Tutorial Proposal: Physical Systems for Software Modellers

Hans Vangheluwe
University of Antwerp - Flanders Make
Belgium
Hans.Vangheluwe@uantwerpen.be

Cláudio Gomes
University of Antwerp - Flanders Make
Belgium
Claudio.Gomes@uantwerpen.be

Basic Information

Title: Physical Systems for Software Modellers

Presenters

HANS VANGHELUWE is a Professor at the University of Antwerp (Belgium). He heads the Modeling, Simulation and Design Lab (MSDL). In a variety of projects, often with industrial partners, he develops and applies the model-based theory and techniques of Multi-Paradigm Modeling (MPM). He is the chair of COST Action IC1404 "Multi-Paradigm Modelling for Cyber-Physical Systems" (MPM4CPS). He was a founding member of the Modelica® design team and in the 1990s helped develop this standard language for equation-based object-oriented modelling. His e-mail address is Hans.Vangheluwe@uantwerpen.be.

CLÁUDIO GOMES is a PhD student in the Modelling, Simulation and Design Lab (MSDL) at the University of Antwerp (Belgium). He was awarded a scholarship from the Research Foundation - Flanders, to work on the foundations of co-simulation. Since 2016, he has connected the fields of numerical analysis, optimization, and computer science to investigate the effect of different co-simulation algorithms on the quality of full-system overall simulation results, even in the presence of "black-box" Functional Mockup Units. His e-mail address is Claudio.Gomes@uantwerpen.be.

Abstract

The complex engineered systems we build today get their value from the networking of multi-physical (mechanical, electrical, hydraulic, biochemical, ...) and computational (control, signal processing, planning, ...) processes, often interacting with a highly uncertain environment. Software plays a pivotal role, both as a component of such systems, often realizing control laws, deployed on a resource-constrained physical platform, and in the construction of enabling modelling and simulation tools.

This two-part tutorial will introduce software modellers to the two main facets of dealing with physical systems through modelling, simulation and (controller) code synthesis.

In the first part, the different levels at which physical systems may be modelled are introduced. This starts with the technological level. At this level, components are considered that can be physically realized with current materials and production methods. Such components are often available off the shelf. They are characterized by the very specific context (also known as Experimental Frame) in which their models are valid. The next level uses the full knowledge of physics and engineering to describe the behaviour of physical components to study a wide variety of properties. To study the possibly turbulent flow of a viscous liquid through a pipe for example, a Navier-Stokes Partial Differential Equations model will be used. Such models are hard to calibrate and simulate accurately and efficiently. The next level considers the often occurring situation where, for the properties of interest, the spatial distribution of the problem can be abstracted and a lumped-parameter (as opposed to distributed-parameter) model can be used. In a translational mechanical context for example, an object with a complex geometry may still be considered as a point mass characterized by a single parameter "mass". Such models still obey physical conservation laws such as energy conservation. At this level, formalisms such as Bond Graphs that focus on power flow through a system are used. At the next level, the link with physics is weakened and computational components (functions) are added. This leads to the popular Equation-based Object-Oriented modelling languages such as Modelica® and Simscape®. The semantics of such computationally a-causal languages will be explained with particular focus on the process of causality assignment. This leads to the next level at which input-output computational blocks are used. The main disadvantage of this level is that it focuses on "how" to compute the evolution of state variables over time as opposed the focus on "what" the governing equations are in equation-based languages, leaving the "how" to a model compiler.

Even the discretized level is still an idealization as the numerical values as not Real numbers, but are implemented as floating point approximations.

The second part of the tutorial starts from the computationally causal level. The formalisms used are known as Causal Block Diagrams (CBDs) or Synchronous Data Flow (SDF), with Simulink® as the most notable example. Three different semantics of CBDs will be explained, bridging the gap between the equations resulting from causality assignment described in the first part of the tutorial and their realization

in software. This software can either be a simulator (or a Functional Mockup Unit in case of co-simulation) on a digital computer or a controller deployed on a micro-controller or ECU.

A first semantics of such input-output Causal Block Diagrams focuses on algebraic CBDs only. Here, time has been abstracted away, which may lead to "algebraic loops" which need to be detected and revoled. The second semantics focuses on discrete-time CBDs. Time is abstracted as a discrete counter. The introduction of memory in the form of a delay block allows, in combination with feedback loops in the CBD, for the expression of complex dynamics. The third semantics treats time as continuous. To allow for simulation on a digital computer, discretization is required. As such, continuous-time CBDs are approximated and mapped onto discrete-time CBDs. Such approximation introduces numerical errors which must be dealt with. Even the discretized level is still an idealization as the numerical values as not Real numbers, but are implemented as floating point approximations.

Once CBDs are well understood, the tutorial gives a very basic introduction to automatic control. In engineering practice, the behaviour of virtually every physical systems (also known as "plant") is regulated by some form of controller. The principles of automatic control will be explained by means of the most simple Proportional, Integral and Derivative (PID) controller. The effect of the different parts of such a controller will be explained. A PID controller will be modelled in the form of a continuous-time CBD. This is then the basis for discritezation to a discrete-time CBD and subsequent synthesis of control software. To demonstrate the concepts, a PID controller will be developed and its optimal parameters will be estimated for the simple cruise control of a vehicle.

Proposed Length

We propose a full-day (6 hours) tutorial.

The topic of dealing with Physical Systems is a complex one, and quite outside the comfort zone of the intended audience. To fully explain it requires a sufficient amount of time.

The presenters have experience with giving similar tutorials in the past. Notice that, especially for the MODELS community, it is important to cover the "why", the "what" and the "how" of the topic, as the audience is interested in deep understanding.

Past experience, with a tutorial on a-causal modelling (and Modelica in particular) at MODELS 2015 in Ottawa, showed (1) that a proper explanation takes time and (2) that the topic of (PID) control is the "missing link": it demonstrates where code gets generated and deployed (or conversely, where the code running in modern software-intensive systems comes from).

The proposal has two main parts. The first three hour part covers modelling of physical systems down to computational causality assignment resulting in Causal Block Diagrams (CBDs). The second three-hour part starts by explaining the semantics of CBDs. This makes the link between the models of physical systems and their ultimate realization in simulation software (as used in the Functional Mockup Interface standard). The second link between physical systems and software comes from the introduction of controllers which steer a physical system (known as "plant") to a desired behaviour. It is these controllers that are first modelled as CBDs and subsequently discretized and realized as software.

The two parts can be followed independently, but for full understanding of the CBD/PID part, it is best taken after the part focusing on the physics.

Level of the Tutorial

Introductory: Only basic knowledge of object-oriented software design/programming, graph algorithms, and calculus are required. Remembering some undergraduate physics helps.

Target Audience

Modellers with an interest in the link between physical systems and software modelling. Those who wish to understand the broad range of languages available for physical system modeling, and their rationale. In particular, (domain-specific) modelling language engineers may find this a refreshing view on a class of languages not rooted in software.

Description and Intended Outline

This tutorial aims to introduce the families of languages used to model physical systems, to an audience of software modellers. It goes to the essence of the following language families, sorted from the lowest modeling effort to the highest:

1) problem/technology-specific (e.g., a nuclear power plant modeling language);
2) domain-specific (e.g., SimMechanics); power-flow (e.g., Bond Graphs);
3) computationally a-causal (e.g., Modelica and Simscape);
4) computationally causal continuous (e.g., Simulink block diagrams);
5) computationally causal discretized (e.g., discrete-time block diagrams); and
6) black-box causal discretized (e.g., Functional Mockup Units);
7) untimed "algebraic" block diagrams (and their link with synchronous data flow).

Note that the first three topics are covered in the first three hour part of the tutorial. The remaining four topics are covered in the second three hour part of the tutorial.

This second part also introduces, at a very introductory level, control theory in general, and Proportional – Integral – Derivative (PID) control in particular. The introduction is done by means of a simple example of cruise control of an autonomous vehicle. Most importantly, the PID controller is modelled as a continuous-time Causal Block Diagram which allows, after discretization, for the synthesis of controller software.

Having followed this tutorial, the audience will understand how to write or generate software code that interacts with physical systems (e.g., due to inertia, turning off a physical systems does not stop it), including the crucial aspect of control.

To achieve these goals, the tutorial will cover:

- general laws of physics used to derive physical system equations,
- algorithms to transform models across the language families introduced above (e.g., causality assignment to translate a-causal models to causal ones, or numerical discretization to transform continuous models to discrete ones), and
- techniques to integrate and simulate multiple models, even if the contents of these models are protected (e.g., in binary form), or represents physical subsystems (e.g., test rigs). These scenarios are common in industry as externally supplied models contain Intellectual Property.
- PID controllers and how to realize them in software.

Additional Information

Similar Tutorials and Novelty

We presented tutorials on a-causal modelling in 2014 and 2015 at MODELS in Valencia and Ottawa respectively. These tutorials correspond to the first half of the current proposal. The proposed tutorial is also based on a part of the MPM4CPS COST Action Training School held 18 - 21 November 2018 in Pisa, Italy (http://mpm4cps.eu/trainingSchools/pisa2018). The audience at the Training School consisted, like at MODELS, mostly of researchers with a software engineering background. The addition of the PID controller part proved to fill the gap identied earlier during the MODELS tutorials.

Required Infrastructure

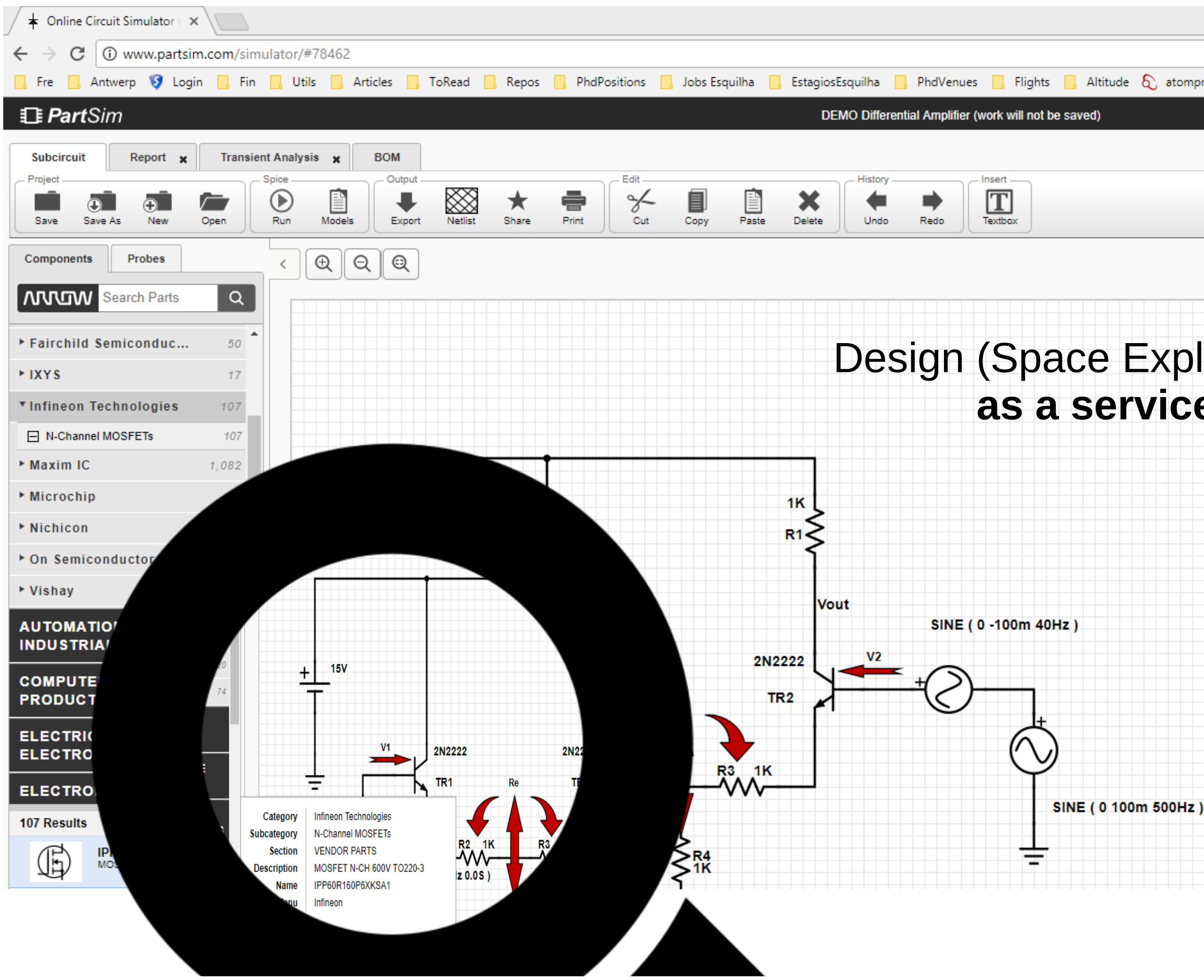
Besides a data projector, a white-board or black-board (or flipchart) is required.

Sample Slides

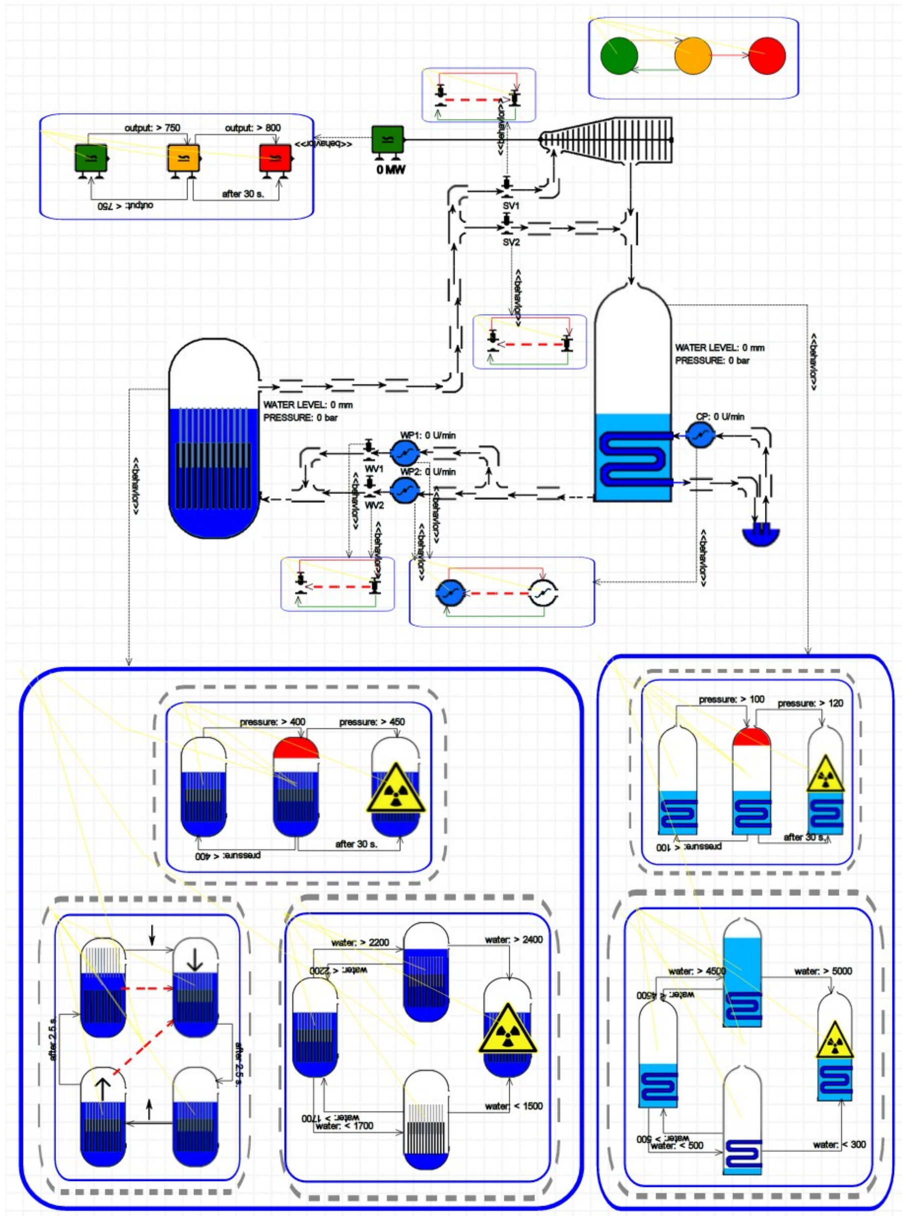
A number of sample slides of the previous versions of the tutorial are attached in the appendix.

