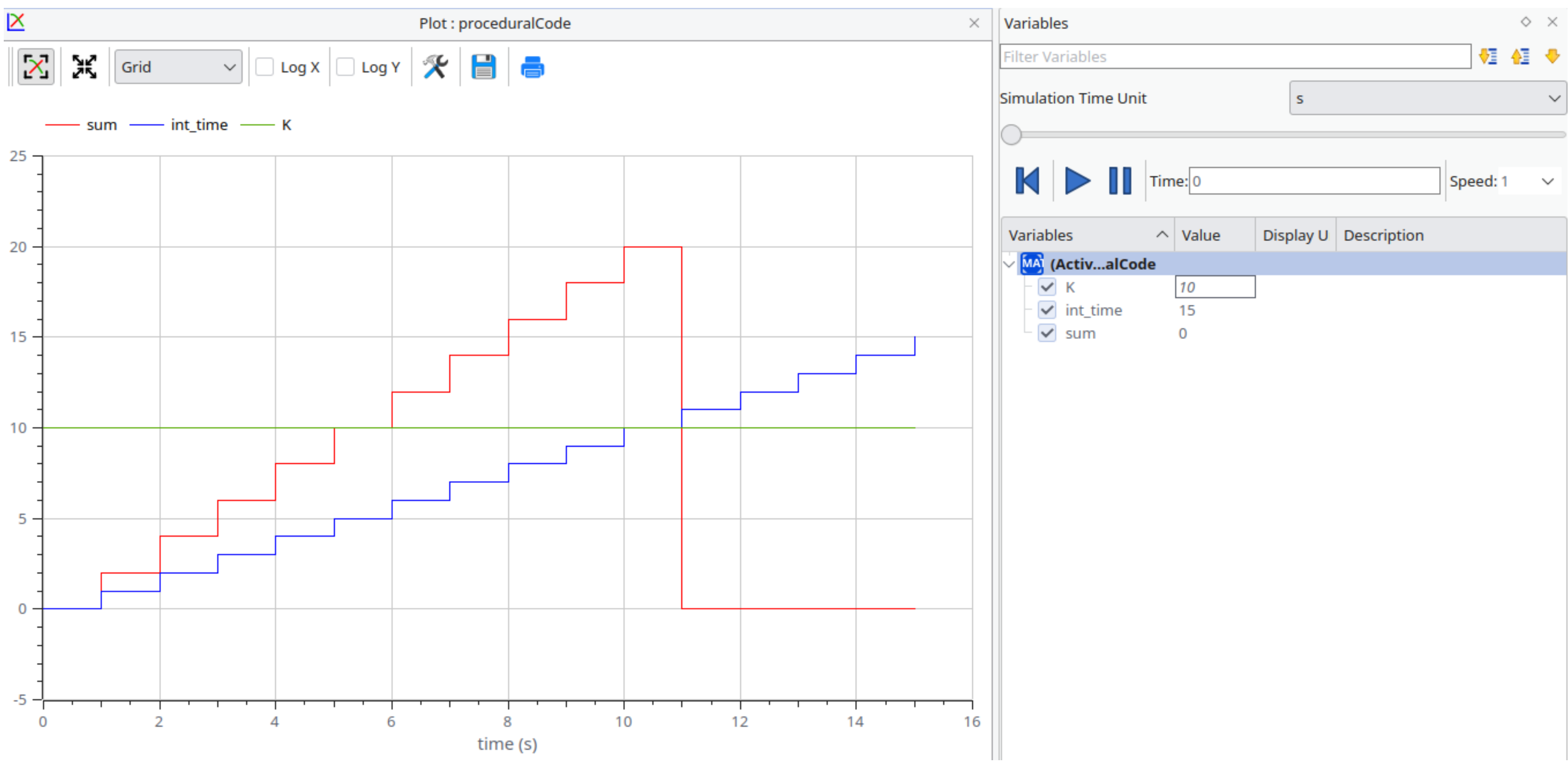




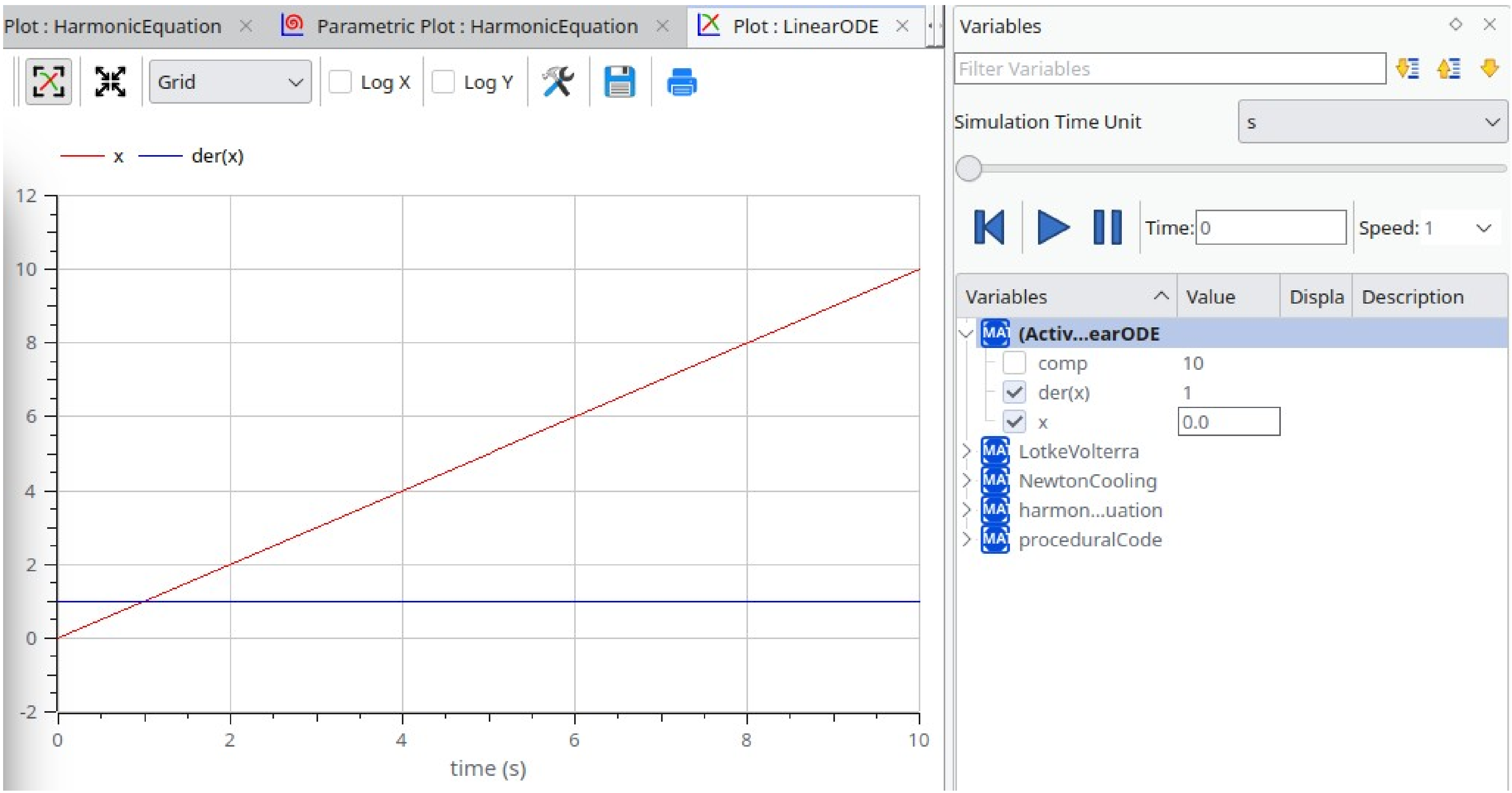
Latest news and events

2024-10-09 Openmodelica v1.24.0 released!
2024-09-16 Openmodelica v1.24.0-dev.beta.0 released!
2024-07-04 Openmodelica v1.23.1 released!
2024-06-06 Openmodelica v1.23.0 released!
2024-05-20 Openmodelica v1.23.0-dev.beta.1 released!
2024-03-12 American Modelica Conference 2024!
2024-03-11 Openmodelica v1.22.3 released!
2024-02-21 Openmodelica v1.22.2 released!
2024-02-05 OpenModelica 2024
2024-02-05 OpenModelica/MODPROD Workshop Feb 5-7, 2024
2023-12-13 Openmodelica v1.22.1 released!
2023-11-08 Openmodelica v1.22.0 released!
2023-04-18 OpenModelica 1.21.0 released!
2023-02-07–2023-02-08 MODPROD 2023
2023-02-06 OpenModelica 2023
2022-12-07 OpenModelica 1.20.0 released!
2022-11-24–2022-11-25 Asian Modelica Conference 2022
2022-11-18 OpenModelica 1.20.0-dev.beta2 released!
2022-10-26–2022-10-28 American Modelica Conference 2022
2022-07-09 OpenModelica 1.19.2 released!

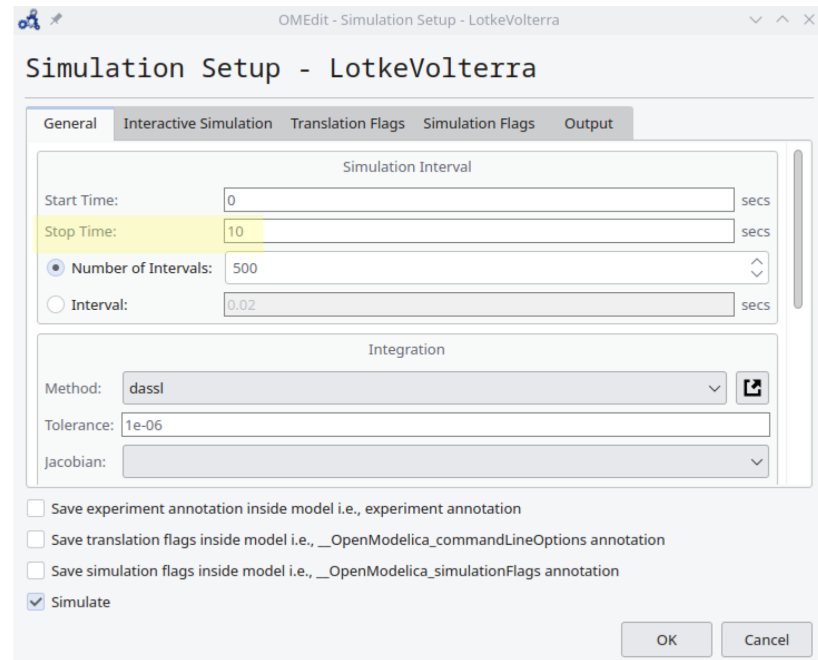
```
package functionExample

function times_two_upto_K "multiplies by two up to K"
  input Integer N  "input";
  input Integer K  "beyond K, result will be 0";
  output Integer result;
algorithm
  result := if N <= K then 2*N else 0;
end times_two_upto_K;

model proceduralCode
  Integer sum(start = 0);
  Integer int_time;
  parameter Integer K = 10;
equation
  int_time = integer(time);
  sum = times_two_upto_K(int_time, K); // or implicit alternative
end proceduralCode;

end functionExample;
```

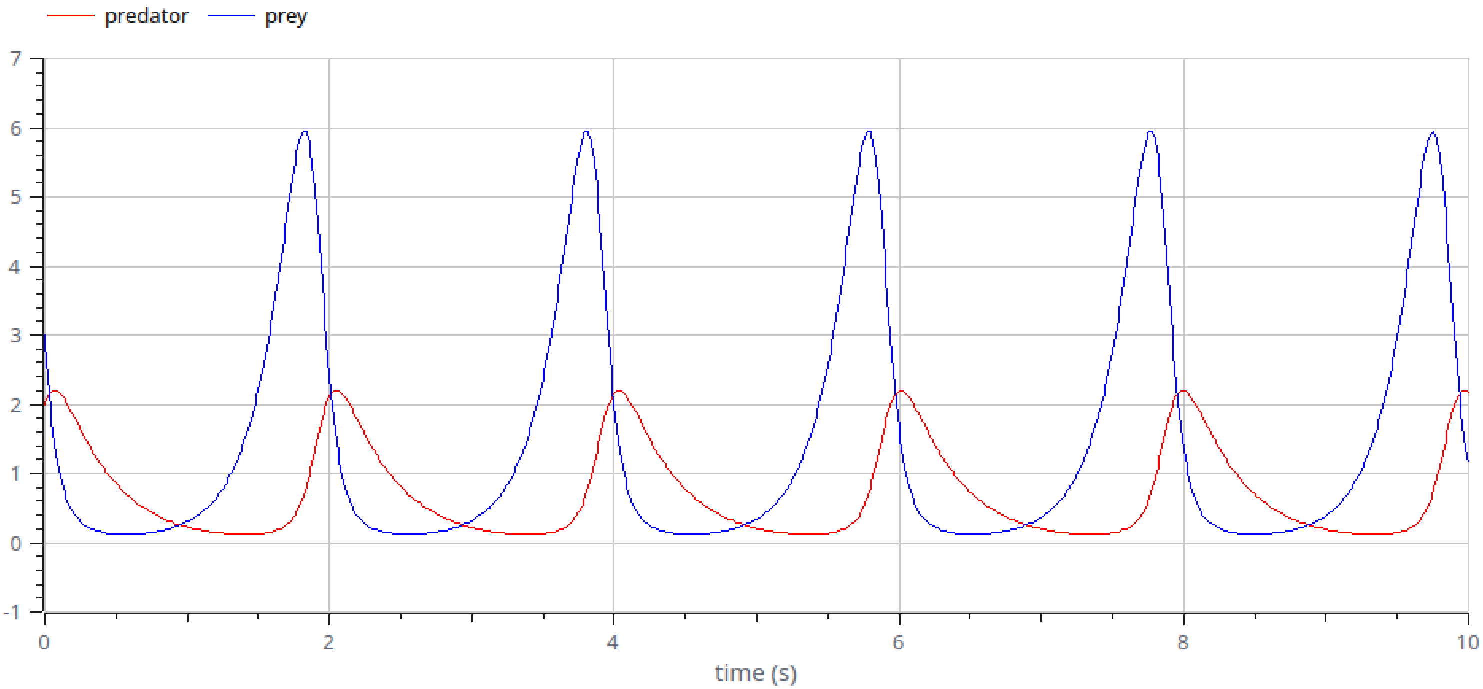


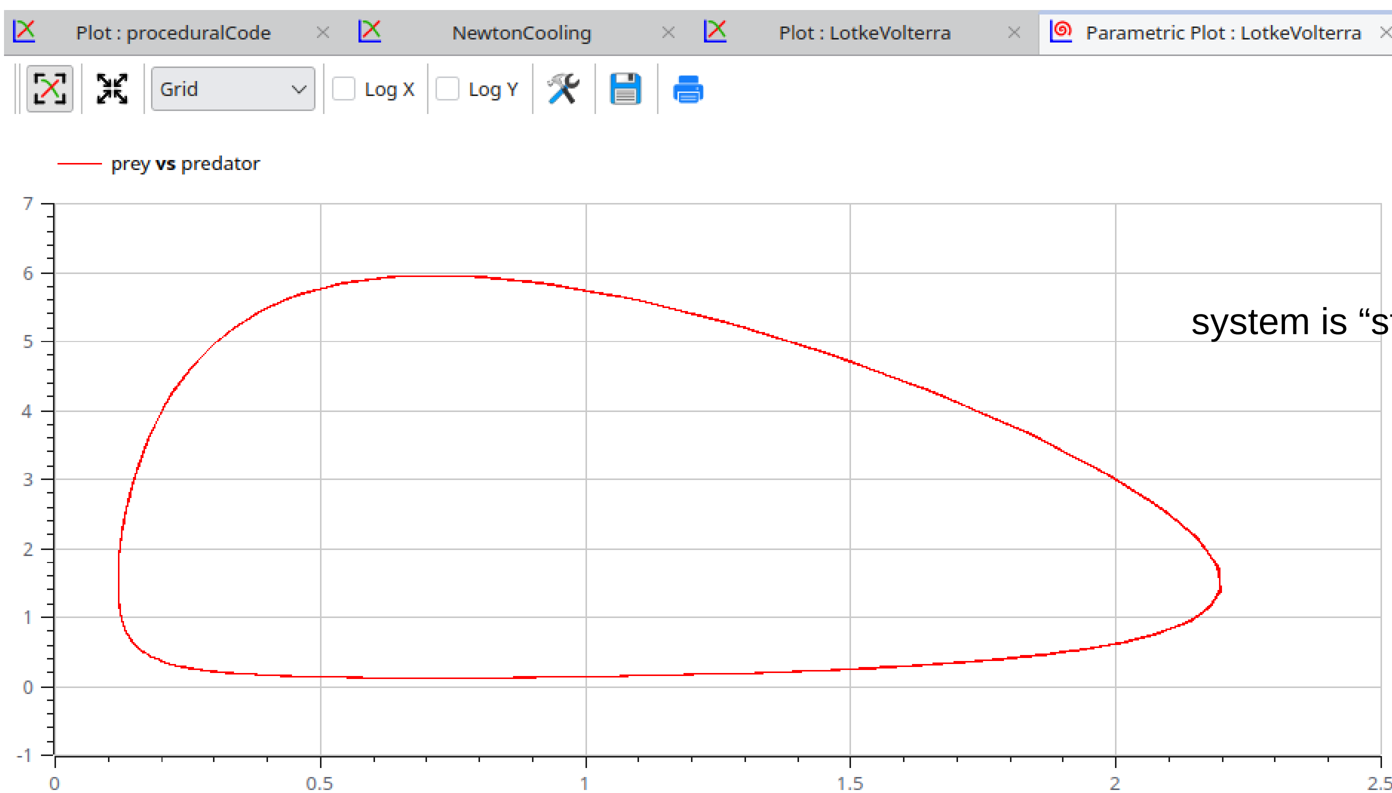
```
model LinearODE
  Real x(start = 0);
  Real comp;
equation
  der(x) =  1;
  comp - x = 0;
end LinearODE;

// x(t) = A * t + B
// x(0) = 0 = B
// dx/dt (0) = A = 1
```



```
model LotkeVolterra "Lotke-Volterra equations modelling a predator-prey system"
  // Types
  type Population=Real(min=0);

  // Parameters
  parameter Population predator_0 = 2 "initial predator population";
  parameter Population prey_0 = 3 "initial prey population";
  parameter Real grazing_factor = 2;
  parameter Real kill_factor    = 7;
  parameter Real excess_death_rate  = 3;
  parameter Real excess_birth_rate  = 5;

  // Variables
  Population predator "predator population";
  Population prey "prey population";

initial equation
  predator = predator_0;
  prey = prey_0;

equation
  der(predator) = -excess_death_rate*predator + grazing_factor*predator*prey;
  der(prey)     =  excess_birth_rate*prey     - kill_factor*predator*prey;
end LotkeVolterra;
```

