Training Session 1: Introduction to Modelica

M. Scott Greenwood, PhD

Oak Ridge, TN

September 11-12, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Modelica I (60 Minutes)

- Overview (5 min)
- Integrated Development Environments (5 min)
- Workflow (10 min)
- Model Creation: Equation and Parameters (10 min)
- Hands-On Example 1: A Simple ODE (30 min)

OAK RIDGE
National Laboratory

Overview

- 5 minutes of general introduction to the Modelica programming language

The Future of "___" Covers a Large Array of Applications

What will be required of modeling and simulation tools?

Flexible and Adaptable

- Tools must be able to be used for a variety of applications
- Tools must be modifiable for new uses

- Advanced Reactor Technologies
  - HTGRs, LMRs, MSRs
- Integrated Energy Systems
  - Desalination, Hydrogen, Oil-recovery

Rapid Development

- Users need the ability to "fail fast" and mature analysis
- Modeler has control over level of fidelity

- Deployable on a range of machines
  - PCs, clusters
- Advanced languages and features
  - Python, Modelica
  - Acausal, object-oriented

Collaborative

- Domain expertise shareable to leverage skill sets
- Models able to communicate with other tools and frameworks

- Models should be shareable/exportable
  - Open-source or "black-box" capable
- Ability to integrate at different "scales"
  - System, CFD

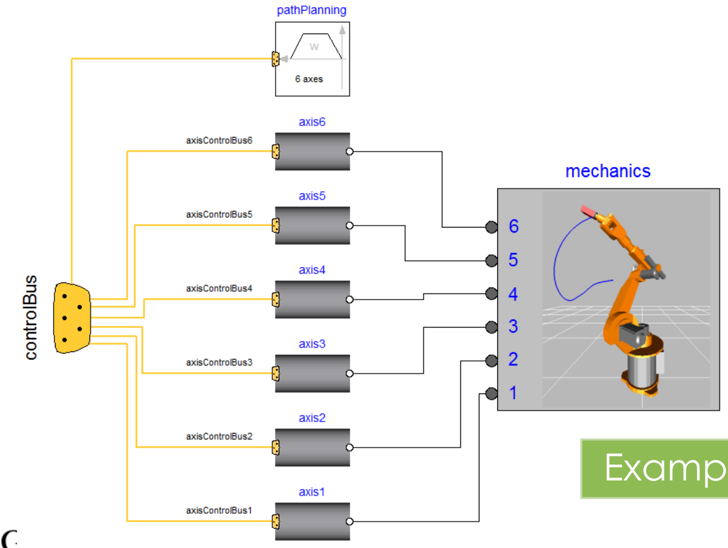
Modelica is …

Equation Based (acausal)

- Order of computations is not decided at modeling time
- Equations do not specify input/outputs
  - $x + y = z^x + yz$
- Solutions direction adapts to data flow

Built for Dynamic Problems

- Time integration handled by solver
  - $der(v) = a + bx(t)$

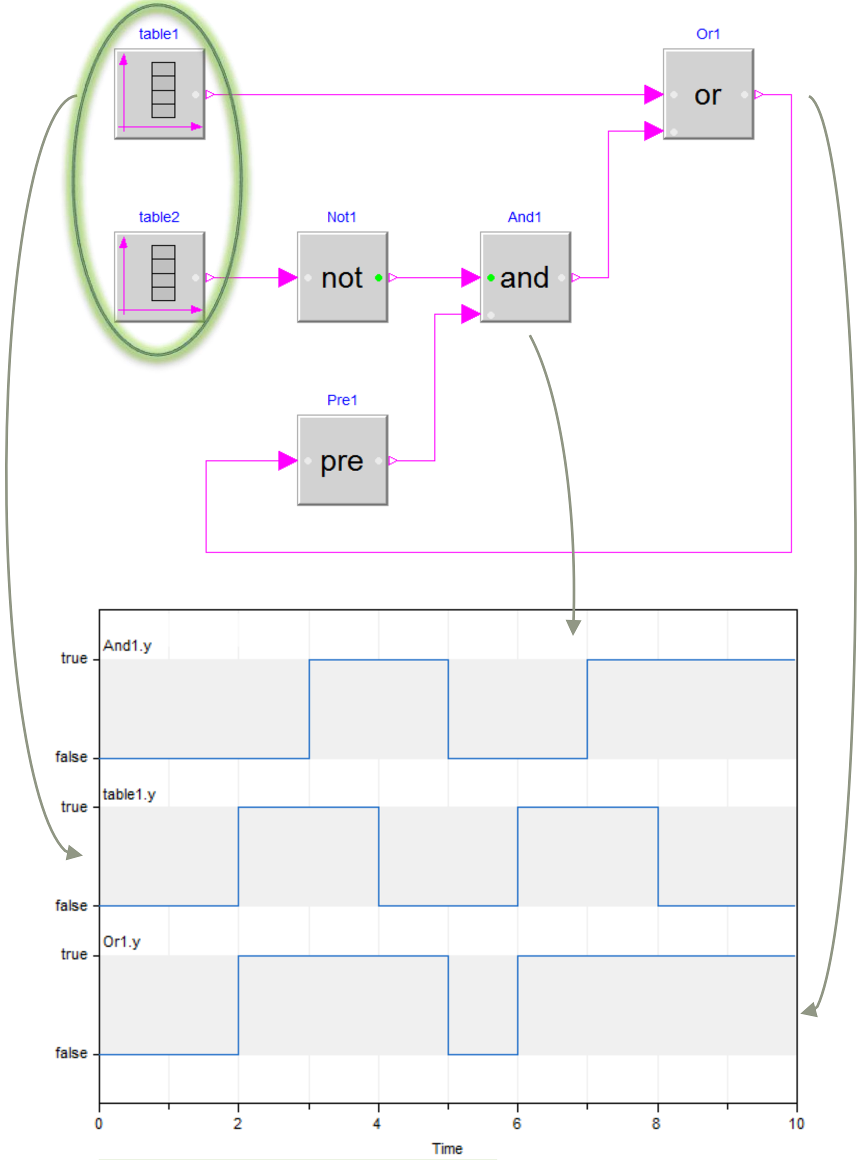


Example from the Modelica Standard Library

Simple Example of Dynamics

- Time dependent aspects of a system
- Concerned with concepts of:
  - States: Attributes described at a point in time
  - Events: Occurrences that trigger a state transition
  - Transitions: A change in the state of an object
  - Actions: Instantaneous operation that results due to an event
  - Activities: Ongoing operations upon the state of an object

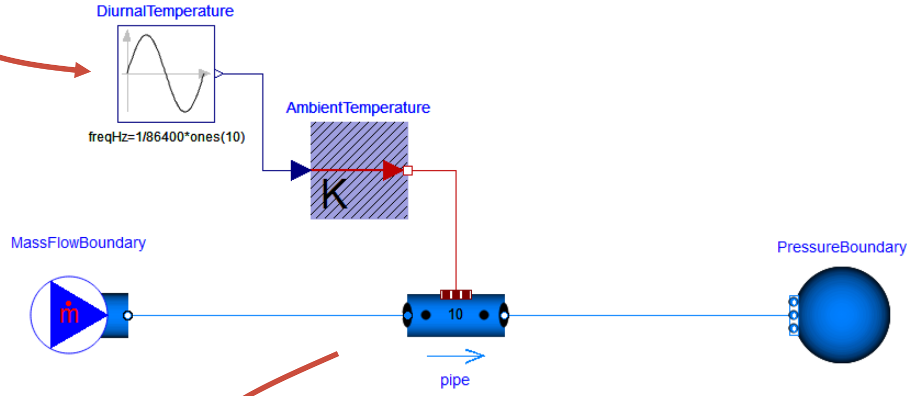
Table values change with time



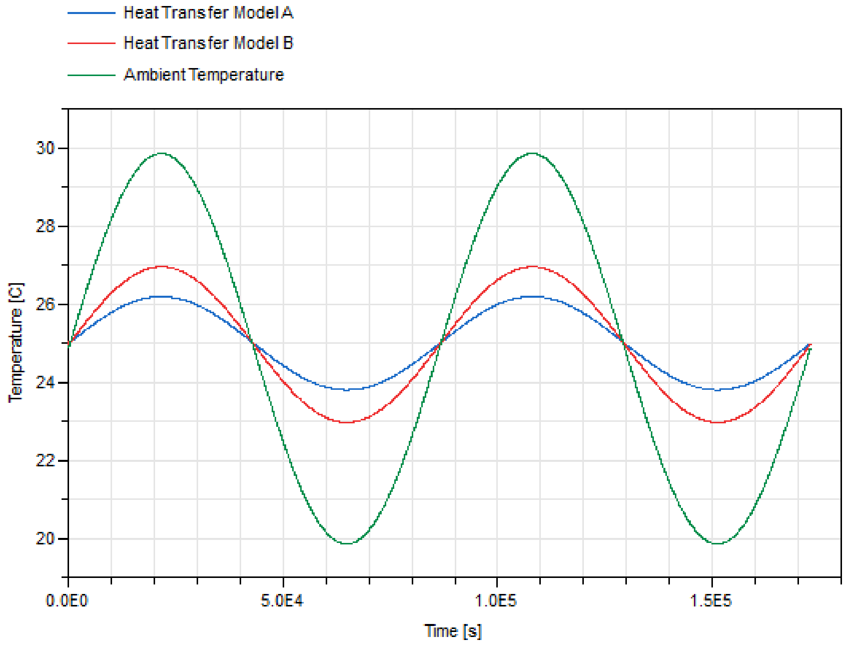
Example of a dynamic problem

“Replaceable” Model Concept

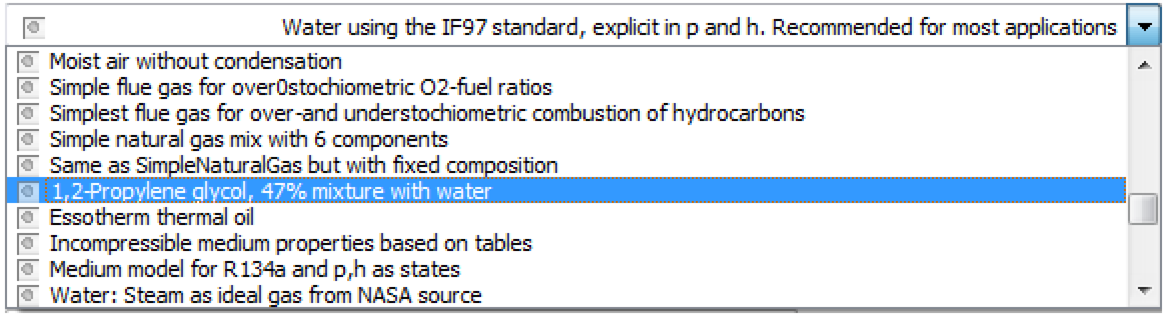
A pipe subjected to cyclic ambient temperature