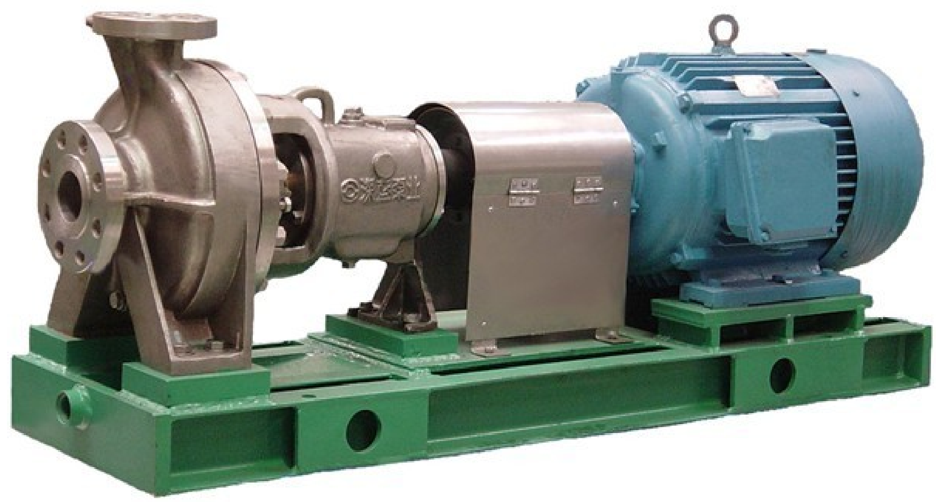
Virtual Build (technological)

http://www.partsim.com/



Design (Space Exploration)
as a service


output: > 750
output: > 800
after 30 s.
output: < 750
<<behavior>>
0 MW
SV1
SV2
WATER LEVEL: 0 mm
PRESSURE: 0 bar
WATER LEVEL: 0 mm
PRESSURE: 0 bar
CP: 0 U/min
WP1: 0 U/min
WP2: 0 U/min
WV1
WV2
pressure: > 400
pressure: > 450
after 30 s.
pressure: < 400
pressure: > 100
pressure: > 120
after 30 s.
pressure: < 100
after 2.5 s.
after 2.5 s.
water: > 2200
water: < 2200
water: > 2400
water: < 1500
water: > 1700
water: < 1700
water: > 4500
water: < 4500
water: > 5000
water: > 500
water: < 500
water: < 300

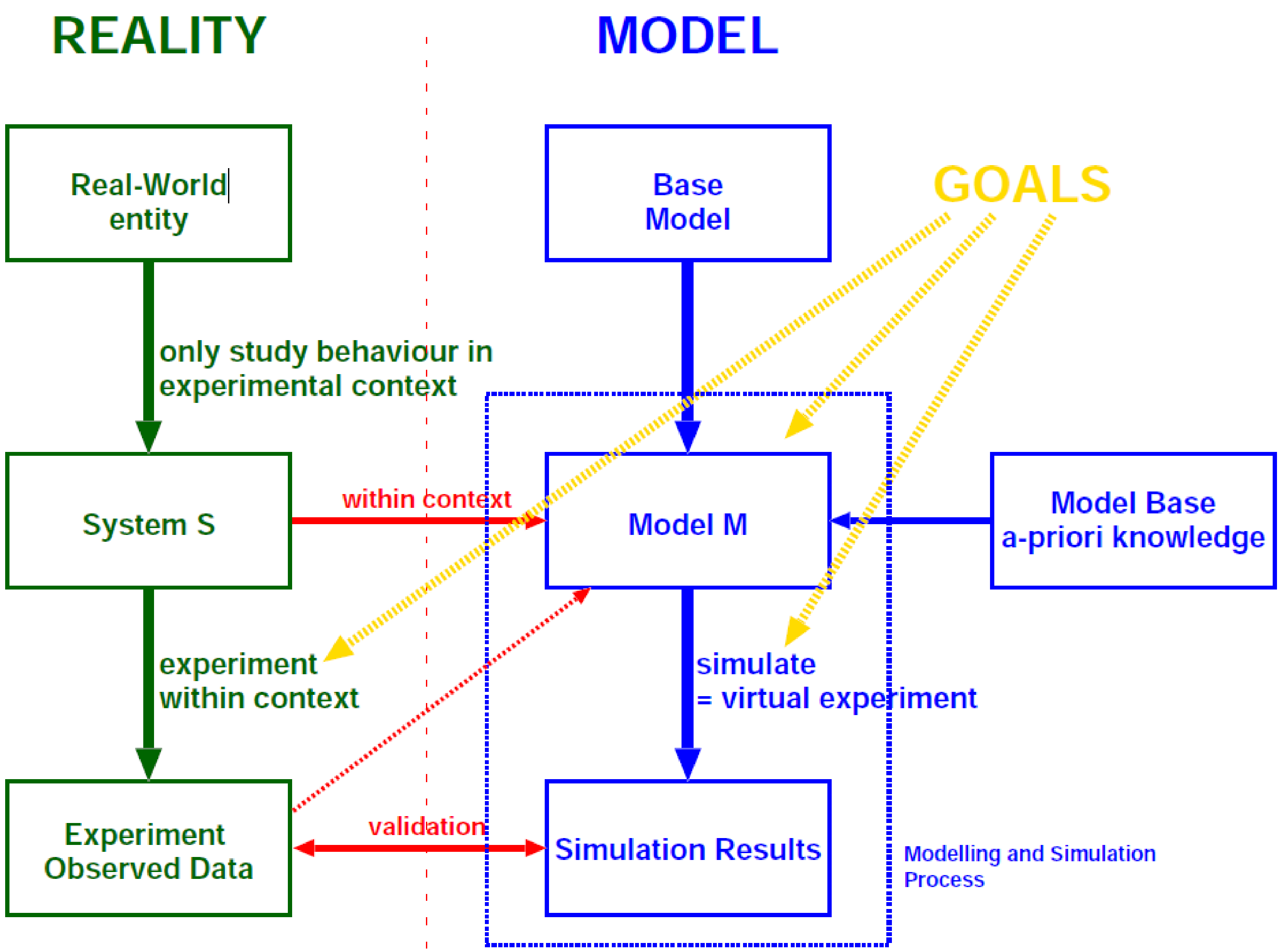


Boric Acid Transportation Pump

Product parameters

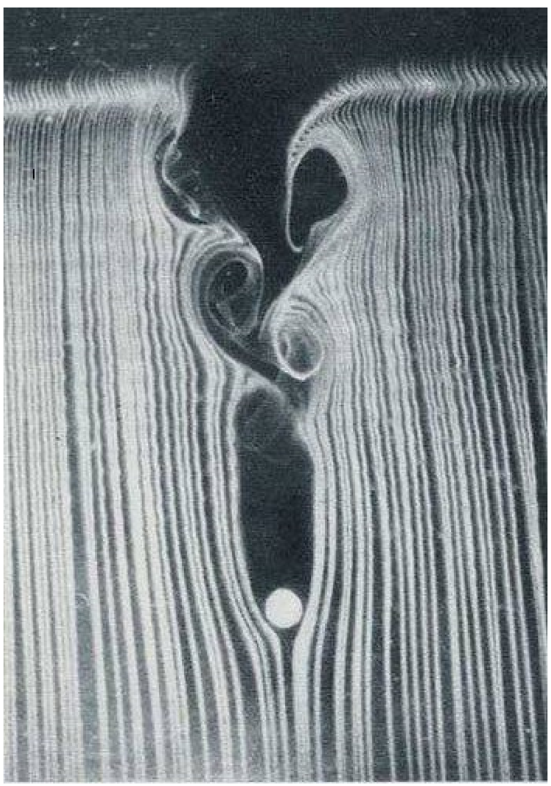
Design standards： RCC-M
Flow： 16.6m3/h
Head： 85m
Temperature： ~80°C
Pressure： 1.6MPa

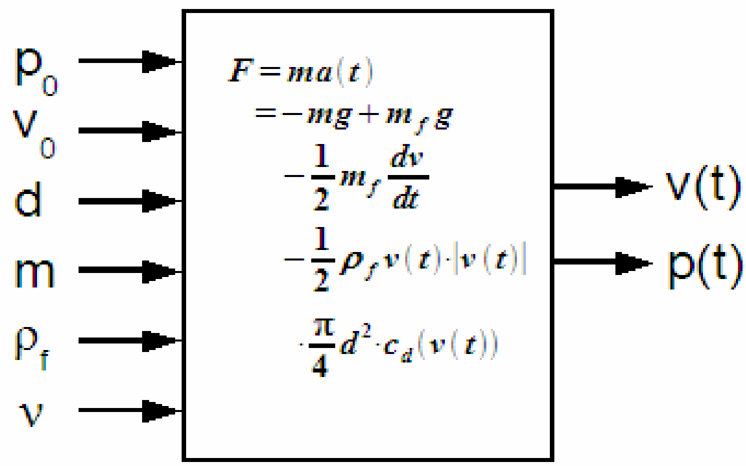
Used in 600MWe 、 900MWe 、 1000MWe PWR nuclear power plant boric acid transportation system.



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

Model Validity … Context?





Spiegel, M., Reynolds, P. F., & Brogan, D. C.
A Case Study of Model Context for Simulation Composability and Reusability.
In *Proceedings of the Winter Simulation Conference, 2005.* (Vol. 2005, pp. 437–444). IEEE.
http://doi.org/10.1109/WSC.2005.1574279

1. Invariant Constraints

1.a Sphere Attributes

1. Sphere Property - The body is a sphere and it remains spherical.
2. Smooth Property - The body is smooth and it remains smooth.
3. Impermeable Property - The body is completely impermeable.
4. Initial Velocity - The body has an initial velocity of v_0 that has no horizontal component of motion.
5. Angular Velocity - The body has no initial angular velocity.
6. Constant Mass - The mass of the body remains constant over time. The body does not experience ablation or accretion.
7. Constant Diameter - The diameter of the body remains constant over time.
8. Distribution of Mass - The body has a centrally symmetric mass distribution that remains constant over time.
9. Uncertainty Principle - The diameter of the body is much greater than the Plank length.
10. Brownian Motion - The mass and diameter of the body are large enough such that Brownian motion of the fluid has negligible impact on the body.
11. General Relativity - The mass of the body is low enough to ignore the gravitational curvature of space-time.

Implicit Assumptions!

1.c Earth Attributes

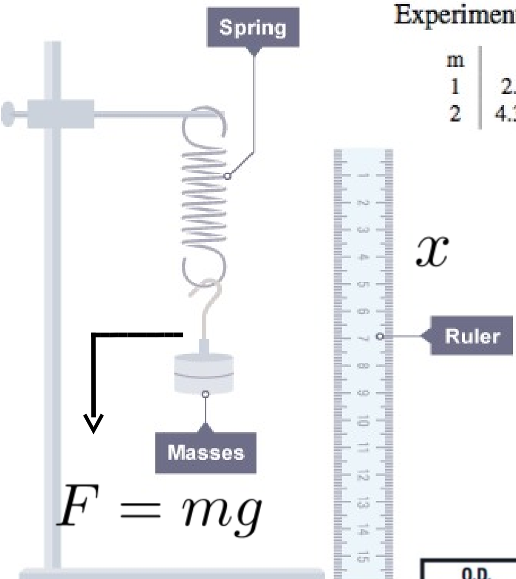
18. Flat Terrain - The ground does not have terrain and remains flat for all $t > 0$.
19. Coriolis Effect - The Earth is not rotating. We ignore the Coriolis effect.

2. Dynamic Constraints

20. Mach Speed - The velocity of the body is sufficiently less than the speed of sound for that medium.
21. Special Relativity - The velocity of the body is sufficiently less than the speed of light for that medium.
22. Reynolds Number - The Reynolds number remains between 10^{-2} and 10^{7} for all $t > 0$. The Reynolds number is a function of velocity.

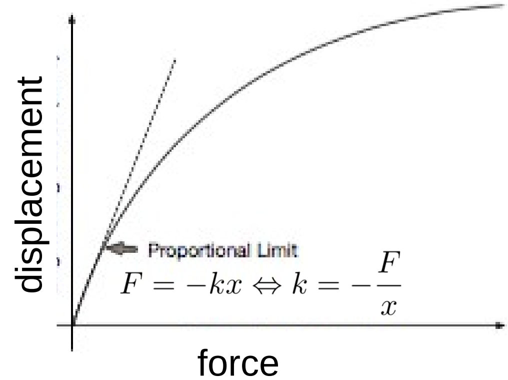
3. Inter-Object Constraints

23. Sphere/Fluid Interaction - The body and the fluid interact only through buoyancy and drag. For example, the body cannot dissolve in the fluid, nor can the body transfer heat to the fluid.
24. Sphere/Earth Interaction - The body and the earth interact only through the gravitational force.
25. Fluid/Earth Interaction - The fluid and the earth do not interact.
26. Closed System - The Earth, sphere, and fluid do not interact with any other objects.
27. Simple Gravity - Gravity is a constant downward force of $9.8\ m/s^2$.
28. One-Sided Gravity - The mass of the body is much less than the mass of the Earth. The Earth is not affected by the gravitational pull of the body.
29. Inelastic Collision - The collision between the sphere and the ground is perfectly inelastic.



Experimental spring results, with mass m in kg and displacement x (± 0.0001) in cm

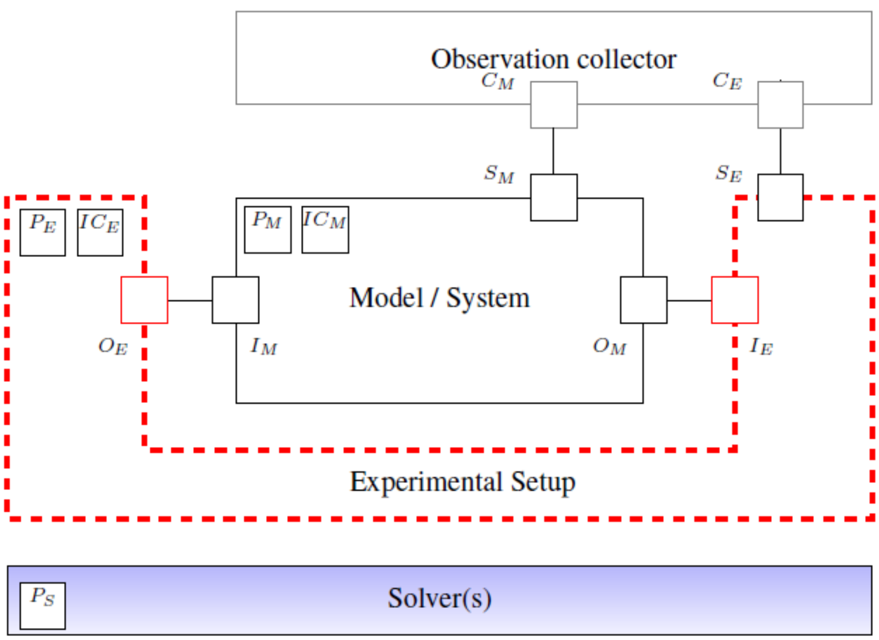
m	x	m	x	m	x	m	x	m	x
1	2.100	3	6.3749	5	10.4915	7	14.6081	9	19.0012
2	4.3166	4	8.4332	6	12.5489	8	16.7774		