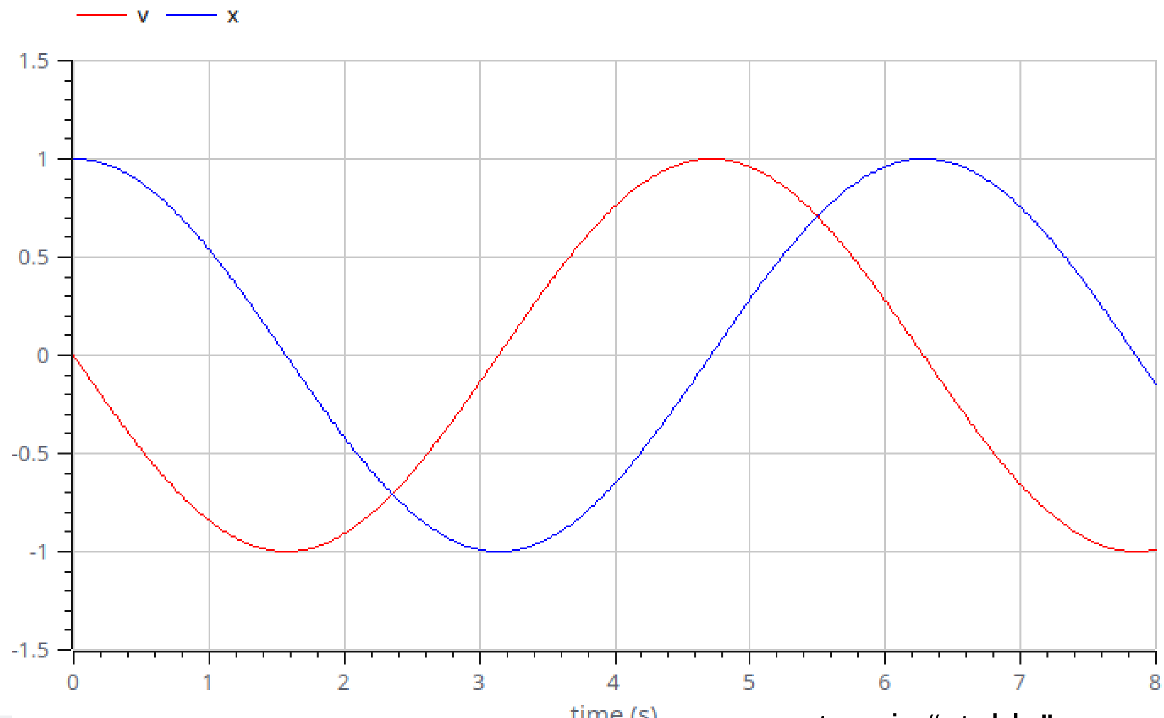




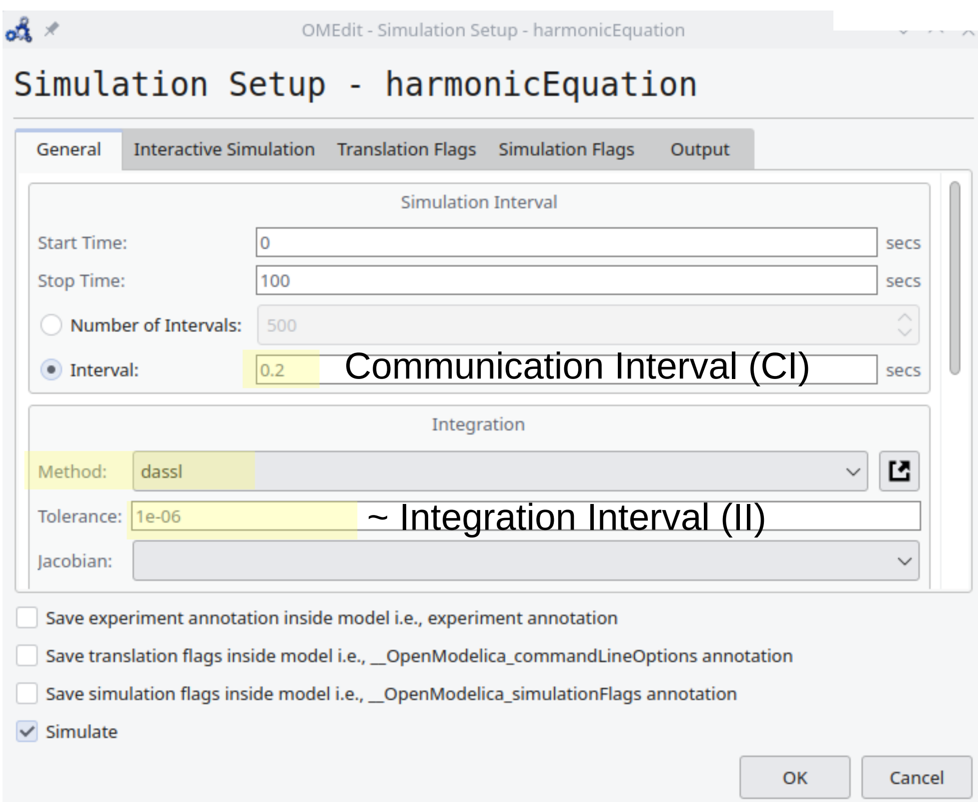


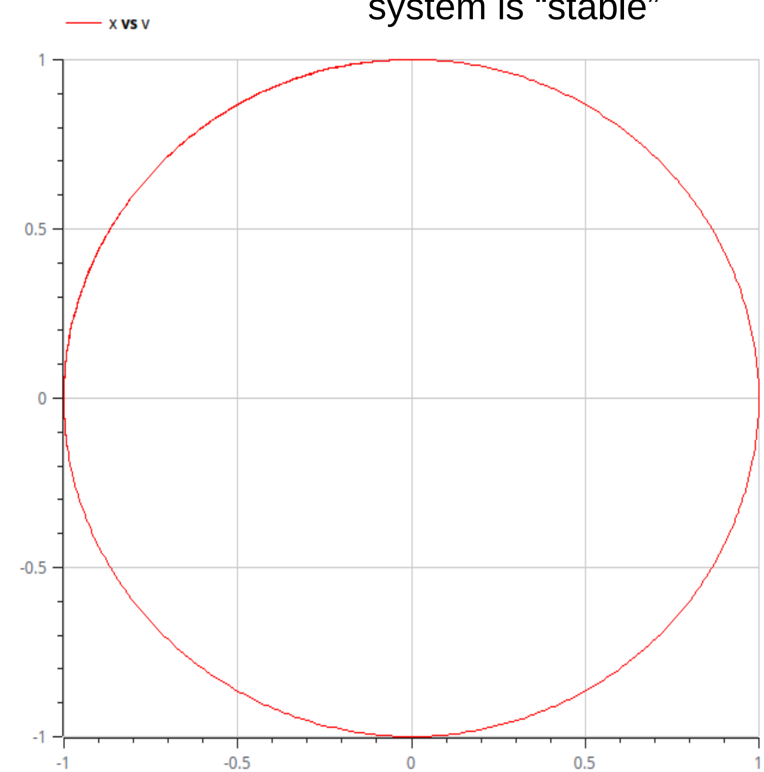
system is “stable”

```
model harmonicEquation
  Real x(start = 1);
  Real v(start = 0);
  Real comp;
equation
  der(x) =  v;
  der(v) = -x;
  comp - x = 0;
end harmonicEquation;

// x(t) = A * sin(t) + B * cos(t)
// x(0) = 1 = Boolean
// v(t) = A * cos(t) - B * sin(t)
// v(0) = 0 = A
//
// x(t) = cos(t)
```

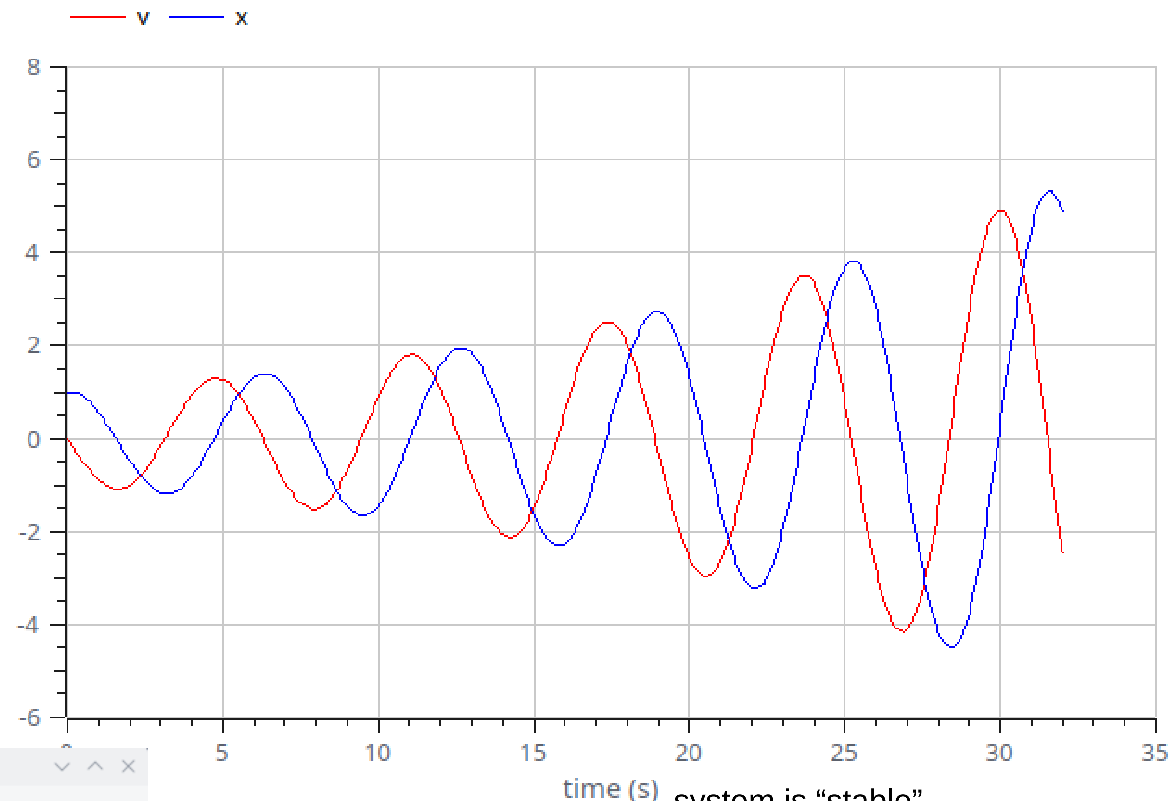


system is “stable”

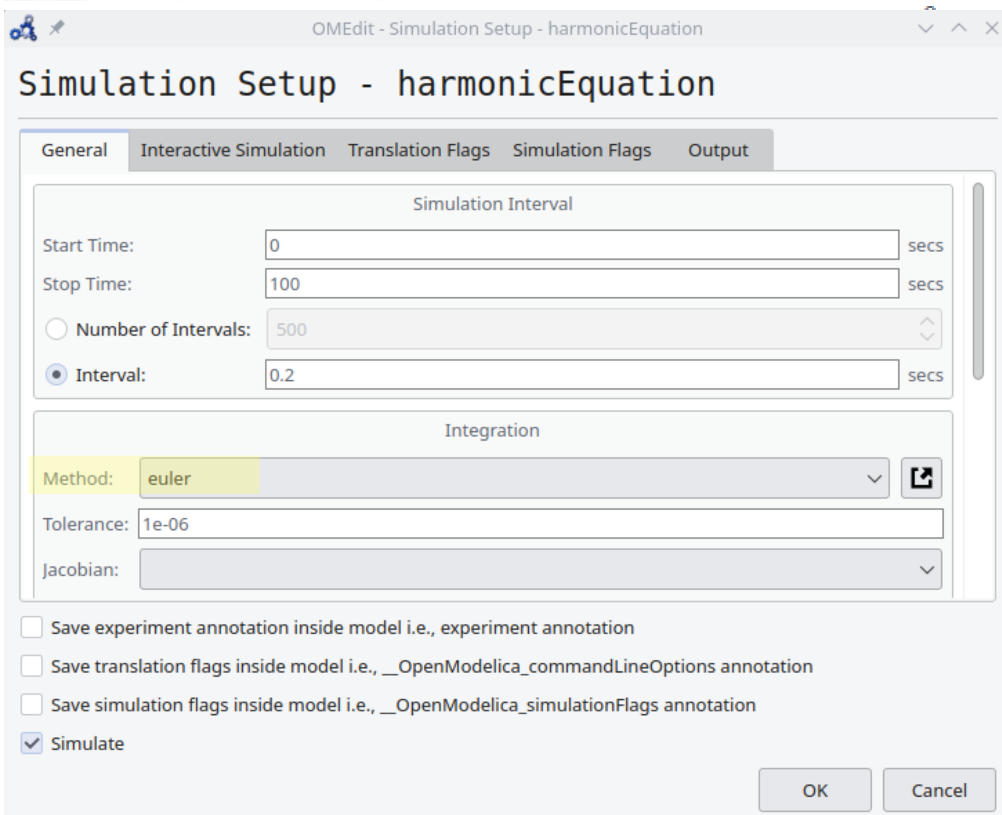


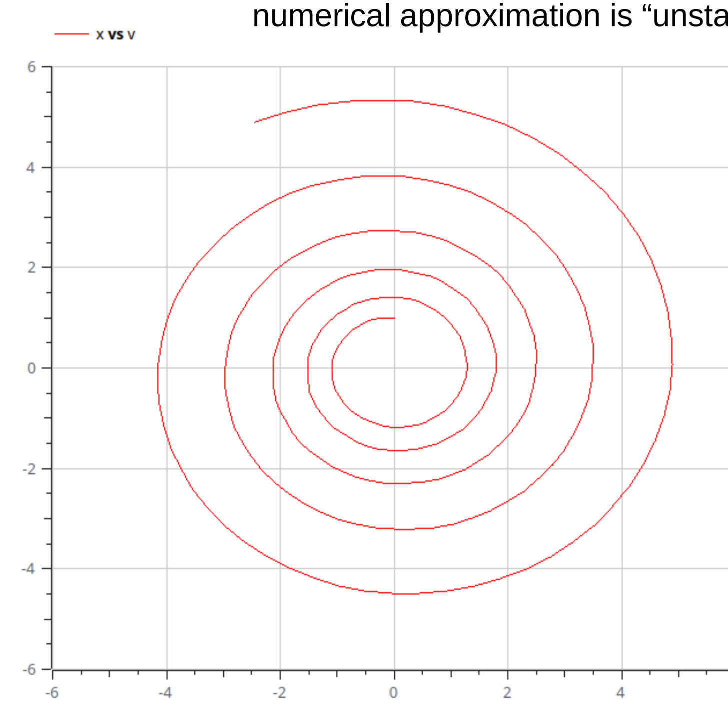


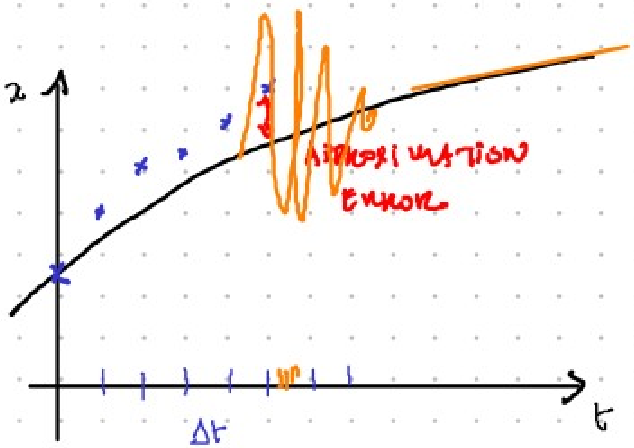
```
model harmonicEquation
  Real x(start = 1);
  Real v(start = 0);
  Real comp;
equation
  der(x) =  v;
  der(v) = -x;
  comp - x = 0;
end harmonicEquation;

// x(t) = A * sin(t) + B * cos(t)
// x(0) = 1 = Boolean
// v(t) = A * cos(t) - B * sin(t)
// v(0) = 0 = A
//
// x(t) = cos(t)
```



system is “stable”
numerical approximation is “unstable”







RATE OF CHANGE

SLOPE OF x(t) at t

$$\begin{cases} \dfrac{d\bar{x}}{dt} = f(\bar{x}, t) \\ \bar{x}(0) = \bar{x}_0 \end{cases}$$

ODE, IVP

DAE

$\bar{x}(t)$?

$$\frac{dx}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

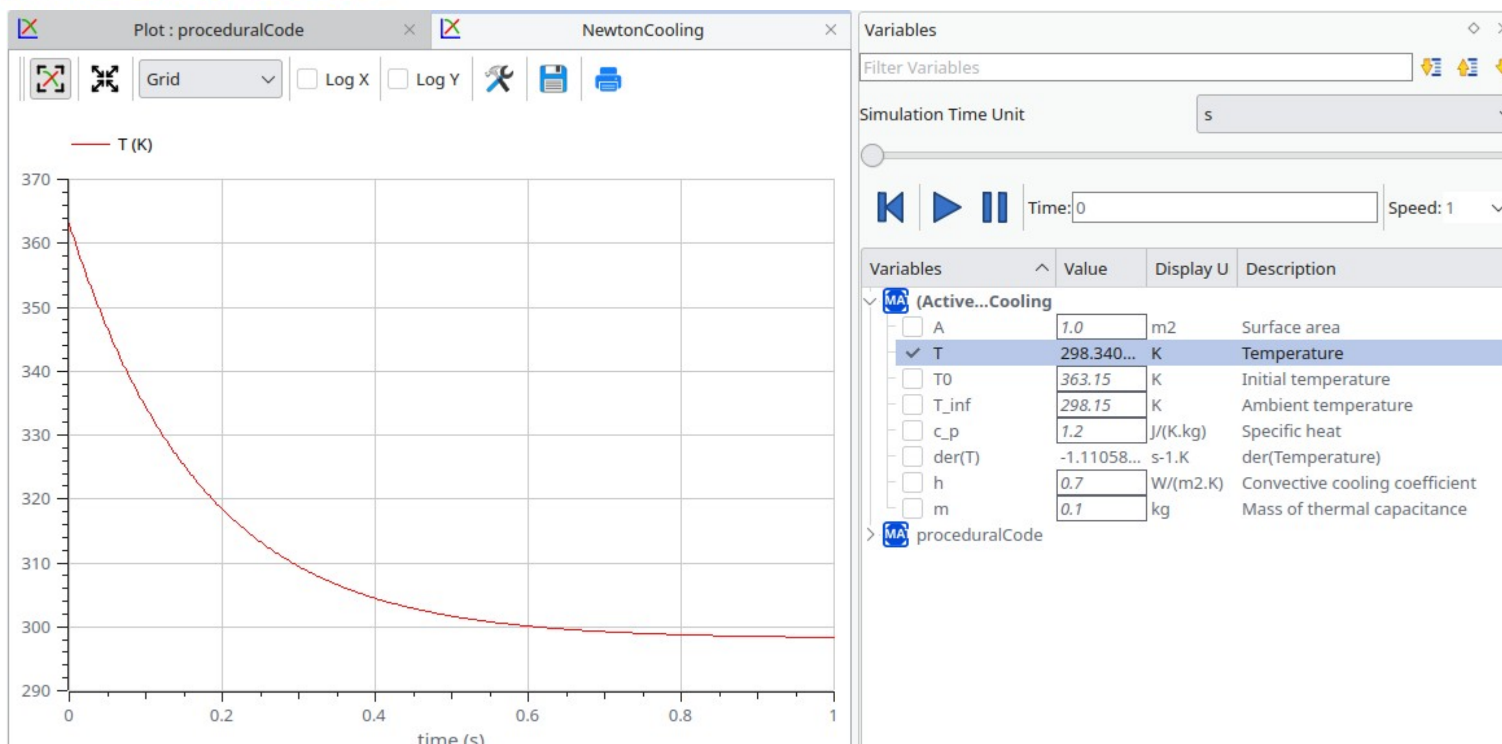
$\Rightarrow$

$$\frac{dx_{APPROX}}{dt} \underset{(\Delta t \ll)}{\approx} \frac{x_A(t+\Delta t) - x_A(t)}{\Delta t}$$

$$x_A(t+\Delta t) = x_A(t) + \Delta t \cdot \frac{dx_A}{dt} \quad \text{“} f(x_A, t) \text{”}$$

"EULER"

```
model NewtonCooling "Cooling example with physical types"
  // Types
  type Temperature=Real(unit="K", min=0);
  type ConvectionCoefficient=Real(unit="W/(m2.K)", min=0);
  type Area=Real(unit="m2", min=0);
  type Mass=Real(unit="kg", min=0);
  type SpecificHeat=Real(unit="J/(K.kg)", min=0);

  // Parameters
  parameter Temperature T_inf=298.15 "Ambient temperature";
  parameter Temperature T0=363.15 "Initial temperature";
  parameter ConvectionCoefficient h=0.7 "Convective cooling coefficient";
  parameter Area A=1.0 "Surface area";
  parameter Mass m=0.1 "Mass of thermal capacitance";
  parameter SpecificHeat c_p=1.2 "Specific heat";

  // Variables
  Temperature T "Temperature";
initial equation
  T = T0 "Specify initial value for T";
equation
  m*c_p*der(T) = h*A*(T_inf-T) "Newton's law of cooling";
end NewtonCooling;
```



Calibration / Parameter Estimation

Newton Cooling Model

```
model NewtonCooling "Cooling example with physical types"
  // Types
  type Temperature=Real(unit="K", min=0);
  type ConvectionCoefficient=Real(unit="W/(m2.K)", min=0);
  type Area=Real(unit="m2", min=0);
  type Mass=Real(unit="kg", min=0);
  type SpecificHeat=Real(unit="J/(K.kg)", min=0);

  // Parameters
  parameter Temperature T_inf=298.15 "Ambient temperature";
  parameter Temperature T0=363.15 "Initial temperature";
  parameter ConvectionCoefficient h=0.7 "Convective cooling coefficient";
  parameter Area A=1.0 "Surface area";
  parameter Mass m=0.1 "Mass of thermal capacitance";
  parameter SpecificHeat c_p=1.2 "Specific heat";

  // Variables
  Temperature T "Temperature";
initial equation
  T = T0 "Specify initial value for T";
equation
  m*c_p*der(T) = h*A*(T_inf-T) "Newton's law of cooling";
end NewtonCooling;
```

Parametrized model

Newton Cooling Model

```
model NewtonCooling "Cooling example with physical types"
  // Types
  type Temperature=Real(unit="K", min=0);
  type ConvectionCoefficient=Real(unit="W/(m2.K)", min=0);
  type Area=Real(unit="m2", min=0);
  type Mass=Real(unit="kg", min=0);
  type SpecificHeat=Real(unit="J/(K.kg)", min=0);

  // Parameters
  parameter Temperature T_inf=298.15 "Ambient temperature";
  parameter Temperature T0=363.15 "Initial temperature";
  parameter ConvectionCoefficient h=0.7 "Convective cooling coefficient";
  parameter Area A=1.0 "Surface area";
  parameter Mass m=0.1 "Mass of thermal capacitance";
  parameter SpecificHeat c_p=1.2 "Specific heat";

  // Variables
  Temperature T "Temperature";
initial equation
  T = T0 "Specify initial value for T";
equation
  m*c_p*der(T) = h*A*(T_inf-T) "Newton's law of cooling";
end NewtonCooling;
```

can be measured

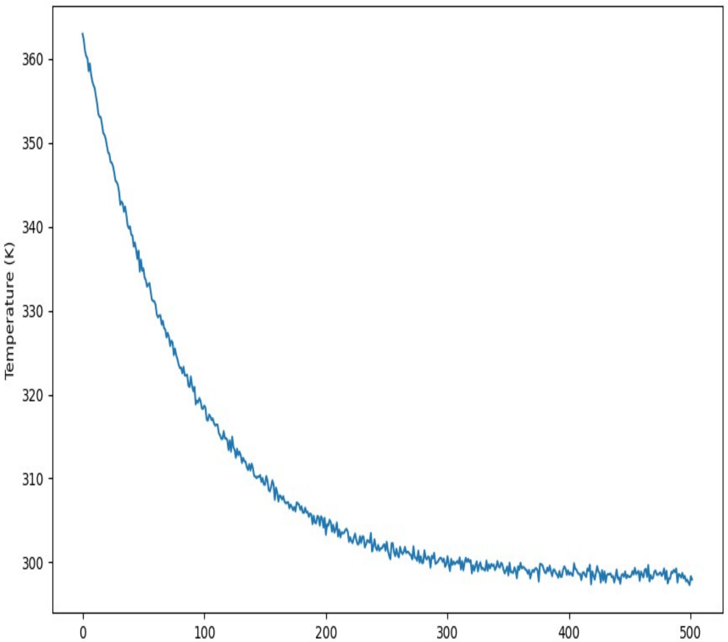
can be calculated

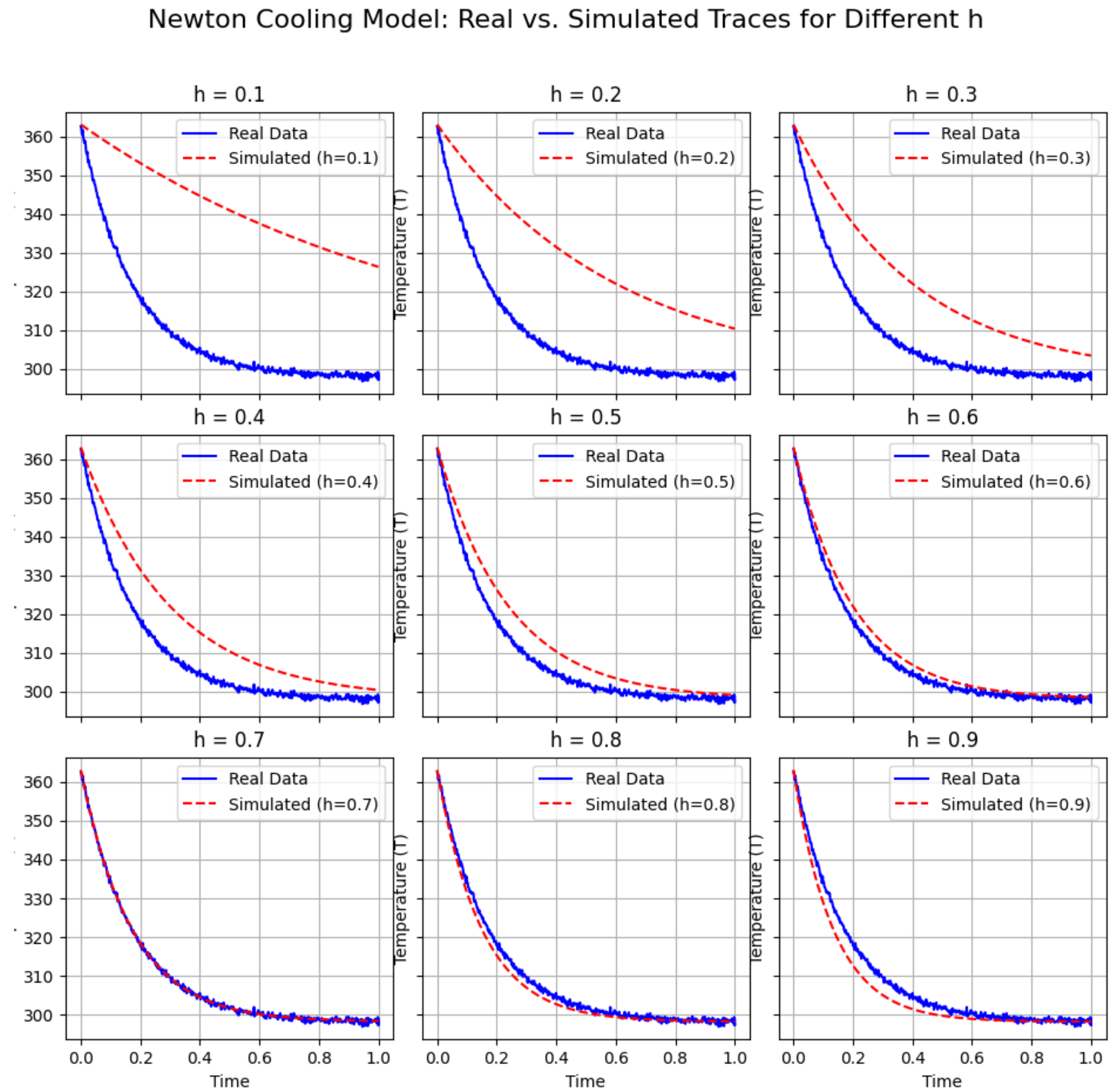
Newton Cooling Model

```
model NewtonCooling "Cooling example with physical types"
  // Types
  type Temperature=Real(unit="K", min=0);
  type ConvectionCoefficient=Real(unit="W/(m2.K)", min=0);
  type Area=Real(unit="m2", min=0);
  type Mass=Real(unit="kg", min=0);
  type SpecificHeat=Real(unit="J/(K.kg)", min=0);

  // Parameters
  parameter Temperature T_inf=298.15 "Ambient temperature";
  parameter Temperature T0=363.15 "Initial temperature";
  parameter ConvectionCoefficient h=0.7 "Convective cooling coefficient";
  parameter Area A=1.0 "Surface area";
  parameter Mass m=0.1 "Mass of thermal capacitance";
  parameter SpecificHeat c_p=1.2 "Specific heat";

  // Variables
  Temperature T "Temperature";
initial equation
  T = T0 "Specify initial value for T";
equation
  m*c_p*der(T) = h*A*(T_inf-T) "Newton's law of cooling";
end NewtonCooling;
```

