

## Introduction to the Modelica language and the simulating environment Dymola, part I

Staffan Haugwitz

Dept. of Automatic Control  
Lund University

Based on material from Modelon

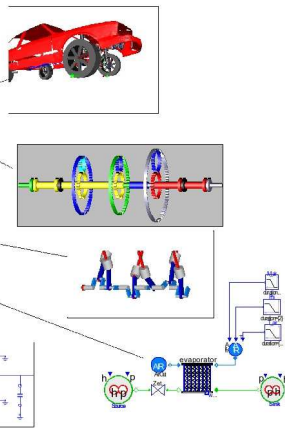
- Introduction, relation between Modelica and Dymola, applications, overview of Dymola
- Introduction to the Modelica language, syntax and constructs
- Traps and pitfalls (high index problem, stiff systems)

## Modelica in Vehicle Modeling

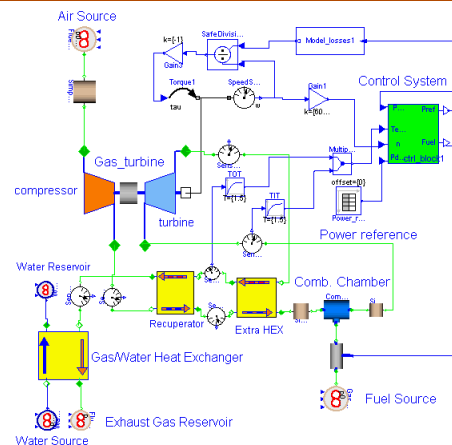
### Example:

Modeling of vehicle components with Modelica and Dymola

- Vehicle dynamics (3D mechanics)
- Power train (1D mechanics)
- Hydraulics
- Combustion
- Air condition
- Electrical/electronic systems
- Control (Input/output blocks)



## Modelica in Microturbine Systems



## Modelica Features

- Declarative equations  
Write the model equations as they are derived from physics, instead of deriving the right hand side.

$$C \frac{dP_2}{dt} = F_4 - F_5; \quad \frac{dP_2}{dt} = \frac{1}{C} \left( \frac{F_1 c_p (T_2 - T_1)}{\lambda} - F_5 \right);$$

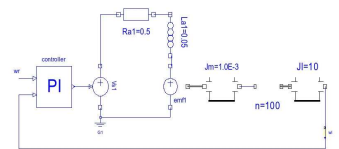
$$\lambda F_4 = F_1 c_p (T_2 - T_1);$$

- Object-oriented modeling language (classes, instances, inheritance...)
- Component models  
Easy to build libraries of reusable code. Modelica standard library contains 740 different models, ready to use or modify.

## Modelica vs Simulink

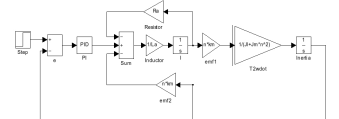
### Acausal modeling

Declarative languages just require the developer to define the problem at a higher level and leaves the solution to the simulation tool

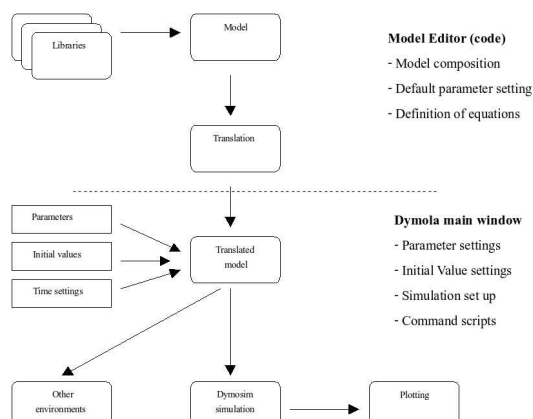


### Block-oriented modeling

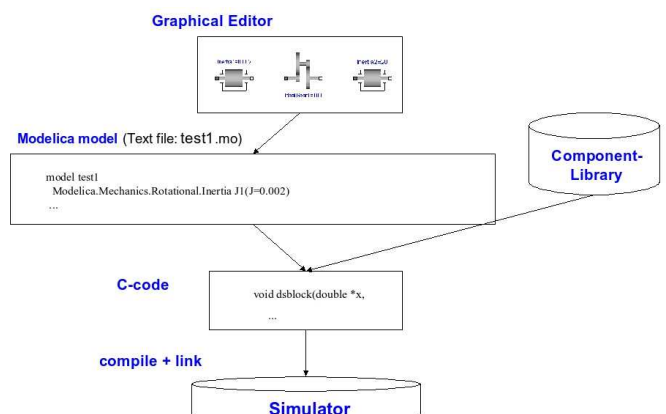
Procedural programmes require the developer to define the order that calculations are to be done in