Simulate



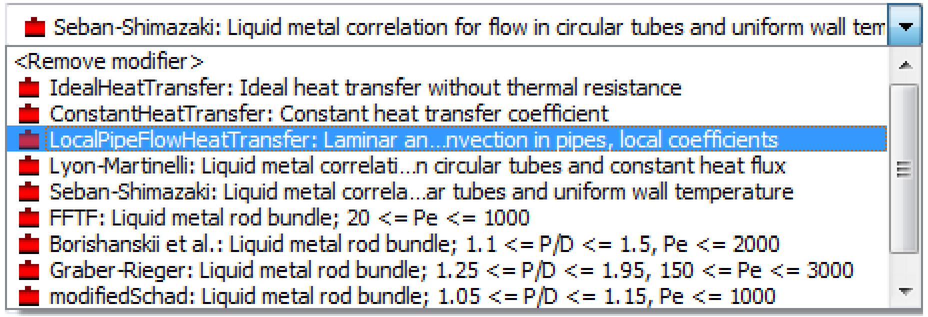
Double click on the component of interest and change the fluid or heat transfer correlation



Water using the IF97 standard, explicit in p and h. Recommended for most applications

- Moist air without condensation
- Simple flue gas for over0stochiometric O2-fuel ratios
- Simplest flue gas for over-and understochiometric combustion of hydrocarbons
- Simple natural gas mix with 6 components
- Same as SimpleNaturalGas but with fixed composition
- 1,2-Propylene glycol, 47% mixture with water
- Essotherm thermal oil
- Incompressible medium properties based on tables
- Medium model for R134a and p,h as states
- Water: Steam as ideal gas from NASA source

Fluid Media



Seban-Shimazaki: Liquid metal correlation for flow in circular tubes and uniform wall ten

- <Remove modifier>
- IdealHeatTransfer: Ideal heat transfer without thermal resistance
- ConstantHeatTransfer: Constant heat transfer coefficient
- LocalPipeFlowHeatTransfer: Laminar an...nvection in pipes, local coefficients
- Lyon-Martinelli: Liquid metal correlati...n circular tubes and constant heat flux
- Seban-Shimazaki: Liquid metal correla...ar tubes and uniform wall temperature
- FFTF: Liquid metal rod bundle; 20 <= Pe <= 1000
- Borishanskii et al.: Liquid metal rod bundle; 1.1 <= P/D <= 1.5, Pe <= 2000
- Graber-Rieger: Liquid metal rod bundle; 1.25 <= P/D <= 1.95, 150 <= Pe <= 3000
- modifiedSchad: Liquid metal rod bundle; 1.05 <= P/D <= 1.15, Pe <= 1000

Heat Transfer Correlation

Integrated Development Environments

- 5 min - icon, diagram, info, code, GUI, parameter GUI, editor, command shell, package browser, result viewer, model creation
- Several modeling environments available
  - https://www.modelica.org/tools
- Dymola (Dassault Systemes)
  - The IDE used in this training

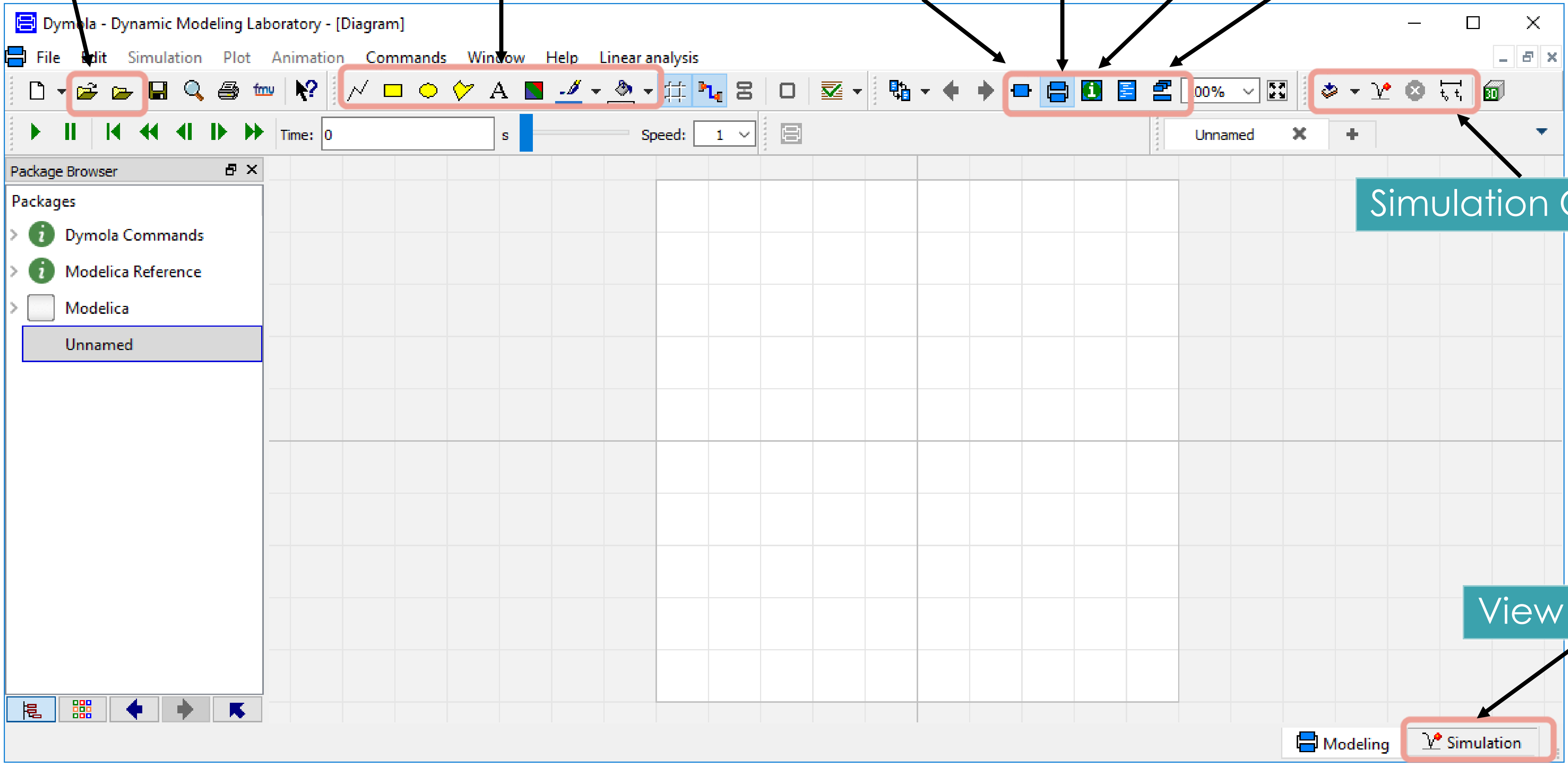
IDE – Navigating the GUI

Access Different Model “Layers”