has to be estimated

Newton Cooling Model



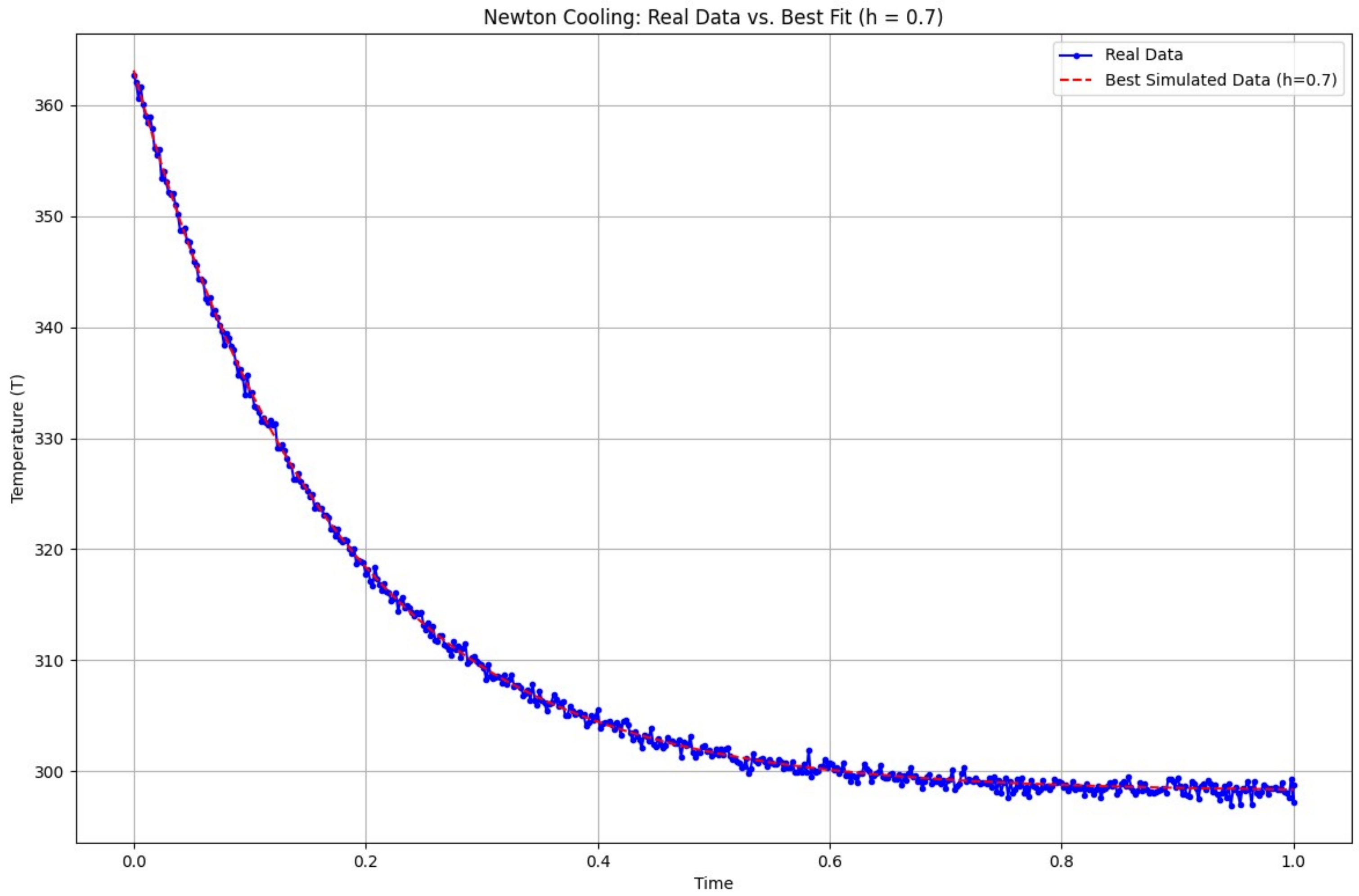


Trace from experiment on real system data with sensor inaccuracies, noise ...

Compare

Multiple traces from simulation with different parameter values for *h*

- distance metric
- search for parameter that yields the smallest distance



Best matching trace →
The corresponding parameter from that simulation →
The best estimate for the parameter value

OO Modelling of Physical Systems

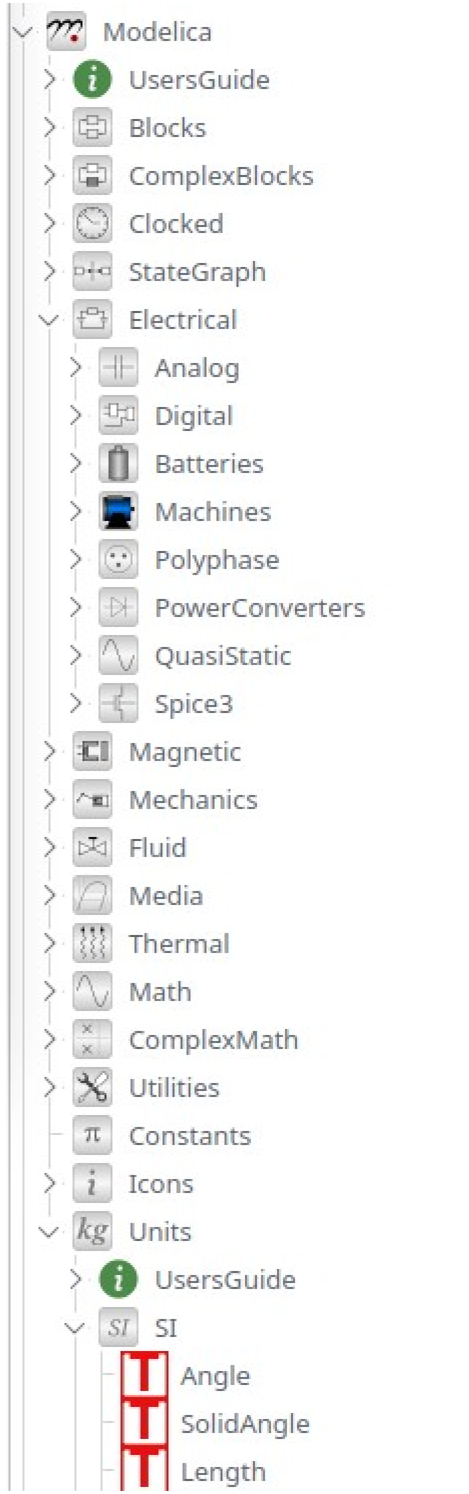
Electrical Types

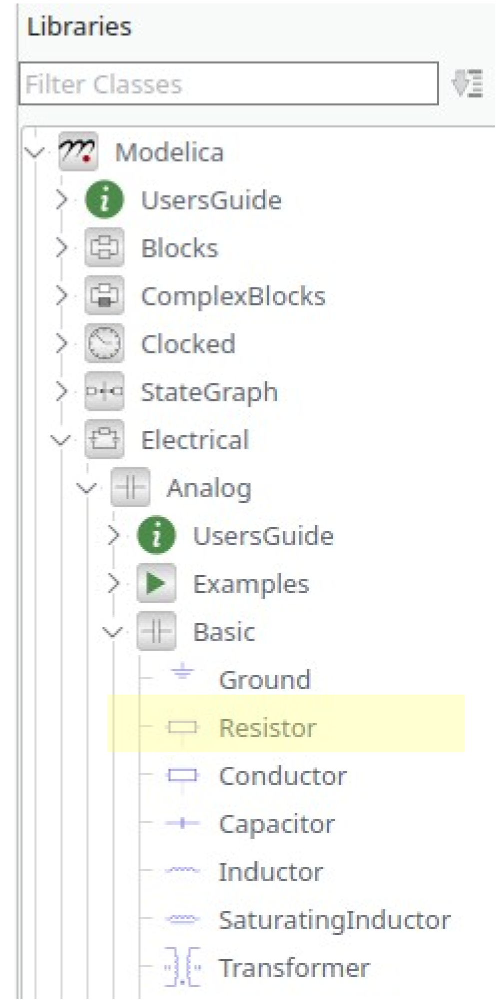
```
type Time = Real (final quantity="Time", final unit="s");
type ElectricPotential = Real (final quantity="ElectricPotential",
                               final unit="V");
type Voltage = ElectricPotential;
type ElectricCurrent = Real (final quantity="ElectricCurrent",
                             final unit="A");
type Current = ElectricCurrent;
```

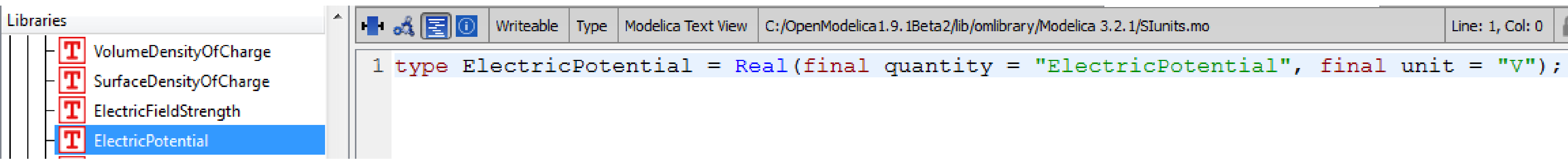
Beware: variables are **signals** (functions of **time**)!

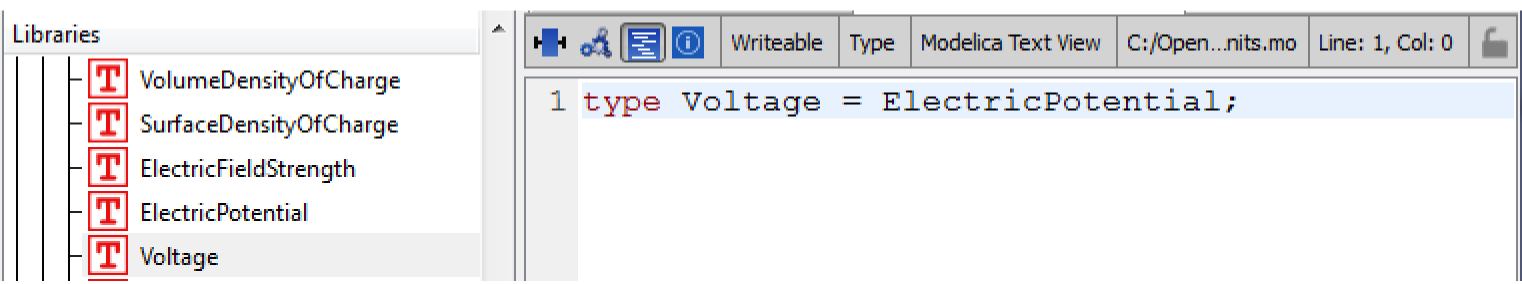
MODELICA

Standard Library (MSL)



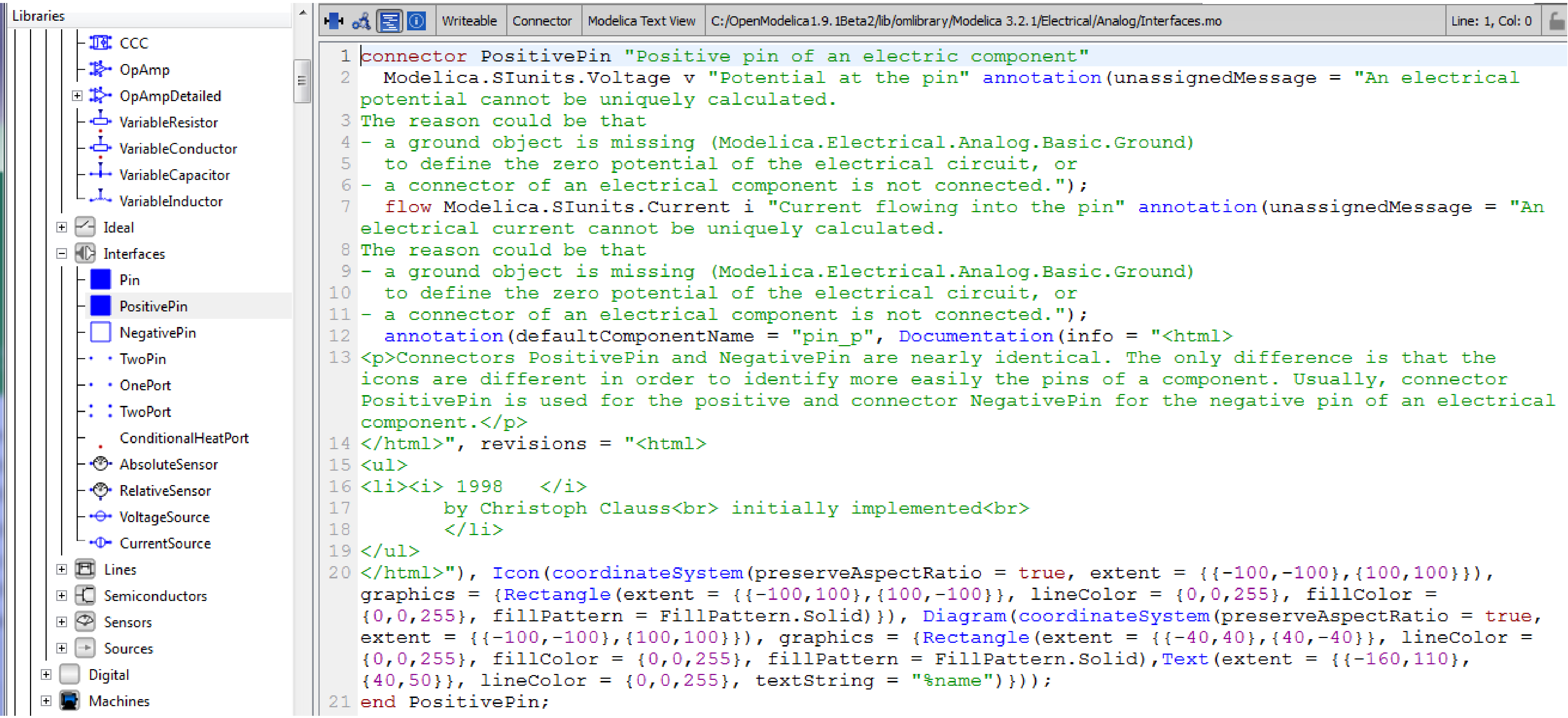



Libraries
VolumeDensityOfCharge
SurfaceDensityOfCharge
ElectricFieldStrength
ElectricPotential
Writeable
Type
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/SIunits.mo
Line: 1, Col: 0
1 type ElectricPotential = Real(final quantity = "ElectricPotential", final unit = "V");


Libraries
VolumeDensityOfCharge
SurfaceDensityOfCharge
ElectricFieldStrength
ElectricPotential
Voltage
Writeable
Type
Modelica Text View
C:/Open...nits.mo
Line: 1, Col: 0
1 type Voltage = ElectricPotential;

Electrical Pin Interface

```
connector PositivePin "Positive pin of an electric component"
       Voltage v "Potential at the pin";
  flow Current i "Current flowing into the pin";
end PositivePin;
```



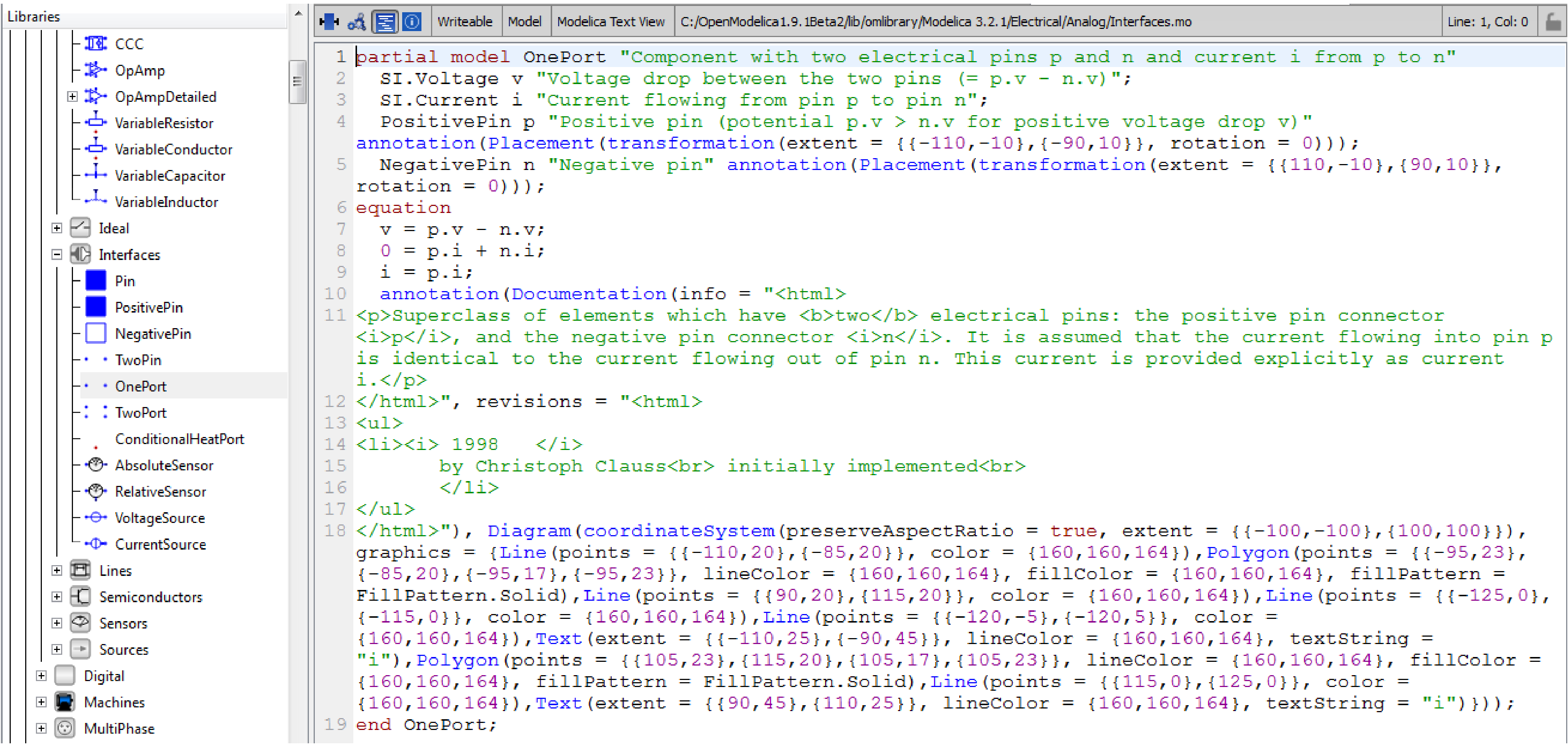
Libraries
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Pin
PositivePin
NegativePin
TwoPin
OnePort
TwoPort
ConditionalHeatPort
AbsoluteSensor
RelativeSensor
VoltageSource
CurrentSource
Lines
Semiconductors
Sensors
Sources
Digital
Machines
Writeable
Connector
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Interfaces.mo
Line: 1, Col: 0
connector PositivePin "Positive pin of an electric component"
Modelica.SIunits.Voltage v "Potential at the pin" annotation(unassignedMessage = "An electrical potential cannot be uniquely calculated.
The reason could be that
- a ground object is missing (Modelica.Electrical.Analog.Basic.Ground)
to define the zero potential of the electrical circuit, or
- a connector of an electrical component is not connected.");
flow Modelica.SIunits.Current i "Current flowing into the pin" annotation(unassignedMessage = "An electrical current cannot be uniquely calculated.
The reason could be that
- a ground object is missing (Modelica.Electrical.Analog.Basic.Ground)
to define the zero potential of the electrical circuit, or
- a connector of an electrical component is not connected.");
annotation(defaultComponentName = "pin_p", Documentation(info = "<html>
<p>Connectors PositivePin and NegativePin are nearly identical. The only difference is that the icons are different in order to identify more easily the pins of a component. Usually, connector PositivePin is used for the positive and connector NegativePin for the negative pin of an electrical component.</p>
</html>", revisions = "<html>