## Modelica vs Dymola



## Dymola Overview 1



The diagram illustrates the Dymola Program Architecture, centered around the **Dymola Program** block, which is divided into four main functional areas: **Editor**, **Symbolic Kernel**, **Experimentation**, and **Plot and Animation** / **Reporting**.

**Modeling Phase (Green Background):**

- Inputs:** Experimental Data and Model Parameters feed into the Editor.
- Components:** User Models and Modelica Libraries are linked to the Editor via bidirectional arrows.
- Outputs:** The Editor outputs CAD (DXF, STL, topology, properties) and External Graphics (vector, bitmap).

**Simulation Phase (Purple Background):**

- Inputs:** Simulation results feed into the Symbolic Kernel via a bidirectional arrow.
- Outputs:** The Symbolic Kernel outputs to the Experimentation area.

**Analysis and Reporting Phase (Grey Background):**

- Inputs:** Scripting (Modelica, C Functions, LAPACK) feeds into the Symbolic Kernel via a bidirectional arrow.
- Experimentation:** This area feeds into the Plot and Animation / Reporting area.
- Simulation Environment:** A dashed box labeled **HIL** (Hardware-in-the-Loop) contains **dSPACE** and **xPC**, which are connected to the Experimentation area.
- Reporting:** The Plot and Animation / Reporting area outputs Model doc. and Experiment log (HTML, VRML, PNG, ...).
- Simulation Environment:** A dashed box labeled **Simulink MATLAB** is connected to the Experimentation area.

Library Browser

Selected Model - Diagram Mode

Simulation Mode

A screenshot of the UML modeling toolbar with various icons. Red arrows point from text labels to specific icons: 'Help' points to the question mark icon; 'Toggle grid' points to the grid icon; 'Toggle connect mode' points to the connect mode icon; 'Documentation View' points to the book icon; 'Diagram View' points to the diagram icon; 'Icon View' points to the icon icon; 'Text View' points to the text icon; 'Zoom Level' points to the zoom slider; and 'Used Classes View' points to the 'Used Classes' icon.

The screenshot shows the Synopsys Dynamic Modeling Laboratory (Diagram) window. The 'Export' menu is open, displaying the following options: 'To Clipboard', 'Image...', 'Zimpl.d...', 'Setup HTML...', and 'HTML...'. The 'Zimpl.d...' option is currently selected. The background shows a grid-based diagram area and a toolbar with various icons.

Dynamic Modeling Library - Diagram

File Edit Simulation Set Interface Commands Window Help

Library Browser

New Model

Component Browser

Modeling Mode

[illegible]

The screenshot displays the software's toolbar with the following labeled functions:

- File Commands** (Grouped by a blue bracket):
  - Show diagram layer
  - Previous level (diagram layer)
- Diagram Commands** (Grouped by a blue bracket):
  - Delete diagram
  - New diagram
  - New plot window
- Plot Commands** (Grouped by a red bracket):
  - Rescale Curves
  - Erase Curves
  - Toggle grid
  - Visualize 3D objects
- Animation Commands** (Grouped by a green bracket):
  - Play
  - Pause
  - Rewind to start
  - Reverse
  - Back step
  - Forward step
  - Forward
- Simulation Commands** (Grouped by a red bracket):
  - Run script
  - Translate model
  - Simulate model
  - Stop action
  - Experiment setup
- Help** (Grouped by a red bracket):
  - Help