Open/Load Files

Draw and Style

Icon, Diagram, Info, Text



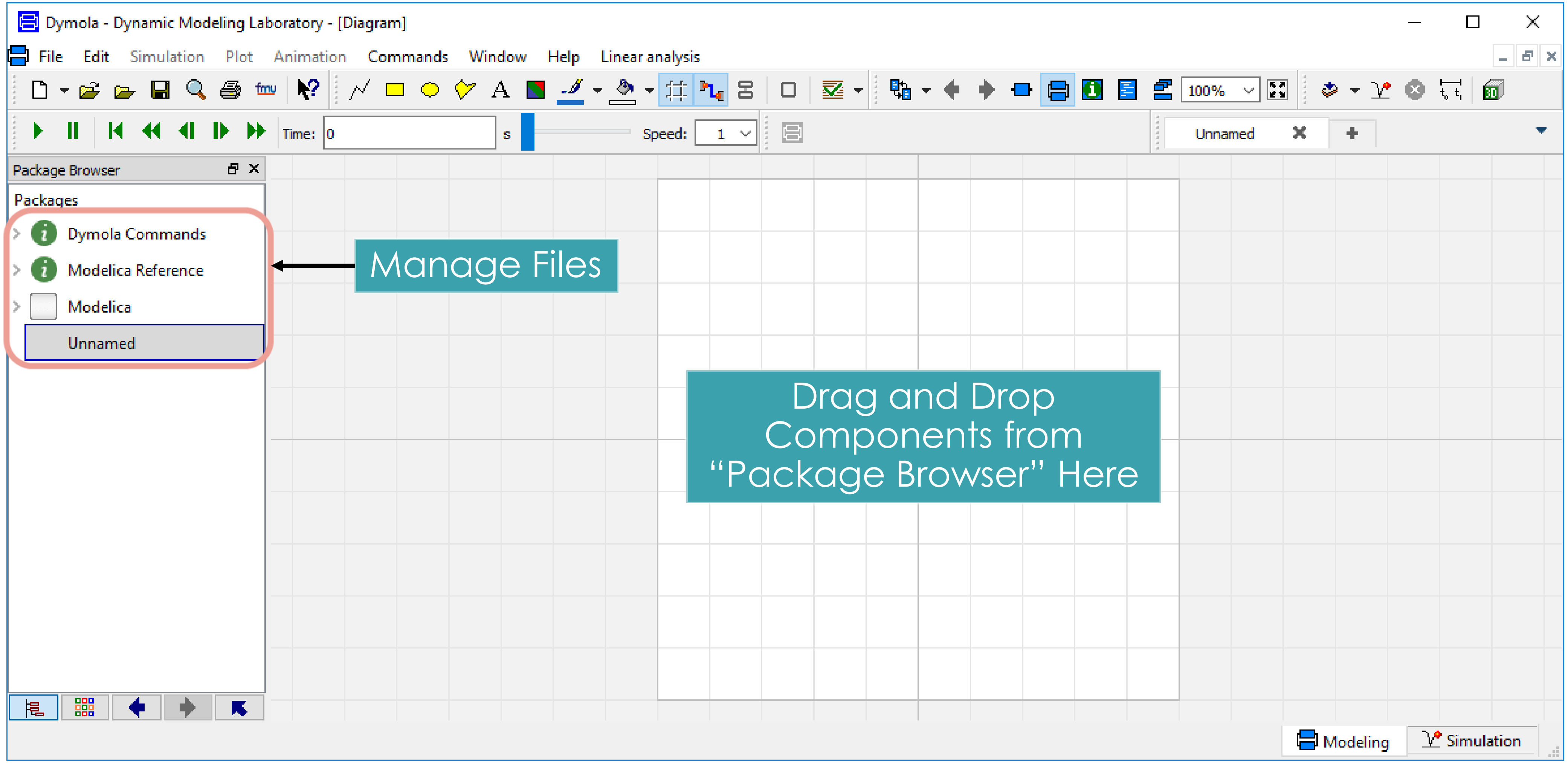
Time: 0 s

Speed: 1

Simulation Control

View Results

IDE – Navigating the GUI



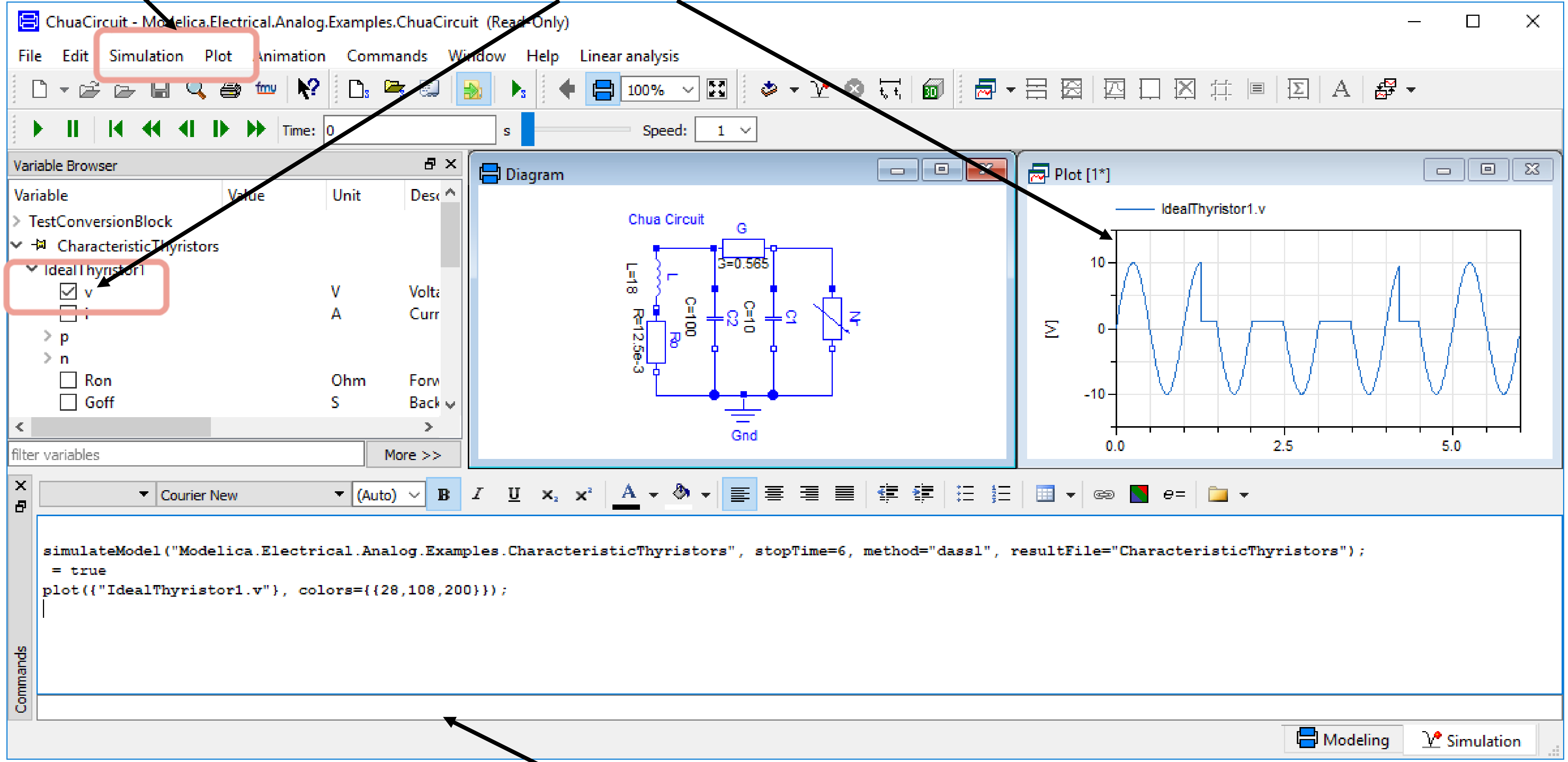
Manage Files

Drag and Drop Components from “Package Browser” Here

IDE – Navigating the GUI

Load Results/Run Scripts

Plot Results

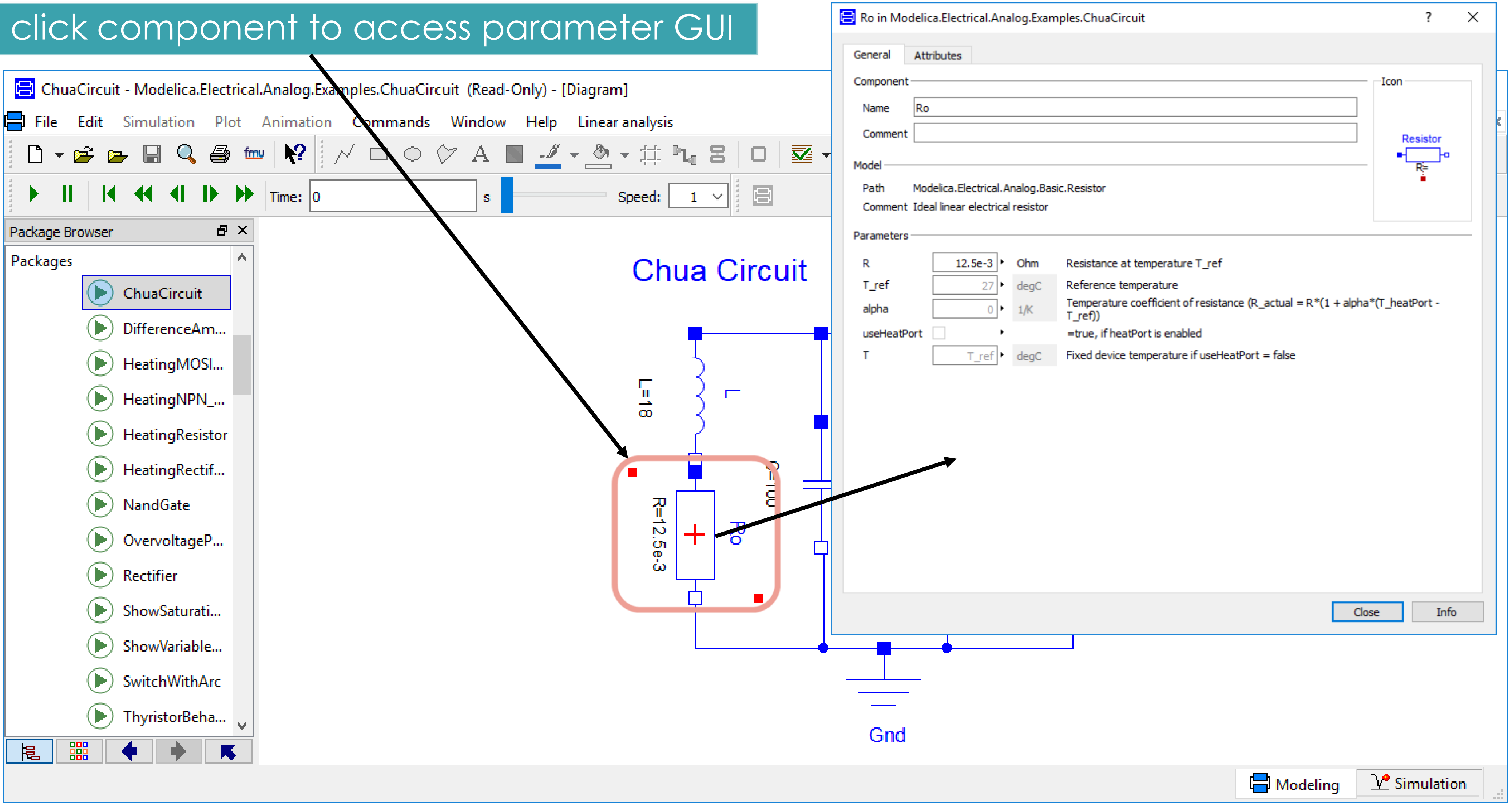


```
simulateModel("Modelica.Electrical.Analog.Examples.CharacteristicThyristors", stopTime=6, method="dassl", resultFile="CharacteristicThyristors");
 = true
plot({"IdealThyristor1.v"}, colors={{28,108,200}});
```