<i> 1998 </i>
by Christoph Clauss
 initially implemented

</html>"), Icon(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-100,100},{100,-100}}, lineColor = {0,0,255}, fillColor = {0,0,255}, fillPattern = FillPattern.Solid)}), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-40,40},{40,-40}}, lineColor = {0,0,255}, fillColor = {0,0,255}, fillPattern = FillPattern.Solid),Text(extent = {{-160,110},{40,50}}, lineColor = {0,0,255}, textString = "%name")}));
end PositivePin;

Electrical Port

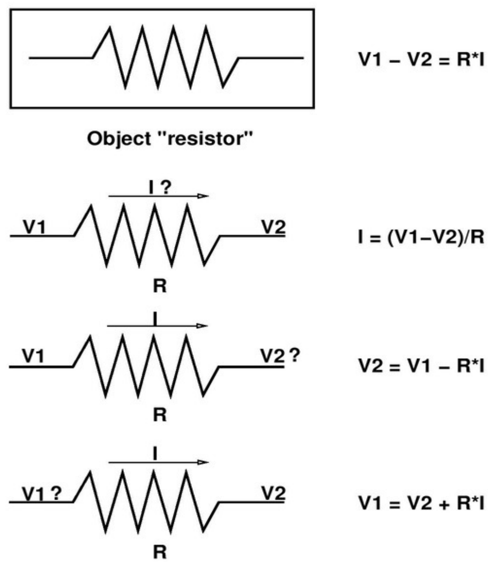
```
partial model OnePort
  "Component with two electrical pins p and n
   and current i from p to n"
  Voltage v "Voltage drop between the two pins (= p.v - n.v)";
  Current i "Current flowing from pin p to pin n";
  PositivePin p;
  NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```


Libraries
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Pin
PositivePin
NegativePin
TwoPin
OnePort
TwoPort
ConditionalHeatPort
AbsoluteSensor
RelativeSensor
VoltageSource
CurrentSource
Lines
Semiconductors
Sensors
Sources
Digital
Machines
MultiPhase
Writeable
Model
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Interfaces.mo
Line: 1, Col: 0
partial model OnePort "Component with two electrical pins p and n and current i from p to n"
SI.Voltage v "Voltage drop between the two pins (= p.v - n.v)";
SI.Current i "Current flowing from pin p to pin n";
PositivePin p "Positive pin (potential p.v > n.v for positive voltage drop v)" annotation(Placement(transformation(extent = {{-110,-10},{-90,10}}, rotation = 0)));
NegativePin n "Negative pin" annotation(Placement(transformation(extent = {{110,-10},{90,10}}, rotation = 0)));
equation
v = p.v - n.v;
0 = p.i + n.i;
i = p.i;
annotation(Documentation(info = "<html>
<p>Superclass of elements which have two electrical pins: the positive pin connector <i>p</i>, and the negative pin connector <i>n</i>. It is assumed that the current flowing into pin p is identical to the current flowing out of pin n. This current is provided explicitly as current i.</p>
</html>", revisions = "<html>

<i> 1998 </i>
by Christoph Clauss
 initially implemented

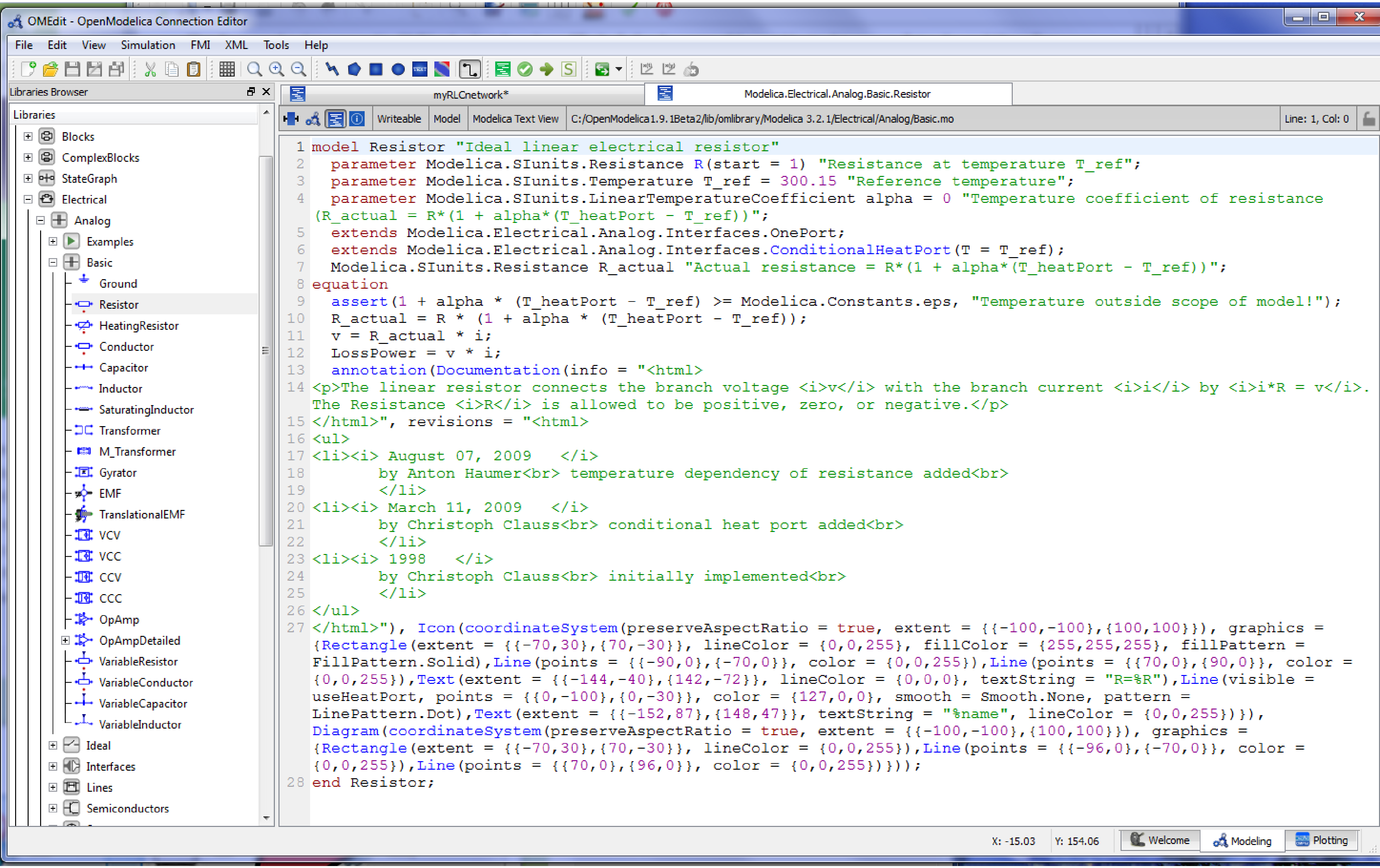
</html>"), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Line(points = {{-110,20},{-85,20}}, color = {160,160,164}),Polygon(points = {{-95,23},{-85,20},{-95,17},{-95,23}}, lineColor = {160,160,164}, fillColor = {160,160,164}, fillPattern = FillPattern.Solid),Line(points = {{90,20},{115,20}}, color = {160,160,164}),Line(points = {{-125,0},{-115,0}}, color = {160,160,164}),Line(points = {{-120,-5},{-120,5}}, color = {160,160,164}),Text(extent = {{-110,25},{-90,45}}, lineColor = {160,160,164}, textString = "i"),Polygon(points = {{105,23},{115,20},{105,17},{105,23}}, lineColor = {160,160,164}, fillColor = {160,160,164}, fillPattern = FillPattern.Solid),Line(points = {{115,0},{125,0}}, color = {160,160,164}),Text(extent = {{90,45},{110,25}}, lineColor = {160,160,164}, textString = "i")}));
end OnePort;

Object-oriented re-use and causality



Electrical Resistor

```
model Resistor "Ideal linear electrical resistor"
  extends OnePort;
  parameter Resistance R=1 "Resistance";
  equation
    R*i = v;
end Resistor;
```

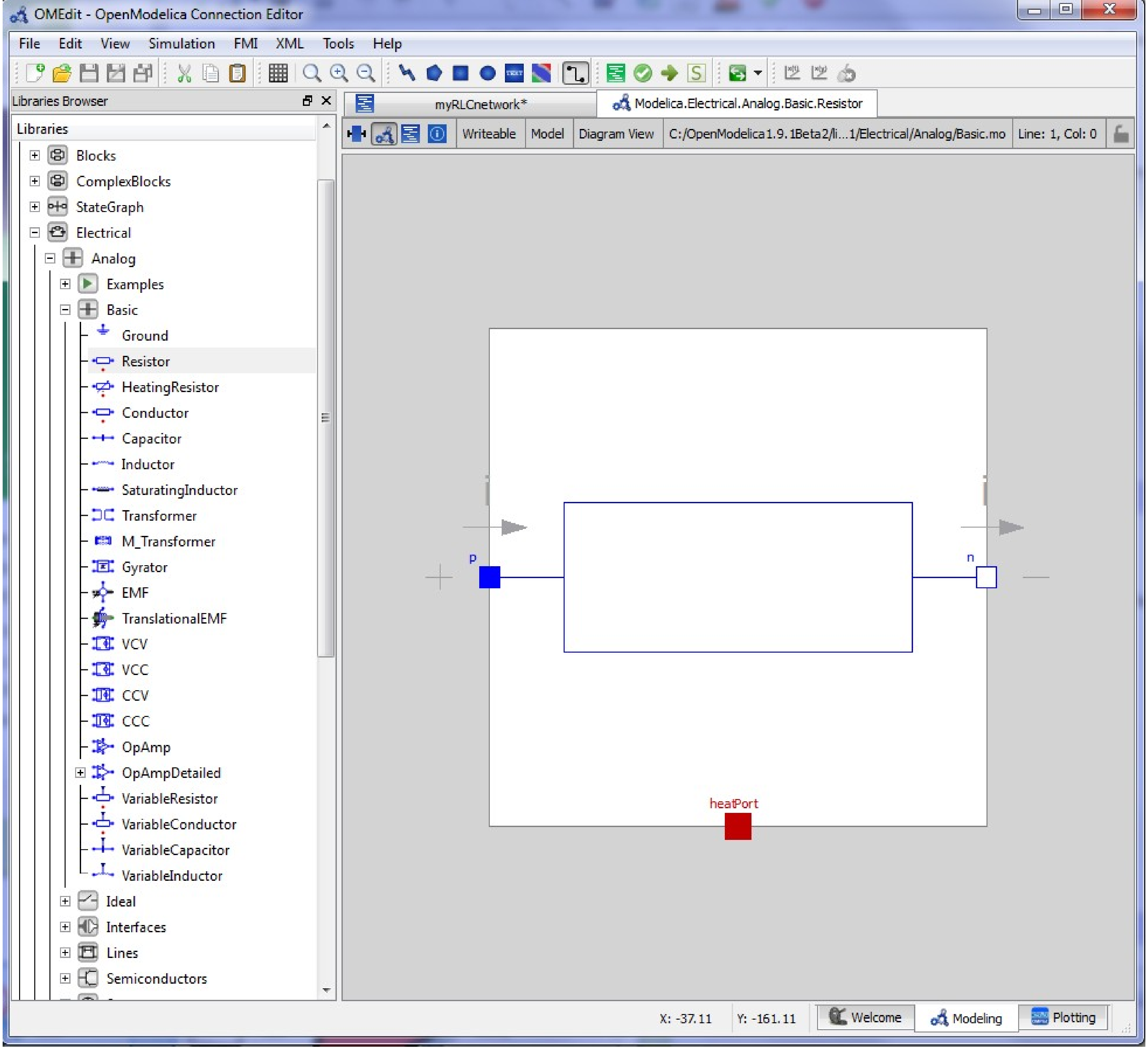

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
HeatingResistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
EMF
TranslationalEMF
VCV
VCC
CCV
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Lines
Semiconductors
myRLCnetwork*
Modelica.Electrical.Analog.Basic.Resistor
Writeable
Model
Modelica Text View
C:/OpenModelica1.9.1Beta2/lib/omlibrary/Modelica 3.2.1/Electrical/Analog/Basic.mo
Line: 1, Col: 0
model Resistor "Ideal linear electrical resistor"
parameter Modelica.SIunits.Resistance R(start = 1) "Resistance at temperature T_ref";
parameter Modelica.SIunits.Temperature T_ref = 300.15 "Reference temperature";
parameter Modelica.SIunits.LinearTemperatureCoefficient alpha = 0 "Temperature coefficient of resistance (R_actual = R*(1 + alpha*(T_heatPort - T_ref))";
extends Modelica.Electrical.Analog.Interfaces.OnePort;
extends Modelica.Electrical.Analog.Interfaces.ConditionalHeatPort(T = T_ref);
Modelica.SIunits.Resistance R_actual "Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))";
equation
assert(1 + alpha * (T_heatPort - T_ref) >= Modelica.Constants.eps, "Temperature outside scope of model!");
R_actual = R * (1 + alpha * (T_heatPort - T_ref));
v = R_actual * i;
LossPower = v * i;
annotation(Documentation(info = "<html>
<p>The linear resistor connects the branch voltage <i>v</i> with the branch current <i>i</i> by <i>i*R = v</i>. The Resistance <i>R</i> is allowed to be positive, zero, or negative.</p>
</html>", revisions = "<html>

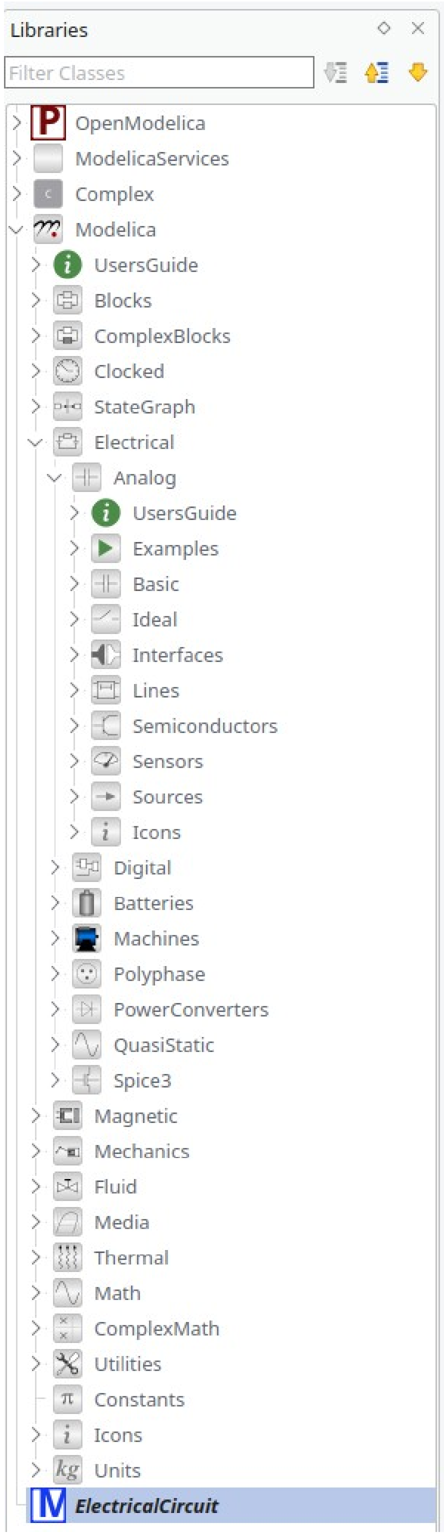
<i> August 07, 2009 </i>
by Anton Haumer
 temperature dependency of resistance added

<i> March 11, 2009 </i>
by Christoph Clauss
 conditional heat port added

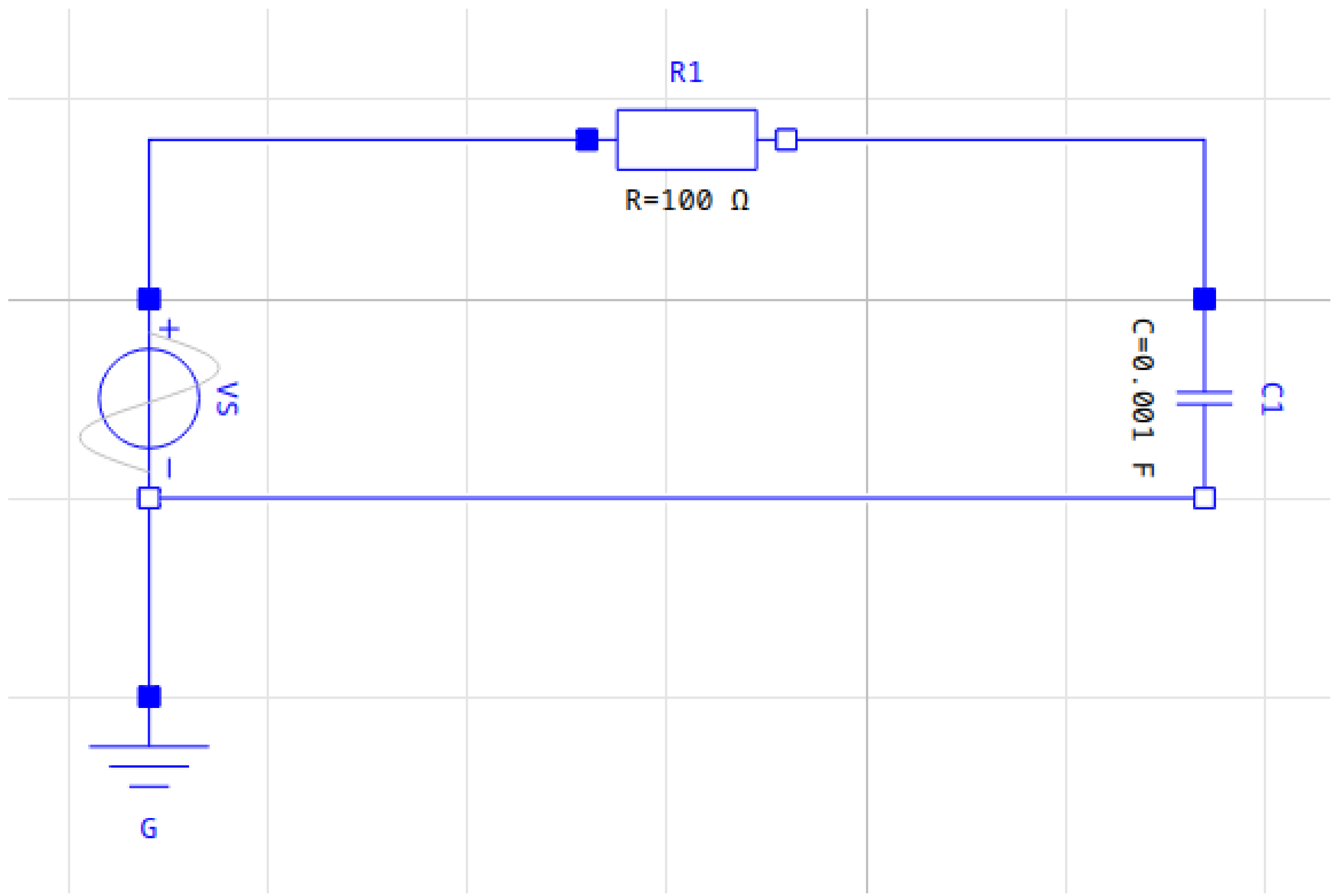
<i> 1998 </i>
by Christoph Clauss
 initially implemented

</html>"), Icon(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-70,30},{70,-30}}, lineColor = {0,0,255}, fillColor = {255,255,255}, fillPattern = FillPattern.Solid),Line(points = {{-90,0},{-70,0}}, color = {0,0,255}),Line(points = {{70,0},{90,0}}, color = {0,0,255}),Text(extent = {{-144,-40},{142,-72}}, lineColor = {0,0,0}, textString = "R=%R"),Line(visible = useHeatPort, points = {{0,-100},{0,-30}}, color = {127,0,0}, smooth = Smooth.None, pattern = LinePattern.Dot),Text(extent = {{-152,87},{148,47}}, textString = "%name", lineColor = {0,0,255})}), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}), graphics = {Rectangle(extent = {{-70,30},{70,-30}}, lineColor = {0,0,255}),Line(points = {{-96,0},{-70,0}}, color = {0,0,255}),Line(points = {{70,0},{96,0}}, color = {0,0,255})}));
end Resistor;
X: -15.03
Y: 154.06
Welcome
Modeling
Plotting

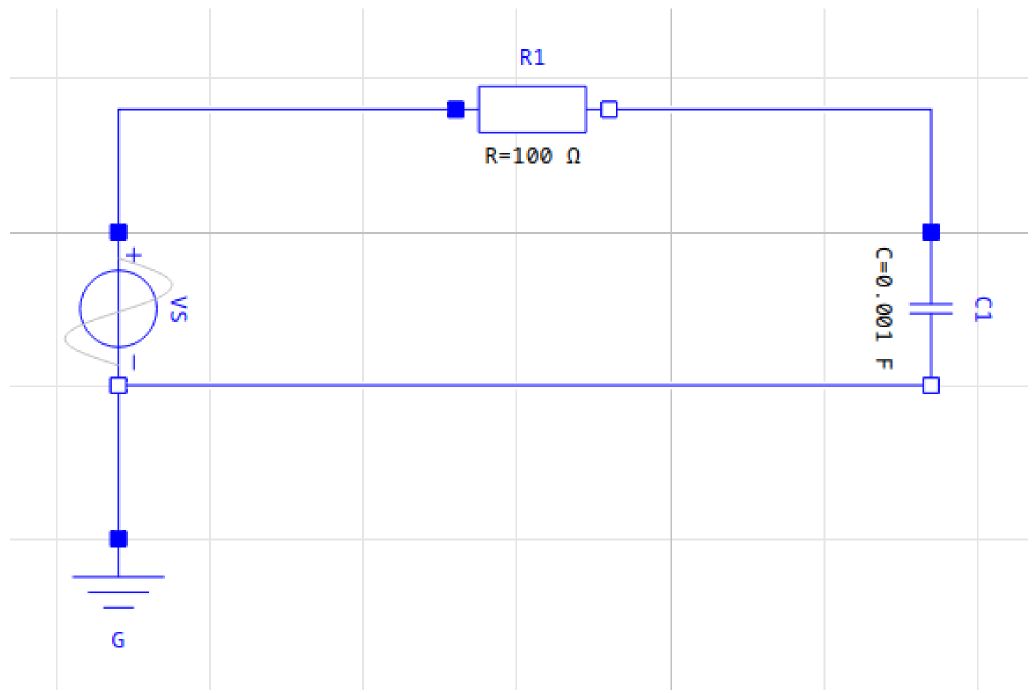

OMEdit - OpenModelica Connection Editor
File
Edit
View
Simulation
FMI
XML
Tools
Help
Libraries Browser
Libraries
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
HeatingResistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
EMF
TranslationalEMF
VCV
VCC
CCV
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Lines
Semiconductors
myRLCnetwork*
Modelica.Electrical.Analog.Basic.Resistor
Writeable
Model
Diagram View
C:/OpenModelica1.9.1Beta2/li...1/Electrical/Analog/Basic.mo
Line: 1, Col: 0
p
n
heatPort
X: -37.11
Y: -161.11
Welcome
Modeling
Plotting



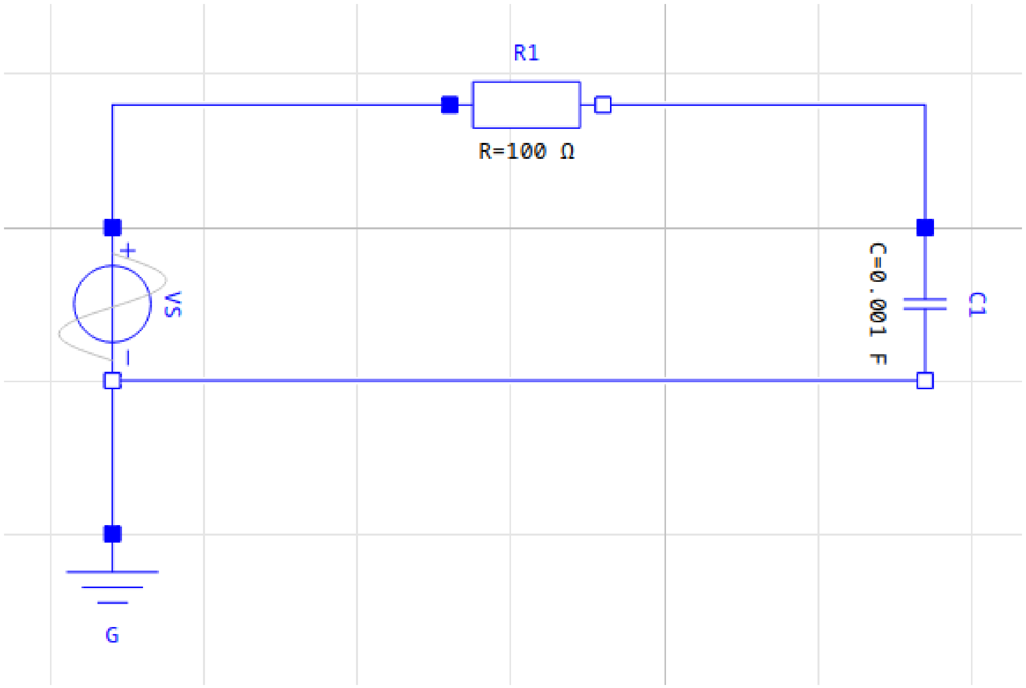
“low (frequency) pass filter”



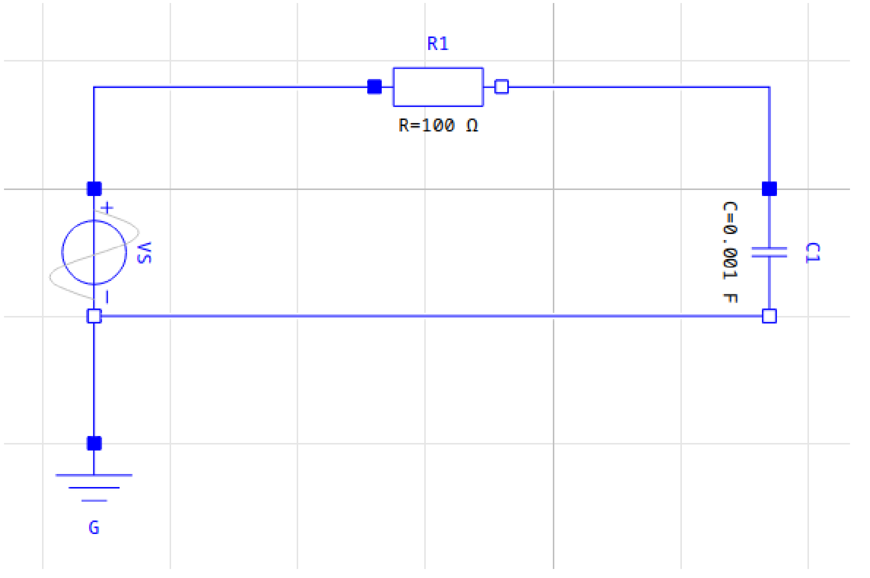
```
model ElectricalCircuit
  Modelica.Electrical.Analog.Basic.Ground G annotation( ...);
  Modelica.Electrical.Analog.Basic.Resistor R1(R = 100)  annotation( ...);
  Modelica.Electrical.Analog.Basic.Capacitor C1(C = 0.001)  annotation( ...);
  Modelica.Electrical.Analog.Sources.SineVoltage VS(V = 220, f (displayUnit = "Hz")= 50)
annotation( ...);
equation
  connect(R1.p, VS.p) annotation( ...);
  connect(VS.n, G.p) annotation( ...);
  connect(VS.n, C1.n) annotation( ...);
  connect(R1.n, C1.p) annotation( ...);