Text View and Graphical View are equivalent

```

model Unnamed
  package Mechanics.Rotational.Inertia Axis #:
    package Mechanics.Rotational.Spring spring #:
      equation
        connect(Axis.flange_b, spring.flange_a) #:
      end Unnamed;
    end Mechanics.Rotational.Inertia Axis #:
  end package Mechanics.Rotational.Spring spring #:
end Unnamed
  
```

### Build model from Libraries (Modeling Mode)

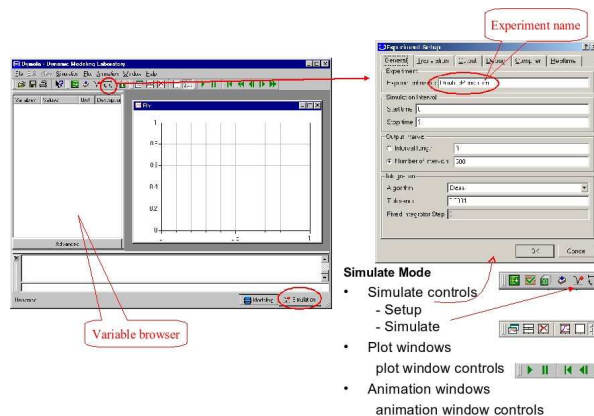
- Advanced: build packages to organize own libraries
- Advanced: Use Modelica language for own models

### Run model (Simulation Mode)

- Single run
- Multiple runs via scripts
- Optimize

### Post-process results (Simulation Mode)

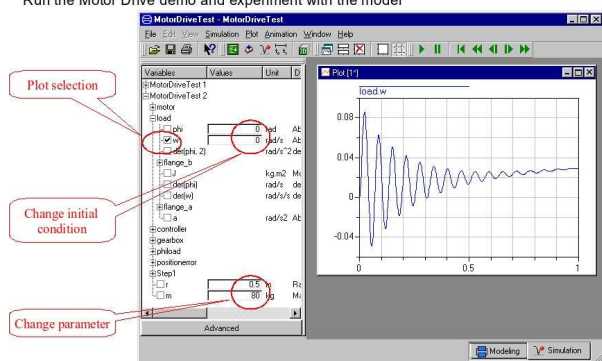
- Plot
- Animate
- Analyze (scripting, Modelica language, external tools)



## Simulation in Dymola 2

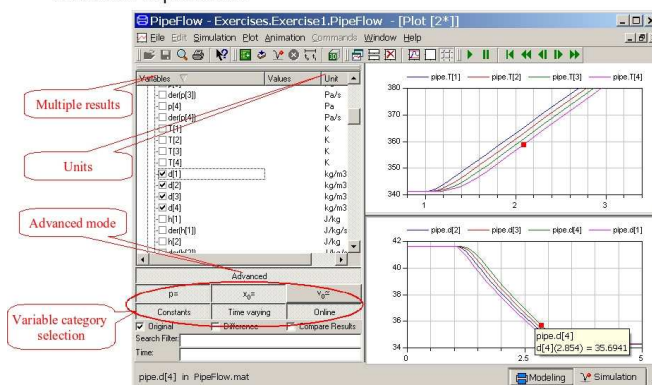
### Simulate an existing model

Run the Motor Drive demo and experiment with the model



## Simulation in Dymola 3

### Advanced experiments



## Live Demo 1 - Pendulum

### Build model by editing text equations

- Simple planar pendulum
- Two states

$$\ddot{\theta} = -\frac{g}{L} \sin \theta \quad (1)$$

model SimplePendulum

```
constant Real g = 9.81;
parameter Real L = 1;
Real Theta(start = 0.1);
Real ThetaDot;
```

equation

```
ThetaDot = der(Theta);
der(ThetaDot) = - g/L*sin(Theta);
```

end SimplePendulum;

## Dymola features - Linearization

- Linearize model, from "Simulation"-menu, beware linearization is at the start time  $t = 0$ !
- Simulate model to the desired linearization point
- Set the final state as the initial state with `importInitial()`;
- Linearize as usual
- Load file `ds1in.mat` into Matlab with e.g. `tloadlin`, many specific matlab routines delivered with Dymola.

## Live Demo 2 - MotorDrive

### Build model by using existing components

- Simple DC motor with ideal gearbox and inertia
- Three states