Command Terminal

IDE – Parameter GUI

Double click component to access parameter GUI



Component

Name: Ro

Comment:

Model

Path: Modelica.Electrical.Analog.Basic.Resistor

Comment: Ideal linear electrical resistor

Parameters

R	12.5e-3	Ohm	Resistance at temperature T_ref
T_ref	27	degC	Reference temperature
alpha	0	1/K	Temperature coefficient of resistance (R_actual = R*(1 + alpha*(T_heatPort - T_ref))
useHeatPort	□		=true, if heatPort is enabled
T	T_ref	degC	Fixed device temperature if useHeatPort = false

Workflow

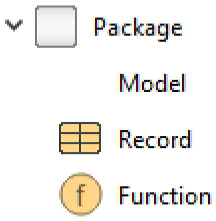
- 10 min - packages, code, translate, compile, sim settings, solve, view results

Workflow – Current Working Directory

- Current working directory
  - Upon simulation, all files are generated in the current working directory
  - Recommend a dedicated working directory for results (i.e., /Documents/Dymola)
- Open vs Load
  - Open: Changes the current working directory to the location of the file opened
  - Load: Adds the file to the path… keeps the current working directory unchanged

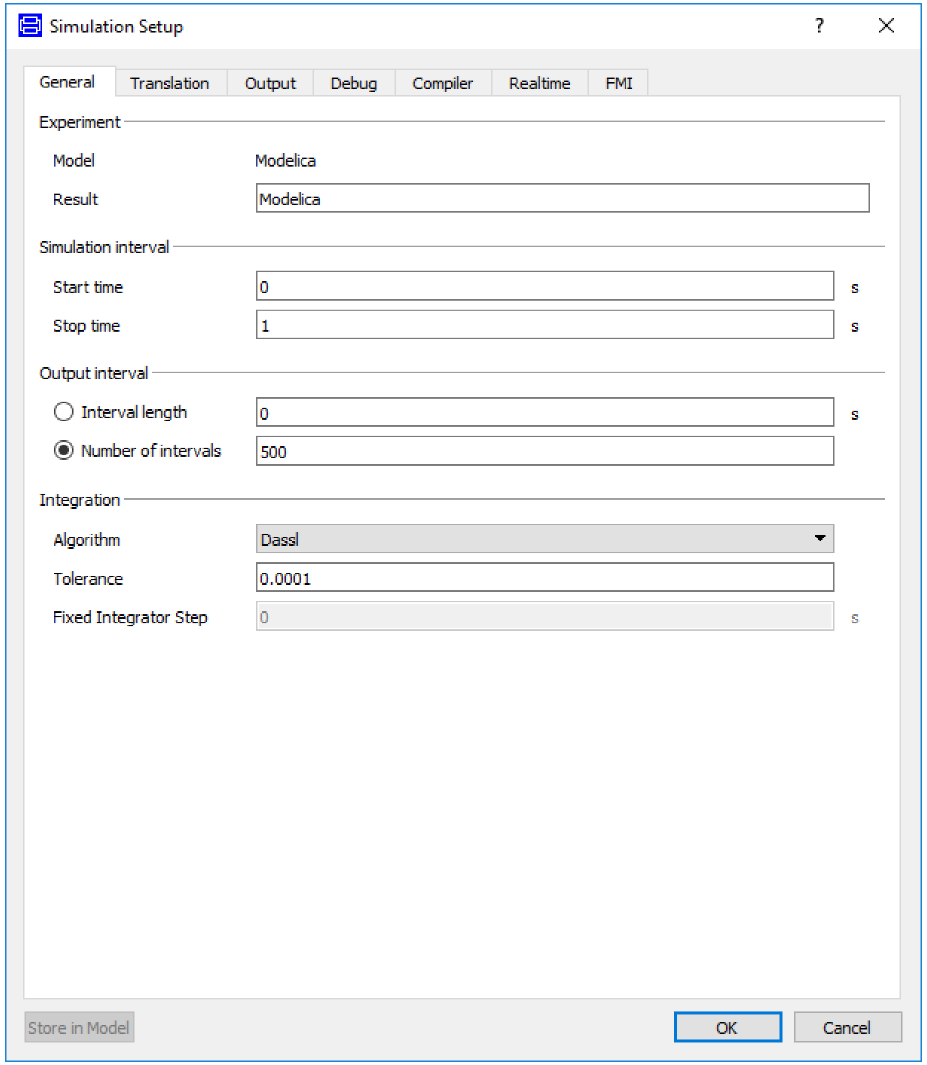
Workflow – Common Classes

- Difference between
  - Package
    - Analogous to folder or directory
  - Model
    - Principle method for creating models
    - Location of “equation” section
  - Function
    - Behaves similar to traditional programming languages (e.g., Matlab)
    - *imperative* – used only for special cases
    - Location of “algorithm” section
  - Record
    - Used to define common types that are reused in various locations
    - e.g., common input parameters to multiple models



Workflow – Simulation Settings

- Variety of options
  - Start/Stop, intervals, solver, global tolerance
  - Translation/Debug flags
  - Change compiler
  - Realtime and Model Export (FMI) options



Simulation Setup

Experiment

Model: Modelica

Result: Modelica

Simulation interval

Start time: 0 s

Stop time: 1 s

Output interval

○ Interval length: 0 s

● Number of intervals: 500

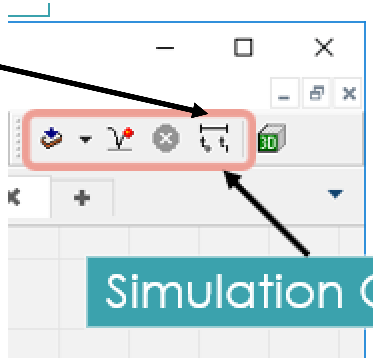
Integration

Algorithm: Dassl

Tolerance: 0.0001

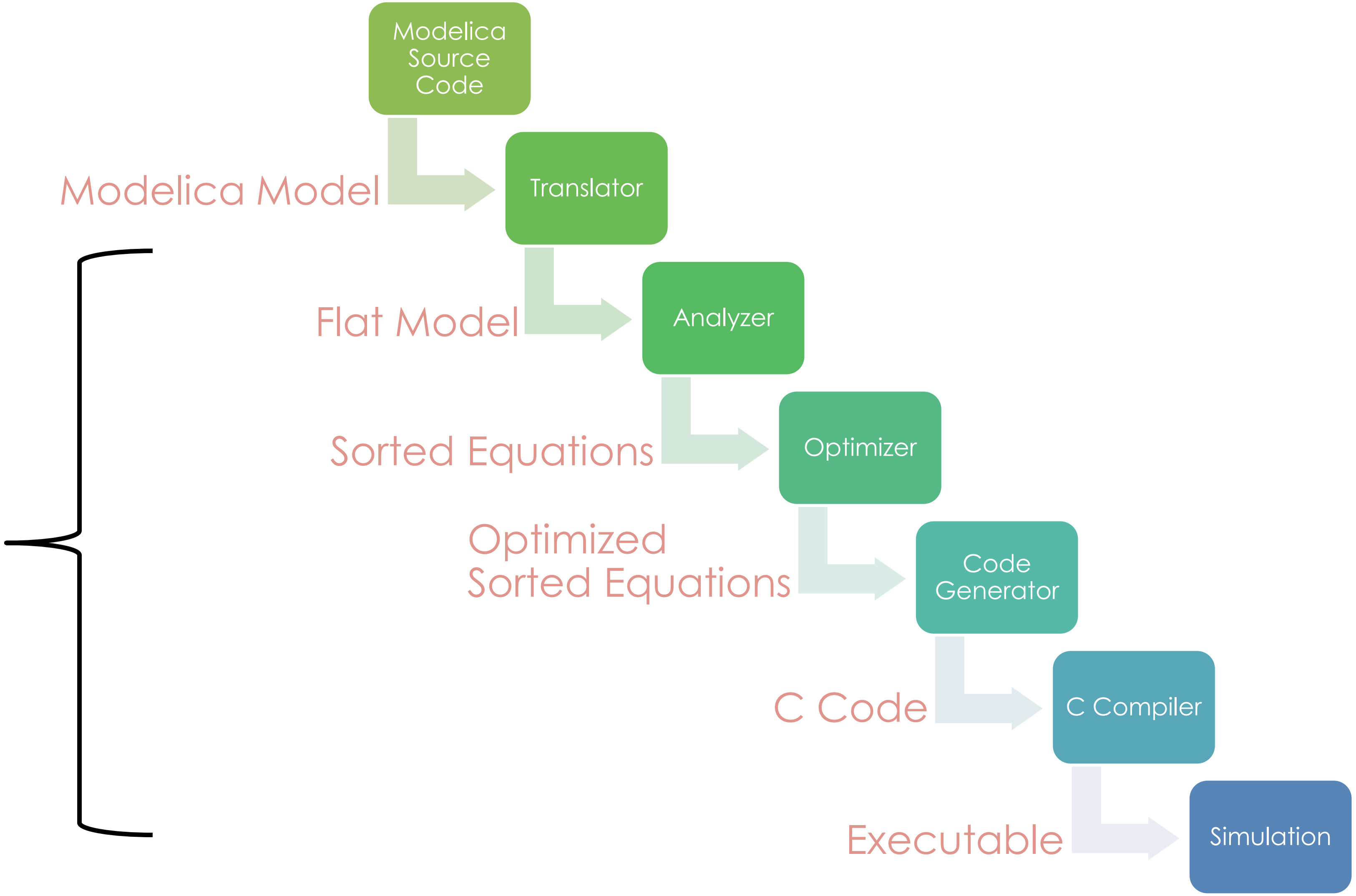
Fixed Integrator Step: 0 s

Click here to control simulation settings



Simulation Control

Workflow – What happens when you push “simulate”?