annotation( ...);
end ElectricalCircuit;
```



```
model ElectricalCircuit
  Modelica.Electrical.Analog.Basic.Ground G annotation(
    Placement(visible = true, transformation(origin = {-72, -50}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Modelica.Electrical.Analog.Basic.Resistor R1(R = 100)  annotation(
    Placement(transformation(origin = {-18, 16}, extent = {{-10, -10}, {10, 10}})));
  Modelica.Electrical.Analog.Basic.Capacitor C1(C = 0.001)  annotation(
    Placement(visible = true, transformation(origin = {34, -10}, extent = {{-10, -10}, {10, 10}}, rotation = -90)));
  Modelica.Electrical.Analog.Sources.SineVoltage VS(V = 220, f (displayUnit = "Hz")= 50)  annotation(
    Placement(visible = true, transformation(origin = {-72, -10}, extent = {{-10, -10}, {10, 10}}, rotation = -90)));
equation
  connect(R1.p, VS.p) annotation(
    Line(points = {{-28, 16}, {-72, 16}, {-72, 0}}, color = {0, 0, 255}));
  connect(VS.n, G.p) annotation(
    Line(points = {{-72, -20}, {-72, -20}, {-72, -40}, {-72, -40}}, color = {0, 0, 255}));
  connect(VS.n, C1.n) annotation(
    Line(points = {{-72, -20}, {34, -20}}, color = {0, 0, 255}));
  connect(R1.n, C1.p) annotation(
    Line(points = {{-8, 16}, {34, 16}, {34, 0}}, color = {0, 0, 255}));

annotation(
    uses(Modelica(version = "4.0.0")));
end ElectricalCircuit;
```



```
model ElectricalCircuit
  Modelica.Electrical.Analog.Basic.Ground G annotation( ...);
  Modelica.Electrical.Analog.Basic.Resistor R1(R = 100)  annotation( ...);
  Modelica.Electrical.Analog.Basic.Capacitor C1(C = 0.001)  annotation( ...);
  Modelica.Electrical.Analog.Sources.SineVoltage VS(V = 220, f (displayUnit = "Hz")= 50)
annotation( ...);
equation
  connect(R1.p, VS.p) annotation( ...);
  connect(VS.n, G.p) annotation( ...);
  connect(VS.n, C1.n) annotation( ...);
  connect(R1.n, C1.p) annotation( ...);

annotation( ...);
end ElectricalCircuit;
```



Meaning: set of Differential Algebraic Equations (DAEs) obtained by

1.a. expanding inheritance
1.b. instantiation of classes
2. flattening hierarchy, constructing unique names
3. expanding connect() into equations (across vs. flow)

Meaning: set of Differential Algebraic Equations (DAEs) obtained by

1.a. expanding inheritance
1.b. instantiation of classes
2. flattening hierarchy, constructing unique names
3. expanding connect() into equations (across vs. flow)

[1] 00:29:59 Scripting Notification
Check of ElectricalCircuit completed successfully.
Class ElectricalCircuit has 24 equation(s) and 24 variable(s).
16 of these are trivial equation(s).

```
class ElectricalCircuit
  Real G.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real G.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Real R1.R(quantity = "Resistance", unit = "Ohm", start = 1.0) = 100.0 "Resistance at temperature T_ref";
  parameter Real R1.T_ref(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal = 300.0) = 300.15 "Reference temperature";
  parameter Real R1.alpha(quantity = "LinearTemperatureCoefficient", unit = "1/K") = 0.0 "Temperature coefficient of resistance (R_actual = R*(1 + alpha*(T_heatPort - T_ref))";
  Real R1.v(quantity = "ElectricPotential", unit = "V") "Voltage drop of the two pins (= p.v - n.v)";
  Real R1.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real R1.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real R1.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real R1.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real R1.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  final parameter Boolean R1.useHeatPort = false "= true, if heatPort is enabled";
  parameter Real R1.T(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal = 300.0) = R1.T_ref "Fixed device temperature if useHeatPort = false";
  Real R1.LossPower(quantity = "Power", unit = "W") "Loss power leaving component via heatPort";
  Real R1.T_heatPort(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal = 300.0) "Temperature of heatPort";
  Real R1.R_actual(quantity = "Resistance", unit = "Ohm") "Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))";
  Real C1.v(quantity = "ElectricPotential", unit = "V", start = 0.0) "Voltage drop of the two pins (= p.v - n.v)";
  Real C1.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real C1.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real C1.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real C1.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real C1.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  parameter Real C1.C(quantity = "Capacitance", unit = "F", min = 0.0, start = 1.0) = 0.001 "Capacitance";
  parameter Real VS.V(quantity = "ElectricPotential", unit = "V", start = 1.0) = 220.0 "Amplitude of sine wave";
  parameter Real VS.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = 0.0 "Phase of sine wave";
  parameter Real VS.f(quantity = "Frequency", unit = "Hz", displayUnit = "Hz", start = 1.0) = 50.0 "Frequency of sine wave";
  Real VS.v(quantity = "ElectricPotential", unit = "V") "Voltage drop of the two pins (= p.v - n.v)";
  Real VS.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real VS.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real VS.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real VS.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real VS.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  final parameter Real VS.signalSource.amplitude = VS.V "Amplitude of sine wave";
  final parameter Real VS.signalSource.f(quantity = "Frequency", unit = "Hz", start = 1.0) = VS.f "Frequency of sine wave";
  final parameter Real VS.signalSource.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = VS.phase "Phase of sine wave";
  Real VS.signalSource.y "Connector of Real output signal";
  final parameter Real VS.signalSource.offset = VS.offset "Offset of output signal y";
  final parameter Real VS.signalSource.startTime(quantity = "Time", unit = "s") = VS.startTime "Output y = offset for time < startTime";
  parameter Real VS.offset(quantity = "ElectricPotential", unit = "V") = 0.0 "Voltage offset";
  parameter Real VS.startTime(quantity = "Time", unit = "s") = 0.0 "Time offset";
```

Meaning: set of Differential Algebraic Equations (DAEs) obtained by

1.a. expanding inheritance
1.b. instantiation of classes
2. flattening hierarchy, constructing unique names
3. expanding connect() into equations (across vs. flow)

[1] 00:29:59 Scripting Notification
Check of ElectricalCircuit completed successfully.
Class ElectricalCircuit has 24 equation(s) and 24 variable(s).
16 of these are trivial equation(s).

```
equation
  R1.p.v = VS.p.v;
  VS.n.v = C1.n.v;
  VS.n.v = G.p.v;
  R1.n.v = C1.p.v;
  VS.n.i + C1.n.i + G.p.i = 0.0;
  VS.p.i + R1.p.i = 0.0;
  C1.p.i + R1.n.i = 0.0;
  G.p.v = 0.0;
  assert(1.0 + R1.alpha * (R1.T_heatPort - R1.T_ref) >= 1e-15, "Temperature outside scope of model!");
  R1.R_actual = R1.R * (1.0 + R1.alpha * (R1.T_heatPort - R1.T_ref));
  R1.v = R1.R_actual * R1.i;
  R1.LossPower = R1.v * R1.i;
  R1.T_heatPort = R1.T;
  0.0 = R1.p.i + R1.n.i;
  R1.i = R1.p.i;
  R1.v = R1.p.v - R1.n.v;
  C1.i = C1.C * der(C1.v);
  0.0 = C1.p.i + C1.n.i;
  C1.i = C1.p.i;
  C1.v = C1.p.v - C1.n.v;
  VS.signalSource.y = VS.signalSource.offset + (if time < VS.signalSource.startTime then 0.0 else VS.signalSource.amplitude *
sin(6.283185307179586 * VS.signalSource.f * (time - VS.signalSource.startTime) + VS.signalSource.phase));
  VS.v = VS.signalSource.y;
  0.0 = VS.p.i + VS.n.i;
  VS.i = VS.p.i;
  VS.v = VS.p.v - VS.n.v;
end ElectricalCircuit;
```

"across" variables are equal in a connection node

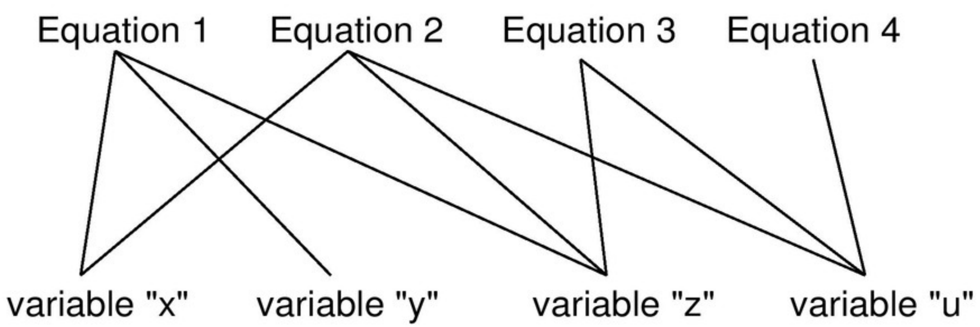
"through/flow" variables sum to 0 in a connection node

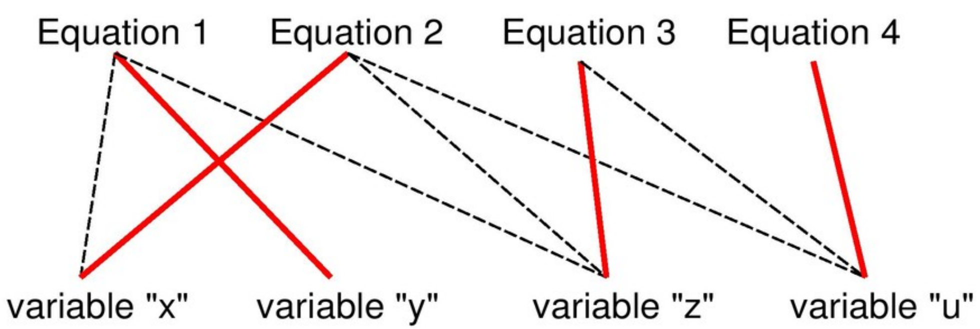
equations with unique variable names, after flattening

Non-causal model
(*e.g.*, from physical conservation laws)

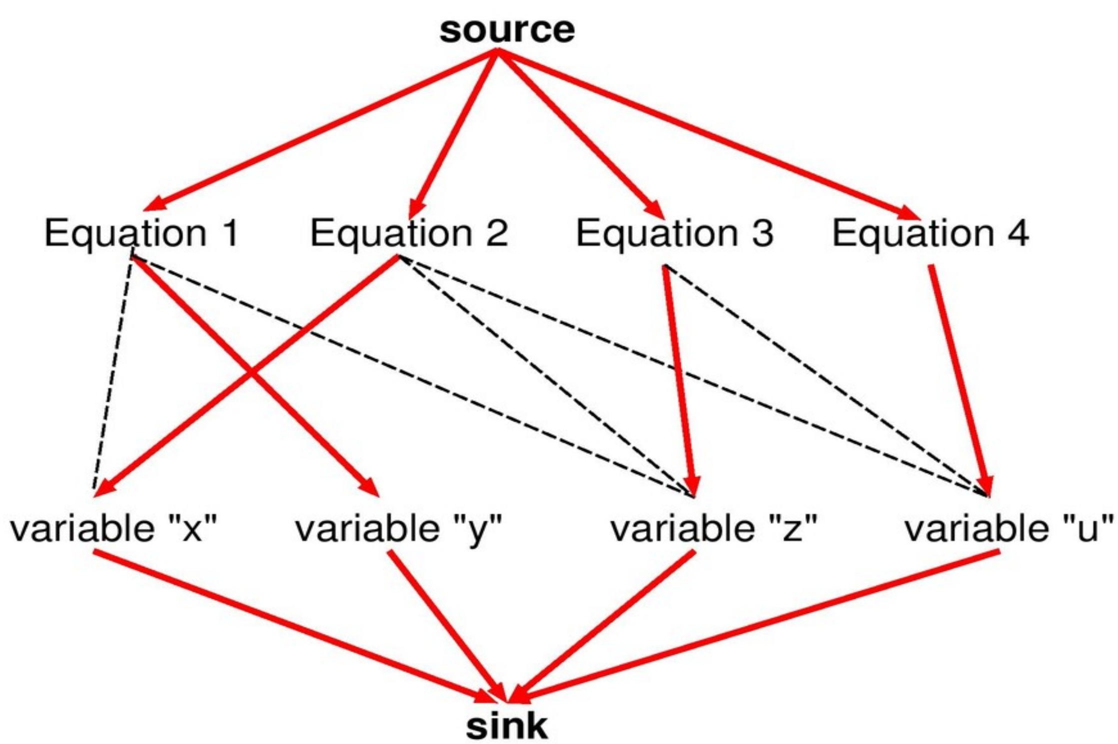
$$\begin{cases} x + y + z & = & 0 & \text{Equation 1} \\ x + 3z + u^2 & = & 0 & \text{Equation 2} \\ z - u - 16 & = & 0 & \text{Equation 3} \\ u - 5 & = & 0 & \text{Equation 4} \end{cases}$$

Causality assignment: bipartite graph, maximum cardinality matching





Causality assignment: network flow



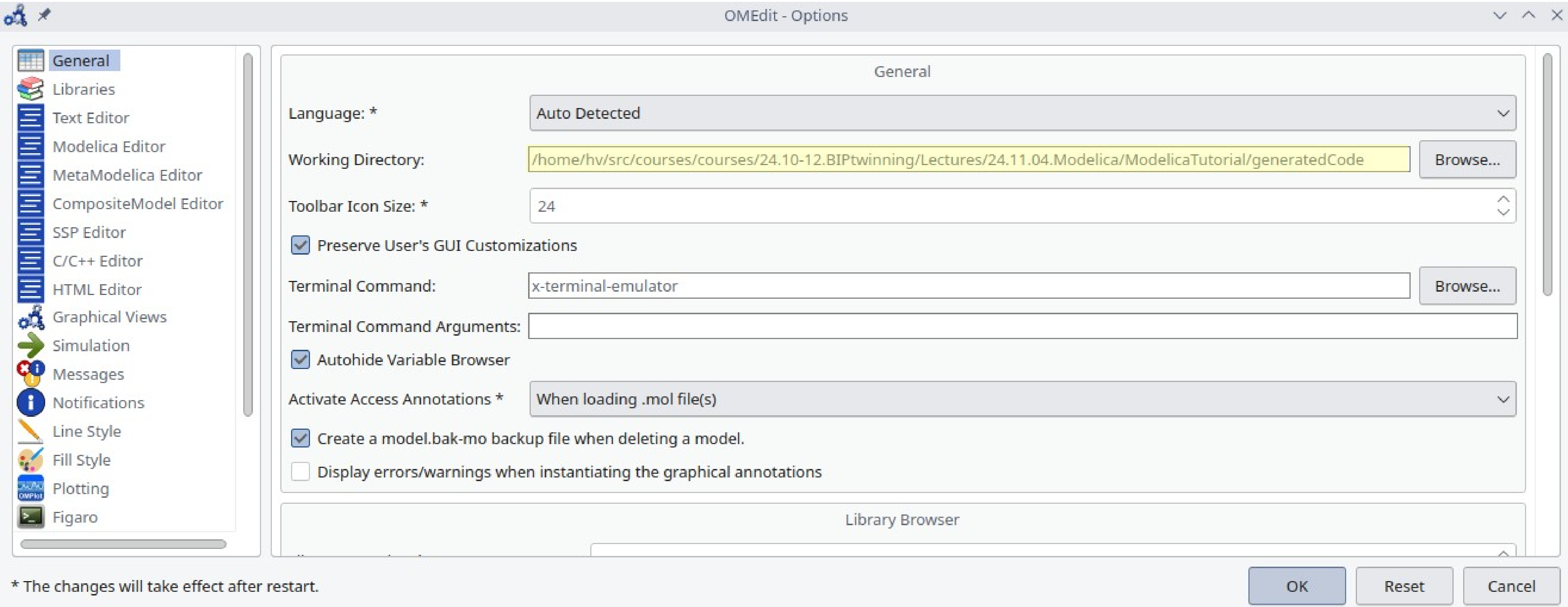
+ weights for "bad inverses"

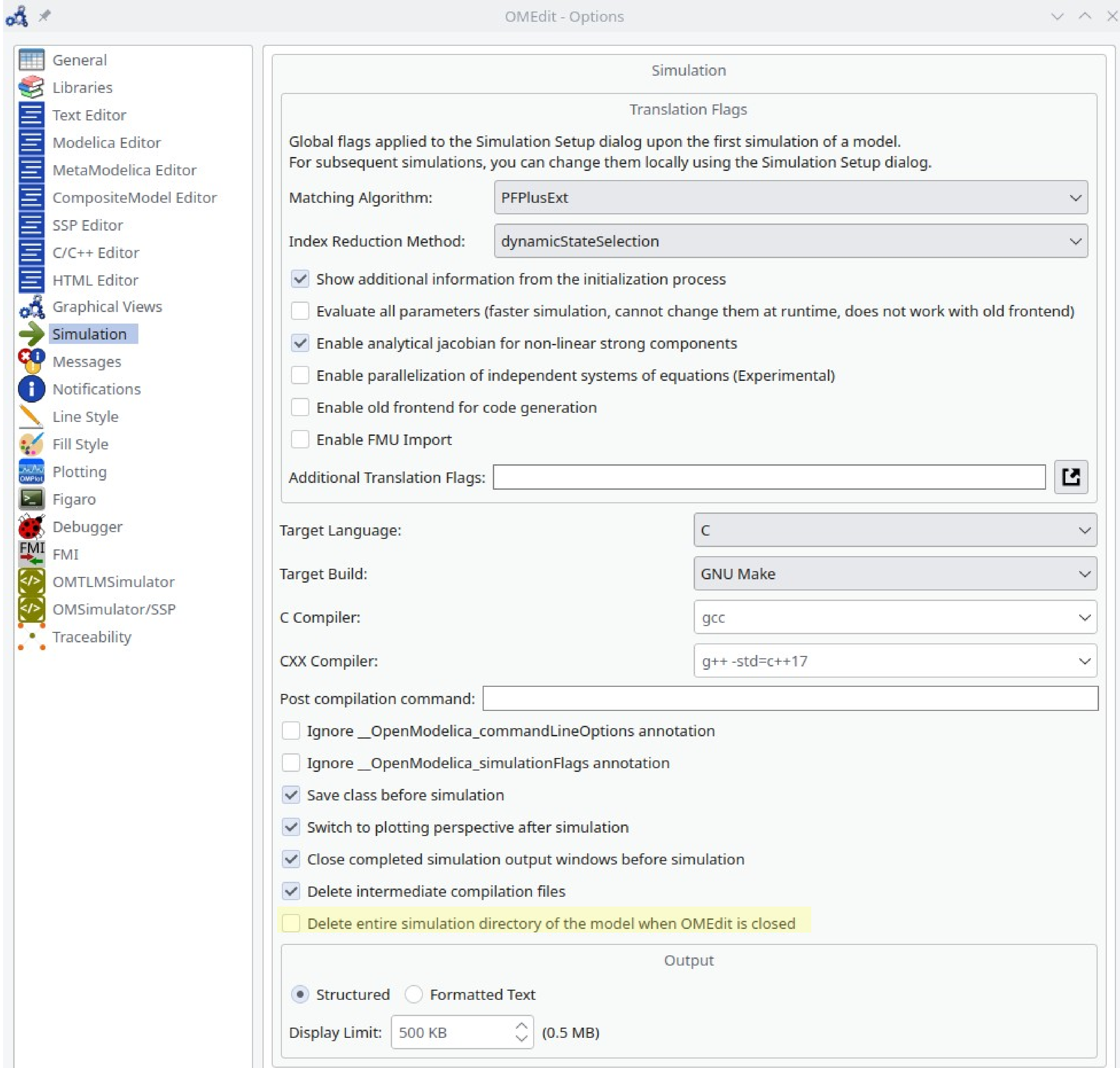
Causality assigned

$$\begin{cases} x + \underline{y} + z & = & 0 & \text{Equation 1} \\ \underline{x} + 3z + u^2 & = & 0 & \text{Equation 2} \\ \underline{z} - u - 16 & = & 0 & \text{Equation 3} \\ \underline{u} - 5 & = & 0 & \text{Equation 4} \end{cases}$$

re-write in causal form (symbolically, using Computer Algebra)

$$\begin{cases} \underline{y} & = & -x - z \\ \underline{x} & = & -3z - u^2 \\ \underline{z} & = & u + 16 \\ \underline{u} & = & 5 \end{cases}$$


OMEdit - Options
General
Libraries
Text Editor
Modelica Editor
MetaModelica Editor
CompositeModel Editor
SSP Editor
C/C++ Editor
HTML Editor
Graphical Views
Simulation
Messages
Notifications
Line Style
Fill Style
Plotting
Figaro
General
Language: *
Auto Detected
Working Directory:
/home/hv/src/courses/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/generatedCode
Browse...
Toolbar Icon Size: *
24
Preserve User's GUI Customizations
Terminal Command:
x-terminal-emulator
Browse...
Terminal Command Arguments:
Autohide Variable Browser
Activate Access Annotations *
When loading .mol file(s)
Create a model.bak-mo backup file when deleting a model.
Display errors/warnings when instantiating the graphical annotations
Library Browser
* The changes will take effect after restart.
OK
Reset
Cancel


OMEdit - Options
General
Libraries
Text Editor
Modelica Editor
MetaModelica Editor
CompositeModel Editor
SSP Editor
C/C++ Editor
HTML Editor
Graphical Views
Simulation
Messages
Notifications
Line Style
Fill Style
Plotting
Figaro
Debugger
FMI
OMTLMSimulator
OMSimulator/SSP
Traceability
Simulation
Translation Flags
Global flags applied to the Simulation Setup dialog upon the first simulation of a model.
For subsequent simulations, you can change them locally using the Simulation Setup dialog.
Matching Algorithm:
PFPlusExt
Index Reduction Method:
dynamicStateSelection
Show additional information from the initialization process
Evaluate all parameters (faster simulation, cannot change them at runtime, does not work with old frontend)
Enable analytical jacobian for non-linear strong components
Enable parallelization of independent systems of equations (Experimental)
Enable old frontend for code generation
Enable FMU Import
Additional Translation Flags:
Target Language:
C
Target Build:
GNU Make
C Compiler:
gcc
CXX Compiler:
g++ -std=c++17
Post compilation command:
Ignore __OpenModelica_commandLineOptions annotation
Ignore __OpenModelica_simulationFlags annotation
Save class before simulation
Switch to plotting perspective after simulation
Close completed simulation output windows before simulation
Delete intermediate compilation files
Delete entire simulation directory of the model when OMEdit is closed
Output
Structured
Formatted Text
Display Limit:
500 KB
(0.5 MB)

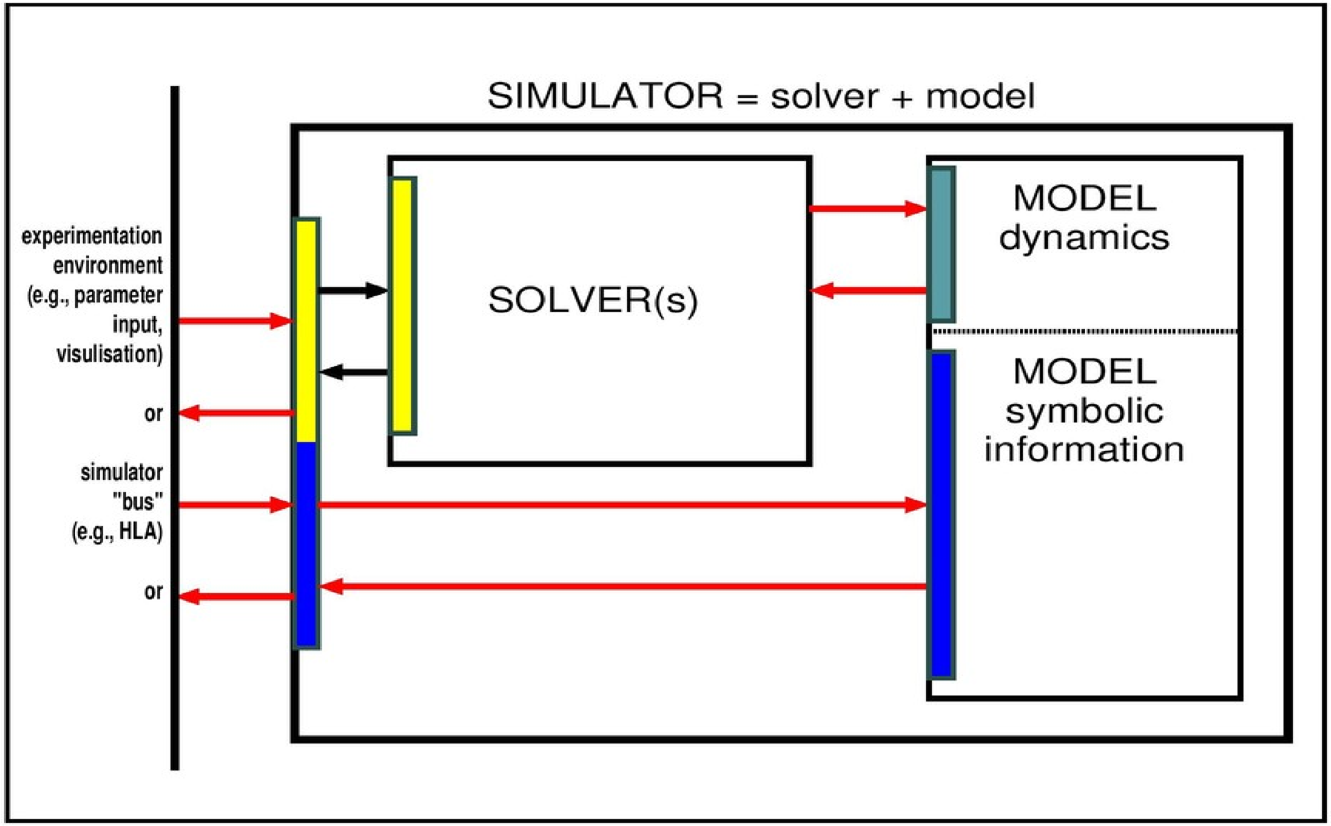
Compilation | Output