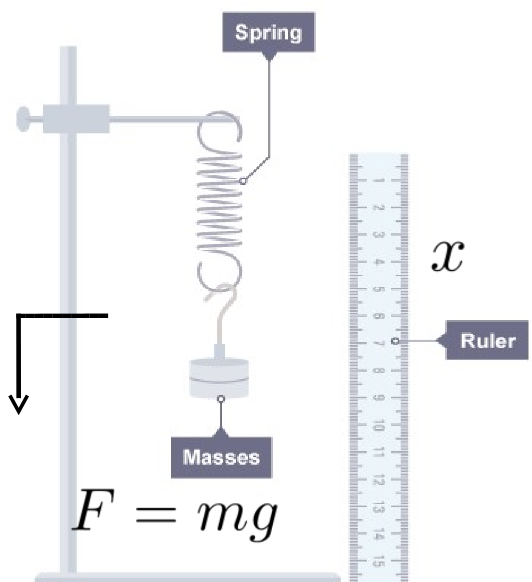


O.D.		CENTURY STOCK NUMBER	FREE LENGTH		I.D.		RATE		SUGG. MAX. DEFL.		SUGG. MAX. LOAD		SOLID LENGTH		WIRE DIA.		TOTAL COILS	MAT'L	ENDS	FNSH
Inches	mm		Inches	mm	Inches	mm	Lbs./In.	N/mm	Inches	mm	Lbs.	N	Inches	mm	Inches	mm				
0.036	.91	10075	.59	15.1	.022	.6	2.6	.46	.15	3.8	.39	1.7	.35	8.9	0.007	0.2	49.0	SST	C	N
0.036	.91	JJ-7	.63	15.9	.024	.6	1.6	.28	.16	4.1	.25	1.1	.25	6.2	0.006	0.2	40.0	SST	C	N
0.040	1.02	2924	.66	16.8	.020	.5	11	2.0	.13	3.2	1.4	6.4	.50	12.6	0.010	0.3	48.5	MW	C	N
0.040	1.02	10778	.69	17.5	.028	.7	1.0	.17	.35	8.9	.35	1.6	.30	7.7	0.006	0.2	49.5	MW	C	N
0.054	1.37	RR-6	.25	6.4	.036	.9	6.2	1.1	.09	2.2	.56	2.5	.16	4.1	0.009	0.2	16.5	SST	C	N
0.054	1.37	10619	.72	18.3	.038	1.0	1.6	.29	.37	9.3	.60	2.7	.32	8.1	0.008	0.2	39.0	MW	C	N
0.057	1.45	70000	.13	3.3	.045	1.1	3.7	.66	.07	1.7	.25	1.1	.04	1.0	0.006	0.2	5.75	MW	C	N
0.057	1.45	70000S	.13	3.3	.045	1.1	3.3	.57	.05	1.3	.17	.74	.04	1.0	0.006	0.2	5.75	SST	C	N
0.057	1.45	70009	.13	3.3	.043	1.1	6.9	1.2	.06	1.5	.40	1.8	.05	1.2	0.007	0.2	6.00	MW	C	N
0.057	1.45	70009S	.13	3.3	.043	1.1	6.0	1.1	.04	1.1	.26	1.2	.05	1.2	0.007	0.2	6.00	SST	C	N
0.057	1.45	70018	.13	3.3	.041	1.0	12	2.1	.05	1.2	.57	2.5	.06	1.4	0.008	0.2	6.13	MW	C	N
0.057	1.45	70018S	.13	3.3	.041	1.0	11	1.8	.03	.88	.37	1.6	.06	1.4	0.008	0.2	6.13	SST	C	N
0.057	1.45	70001	.19	4.8	.045	1.1	2.3	.40	.11	2.8	.25	1.1	.06	1.4	0.006	0.2	8.13	MW	C	N
0.057	1.45	70001S	.19	4.8	.045	1.1	2.0	.35	.08	2.1	.17	.74	.06	1.4	0.006	0.2	8.13	SST	C	N
0.057	1.45	70010	.19	4.8	.043	1.1	4.0	.70	.10	2.5	.40	1.8	.07	1.8	0.007	0.2	8.88	MW	C	N
0.057	1.45	70010S	.19	4.8	.043	1.1	3.5	.61	.07	1.9	.26	1.2	.07	1.8	0.007	0.2	8.88	SST	C	N
0.057	1.45	70019	.19	4.8	.041	1.0	7.4	1.3	.08	2.0	.57	2.5	.08	2.0	0.008	0.2	8.75	MW	C	N
0.057	1.45	70019S	.19	4.8	.041	1.0	6.4	1.1	.06	1.4	.37	1.6	.08	2.0	0.008	0.2	8.75	SST	C	N
0.057	1.45	70002	.25	6.4	.045	1.1	1.7	.30	.15	3.8	.25	1.1	.07	1.7	0.006	0.2	10.3	MW	C	N
0.057	1.45	70002S	.25	6.4	.045	1.1	1.5	.26	.11	2.8	.17	.74	.07	1.7	0.006	0.2	10.3	SST	C	N
0.057	1.45	70011	.25	6.4	.043	1.1	3.1	.54	.13	3.3	.40	1.8	.08	2.1	0.007	0.2	11.0	MW	C	N
0.057	1.45	70011S	.25	6.4	.043	1.1	2.7	.47	.10	2.5	.26	1.2	.08	2.1	0.007	0.2	11.0	SST	C	N
0.057	1.45	70020	.25	6.4	.041	1.0	5.3	.92	.11	2.8	.57	2.5	.10	2.5	0.008	0.2	11.5	MW	C	N
0.057	1.45	70020S	.25	6.4	.041	1.0	4.6	.80	.08	2.0	.37	1.6	.10	2.5	0.008	0.2	11.5	SST	C	N
0.057	1.45	70003	.31	7.9	.045	1.1	1.4	.24	.19	4.7	.25	1.1	.08	2.0	0.006	0.2	12.4	MW	C	N
0.057	1.45	70003S	.31	7.9	.045	1.1	1.2	.21	.14	3.6	.17	.74	.08	2.0	0.006	0.2	12.4	SST	C	N
0.057	1.45	70012	.31	7.9	.043	1.1	2.4	.42	.17	4.2	.40	1.8	.10	2.6	0.007	0.2	13.5	MW	C	N
0.057	1.45	70012S	.31	7.9	.043	1.1	2.1	.37	.12	3.2	.26	1.2	.10	2.6	0.007	0.2	13.5	SST	C	N
0.057	1.45	70021	.31	7.9	.041	1.0	4.1	.72	.14	3.6	.57	2.5	.12	3.1	0.008	0.2	14.3	MW	C	N
0.057	1.45	70021S	.31	7.9	.041	1.0	3.6	.62	.10	2.6	.37	1.6	.12	3.1	0.008	0.2	14.3	SST	C	N

www.centuryspring.com

Validity “Frame” ~ reproducibility





Denil, J., Klikovits, S., Mosterman, P. J., Vallecillo, A., & Vangheluwe, H. (2017).
The experiment model and validity frame in M&S.
In *Proceedings of the Symposium on Theory of Modeling & Simulation* (Vol. 49).

Vanherpen, K., Denil, J., De Meulenaere, P., & Vangheluwe, H. (2016).
Ontological Reasoning as an Enabler of Contract-Based Co-design.
In C. Berger, M. R. Mousavi, & R. Wisniewski (Eds.), *Cyber Physical Systems. Design, Modeling, and Evaluation: 6th International Workshop, CyPhy 2016, Pittsburgh, PA, USA, October 6, 2016, Revised Selected Papers* (pp. 101–115). Cham: Springer International Publishing.
http://doi.org/10.1007/978-3-319-51738-4_8

•

A-Causal Modelling in Context

- Problem-Specific (technological)
- Domain-Specific (e.g., translational mechanical)
- (general) Laws of Physics
- Power Flow/Bond Graphs (physical: energy/power)
- Computationally a-causal
  (Mathematical and Object-Oriented) ← **Modelica**
- Causal Block Diagrams (data flow)
- Numerical (Discrete) Approximations
- Computer Algorithmic + Numerical
  (Floating Point vs. Fixed Point)
- As-Fast-As-Possible vs. Real-time (XiL)
- Hybrid (discrete-continuous) modelling/simulation
- Hiding IP: Composition of Functional Mockup Units (FMI)
- Dynamic Structure

Dymola

Dokumentutgivare
Lund Institute of Technology

Dokumentnamn
REPORT

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

Handläggare
Karl Johan Åström

Utgivningsdatum
May 1978

Ärendebeteckning

Författare
Hilding Elmqvist

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

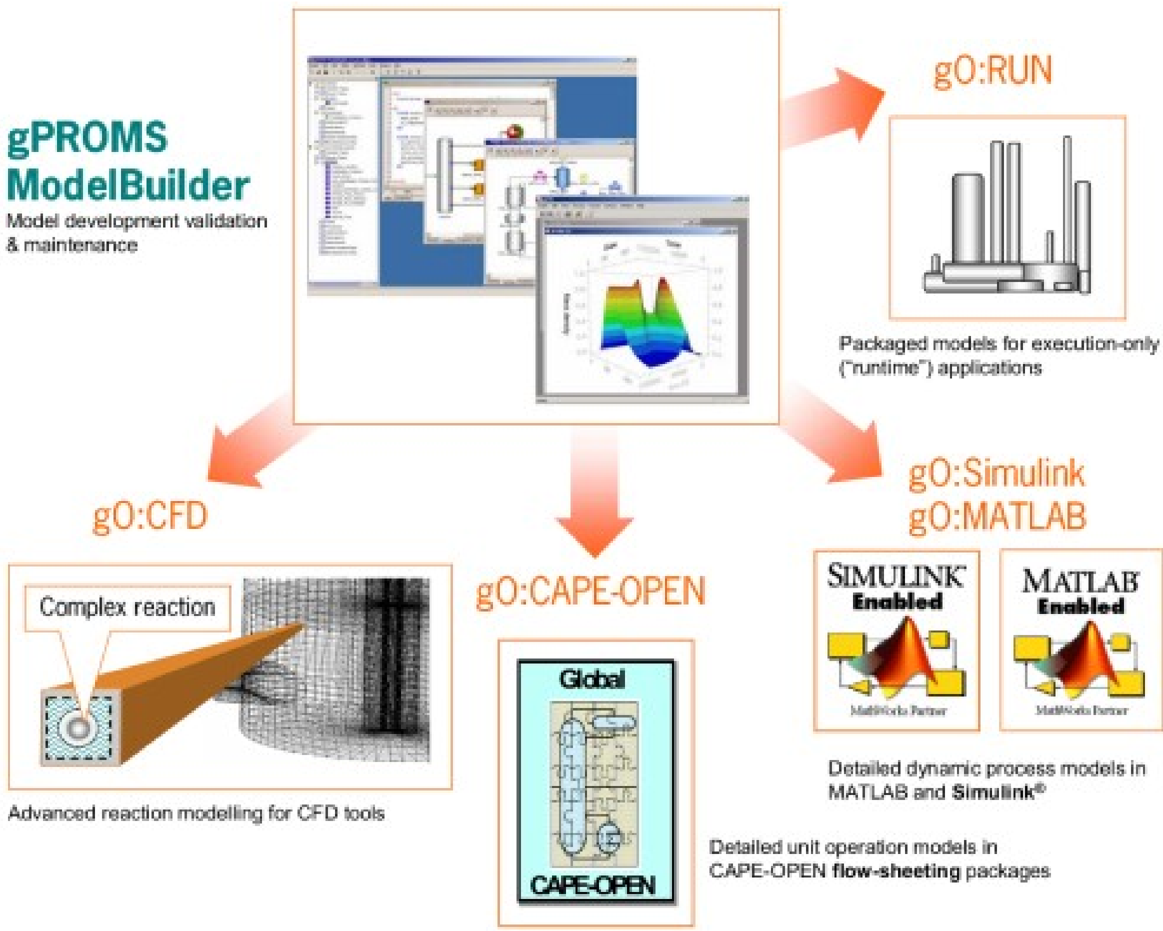
Pris

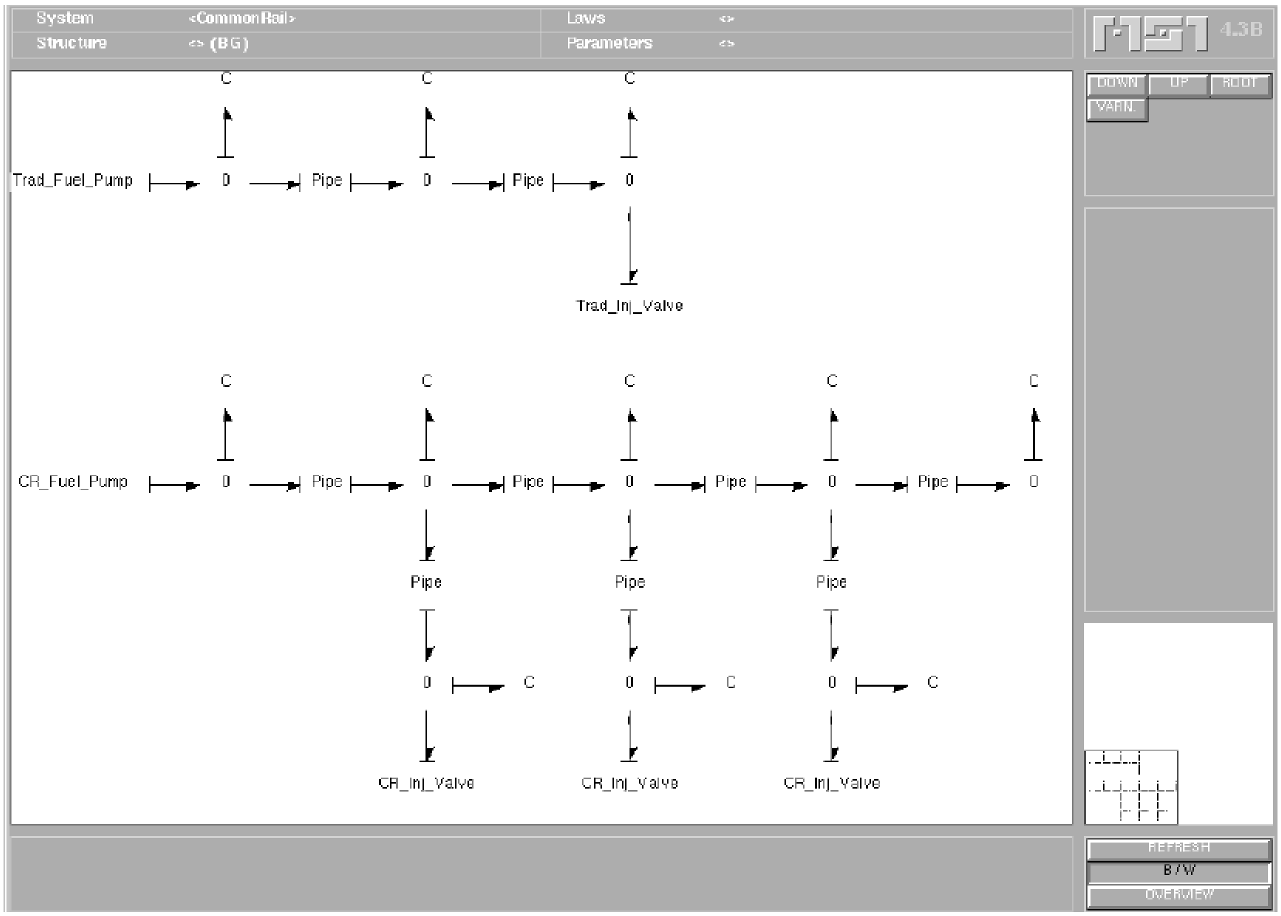
DOKUMENTDATABLAD enligt SIS 62 10 12

SIS-DB 1

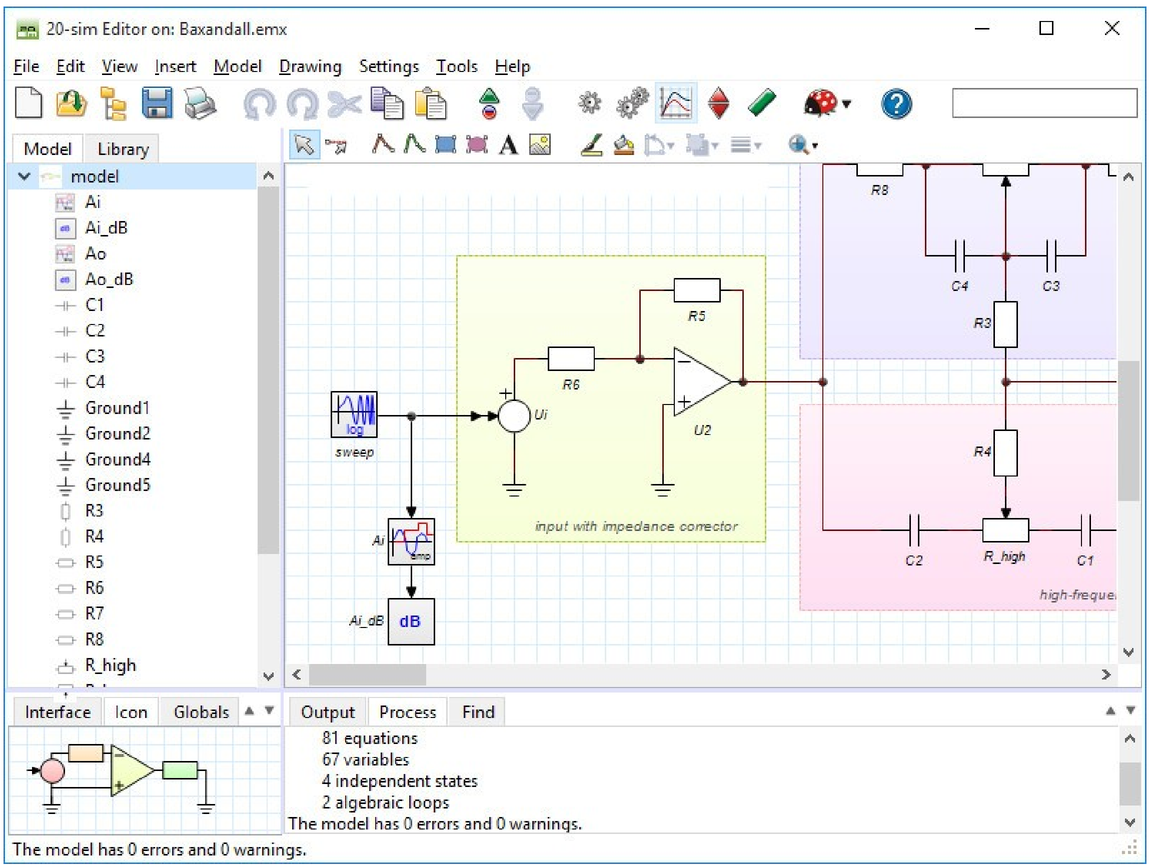
Blankett LU 11:25 1976–07


Simulation in Europe
esprit
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
MODELICA
OpenModelica