Much of this is the focus of the IDE

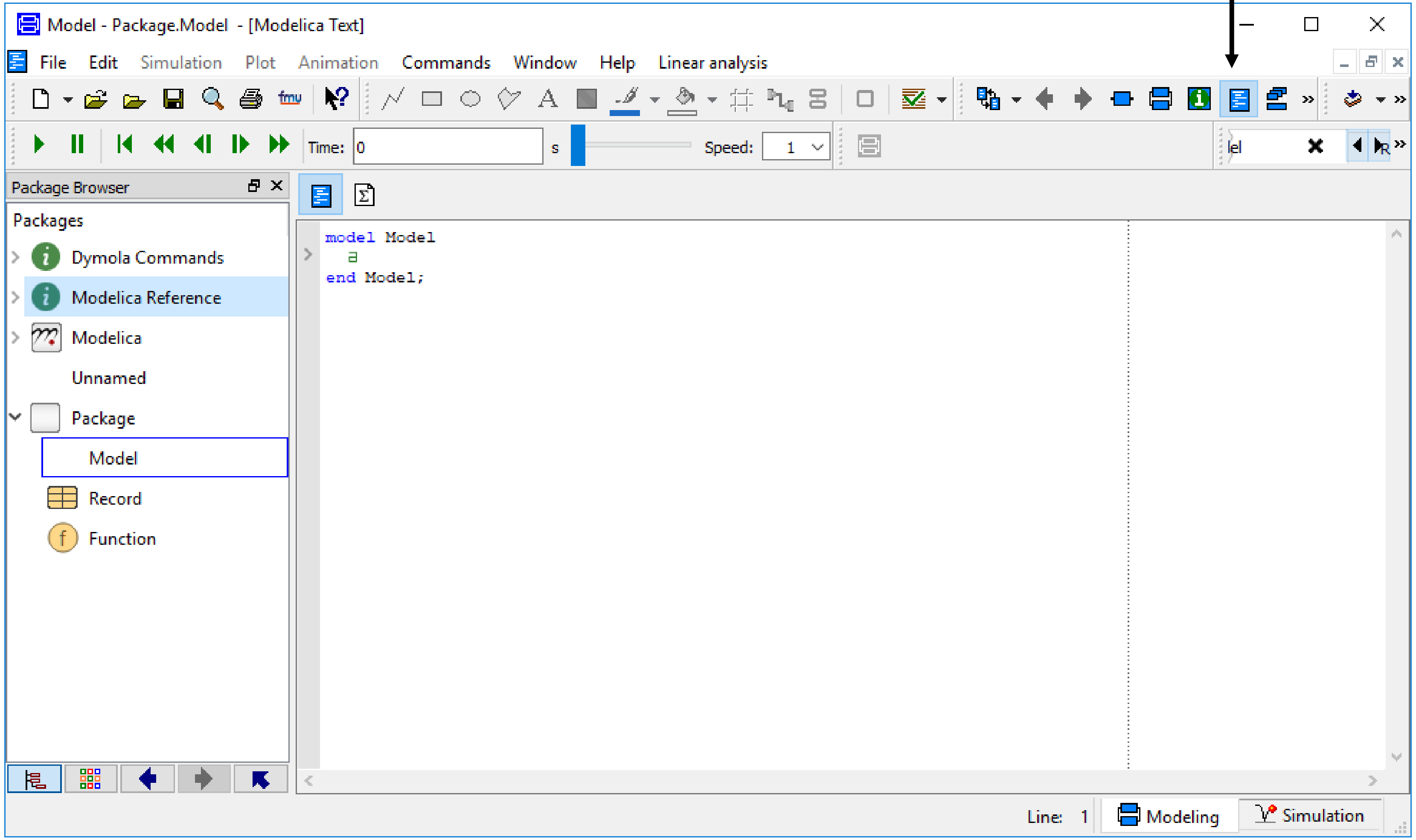
OAK RIDGE National Laboratory

Model Creation: Equation and Parameters

- 10 min - code sections (param, init, eqn), defining parameters, units, equations, initial (start vs init eqn.), der()

Model Creation: Working in the Text Editor

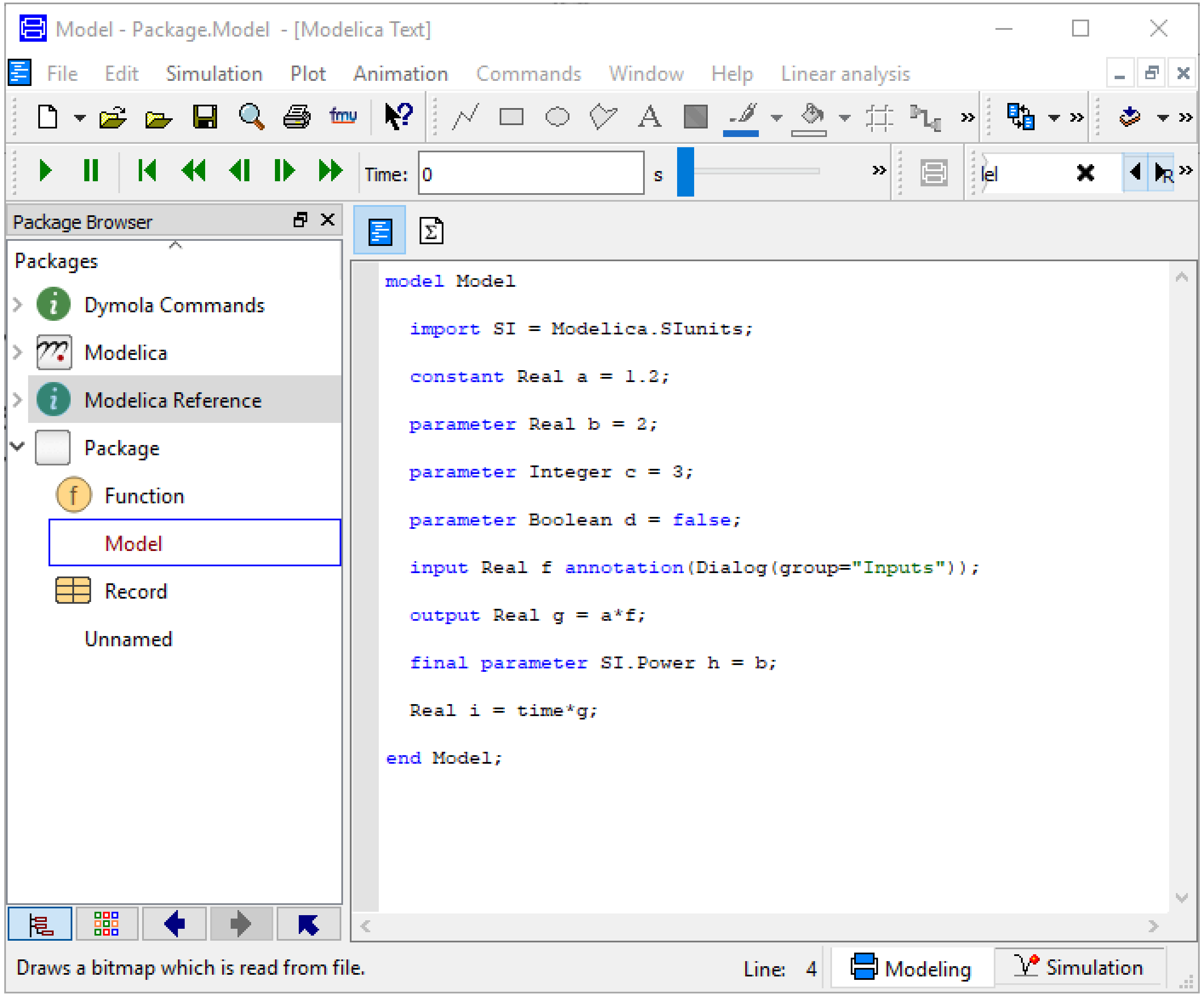
Click here to change to text editor



```
model Model
  
end Model;
```

Model Creation: Variable classes and types

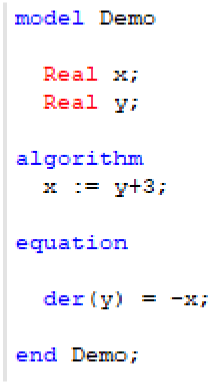
- Classes
  - constant
  - parameter
    - normal and “final”
  - input
  - output
  - unspecified
- Types
  - Real
    - Can define types for units
  - Boolean
  - Integer
- Annotations
  - GUI/translation related



```
model Model
  import SI = Modelica.SIunits;
  constant Real a = 1.2;
  parameter Real b = 2;
  parameter Integer c = 3;
  parameter Boolean d = false;
  input Real f annotation(Dialog(group="Inputs"));
  output Real g = a*f;
  final parameter SI.Power h = b;
  Real i = time*g;
end Model;
```