```
make -j4 -f ElectricalCircuit.makefile
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit.o ElectricalCircuit.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_functions.o ElectricalCircuit_functions.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_records.o ElectricalCircuit_records.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_01exo.o ElectricalCircuit_01exo.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_02nls.o ElectricalCircuit_02nls.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_03lsy.o ElectricalCircuit_03lsy.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_04set.o ElectricalCircuit_04set.c
```

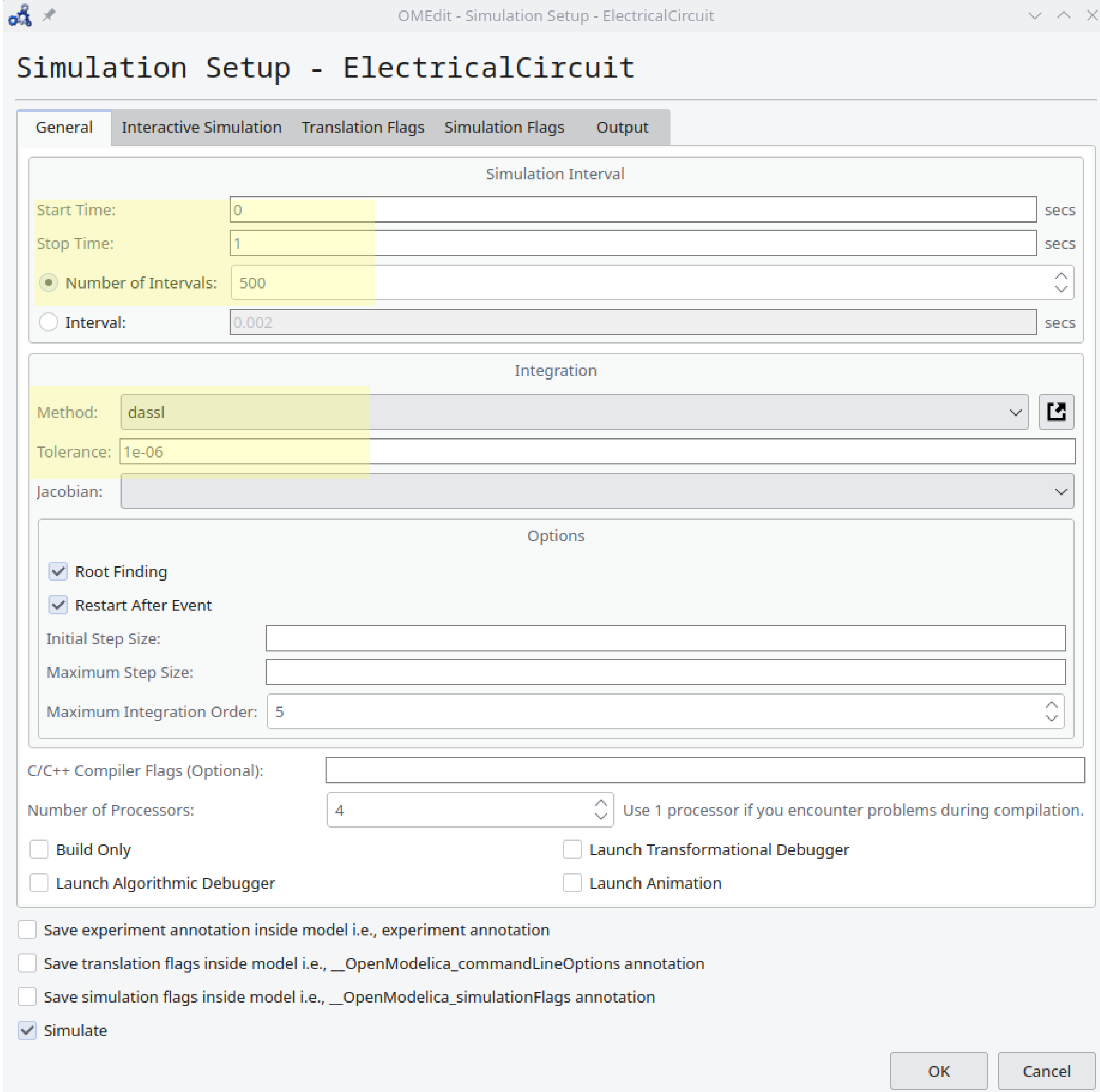
```
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_13opt.o ElectricalCircuit_13opt.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_14lnz.o ElectricalCircuit_14lnz.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_15syn.o ElectricalCircuit_15syn.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_16dae.o ElectricalCircuit_16dae.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_17inl.o ElectricalCircuit_17inl.c
gcc  -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/
openmodelica-nightly/bin/../include/omc" -I. -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -
DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -DOMC_NVAR_STRING=0  -c -o ElectricalCircuit_18spd.o ElectricalCircuit_18spd.c
gcc -I. -o ElectricalCircuit ElectricalCircuit.o ElectricalCircuit_functions.o ElectricalCircuit_records.o ElectricalCircuit_01exo.o ElectricalCircuit_02nls.o ElectricalCircuit_03lsy.o
ElectricalCircuit_04set.o ElectricalCircuit_05evt.o ElectricalCircuit_06inz.o ElectricalCircuit_07dly.o ElectricalCircuit_08bnd.o ElectricalCircuit_09alg.o ElectricalCircuit_10asr.o
ElectricalCircuit_11mix.o ElectricalCircuit_12jac.o ElectricalCircuit_13opt.o ElectricalCircuit_14lnz.o ElectricalCircuit_15syn.o ElectricalCircuit_16dae.o ElectricalCircuit_17inl.o
ElectricalCircuit_18spd.o -L"/home/hv/src/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/OOEquationBased"    -Os -DOM_HAVE_PTHREADS -fPIC -falign-functions -
mfpmath=sse -fno-dollars-in-identifiers -Wno-parentheses-equality     -I"/opt/openmodelica-nightly/bin/../include/omc/c" -I"/opt/openmodelica-nightly/bin/../include/omc" -I. -
DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME -DOMC_MODEL_PREFIX=ElectricalCircuit -DOMC_NUM_MIXED_SYSTEMS=0 -DOMC_NUM_LINEAR_SYSTEMS=0 -DOMC_NUM_NONLINEAR_SYSTEMS=0 -DOMC_NDELAY_EXPRESSIONS=0 -
DOMC_NVAR_STRING=0 -L"/opt/openmodelica-nightly/bin/../lib/x86_64-linux-gnu/omc" -L"/opt/openmodelica-nightly/bin/../lib" -Wl,-rpath,"/opt/openmodelica-nightly/bin/../lib/x86_64-linux-gnu/omc"
-Wl,-rpath,"/opt/openmodelica-nightly/bin/../lib"    -Wl,--no-as-needed -Wl,--disable-new-dtags -lSimulationRuntimeC -llapack -lblas -lm -lomcgc -lryu -lpthread -rdynamic -Wl,--no-undefined
Compilation process finished successfully.
```

```
hv@sanderling 59% pwd
/home/hv/src/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/generatedCode/ElectricalCircuit
hv@sanderling 60% ls
ElectricalCircuit               ElectricalCircuit_08bnd.c  ElectricalCircuit_14lnz.c        ElectricalCircuit_includes.h
ElectricalCircuit_01exo.c       ElectricalCircuit_08bnd.o  ElectricalCircuit_14lnz.o        ElectricalCircuit_info.json
ElectricalCircuit_01exo.o       ElectricalCircuit_09alg.c  ElectricalCircuit_15syn.c        ElectricalCircuit_init.xml
ElectricalCircuit_02nls.c       ElectricalCircuit_09alg.o  ElectricalCircuit_15syn.o        ElectricalCircuit_JacA.bin
ElectricalCircuit_02nls.o       ElectricalCircuit_10asr.c  ElectricalCircuit_16dae.c        ElectricalCircuit_literals.h
ElectricalCircuit_03lsy.c       ElectricalCircuit_10asr.o  ElectricalCircuit_16dae.h        ElectricalCircuit.log
ElectricalCircuit_03lsy.o       ElectricalCircuit_11mix.c  ElectricalCircuit_16dae.o        ElectricalCircuit.makefile
ElectricalCircuit_04set.c       ElectricalCircuit_11mix.h  ElectricalCircuit_17inl.c        ElectricalCircuit_model.h
ElectricalCircuit_04set.o       ElectricalCircuit_11mix.o  ElectricalCircuit_17inl.o        ElectricalCircuit.o
ElectricalCircuit_05evt.c       ElectricalCircuit_12jac.c  ElectricalCircuit_18spd.c        ElectricalCircuit_prof.intdata
ElectricalCircuit_05evt.o       ElectricalCircuit_12jac.h  ElectricalCircuit_18spd.o        ElectricalCircuit_prof.realdata
ElectricalCircuit_06inz.c       ElectricalCircuit_12jac.o  ElectricalCircuit.c              ElectricalCircuit_records.c
ElectricalCircuit_06inz.o       ElectricalCircuit_13opt.c  ElectricalCircuit_functions.c    ElectricalCircuit_records.o
ElectricalCircuit_07dly.c       ElectricalCircuit_13opt.h  ElectricalCircuit_functions.h    ElectricalCircuit_res.mat
ElectricalCircuit_07dly.o       ElectricalCircuit_13opt.o  ElectricalCircuit_functions.o
```

Model-Solver Interface
Simulator-Environment Interface



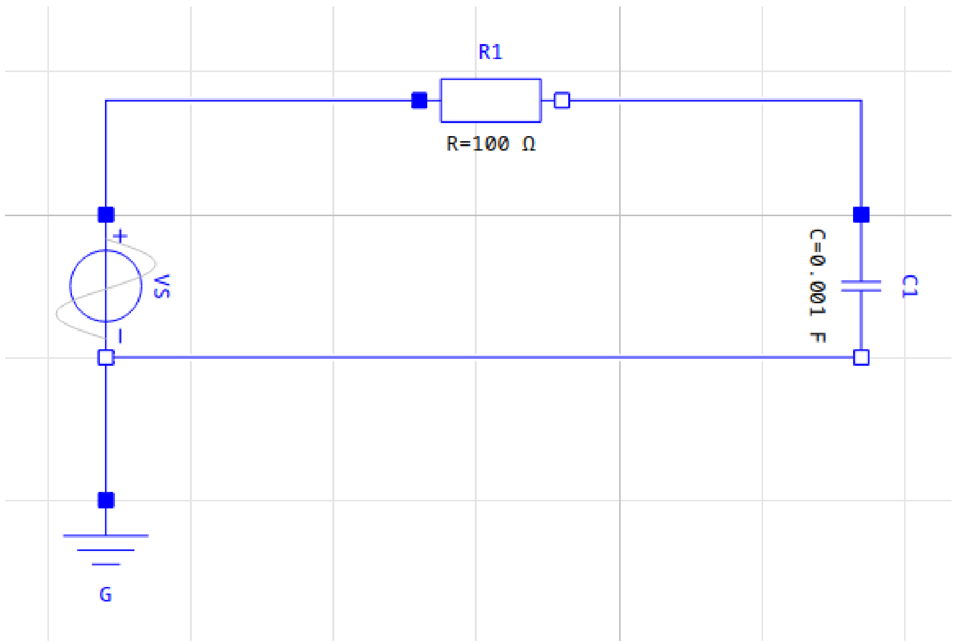
~ co-simulation (and the FMI standard)

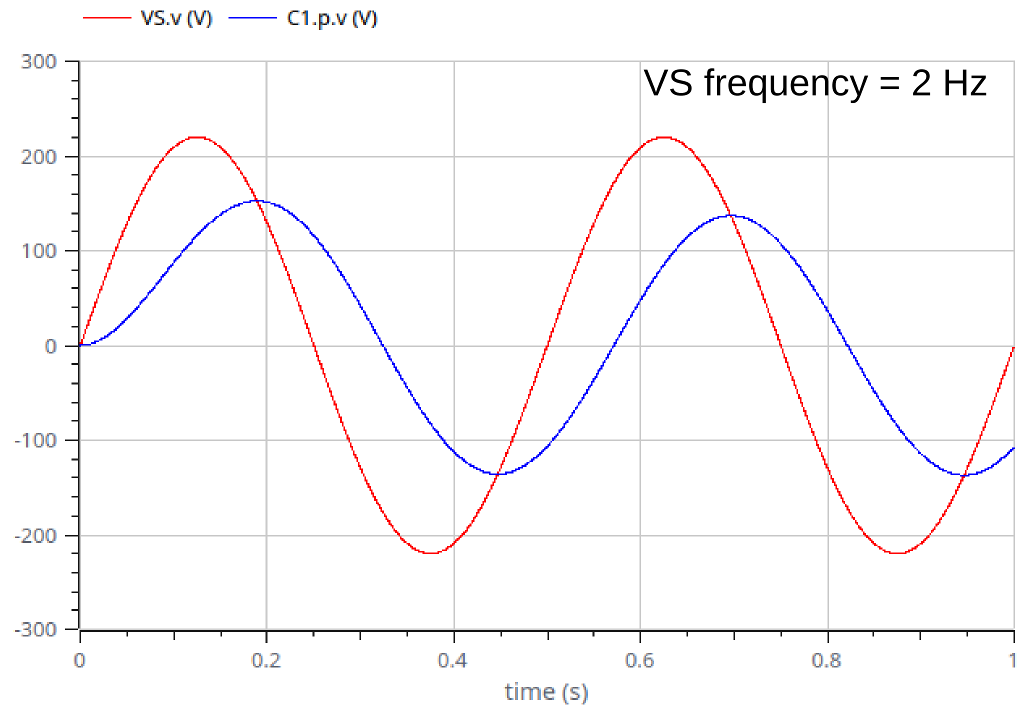

OMEdit - Simulation Setup - ElectricalCircuit
Simulation Setup - ElectricalCircuit
General
Interactive Simulation
Translation Flags
Simulation Flags
Output
Simulation Interval
Start Time:
0
secs
Stop Time:
1
secs
Number of Intervals:
500
Interval:
0.002
secs
Integration
Method:
dassl
Tolerance:
1e-06
Jacobian:
Options
Root Finding
Restart After Event
Initial Step Size:
Maximum Step Size:
Maximum Integration Order:
5
C/C++ Compiler Flags (Optional):
Number of Processors:
4
Use 1 processor if you encounter problems during compilation.
Build Only
Launch Transformational Debugger
Launch Algorithmic Debugger
Launch Animation
Save experiment annotation inside model i.e., experiment annotation
Save translation flags inside model i.e., __OpenModelica_commandLineOptions annotation
Save simulation flags inside model i.e., __OpenModelica_simulationFlags annotation
Simulate
OK
Cancel

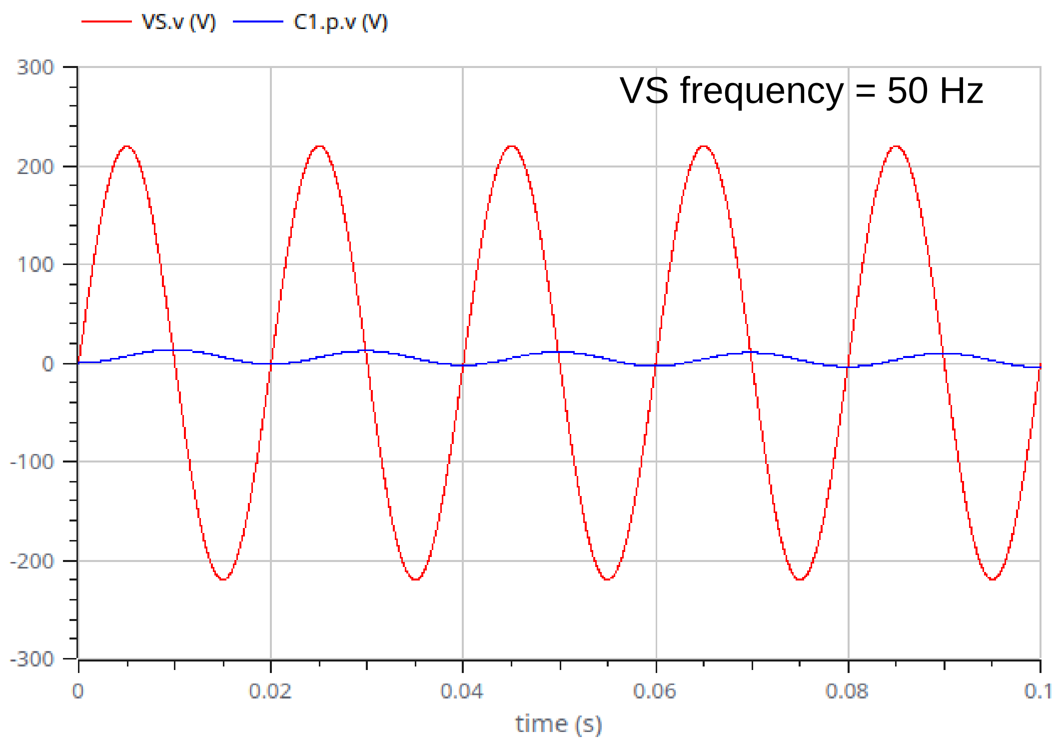
simulation run statistics