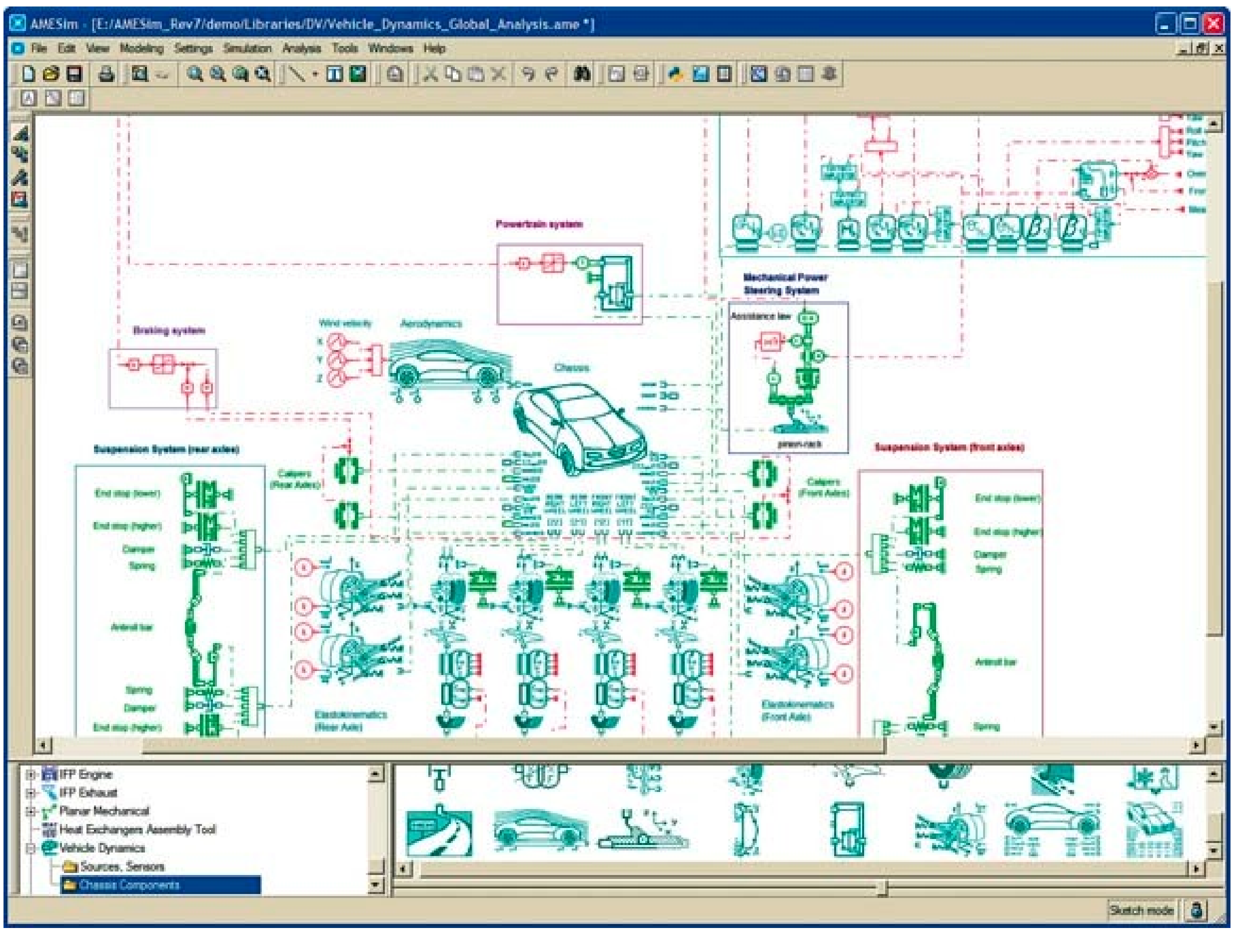

gPROMS
ModelBuilder
Model development validation
& maintenance
gO:RUN
Packaged models for execution-only
("runtime") applications
gO:CFD
Complex reaction
Advanced reaction modelling for CFD tools
gO:CAPE-OPEN
Global
CAPE-OPEN
Detailed unit operation models in
CAPE-OPEN flow-sheeting packages
gO:Simulink
gO:MATLAB
SIMULINK
Enabled
MATLAB
Enabled
Detailed dynamic process models in
MATLAB and Simulink®

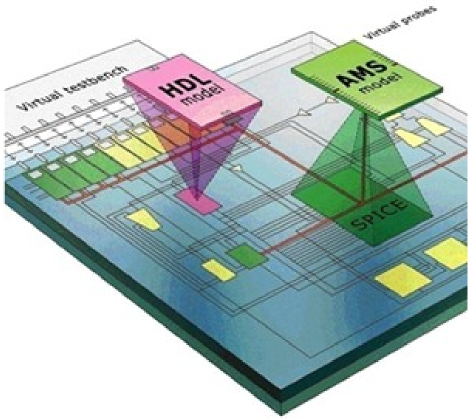


System <CommonRail>
Structure <> (BG)
Laws <>
Parameters <>
4.3B
DOWN
UP
ROOT
VARN.
Trad_Fuel_Pump
Pipe
Pipe
Trad_Inj_Valve
CR_Fuel_Pump
Pipe
Pipe
Pipe
Pipe
CR_Inj_Valve
CR_Inj_Valve
CR_Inj_Valve
REFRESH
B / W
OVERVIEW

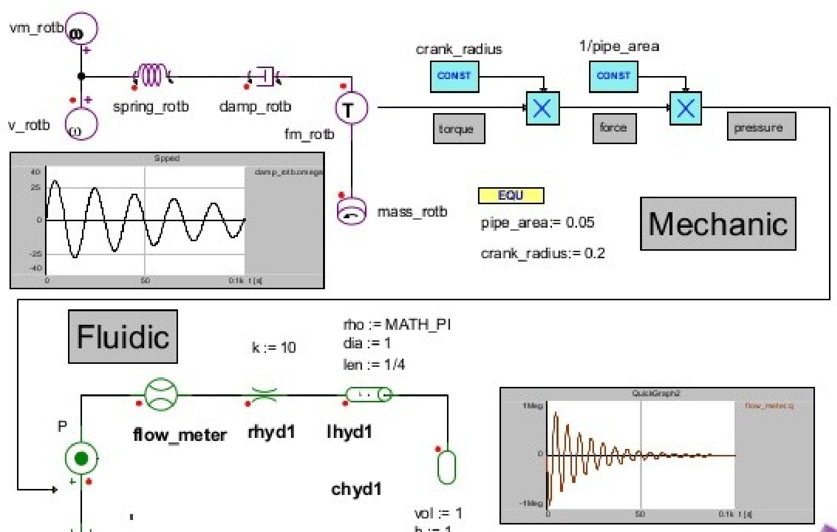

20-sim Editor on: Baxandall.emx
File Edit View Insert Model Drawing Settings Tools Help
Model Library
model
Ai
Ai_dB
Ao
Ao_dB
C1
C2
C3
C4
Ground1
Ground2
Ground4
Ground5
R3
R4
R5
R6
R7
R8
R_high
sweep
Ui
R6
R5
U2
input with impedance corrector
Ai
Ai_dB
dB
R8
C4
C3
R3
R4
C2
R_high
C1
high-freque
Interface 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.

Imagine.Lab AMESim



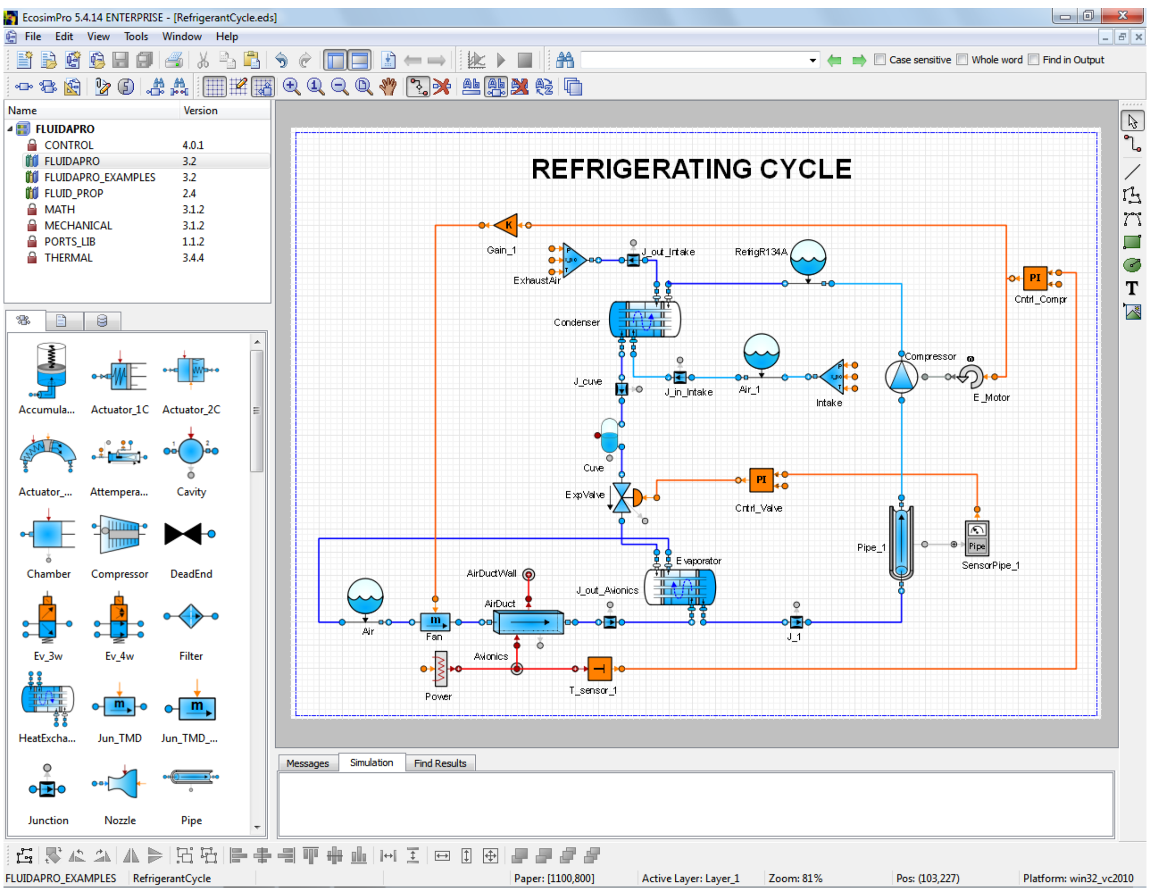


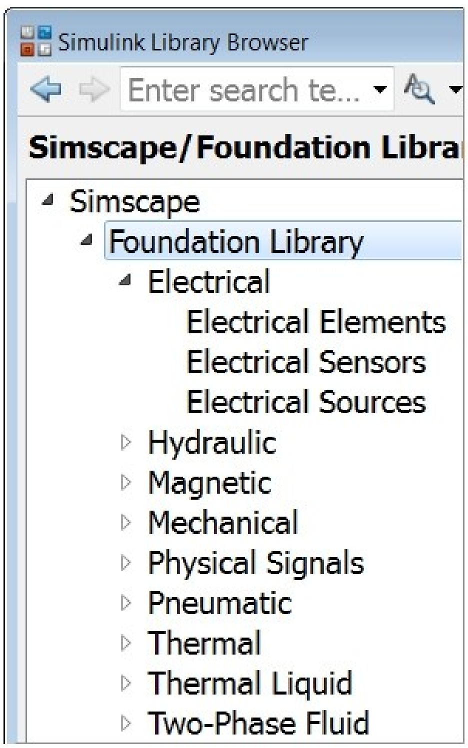
VHDL-AMS Multi Domain Design

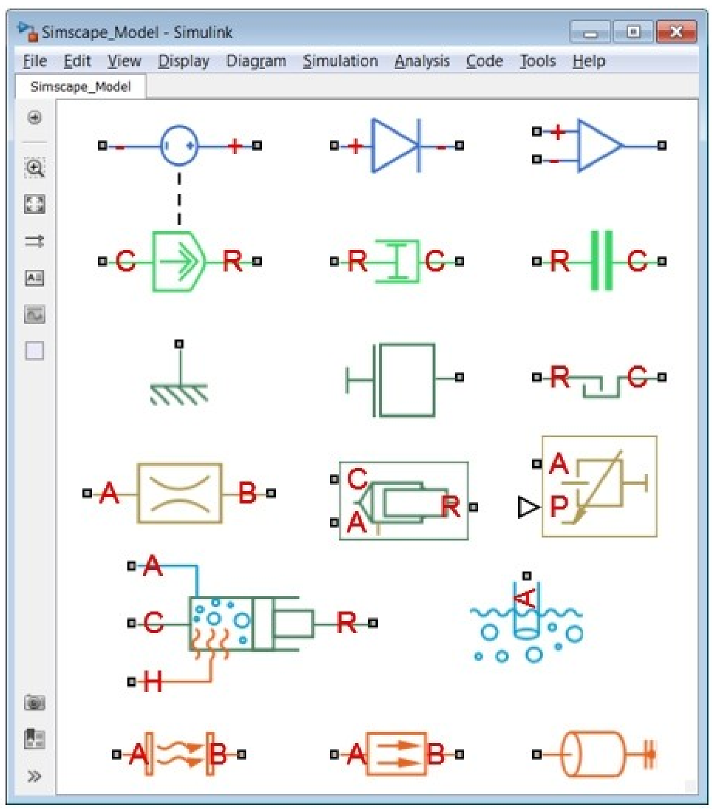


ANSOFT CORPORATION

EcosimPro
Modelling and Simulation Software




Simulink Library Browser
Enter search te...
Simscape/Foundation Libra
Simscape
Foundation Library
Electrical
Electrical Elements
Electrical Sensors
Electrical Sources
Hydraulic
Magnetic
Mechanical
Physical Signals
Pneumatic
Thermal
Thermal Liquid
Two-Phase Fluid


Simscape_Model - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
Simscape_Model

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

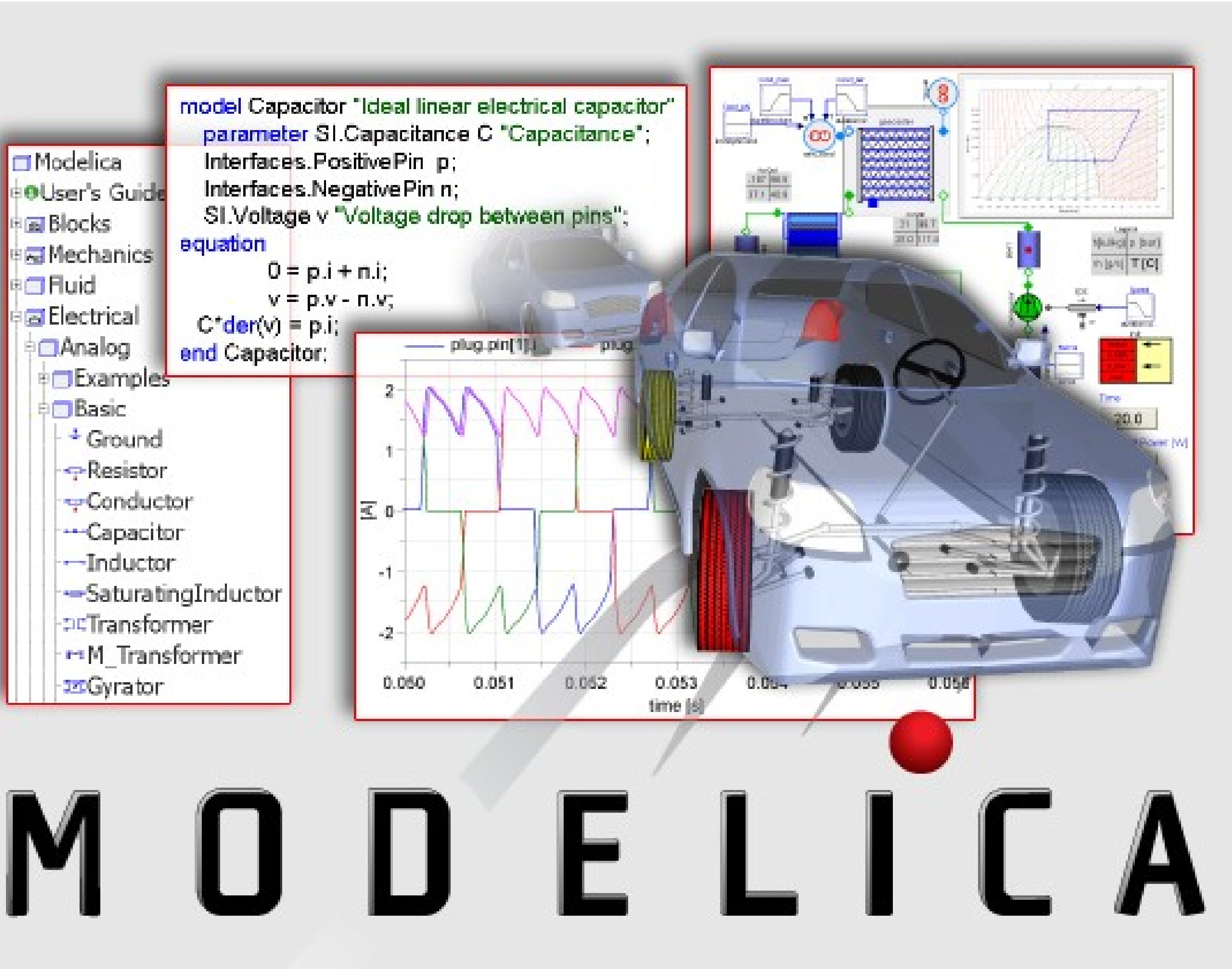
EOOLT 2016 Proceedings (ACM Digital Library)