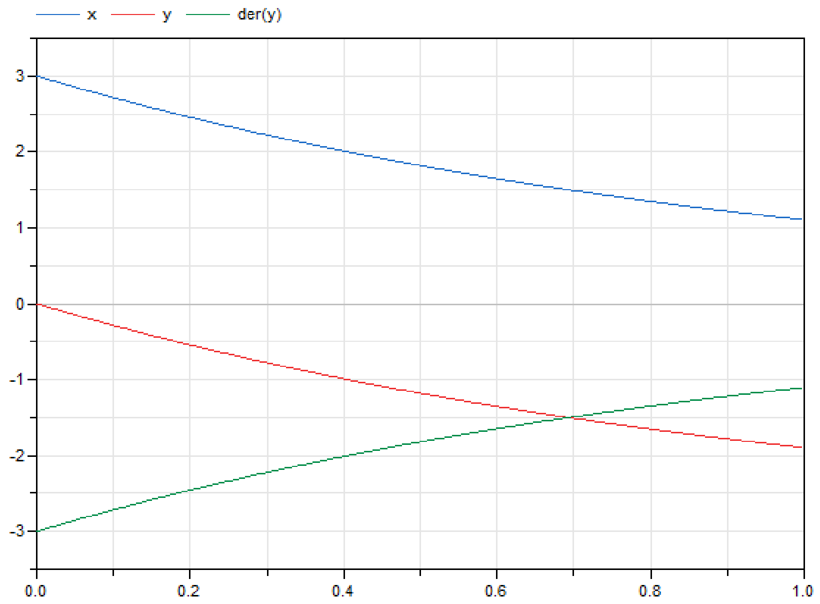
Model Creation: equation and algorithm section

- Equation:
  - Can be used in “model” class
  - Acausal (engineering type equation)
  - Allows translator/solver freedom to manipulate equations
  - Workhorse section for Modelica… default use this over algorithm/functions



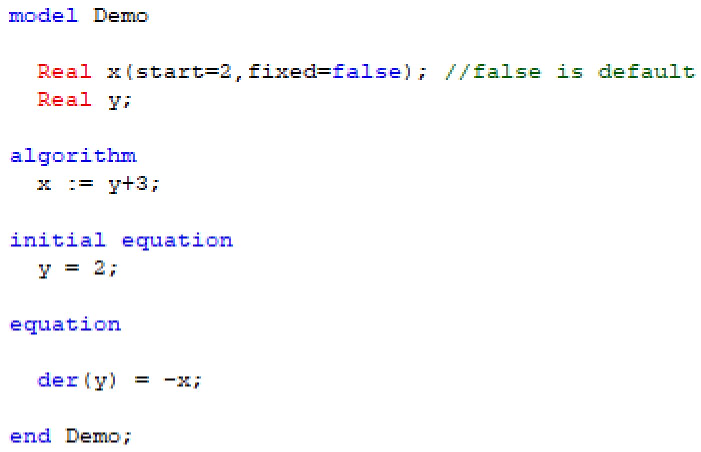
```
model Demo

  Real x;
  Real y;

algorithm
  x := y+3;

equation

  der(y) = -x;

end Demo;
```

- Algorithm
  - Can be used in “model” and “function” class
  - Causal
  - Limits on what the translator can do with equations
  - Can increase solution time/reduce model robustness
  - Sometimes necessary (initialization)

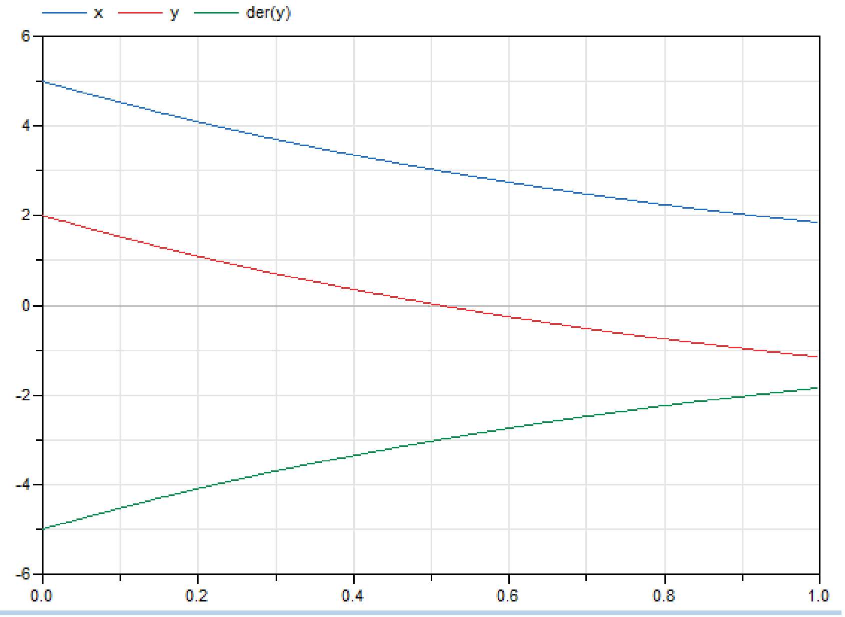


Model Creation: der(), start, and initial equation

- der()
  - Built-in operator for specifying the derivative of the variable
  - “time” is the built-in/associated variable
- start
  - Allows the user to define the initial value
  - Can have a soft (guess) or fixed start value
- initial equation
  - Each variable with a derivative should have a start value
  - Default start value is 0 or der() = 0
  - This sections causes a “fixed” start value
  - Can have der() = 0 be defined



```
model Demo

  Real x(start=2,fixed=false); //false is default
  Real y;

algorithm
  x := y+3;

initial equation
  y = 2;

equation

  der(y) = -x;

end Demo;
```



Hands-On Examples:

Code Based

- 30 min - attendees are given a description of a physical system and provided the mathematical expression and asked to convert to Modelica, practice with troubleshooting, view results, play with simulation settings

Guide to Modeling

- Start simple and gradually build complexity
  - Don’t jump to the end!
- Steps:
  1. Define the equation
  2. Define the variables
  3. Set initial or “start” values
  4. Give values to parameters
  5. Simulate
  6. Extend/adapt as needed

Example 1: A Simple ODE

This example will introduce the user to:

- defining parameters and variables,
- initializing state variables,
- writing ODEs,
- specifying annotations, and
- controlling simulation length

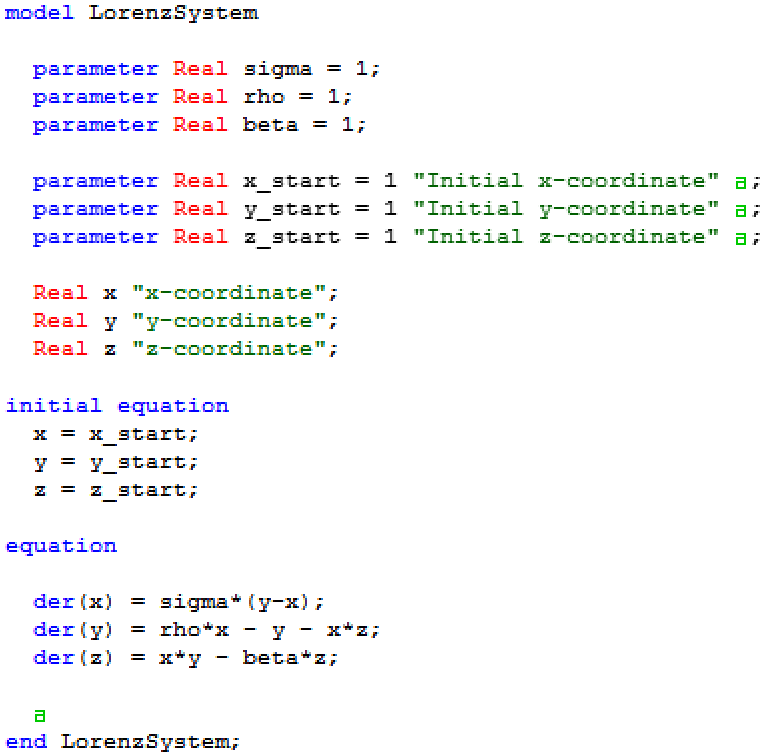
- Try coding a Lorenz System:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = \rho x - y - xz$$

$$\frac{dz}{dt} = xy - \beta z$$

Mathematical Model

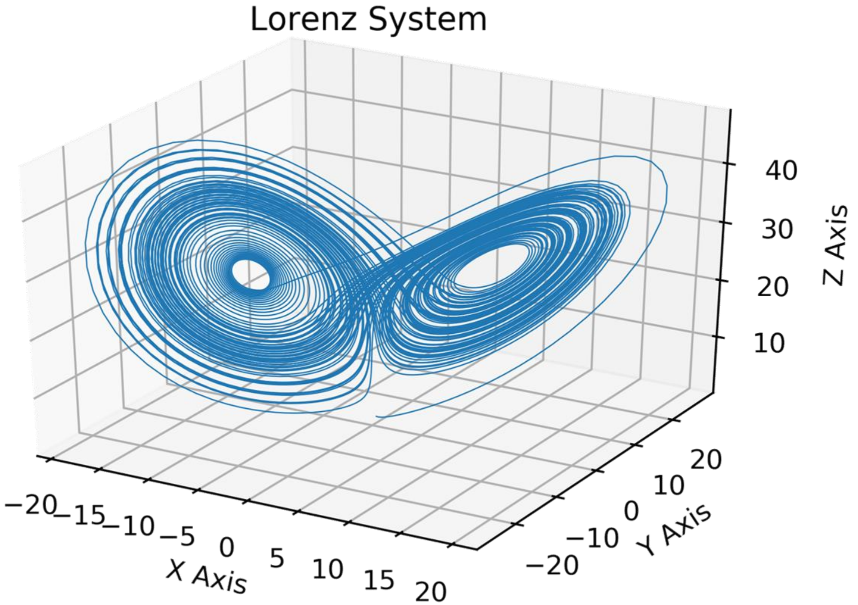


```
model LorenzSystem

  parameter Real sigma = 1;
  parameter Real rho = 1;
  parameter Real beta = 1;

  parameter Real x_start = 1 "Initial x-coordinate" a;
  parameter Real y_start = 1 "Initial y-coordinate" a;
  parameter Real z_start = 1 "Initial z-coordinate" a;

  Real x "x-coordinate";
  Real y "y-coordinate";
  Real z "z-coordinate";

initial equation
  x = x_start;
  y = y_start;
  z = z_start;

equation

  der(x) = sigma*(y-x);
  der(y) = rho*x - y - x*z;
  der(z) = x*y - beta*z;

  a
end LorenzSystem;
```