```
/home/hv/src/courses/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/generatedCode/
ElectricalCircuit/ElectricalCircuit -port=46007 -logFormat=xmltcp -
override=startTime=0,stopTime=1,stepSize=0.002,tolerance=1e-06,solver=dassl,outputFormat=mat,variableFilter=.* -
r=/home/hv/src/courses/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/generatedCode/
ElectricalCircuit/ElectricalCircuit_res.mat -w -lv=LOG_STDOUT,LOG_ASSERT,LOG_STATS -inputPath=/home/hv/src/
courses/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/generatedCode/ElectricalCircuit -
outputPath=/home/hv/src/courses/courses/24.10-12.BIPtwinning/Lectures/24.11.04.Modelica/ModelicaTutorial/
generatedCode/ElectricalCircuit
The initialization finished successfully without homotopy method.
### STATISTICS ###
timer
 0.000811424s          reading init.xml
  9.6951e-05s          reading info.xml
 0.000222972s [  3.7%] pre-initialization
  6.1091e-05s [  1.0%] initialization
    7.91e-06s [  0.1%] steps
    0.001742s [ 28.8%] solver (excl. callbacks)
 0.000352641s [  5.8%] creating output-file
 0.000344505s [  5.7%] event-handling
  8.0611e-05s [  1.3%] overhead
  0.00324135s [ 53.5%] simulation
  0.00605308s [100.0%] total
events
    0 state events
    0 time events
solver: dassl
 2938 steps taken
 3255 calls of functionODE
   44 evaluations of jacobian
   28 error test failures
    0 convergence test failures
1.8071e-05s time of jacobian evaluation
The simulation finished successfully.
```

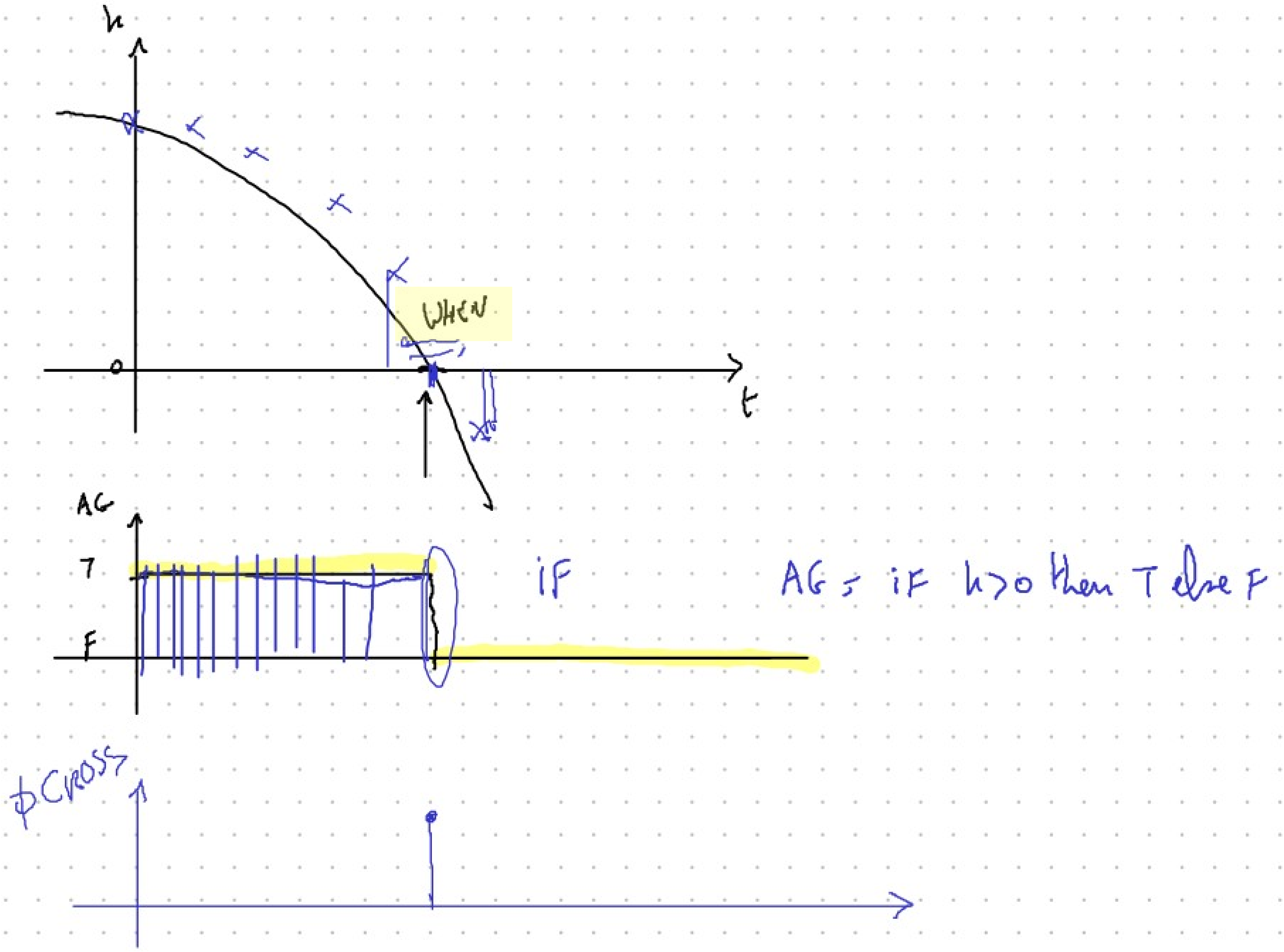
“low (frequency) pass filter”





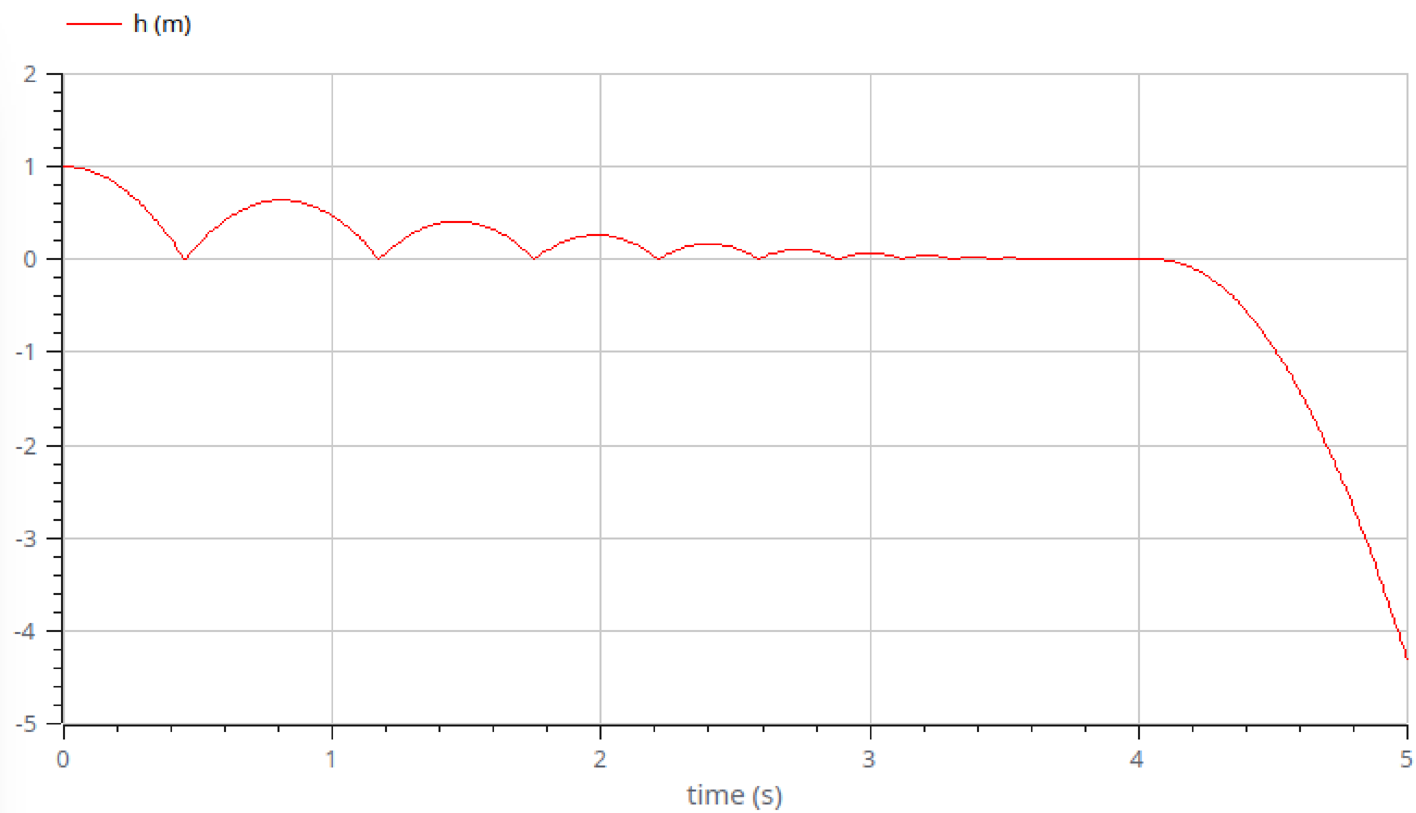


Hybrid (includes discontinuities) → “when” temporal operator



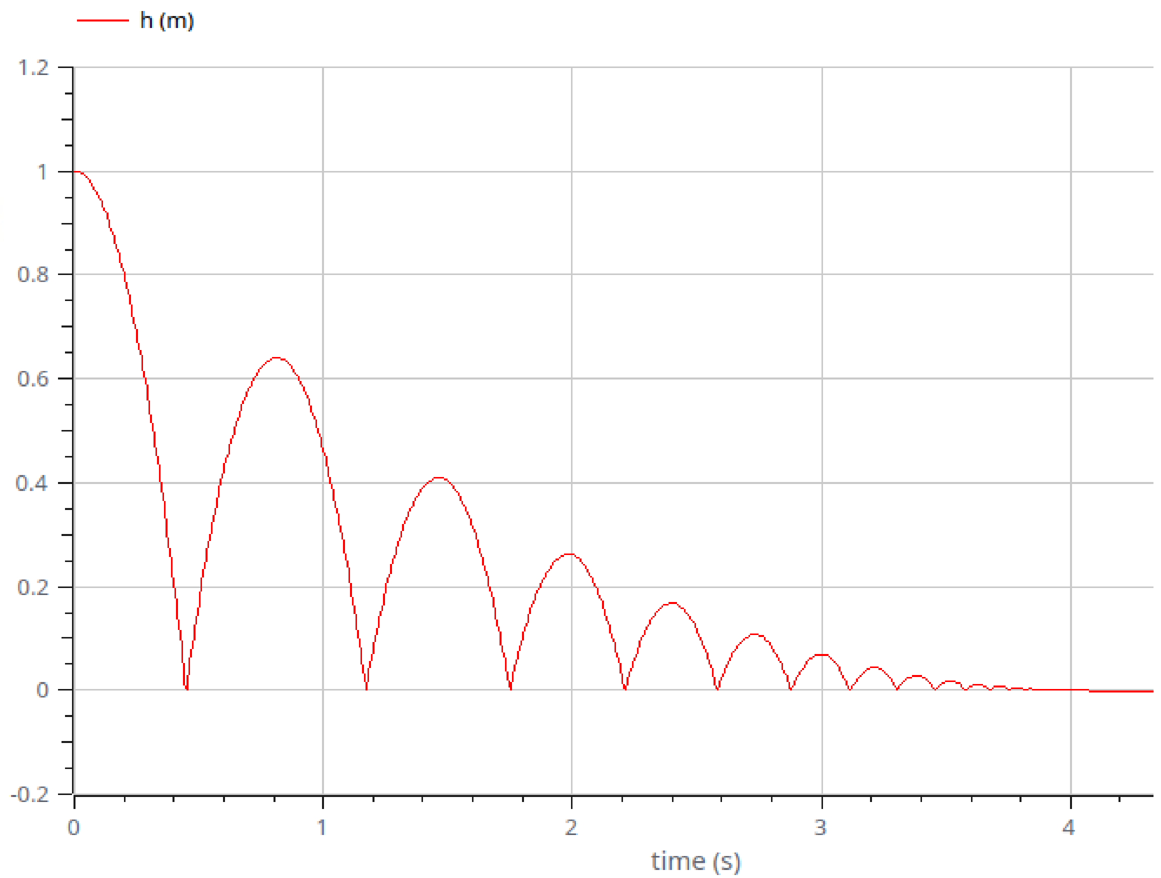
Hybrid (includes discontinuities) -- unstable

```
model unstable_bouncing_ball "The 'classic' bouncing ball model"
  type Height=Real(unit="m");
  type Velocity=Real(unit="m/s");
  parameter Real e=0.8 "Coefficient of restitution";
  parameter Height h0=1.0 "Initial height";
  Height h "Height";
  Velocity v(start=0.0, fixed=true) "Velocity";
initial equation
  h = h0;
equation
  v = der(h);
  der(v) = -9.81;
  when h<0 then
    reinit(v, -e*pre(v));
  end when;
end unstable_bouncing_ball;
```



Hybrid (includes discontinuities) -- stable