Workshop site (archived)

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

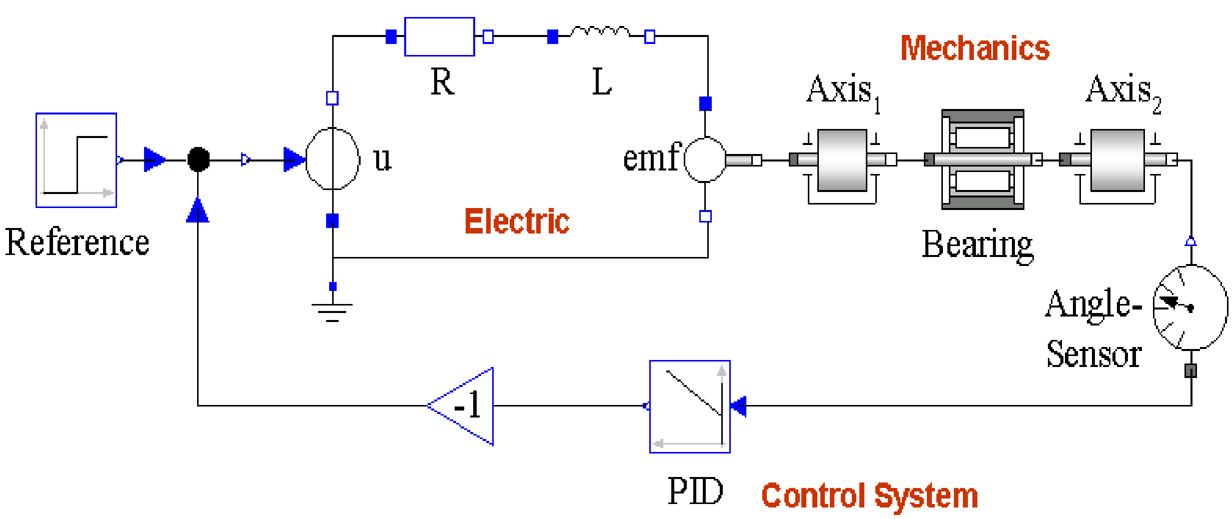
EOOLT 2014 Proceedings (ACM Digital Library)

Workshop site (archived)


Modelica
User's Guide
Blocks
Mechanics
Fluid
Electrical
Analog
Examples
Basic
Ground
Resistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
model Capacitor "Ideal linear electrical capacitor"
parameter SI.Capacitance C "Capacitance";
Interfaces.PositivePin p;
Interfaces.NegativePin n;
SI.Voltage v "Voltage drop between pins";
equation
0 = p.i + n.i;
v = p.v - n.v;
C*der(v) = p.i;
end Capacitor;
plug.pin[1].i
[A]
time [s]
0.050
0.051
0.052
0.053
MODELICA

Multi-Domain Modeling

http://www.modelica.org



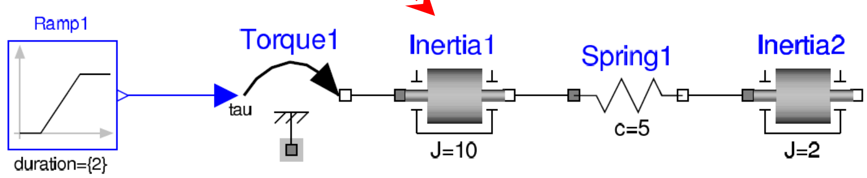
this slide from Peter Fritzson's Modelica tutorial

Multi-Domain Modeling

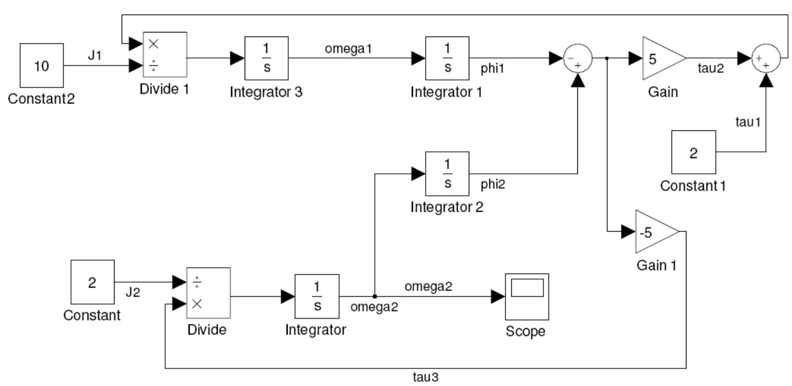
Visual Acausal Hierarchical Component Modeling

Keeps the physical structure

Acausal model (Modelica)



Causal block-based model (Simulink)



this slide from Peter Fritzson's Modelica tutorial

MODELICA

- Model exchange/re-use standard (Modelica Association)
- Modelica Standard Library (MSL)
- Object-oriented, hierarchical; semantics based on flattening
- Computationally a-causal modelling; semantics based on DAEs
- Originated in Hilding Elmquist's 1978 PhD thesis @ Lund
- Early 1990's: Modelica Design Team (started in SiE)

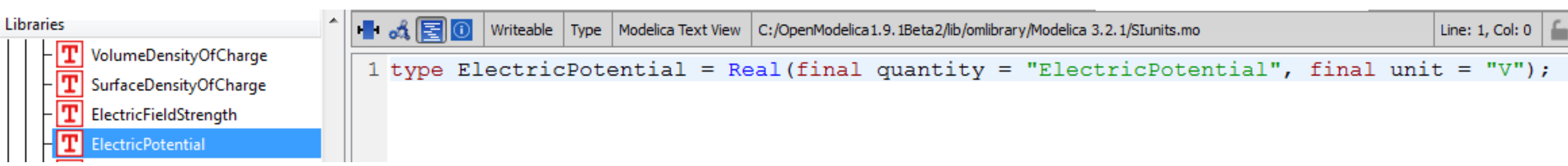
MODELICA

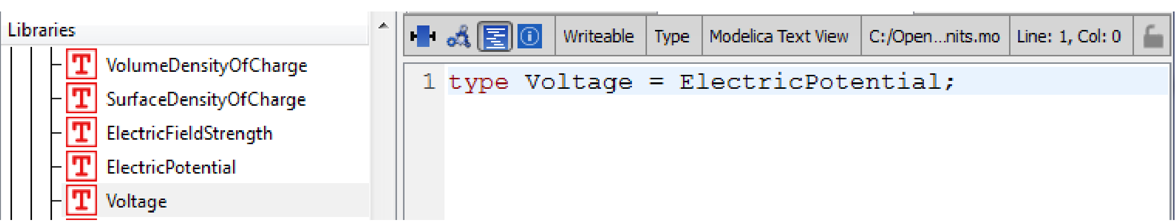
- hybrid (discrete-time/discrete-event) constructs (e.g., used to model network protocols based on TrueTime http://www.control.lth.se/truetime/)
- Limited support for Dynamic Structure models (i.e., no “agents”)
- Separate model from its (numerical) solution ...
- Generate Functional Mockup Interface (FMI) compliant simulation units
- Currently: many commercial and open (e.g., OpenModelica) tools
- Related: Mathworks Simscape, EcosimPro, NMF, gProms, ...

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;
```

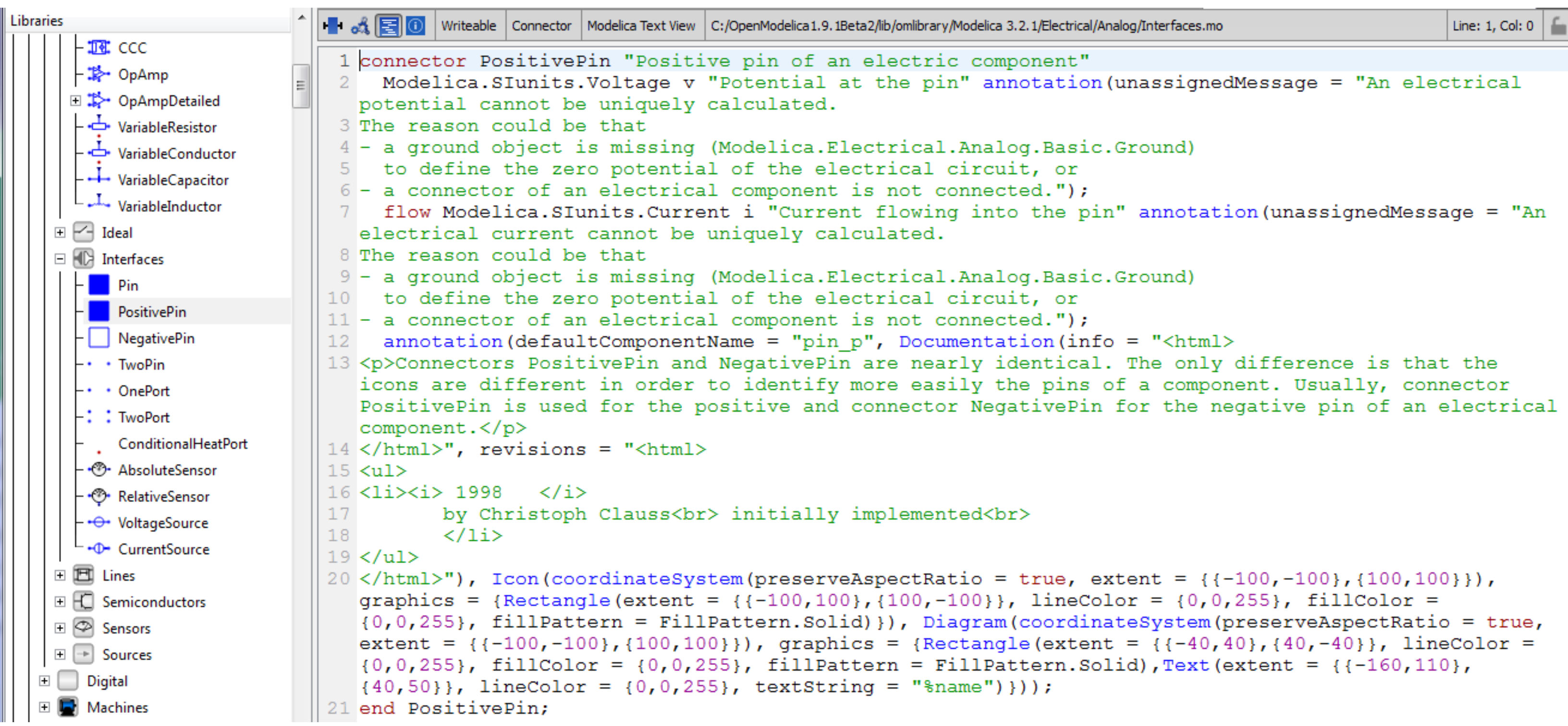
Beware: variables are **signals** (functions of **time**)!


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;
```

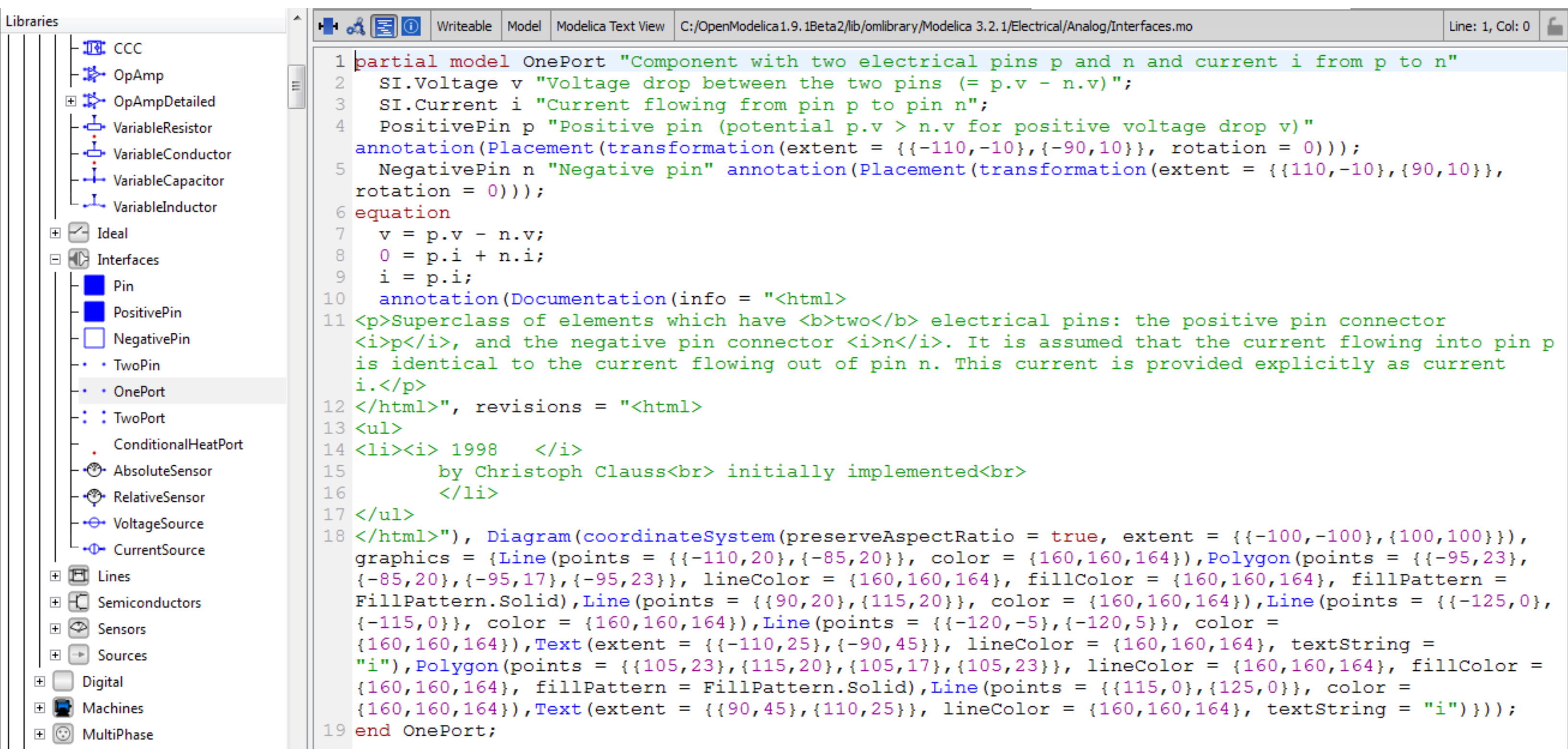

Libraries
CCC
OpAmp
OpAmpDetailed
VariableResistor
VariableConductor
VariableCapacitor
VariableInductor
Ideal
Interfaces
Pin
PositivePin
NegativePin
TwoPin
OnePort
TwoPort
ConditionalHeatPort
AbsoluteSensor
RelativeSensor
VoltageSource
CurrentSource
Lines
Semiconductors
Sensors
Sources
Digital
Machines
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