Implementation in Modelica

$$\sigma = 10; \rho = 28; \beta = 8/3$$

$$x(0) = 0; y(0) = 1; z(0) = 1.05$$



Solution
(plotted with Python)

https://en.wikipedia.org/wiki/Lorenz_system

Example 2: Time to Cool Off

This example introduce the user to:

- specifying units (i.e., importing),
- using custom function (i.e., for unit conversions),
- providing comments to parameters and variables,
- booleans,
- adding components to models, and
- using "if" logic

- Try coding a Newton Cooling problem:

$$m * c_p * \frac{dT}{dt} = h * A * (T_{ambient} - T)$$

$m = mass$

$c_p = specific\ heat\ capacity$

$T = temperature\ of\ lumped\ mass$

$T_{ambient} = ambient\ temperature$

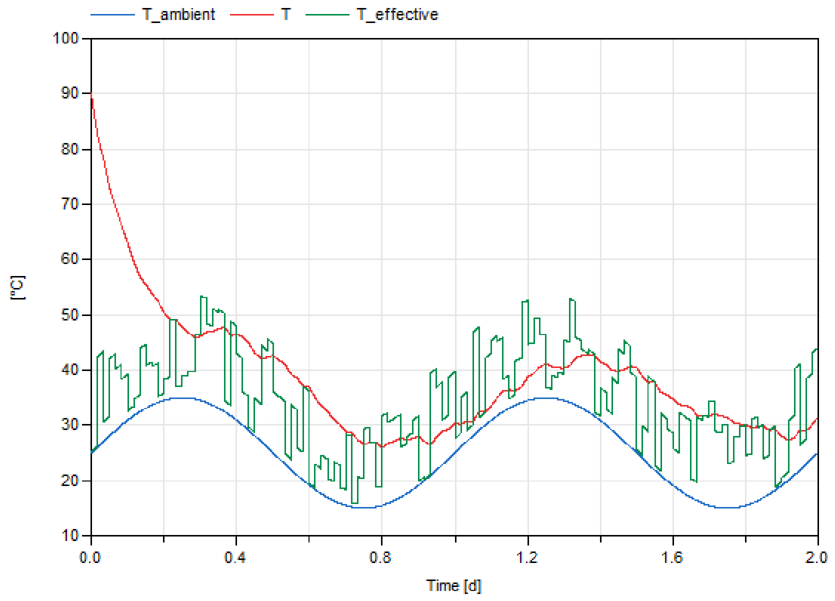
$t = time$

$h = heat\ transfer\ coefficient$

$A = surface\ area$

$h = heat\ transfer\ coefficient$

<u>Challenge Problem:</u>
Try adding a time dependent shape and noise to the ambient temperature?



Example 3: Let There Be decay!

This example will introduce the user to:

- arrays/matrices,
- "for loops", and
- alternative means to specify units

- Try coding Bateman Equations:

$$\frac{dN_1}{dt} = \phi\sigma_1 - \lambda N_1$$

$$\frac{dN_{i..n}}{dt} = \phi\sigma_i - \lambda N_i + \lambda_{i-1}N_{i-1} \quad for\ i = 2 \ldots n$$

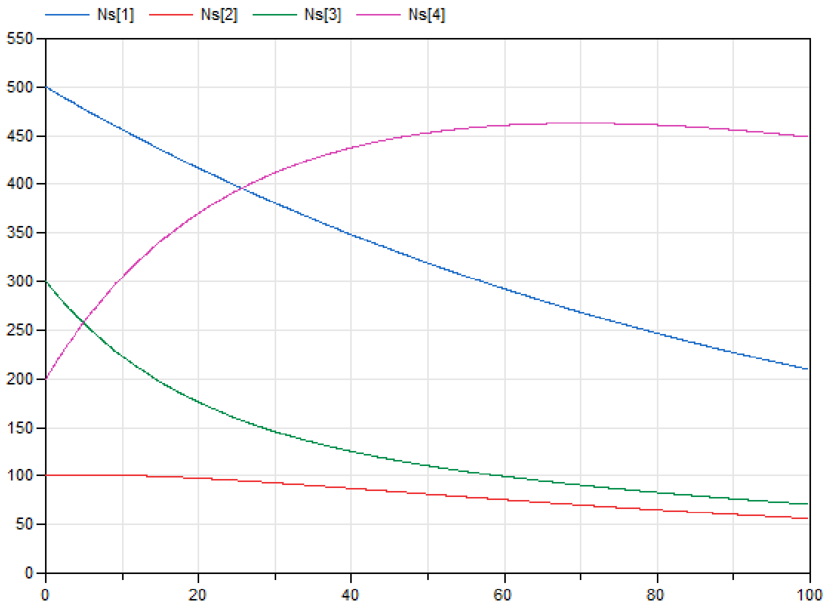
$N_i = atoms\ of\ isotope\ i$

$\phi = flux$

$\sigma = cross\ section$

$\lambda = decay\ constant$

$t = time$



This example will introduce the user to:

- Input connector

Example 4: Aliens vs. Predator!

- Try coding a Lotka-Volterra System:

$$\frac{dx}{dt} = \alpha x - \beta xy + u$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$

$x = Prey\ (Aliens)$

$y = Predator$

$\alpha = prey\ population\ growth$

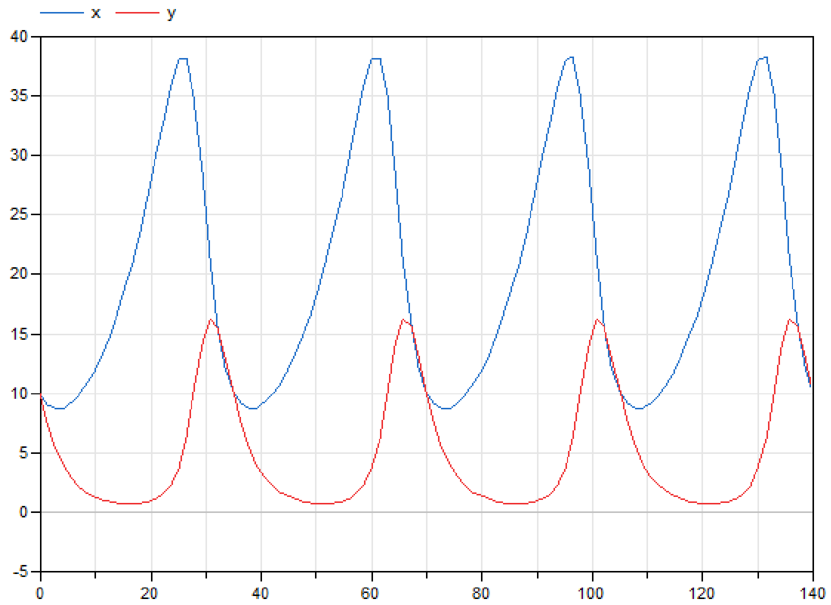
$\beta = predation$

$\delta = predator\ growth$

$\gamma = predator\ death$

$u = input\ connector$

Modelica.Blocks.Interfaces.RealInput



OAK RIDGE National Laboratory

Hands-On Examples:

Component Based

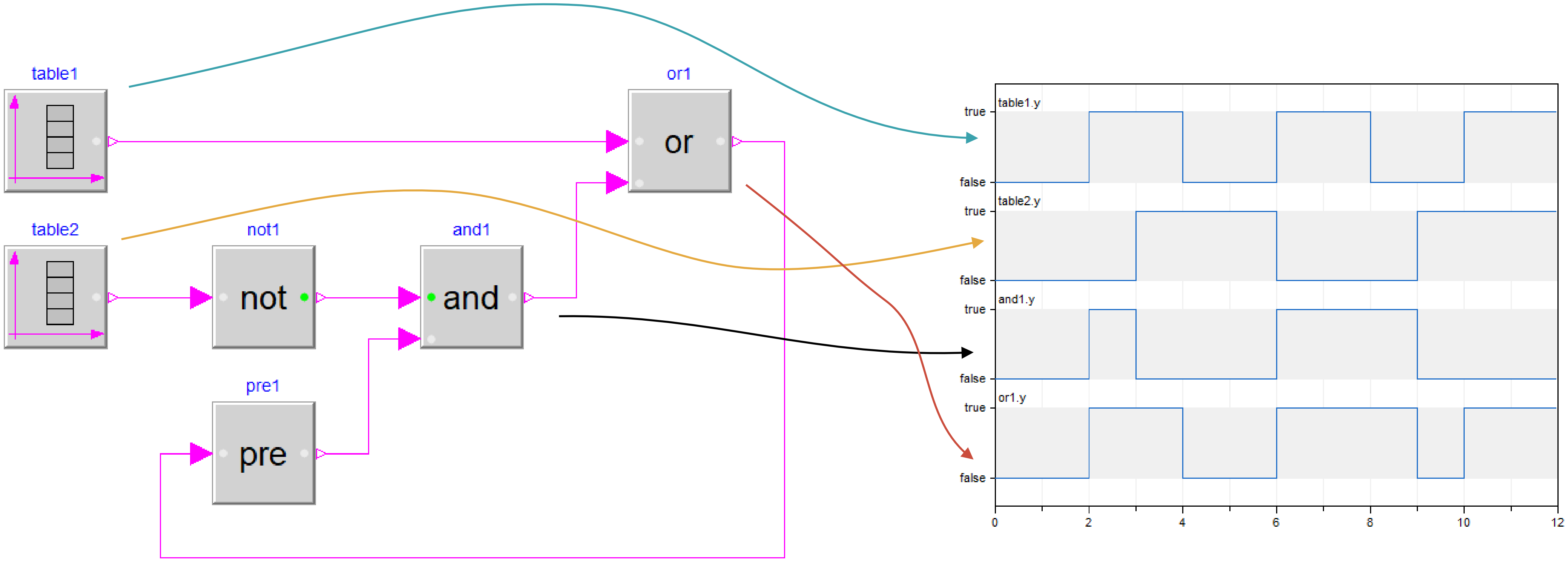
- 10 min – attendees get familiar with navigating “packages” to select components and connect to solve simple problems

Example 1: True or False?