```
model StableBouncingBall
  "The 'classic' bouncing ball model with numerical tolerances"
  type Height=Real(unit="m");
  type Velocity=Real(unit="m/s");
  parameter Real e=0.8 "Coefficient of restitution";
  parameter Height h0=1.0 "Initial height";
  constant Height eps=1e-3 "Small height";
  Boolean done "Flag when to turn off gravity";
  Height h "Height";
  Velocity v(start=0.0, fixed=true) "Velocity";
initial equation
  h = h0;
  done = false;
equation
  v = der(h);
  der(v) = if done then 0 else -9.81;
  when {h<0, h<-eps} then
    done = h<-eps;
    reinit(v, if h<-eps then 0 else -e*pre(v));
  end when;
end StableBouncingBall;
```

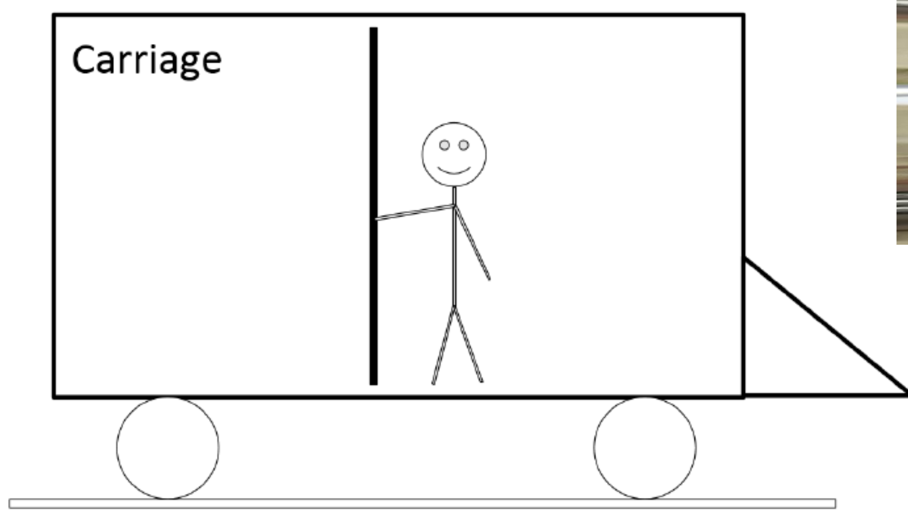


Controller Design and Tuning

Control System

- A *control system* (or "controller") is a system whose purpose is to command, direct, or regulate itself, or another system.
- The System under Control is often called a "plant" (as in "chemical production plant").
- There are open-loop and closed-loop control systems.
  - Closed-loop control system: e.g., human picking an object
    - Eyes are *sensors*.
    - Hands are *actuators*.
    - Brain is the *controller* that estimates the distance between hand and object based on sensor input.
      It determines/computes an appropriate *control action* that satisfies requirements and implements it through the *actuators*.
  - Open-loop control system: e.g., blindfolded picking
    - Only the current state and a model of the plant are used. The output of the system under control is not observed.
- Our example (closed loop): velocity control in rail car

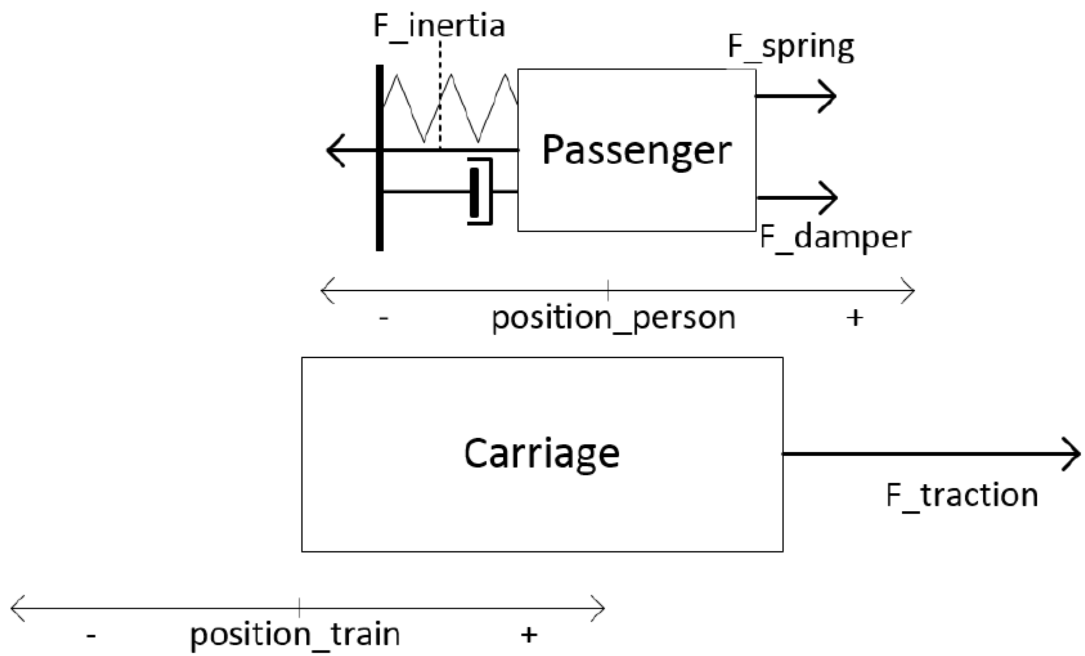
Rail Car Case



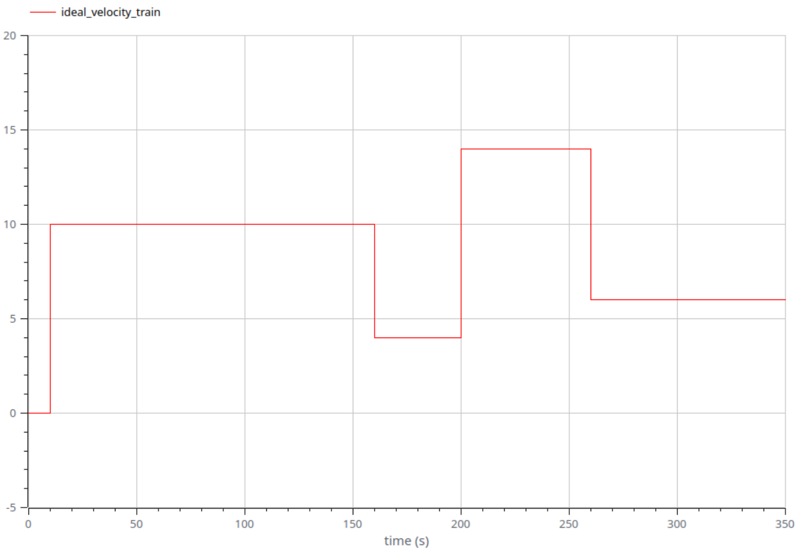
- Build the controller for a driverless rail car.
- The controller determines the acceleration of the train, in an attempt to match (i.e., deviate as little as possible from) a predefined profile of desired velocities.

  The desired (piecewise constant) velocity profile is known beforehand by a central coordinator (and is encoded in a file).
- Passengers should not fall (*i.e.*, accelerate too much).
- Other requirements such as minimizing total energy consumption could be added.

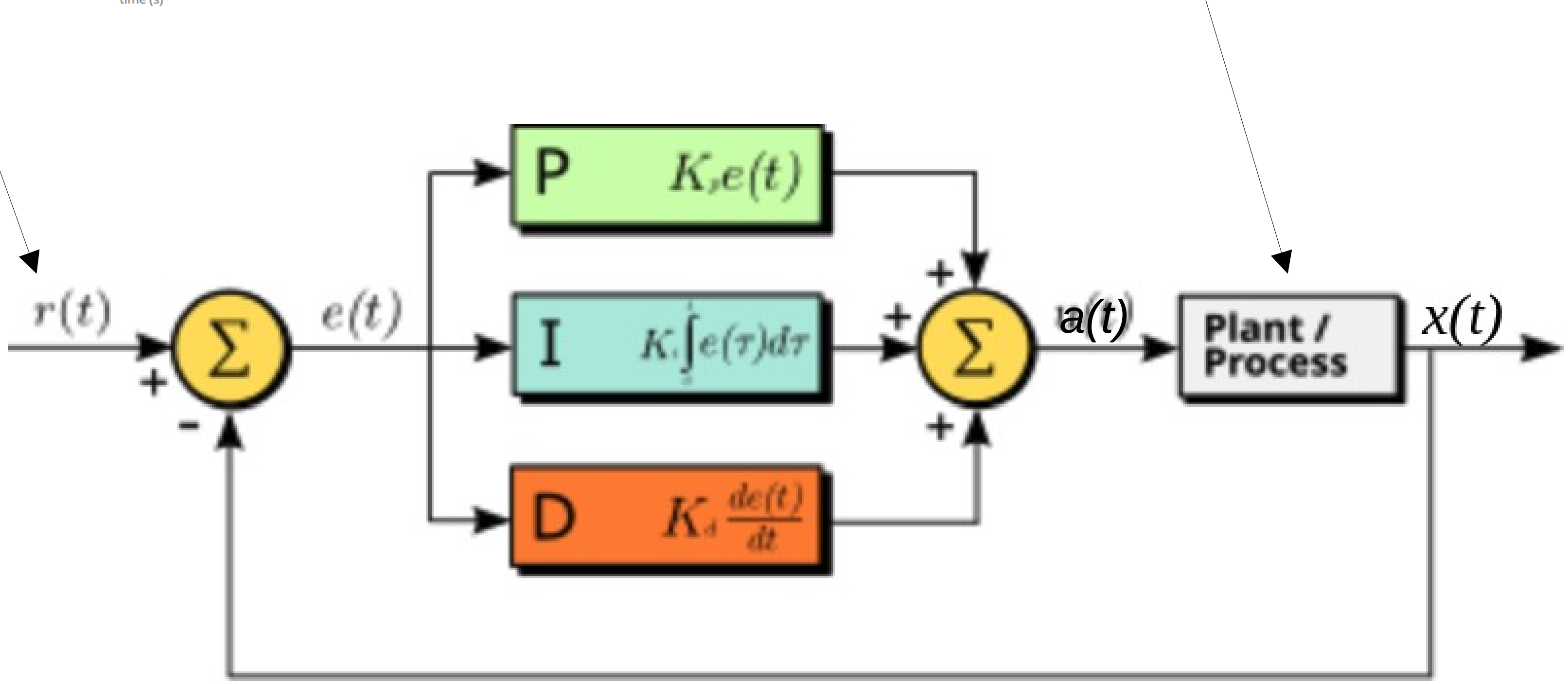
Abstracting Train-and-Passenger ("Plant" model)



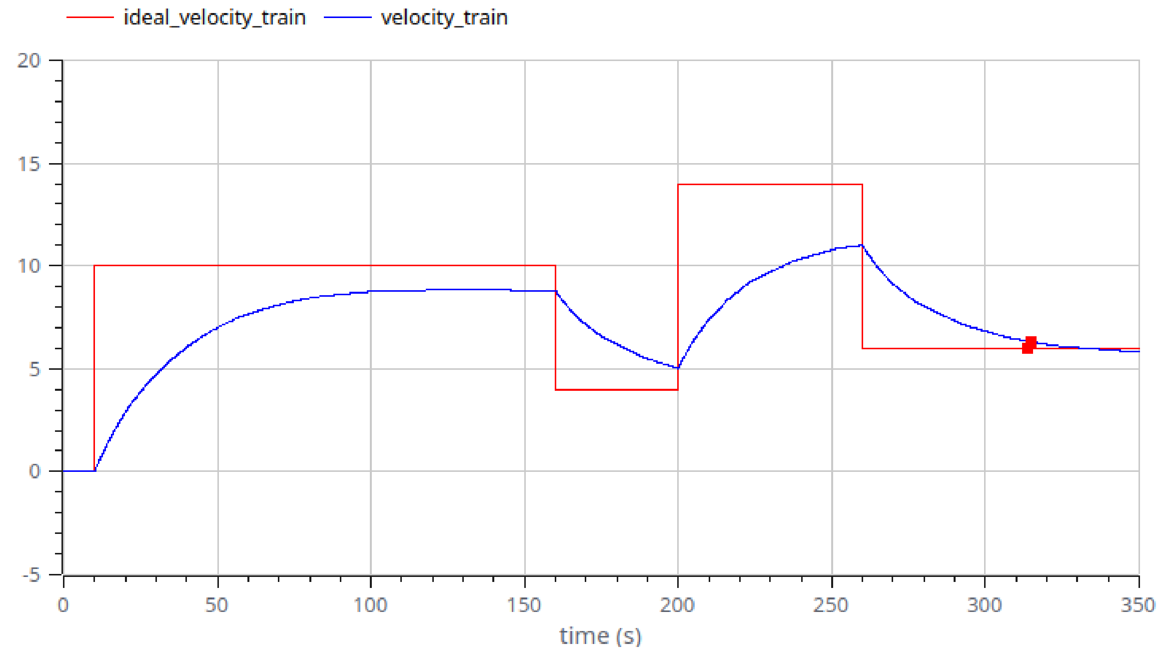
$$
\begin{cases}
m_{passger} * a_{passger} & = k(-x_{passger}) + c(-v_{passger}) - m_{passger} * a_{train} \\
F_{traction} & = (m_{train} + m_{passger}) * a_{train} \\
a_{passger} & = \frac{dv_{passger}}{dt} \\
v_{passger} & = \frac{dx_{passger}}{dt} \\
a_{train} & = \frac{dv_{train}}{dt} \\
v_{train} & = \frac{dx_{train}}{dt}
\end{cases}
$$



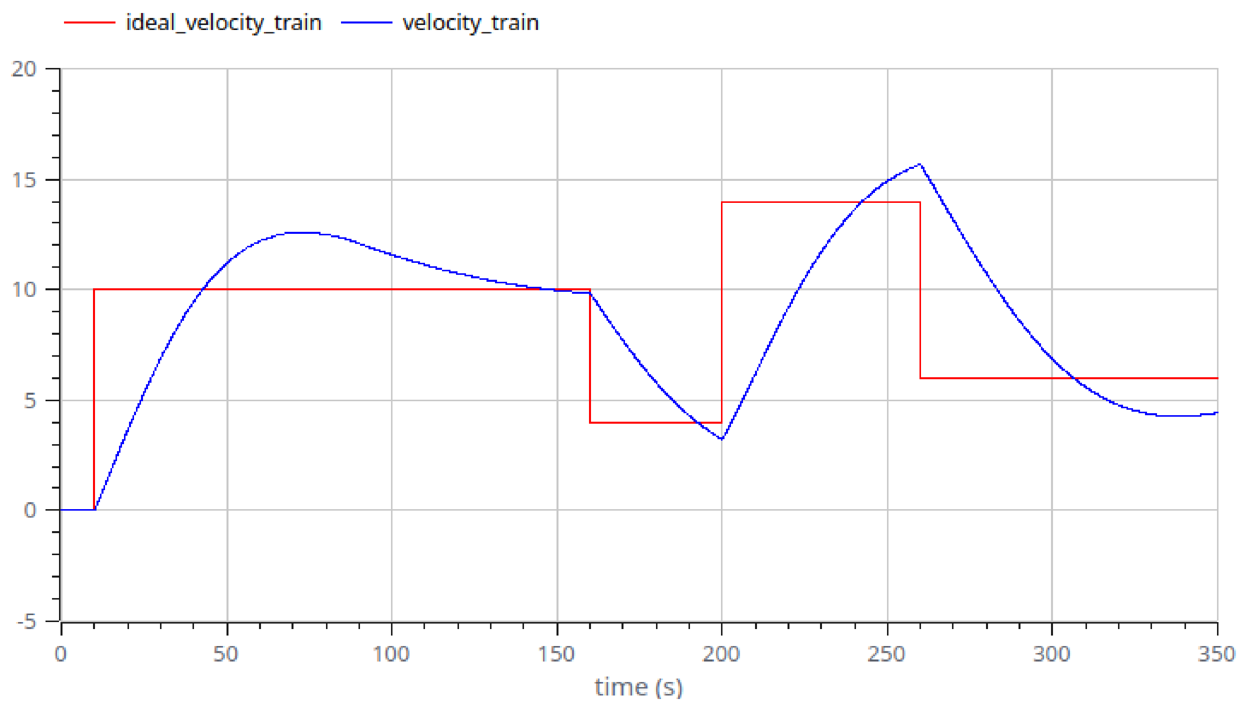
$$
\begin{cases}
m_{passger} * a_{passger} & = k(-x_{passger}) + c(-v_{passger}) - m_{passger} * a_{train} \\
F_{traction} & = (m_{train} + m_{passger}) * a_{train} \\
a_{passger} & = \frac{dv_{passger}}{dt} \\
v_{passger} & = \frac{dx_{passger}}{dt} \\
a_{train} & = \frac{dv_{train}}{dt} \\
v_{train} & = \frac{dx_{train}}{dt}
\end{cases}
$$



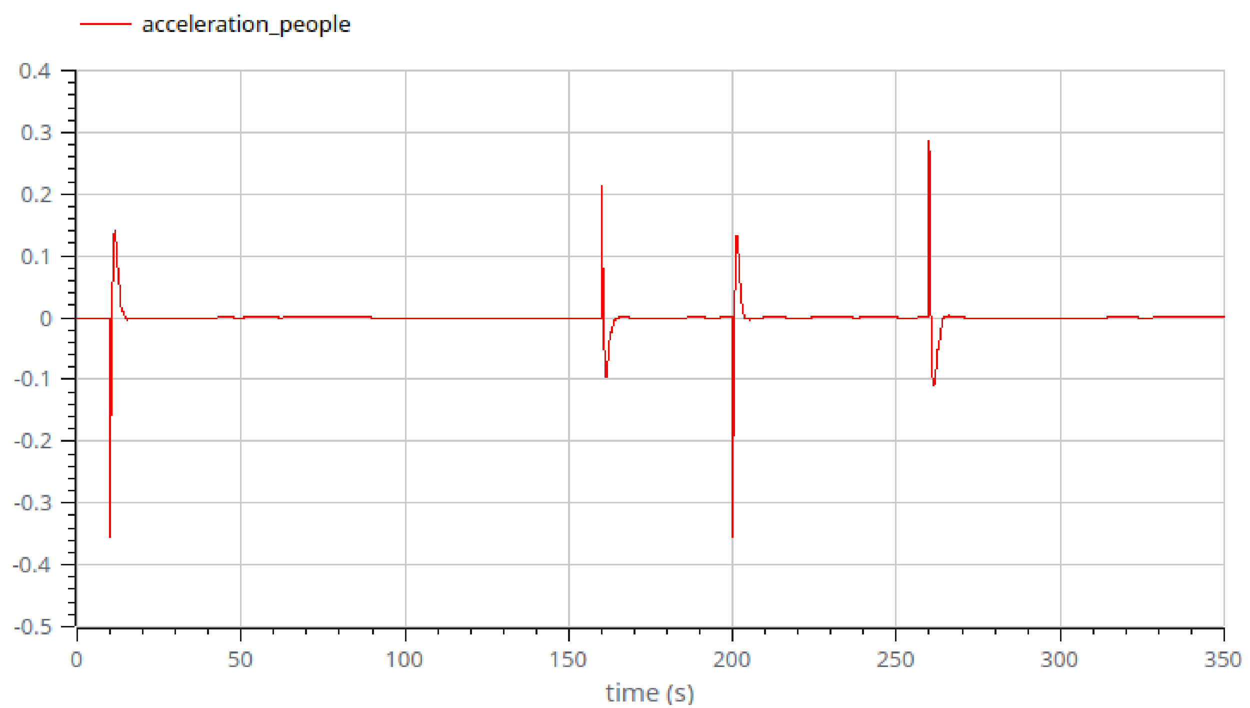
```
equation
  // train and passenger motion part
  der(velocity_people) = ((-mass_people * acceleration_train) + k * (-displacement_people) + c * (-velocity_people)) / mass_people;
  der(displacement_people) = velocity_people;
  der(velocity_people) = acceleration_people;
  drag_force_train = -0.5 * p * velocity_train * velocity_train * Cd * A;
  acceleration_train = (traction_force_train + drag_force_train) / (mass_train + mass_people);
  der(velocity_train) = acceleration_train;

  // control part
  velocity_error_train = ideal_velocity_train - velocity_train;
  symbolic_der_velocity_error_train = -acceleration_train;
  der(accumulated_error_train) = velocity_error_train;
  traction_force_train =
    control_error_proportion * velocity_error_train +
    control_int_error_proportion * accumulated_error_train +
    control_der_error_proportion * symbolic_der_velocity_error_train;

  // external inputs
  // 14 m/s is about 50 km/h
  // the desired velocity profile
  ideal_velocity_train = if time < 10 then 0 else if time < 160 then 10 else if time < 200 then 4 else if time < 260 then 14 else 6;

  // experiment settings
  annotation(experiment(StartTime = 0, StopTime = 350, Tolerance = 0.1, Interval = 0.070014));
```

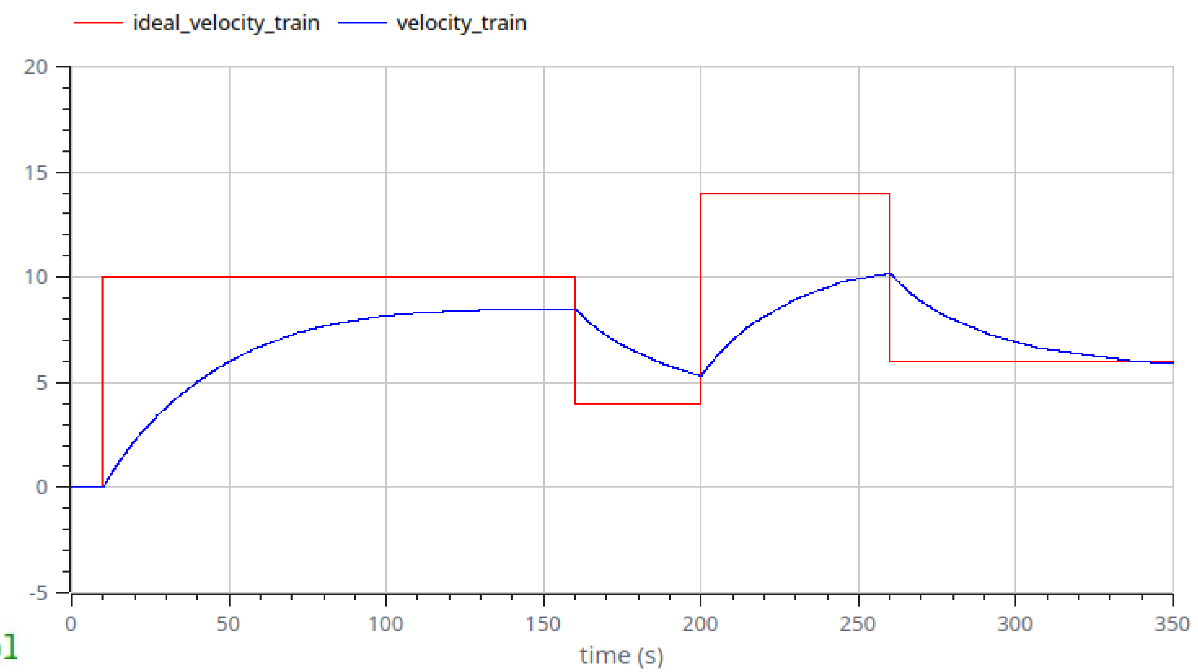



```
  // a Proportional (P) controller
parameter Real control_error_proportion = 200;
parameter Real control_int_error_proportion = 0;
parameter Real control_der_error_proportion = 0;
```

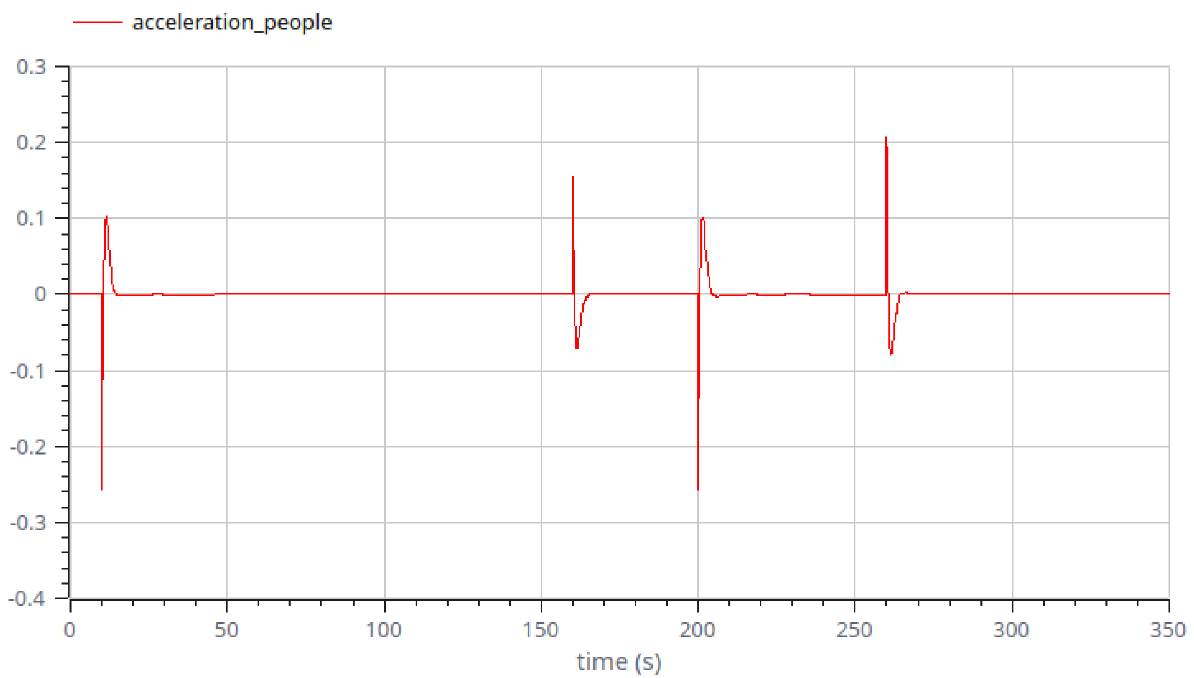



```
// a Proportional and Integral (PI) controller
parameter Real control_error_proportion = 200;
parameter Real control_int_error_proportion = 10;
parameter Real control_der_error_proportion = 0;
```

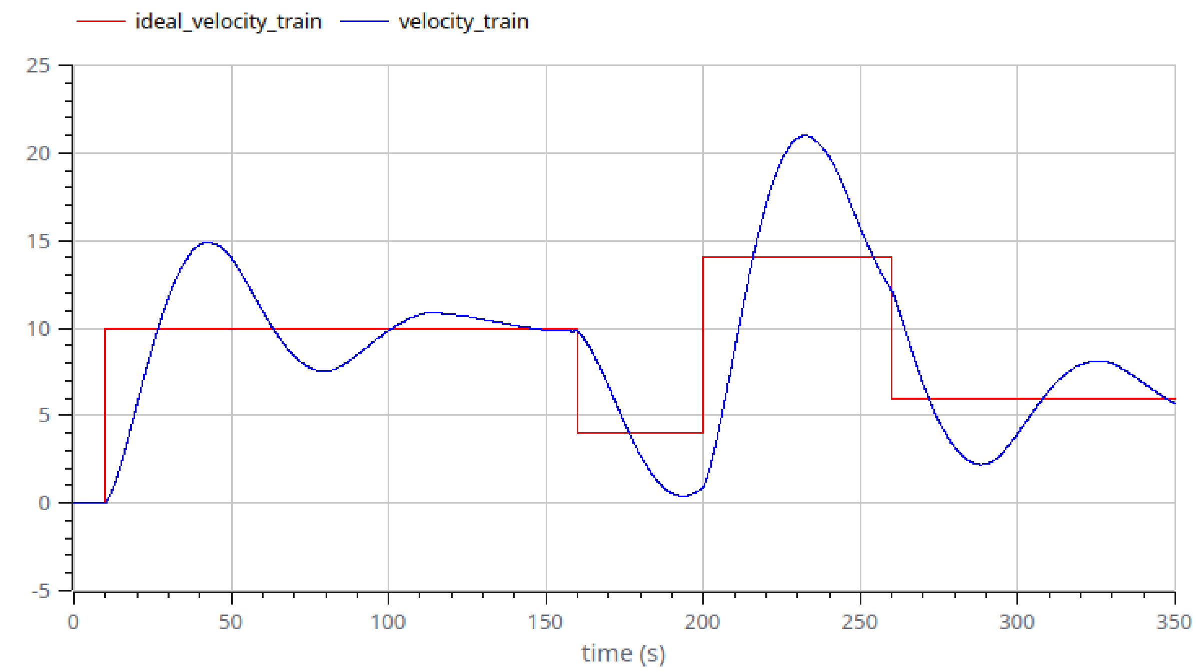





```
  // a Proportional and Derivative (PD) control
parameter Real control_error_proportion = 150;
parameter Real control_int_error_proportion = 0;
parameter Real control_der_error_proportion = 200;
```



PID controller
constrained multi-criteria **optimization**




```
  // a PID controller
parameter Real control_error_proportion = 150;
parameter Real control_int_error_proportion = 50;
parameter Real control_der_error_proportion = 200;
```