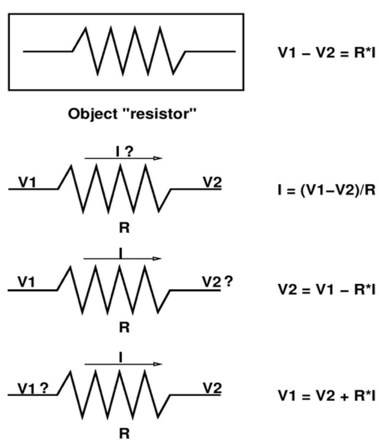
```
partial model OnePort
  "Component with two electrical pins p and n
   and current i from p to n"
  Voltage v "Voltage drop between the two pins (= p.v - n.v)";
  Current i "Current flowing from pin p to pin n";
  PositivePin p;
  NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;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

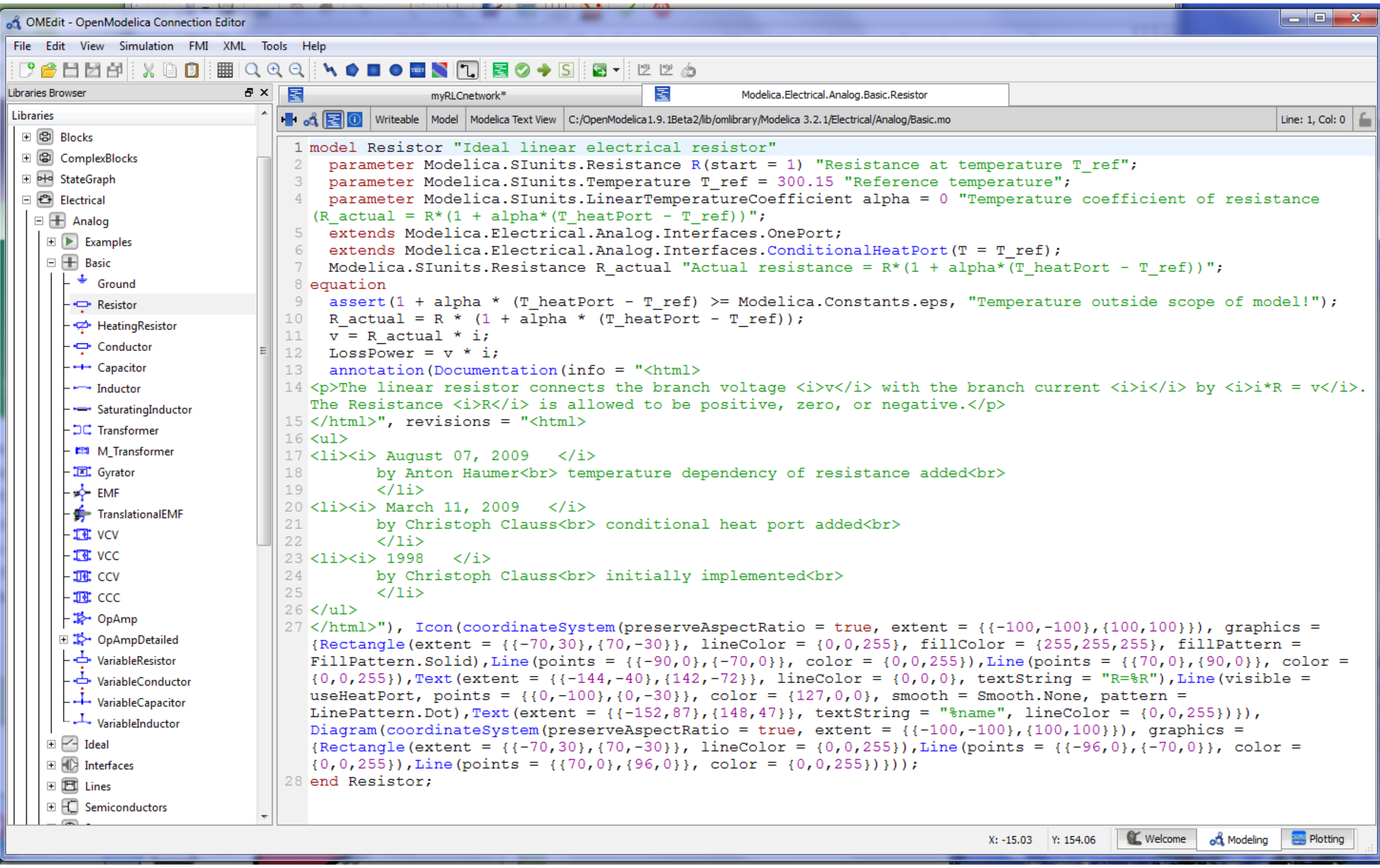
</html>"), Diagram(coordinateSystem(preserveAspectRatio = true, extent = {{-100,-100},{100,100}}),
graphics = {Line(points = {{-110,20},{-85,20}}, color = {160,160,164}),Polygon(points = {{-95,23},
{-85,20},{-95,17},{-95,23}}, lineColor = {160,160,164}, fillColor = {160,160,164}, fillPattern =
FillPattern.Solid),Line(points = {{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;
```

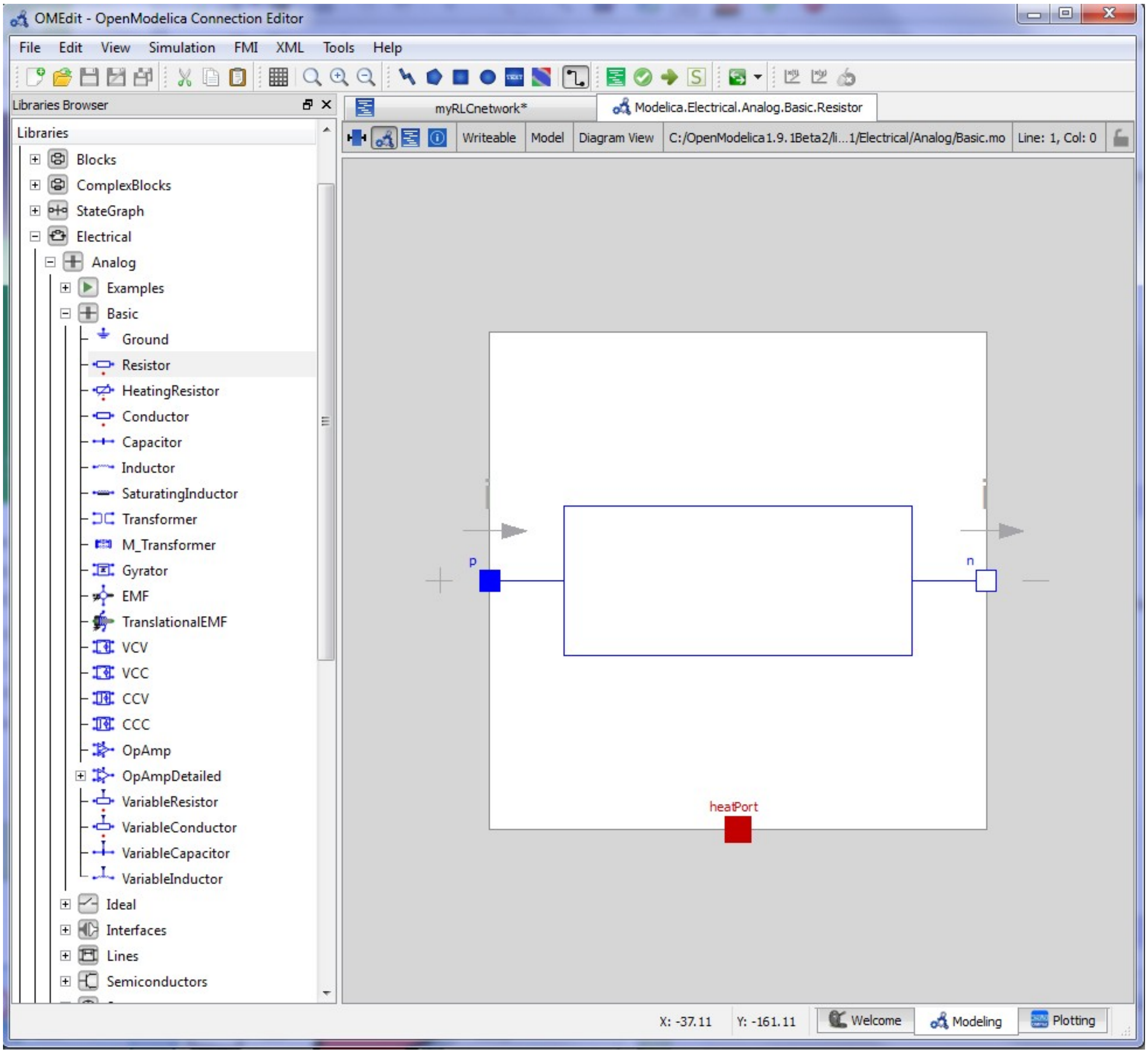

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
HeatingResistor
Conductor
Capacitor
Inductor
SaturatingInductor
Transformer
M_Transformer
Gyrator
EMF
TranslationalEMF
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

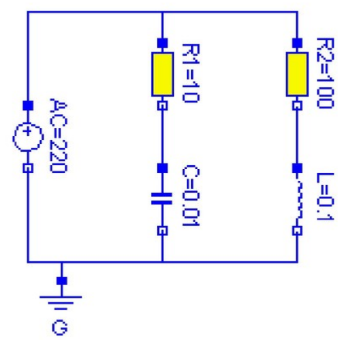
<i> 1998 </i>
by Christoph Clauss
 initially implemented

</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
myRLCnetwork*
Modelica.Electrical.Analog.Basic.Resistor
Writeable
Model
Diagram View
C:/OpenModelica1.9.1Beta2/li...1/Electrical/Analog/Basic.mo
Line: 1, Col: 0
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
p
n
heatPort
X: -37.11
Y: -161.11
Welcome
Modeling
Plotting

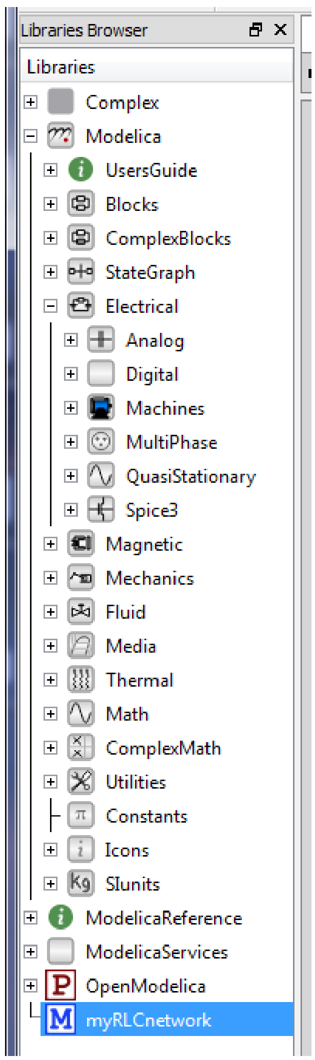
The circuit

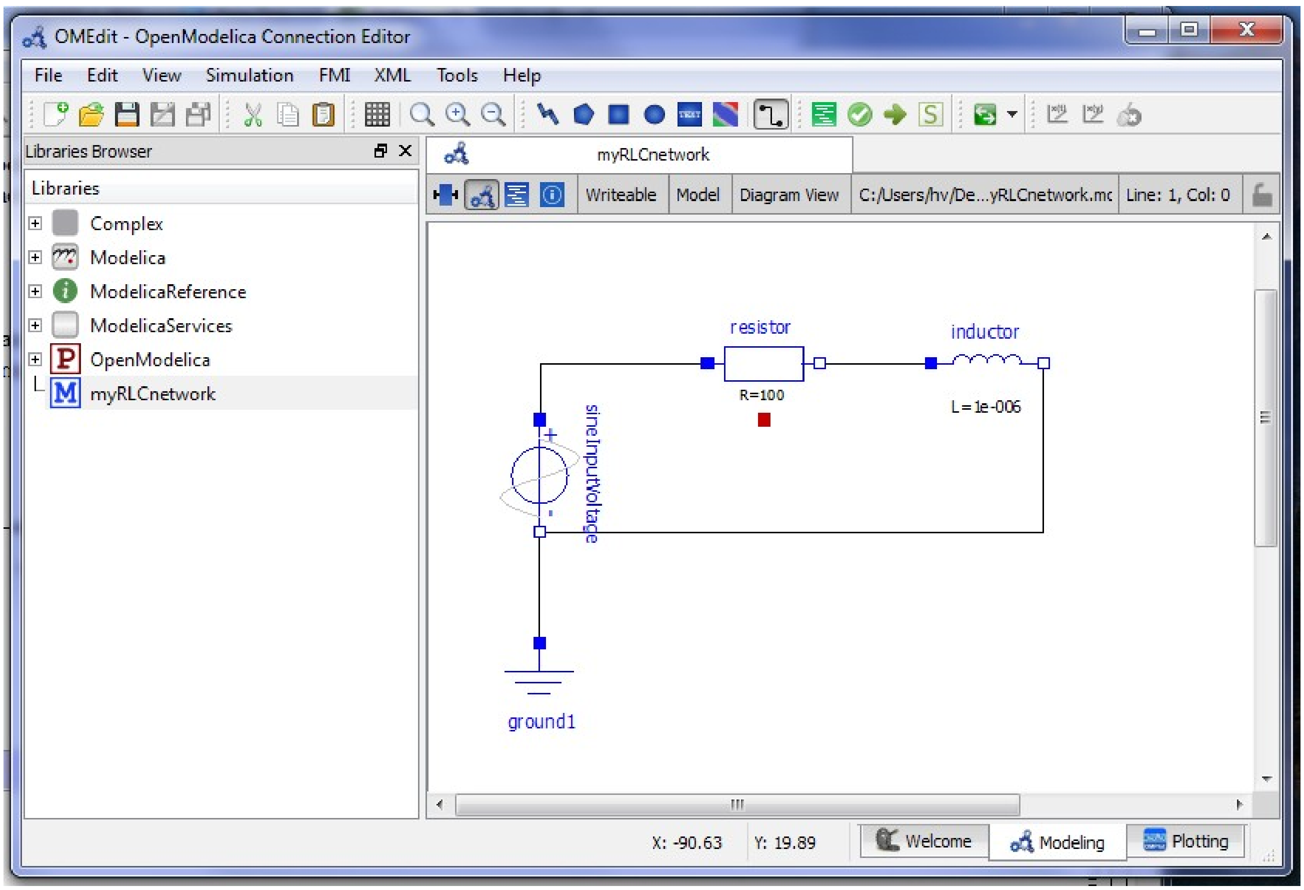
```
model circuit
 Resistor  R1(R=10);
 Capacitor C(C=0.01);
 Resistor  R2(R=100);
 Inductor  L(L=0.1);
 VsourceAC AC;
 Ground    G;
equation
 connect(AC.p, R1.p);
 connect(R1.n, C.p);
 connect(C.n, AC.n);
 connect(R1.p, R2.p);
 connect(R2.n, L.p);
 connect(L.n, C.n);
 connect(AC.n, G.p);
end circuit;
```