This example will introduce the user to:

- Drag and drop components
- Modelica.Blocks.*

- Try creating a simple logic tree:

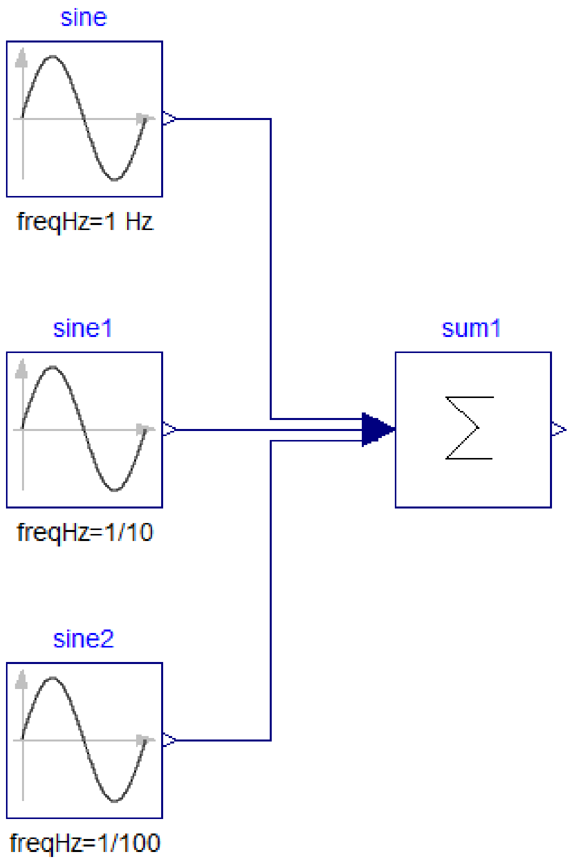


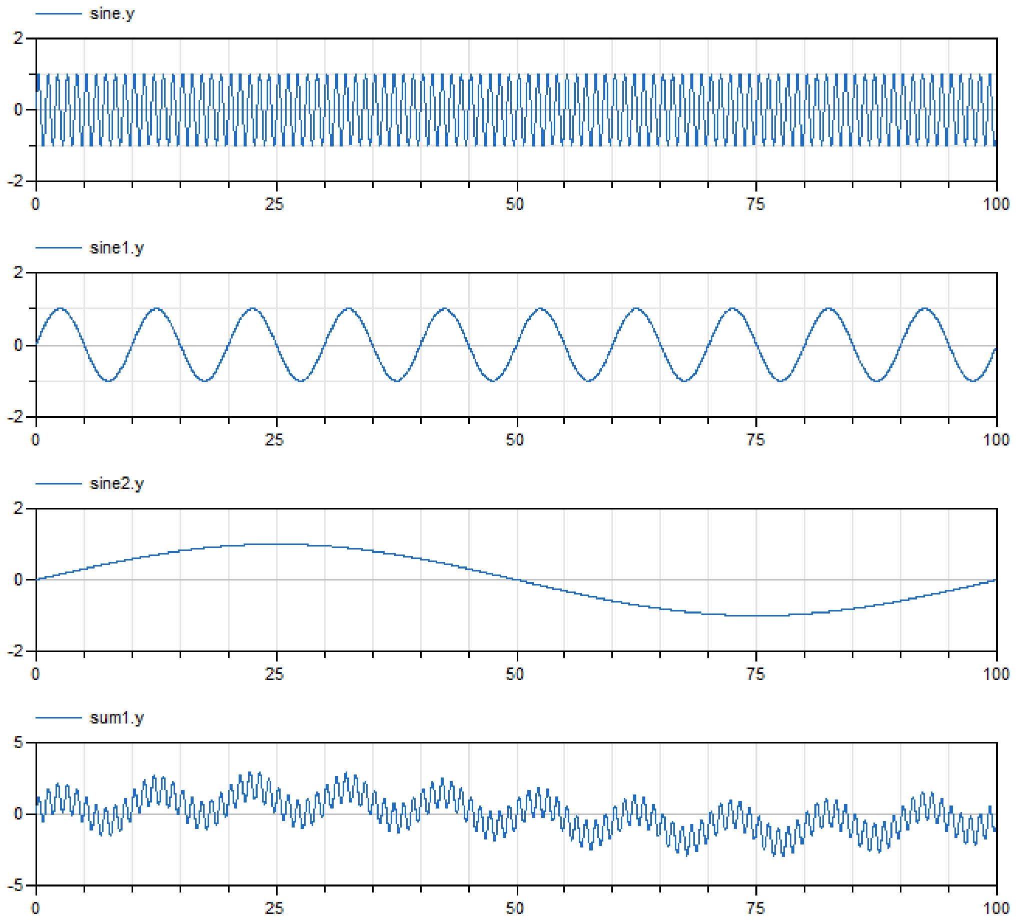
Example 2: Give me a sine!

This example will introduce the user to:

- Drag and drop components
- Modelica.Blocks.*

- Try creating a sum of sines:



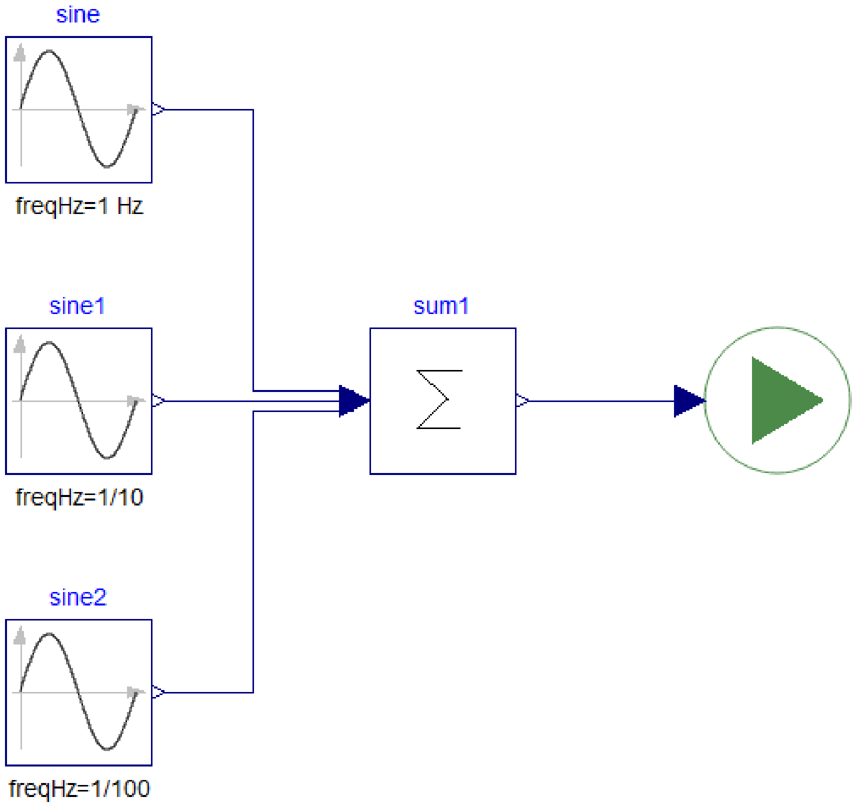


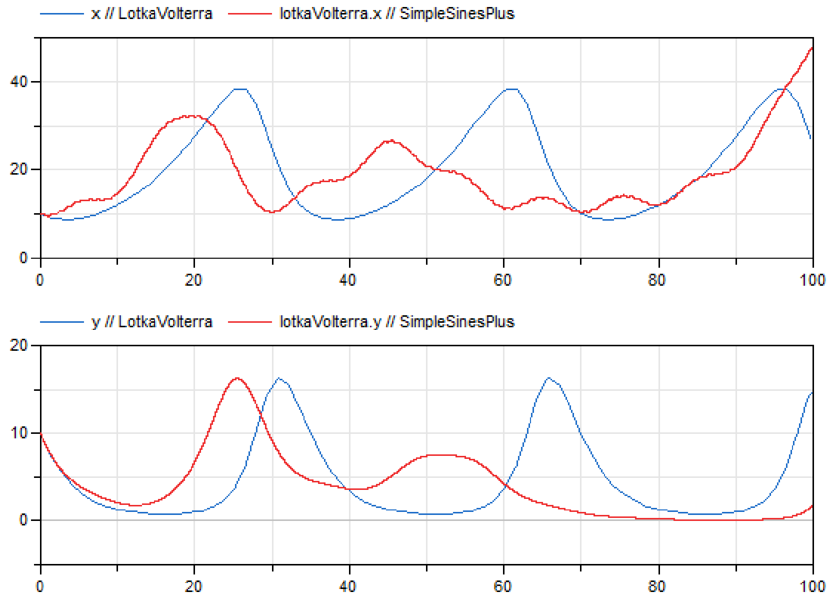
This example will introduce the user to:

- Drag and drop components
- Modelica.Blocks.*
- Input connectors

Example 3: The Circle of Life

- Use the sum of slides as input to the Lotka-Volterra example:





Oak Ridge National Laboratory

Thank you.

Scott Greenwood

greenwoodms@ornl.gov