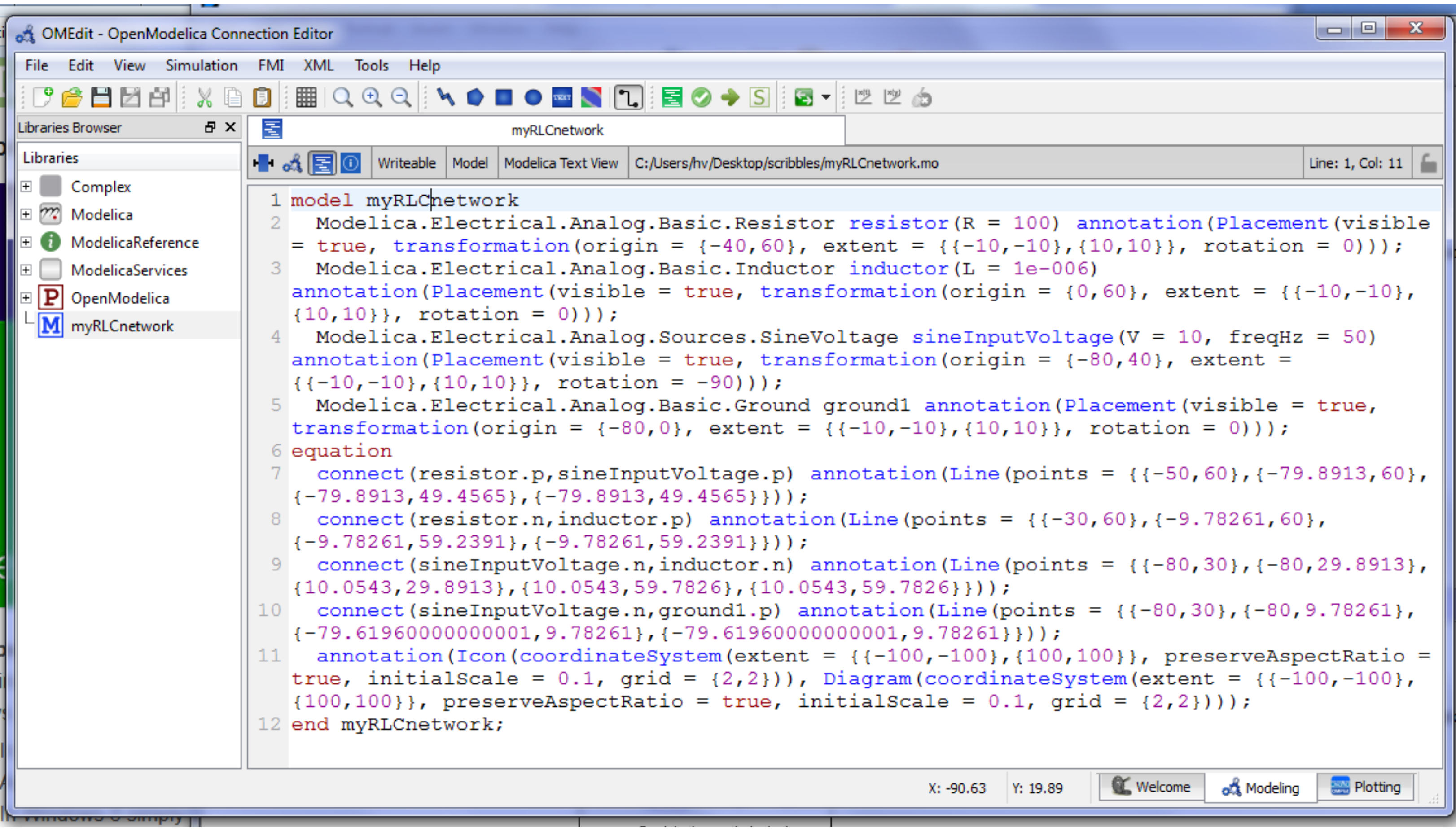


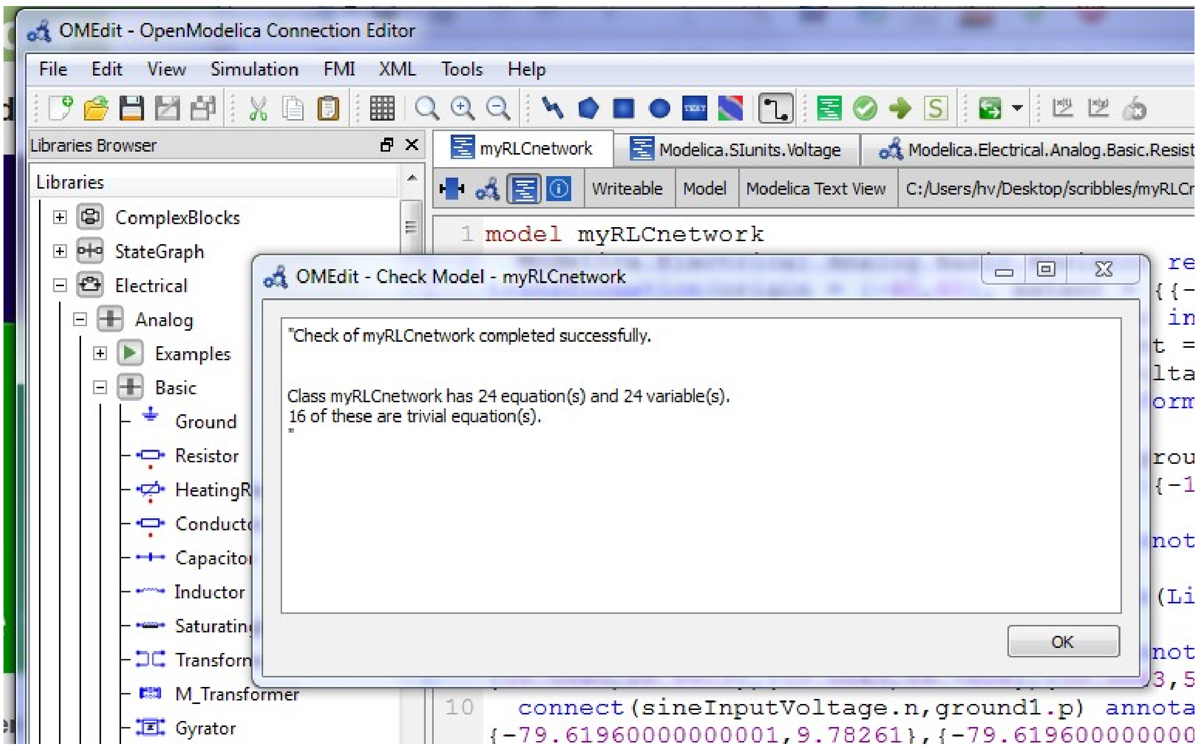
Meaning: set of Differential Algebraic Equations (DAEs) obtained by

1. expanding inheritance/instantiation
2. flattening hierarchy, unique names
3. expanding connect() into equations (across vs. flow)


Libraries Browser
Libraries
Complex
Modelica
UsersGuide
Blocks
ComplexBlocks
StateGraph
Electrical
Analog
Digital
Machines
MultiPhase
QuasiStationary
Spice3
Magnetic
Mechanics
Fluid
Media
Thermal
Math
ComplexMath
Utilities
Constants
Icons
SIunits
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork


OMEdit - OpenModelica Connection Editor
File
Edit
View
Simulation
FMI
XML
Tools
Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
Writeable
Model
Diagram View
C:/Users/hv/De...yRLCnetwork.mo
Line: 1, Col: 0
resistor
R=100
inductor
L=1e-006
sineInputVoltage
ground1
X: -90.63
Y: 19.89
Welcome
Modeling
Plotting

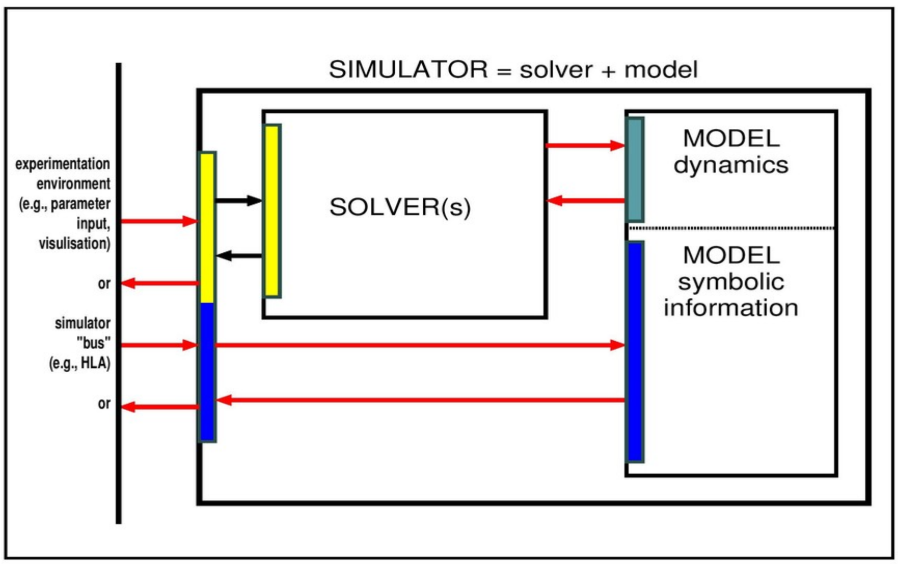

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
Writeable Model Modelica Text View C:/Users/hv/Desktop/scribbles/myRLCnetwork.mo
Line: 1, Col: 11
model myRLCnetwork
Modelica.Electrical.Analog.Basic.Resistor resistor(R = 100) annotation(Placement(visible = true, transformation(origin = {-40,60}, extent = {{-10,-10},{10,10}}, rotation = 0)));
Modelica.Electrical.Analog.Basic.Inductor inductor(L = 1e-006) annotation(Placement(visible = true, transformation(origin = {0,60}, extent = {{-10,-10},{10,10}}, rotation = 0)));
Modelica.Electrical.Analog.Sources.SineVoltage sineInputVoltage(V = 10, freqHz = 50) annotation(Placement(visible = true, transformation(origin = {-80,40}, extent = {{-10,-10},{10,10}}, rotation = -90)));
Modelica.Electrical.Analog.Basic.Ground ground1 annotation(Placement(visible = true, transformation(origin = {-80,0}, extent = {{-10,-10},{10,10}}, rotation = 0)));
equation
connect(resistor.p,sineInputVoltage.p) annotation(Line(points = {{-50,60},{-79.8913,60},{-79.8913,49.4565},{-79.8913,49.4565}}));
connect(resistor.n,inductor.p) annotation(Line(points = {{-30,60},{-9.78261,60},{-9.78261,59.2391},{-9.78261,59.2391}}));
connect(sineInputVoltage.n,inductor.n) annotation(Line(points = {{-80,30},{-80,29.8913},{10.0543,29.8913},{10.0543,59.7826},{10.0543,59.7826}}));
connect(sineInputVoltage.n,ground1.p) annotation(Line(points = {{-80,30},{-80,9.78261},{-79.61960000000001,9.78261},{-79.61960000000001,9.78261}}));
annotation(Icon(coordinateSystem(extent = {{-100,-100},{100,100}}, preserveAspectRatio = true, initialScale = 0.1, grid = {2,2})), Diagram(coordinateSystem(extent = {{-100,-100},{100,100}}, preserveAspectRatio = true, initialScale = 0.1, grid = {2,2})));
end myRLCnetwork;
X: -90.63 Y: 19.89
Welcome Modeling Plotting

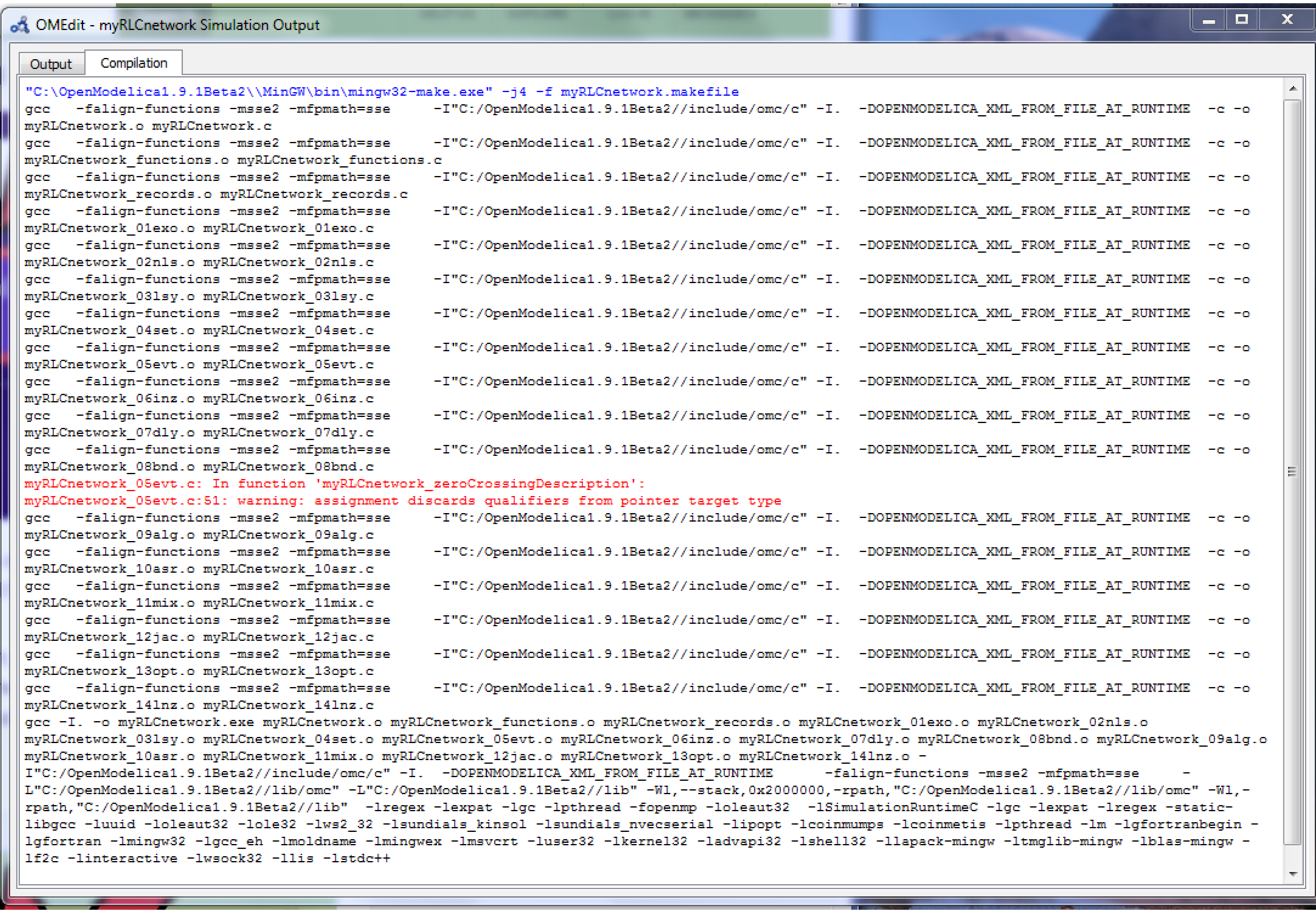

OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
ComplexBlocks
StateGraph
Electrical
Analog
Examples
Basic
Ground
Resistor
Inductor
M_Transformer
Gyrator
myRLCnetwork
Modelica.SIunits.Voltage
Writeable
Model
Modelica Text View
C:/Users/hv/Desktop/scribbles/myRLCr
1 model myRLCnetwork
10 connect(sineInputVoltage.n,ground1.p) annota
{-79.61960000000001,9.78261},{-79.619600000000
OMEdit - Check Model - myRLCnetwork
"Check of myRLCnetwork completed successfully.
Class myRLCnetwork has 24 equation(s) and 24 variable(s).
16 of these are trivial equation(s).
OK

```
class myRLCnetwork
  Real resistor.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real resistor.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  Real resistor.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real resistor.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real resistor.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real resistor.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Boolean resistor.useHeatPort = false "=true, if HeatPort is enabled";
  Real resistor.LossPower(quantity = "Power", unit = "W") "Loss power leaving component via HeatPort";
  Real resistor.T_heatPort(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal =
300.0) "Temperature of HeatPort";
  parameter Real resistor.R(quantity = "Resistance", unit = "Ohm", start = 1.0) = 100.0 "Resistance at temperature T_ref";
  parameter Real resistor.T_ref(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15,
nominal = 300.0) = 300.15 "Reference temperature";
  parameter Real resistor.alpha(quantity = "LinearTemperatureCoefficient", unit = "1/K") = 0.0 "Temperature coefficient of resistance
(R_actual = R*(1 + alpha*(T_heatPort - T_ref))";
  Real resistor.R_actual(quantity = "Resistance", unit = "Ohm") "Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))";
  parameter Real resistor.T(quantity = "ThermodynamicTemperature", unit = "K", displayUnit = "degC", min = 0.0, start = 288.15, nominal =
300.0) = resistor.T_ref "Fixed device temperature if useHeatPort = false";
  Real inductor.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real inductor.i(quantity = "ElectricCurrent", unit = "A", start = 0.0) "Current flowing from pin p to pin n";
  Real inductor.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real inductor.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real inductor.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real inductor.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Real inductor.L(quantity = "Inductance", unit = "H", start = 1.0) = 1e-006 "Inductance";
  Real sineInputVoltage.v(quantity = "ElectricPotential", unit = "V") "Voltage drop between the two pins (= p.v - n.v)";
  Real sineInputVoltage.i(quantity = "ElectricCurrent", unit = "A") "Current flowing from pin p to pin n";
  Real sineInputVoltage.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real sineInputVoltage.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  Real sineInputVoltage.n.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real sineInputVoltage.n.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
  parameter Real sineInputVoltage.offset(quantity = "ElectricPotential", unit = "V") = 0.0 "Voltage offset";
  parameter Real sineInputVoltage.startTime(quantity = "Time", unit = "s") = 0.0 "Time offset";
  parameter Real sineInputVoltage.V(quantity = "ElectricPotential", unit = "V", start = 1.0) = 10.0 "Amplitude of sine wave";
  parameter Real sineInputVoltage.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = 0.0 "Phase of sine wave";
  parameter Real sineInputVoltage.freqHz(quantity = "Frequency", unit = "Hz", start = 1.0) = 50.0 "Frequency of sine wave";
  output Real sineInputVoltage.signalSource.y "Connector of Real output signal";
  parameter Real sineInputVoltage.signalSource.amplitude = sineInputVoltage.V "Amplitude of sine wave";
  parameter Real sineInputVoltage.signalSource.freqHz(quantity = "Frequency", unit = "Hz", start = 1.0) = sineInputVoltage.freqHz
"Frequency of sine wave";
  parameter Real sineInputVoltage.signalSource.phase(quantity = "Angle", unit = "rad", displayUnit = "deg") = sineInputVoltage.phase
"Phase of sine wave";
  parameter Real sineInputVoltage.signalSource.offset = sineInputVoltage.offset "Offset of output signal";
  parameter Real sineInputVoltage.signalSource.startTime(quantity = "Time", unit = "s") = sineInputVoltage.startTime "Output = offset for
time < startTime";
  protected constant Real sineInputVoltage.signalSource.pi = 3.141592653589793;
  Real ground1.p.v(quantity = "ElectricPotential", unit = "V") "Potential at the pin";
  Real ground1.p.i(quantity = "ElectricCurrent", unit = "A") "Current flowing into the pin";
```

```
equation
  assert(1.0 + resistor.alpha * (resistor.T_heatPort - resistor.T_ref) >= 1e-015,"Temperature outside scope of model!");
  resistor.R_actual = resistor.R * (1.0 + resistor.alpha * (resistor.T_heatPort - resistor.T_ref));
  resistor.v = resistor.R_actual * resistor.i;
  resistor.LossPower = resistor.v * resistor.i;
  resistor.v = resistor.p.v - resistor.n.v;
  0.0 = resistor.p.i + resistor.n.i;
  resistor.i = resistor.p.i;
  resistor.T_heatPort = resistor.T;
  inductor.L * der(inductor.i) = inductor.v;
  inductor.v = inductor.p.v - inductor.n.v;
  0.0 = inductor.p.i + inductor.n.i;
  inductor.i = inductor.p.i;
  sineInputVoltage.signalSource.y = sineInputVoltage.signalSource.offset + (if time <
sineInputVoltage.signalSource.startTime then 0.0 else sineInputVoltage.signalSource.amplitude * sin(6.283185307179586 *
sineInputVoltage.signalSource.freqHz * (time - sineInputVoltage.signalSource.startTime) +
sineInputVoltage.signalSource.phase));
  sineInputVoltage.v = sineInputVoltage.signalSource.y;
  sineInputVoltage.v = sineInputVoltage.p.v - sineInputVoltage.n.v;
  0.0 = sineInputVoltage.p.i + sineInputVoltage.n.i;
  sineInputVoltage.i = sineInputVoltage.p.i;
  ground1.p.v = 0.0;
  resistor.p.i + sineInputVoltage.p.i = 0.0;
  resistor.n.i + inductor.p.i = 0.0;
  inductor.n.i + sineInputVoltage.n.i + ground1.p.i = 0.0;
  resistor.p.v = sineInputVoltage.p.v;
  inductor.p.v = resistor.n.v;
  ground1.p.v = inductor.n.v;
  ground1.p.v = sineInputVoltage.n.v;
end myRLCnetwork;
```

Model-Solver Interface
Simulator-Environment Interface



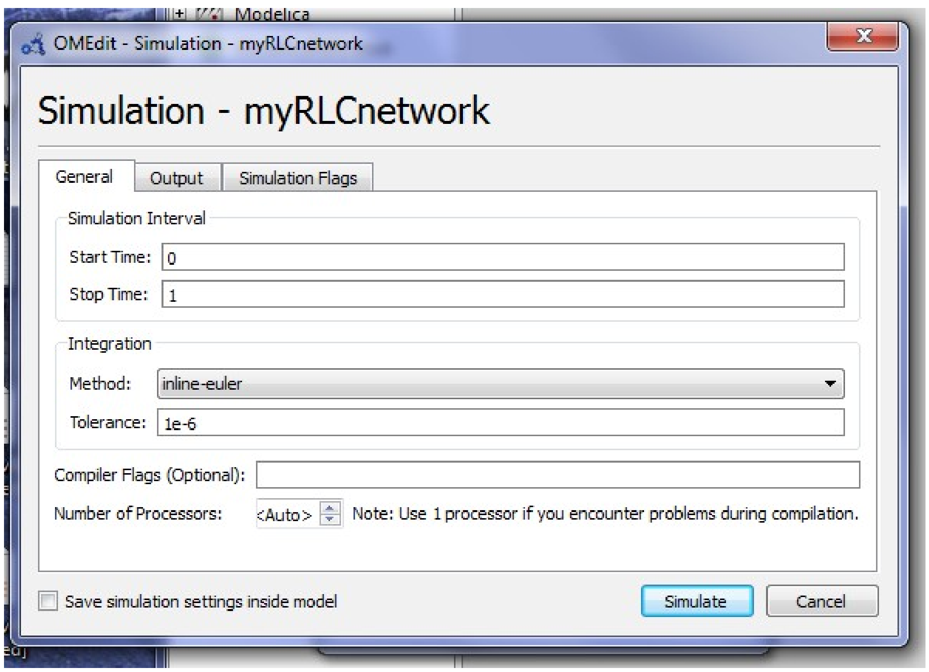


OMEdit - myRLCnetwork Simulation Output

Output | Compilation

```
"C:\OpenModelica1.9.1Beta2\\MinGW\bin\mingw32-make.exe" -j4 -f myRLCnetwork.makefile
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork.o myRLCnetwork.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_functions.o myRLCnetwork_functions.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_records.o myRLCnetwork_records.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_01exo.o myRLCnetwork_01exo.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_02nls.o myRLCnetwork_02nls.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_03lsy.o myRLCnetwork_03lsy.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_04set.o myRLCnetwork_04set.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_05evt.o myRLCnetwork_05evt.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_06inz.o myRLCnetwork_06inz.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_07dly.o myRLCnetwork_07dly.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_08bnd.o myRLCnetwork_08bnd.c
myRLCnetwork_05evt.c: In function 'myRLCnetwork_zeroCrossingDescription':
myRLCnetwork_05evt.c:51: warning: assignment discards qualifiers from pointer target type
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_09alg.o myRLCnetwork_09alg.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_10asr.o myRLCnetwork_10asr.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_11mix.o myRLCnetwork_11mix.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_12jac.o myRLCnetwork_12jac.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_13opt.o myRLCnetwork_13opt.c
gcc   -falign-functions -msse2 -mfpmath=sse     -I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME  -c -o
myRLCnetwork_14lnz.o myRLCnetwork_14lnz.c
gcc -I. -o myRLCnetwork.exe myRLCnetwork.o myRLCnetwork_functions.o myRLCnetwork_records.o myRLCnetwork_01exo.o myRLCnetwork_02nls.o
myRLCnetwork_03lsy.o myRLCnetwork_04set.o myRLCnetwork_05evt.o myRLCnetwork_06inz.o myRLCnetwork_07dly.o myRLCnetwork_08bnd.o myRLCnetwork_09alg.o
myRLCnetwork_10asr.o myRLCnetwork_11mix.o myRLCnetwork_12jac.o myRLCnetwork_13opt.o myRLCnetwork_14lnz.o -
I"C:/OpenModelica1.9.1Beta2//include/omc/c" -I.  -DOPENMODELICA_XML_FROM_FILE_AT_RUNTIME      -falign-functions -msse2 -mfpmath=sse     -
L"C:/OpenModelica1.9.1Beta2//lib/omc" -L"C:/OpenModelica1.9.1Beta2//lib" -Wl,--stack,0x2000000,-rpath,"C:/OpenModelica1.9.1Beta2//lib/omc" -Wl,-
rpath,"C:/OpenModelica1.9.1Beta2//lib"  -lregex -lexpat -lgc -lpthread -fopenmp -loleaut32  -lSimulationRuntimeC -lgc -lexpat -lregex -static-
libgcc -luuid -loleaut32 -lole32 -lws2_32 -lsundials_kinsol -lsundials_nvecserial -lipopt -lcoinmumps -lcoinmetis -lpthread -lm -lgfortranbegin -
lgfortran -lmingw32 -lgcc_eh -lmoldname -lmingwex -lmsvcrt -luser32 -lkernel32 -ladvapi32 -lshell32 -llapack-mingw -ltmglib-mingw -lblas-mingw -
lf2c -linteractive -lwsock32 -llis -lstdc++
```

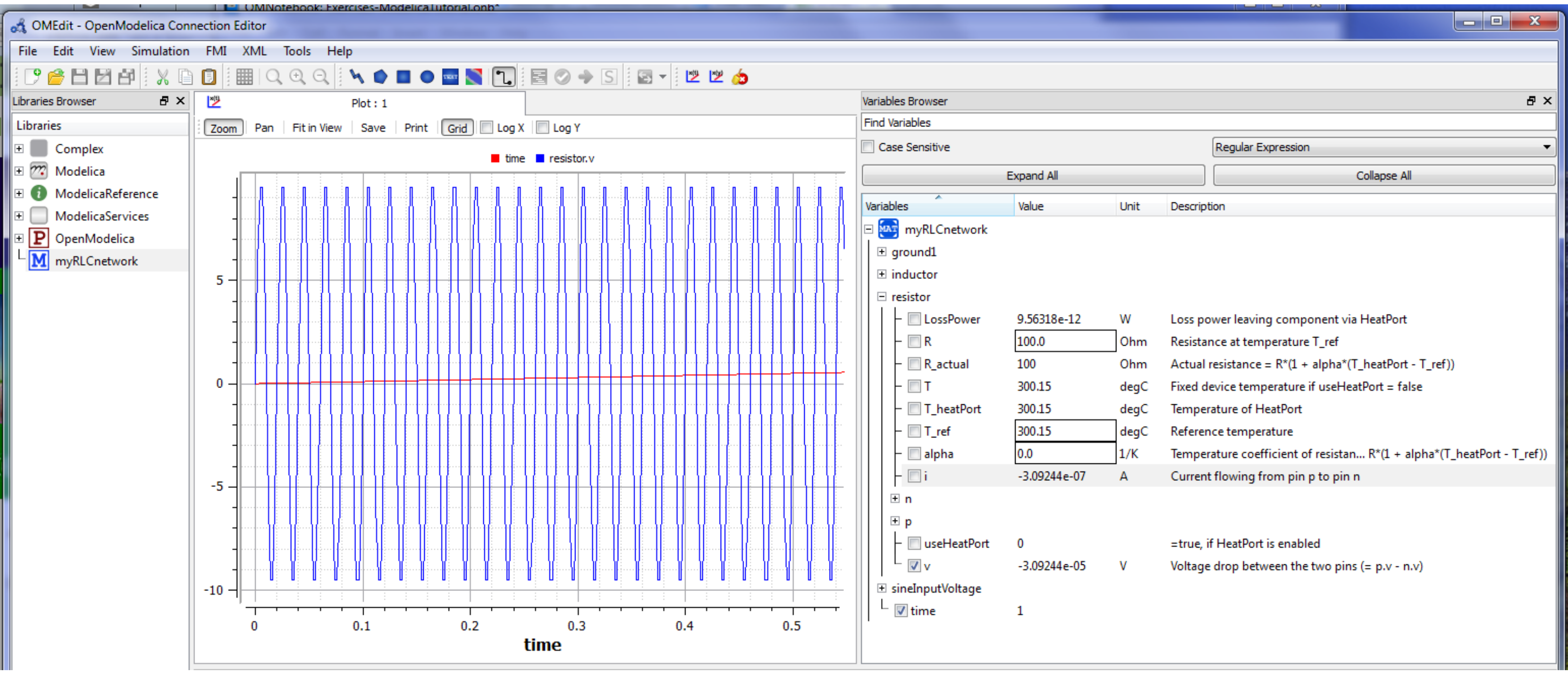
29/09/20...	C File	myRLCnetwork	14 KB
29/09/20...	Application	myRLCnetwork	9,960 KB
29/09/20...	LIBS File	myRLCnetwork.libs	0 KB
29/09/20...	Text Document	myRLCnetwork	0 KB
29/09/20...	MAKEFILE File	myRLCnetwork.makefile	2 KB
29/09/20...	O File	myRLCnetwork.o	17 KB
29/09/20...	C File	myRLCnetwork_01exo	2 KB
29/09/20...	O File	myRLCnetwork_01exo.o	2 KB
29/09/20...	C File	myRLCnetwork_02nls	2 KB
29/09/20...	O File	myRLCnetwork_02nls.o	1 KB
29/09/20...	C File	myRLCnetwork_03lsy	2 KB
29/09/20...	O File	myRLCnetwork_03lsy.o	1 KB
29/09/20...	C File	myRLCnetwork_04set	2 KB
29/09/20...	O File	myRLCnetwork_04set.o	1 KB
29/09/20...	C File	myRLCnetwork_05evt	3 KB
29/09/20...	O File	myRLCnetwork_05evt.o	2 KB
29/09/20...	C File	myRLCnetwork_06inz	7 KB
29/09/20...	O File	myRLCnetwork_06inz.o	5 KB
29/09/20...	C File	myRLCnetwork_07dly	2 KB
29/09/20...	O File	myRLCnetwork_07dly.o	1 KB
29/09/20...	C File	myRLCnetwork_08bnd	7 KB
29/09/20...	O File	myRLCnetwork_08bnd.o	5 KB
29/09/20...	C File	myRLCnetwork_09alg	2 KB
29/09/20...	O File	myRLCnetwork_09alg.o	1 KB
29/09/20...	C File	myRLCnetwork_10asr	2 KB
29/09/20...	O File	myRLCnetwork_10asr.o	1 KB
29/09/20...	C File	myRLCnetwork_11mix	2 KB
29/09/20...	H File	myRLCnetwork_11mix.h	0 KB
29/09/20...	O File	myRLCnetwork_11mix.o	1 KB
29/09/20...	C File	myRLCnetwork_12jac	4 KB
29/09/20...	H File	myRLCnetwork_12jac.h	2 KB


Modelica
OMEdit - Simulation - myRLCnetwork
Simulation - myRLCnetwork
General
Output
Simulation Flags
Simulation Interval
Start Time: 0
Stop Time: 1
Integration
Method: inline-euler
Tolerance: 1e-6
Compiler Flags (Optional):
Number of Processors: <Auto> Note: Use 1 processor if you encounter problems during compilation.
Save simulation settings inside model
Simulate
Cancel

OMEdit - myRLCnetwork Simulation Output

Output | Compilation

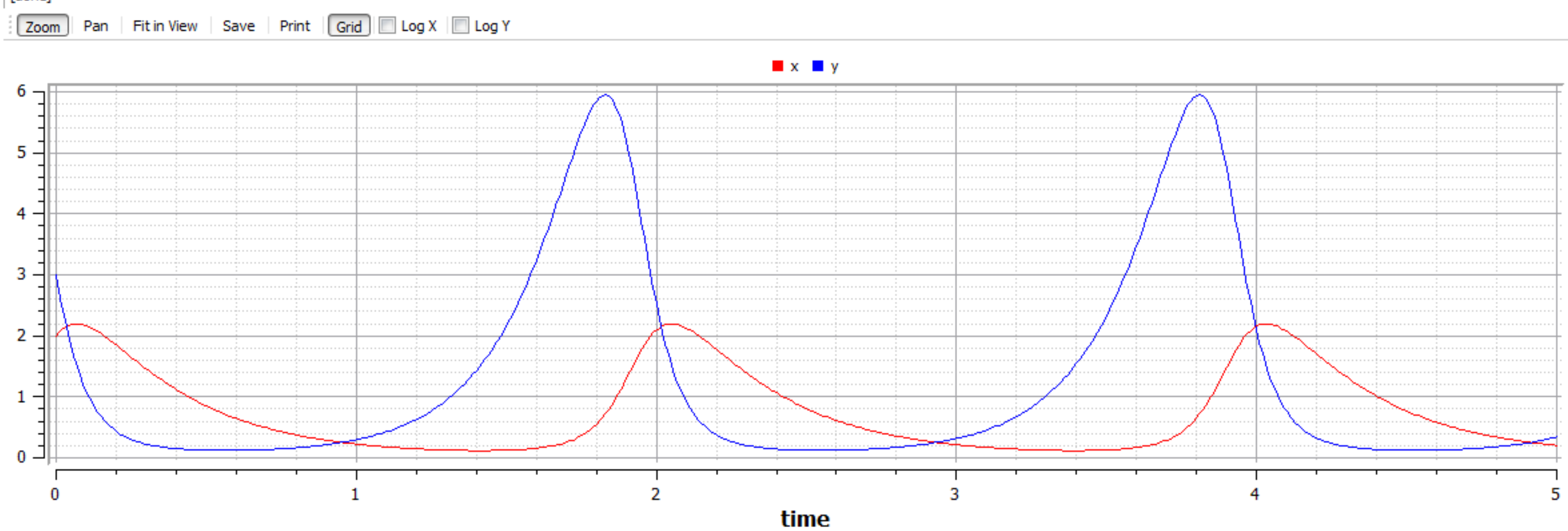
```
C:/Users/hv/AppData/Local/Temp/OpenModelica/OMEdit/myRLCnetwork.exe -port=49502 -logFormat=xml -w -lv=LOG_STATS
LOG_STATS                | info      | ### STATISTICS ###
LOG_STATS                | info      | timer
|                        | |         | |     0.0150538s [ 46.9%] pre-initialization
|                        | |         | | 4.18139e-005s [  0.1%] initialization
|                        | |         | |  2.0907e-005s [  0.1%] steps
|                        | |         | |     0.0157118s [ 49.0%] creating output-file
|                        | |         | |  0.000115558s [  0.4%] event-handling
|                        | |         | |  0.000295738s [  0.9%] overhead
|                        | |         | |  0.000824114s [  2.6%] simulation
|                        | |         | |     0.0320637s [100.0%] total
LOG_STATS                | info      | events
|                        | |         | |      0 state events
|                        | |         | |      0 time events
LOG_STATS                | info      | solver: DASSL
|                        | |         | |  2431 steps taken
|                        | |         | |  3266 calls of functionODE
|                        | |         | |   165 evaluations of jacobian
|                        | |         | |    73 error test failures
|                        | |         | |     0 convergence test failures
LOG_STATS                | info      | ### END STATISTICS ###
```


OMEdit - OpenModelica Connection Editor
File Edit View Simulation FMI XML Tools Help
Libraries Browser
Libraries
Complex
Modelica
ModelicaReference
ModelicaServices
OpenModelica
myRLCnetwork
Plot : 1
Zoom Pan Fit in View Save Print Grid Log X Log Y
time resistor.v
5
0
-5
-10
0 0.1 0.2 0.3 0.4 0.5
time
Variables Browser
Find Variables
Case Sensitive
Regular Expression
Expand All
Collapse All
Variables Value Unit Description
myRLCnetwork
ground1
inductor
resistor
LossPower 9.56318e-12 W Loss power leaving component via HeatPort
R 100.0 Ohm Resistance at temperature T_ref
R_actual 100 Ohm Actual resistance = R*(1 + alpha*(T_heatPort - T_ref))
T 300.15 degC Fixed device temperature if useHeatPort = false
T_heatPort 300.15 degC Temperature of HeatPort
T_ref 300.15 degC Reference temperature
alpha 0.0 1/K Temperature coefficient of resistan... R*(1 + alpha*(T_heatPort - T_ref))
i -3.09244e-07 A Current flowing from pin p to pin n
n
p
useHeatPort 0 =true, if HeatPort is enabled
v -3.09244e-05 V Voltage drop between the two pins (= p.v - n.v)
sineInputVoltage
time 1

```
model mySimpleEqnSet "simple equation set"
  Real x(start=2, fixed=true);
  Real y(start=3, fixed=true);
equation
  der(x) = 2*x*y-3*x;
  der(y) = 5*y-7*x*y;
end mySimpleEqnSet;
```

```
plot({x,y})
```

[done]




```
plotParametric(x,y)
```

[done]