Modeling Technical Systems

Mag MA MA Schweiger Gerald
TU Graz
Spring 2017
Outline

• Scope of the course

• Introduction

• Modelica Basics

• Introduction to Dymola
Scope of the course

• Introduction to Modelica

• Differential algebraic equations and numerical methods

• Co-Simulation

• Dynamic optimization of Modelica models
Introduction
**Introduction**

**Systems**

What is a system? *A system can be almost anything*...

- The universe, a space shuttle, a car, an energy system, etc.
- A system can contain subsystems that are themselves systems

**Definition of a system:**

*A system is an object or collection of objects whose properties we want to study.*

**Reasons to study systems?**

- Understand it in order to build it → engineering point of view
- Satisfy human curiosity (f.i. to understand more about the nature)
We must be highly selective in the system definition, depending on what aspects we want to study.

An important property of systems is that they should be observable: We can influence their behavior through inputs:

The inputs of a system are variables of the environment that influence the behavior of the system. These inputs may or may not be controllable by us.

The outputs of a system are variables that are determined by the system.
In many systems the same variables act as both inputs and outputs.

Acausal behavior: the relationships between variables do not have a causal direction, which is the case for relationships described by equations.

Example: In a mechanical system the forces from the environment influence the displacement of an object, but on the other hand the displacement of the object influences the forces between the object and environment. What is input and what is output in this case is primarily a choice by the observer, guided by what is interesting to study.
Experiments

• Observability is essential in order to study a system.
• We must at least be able to observe some outputs of a system.
• We can learn even more if it is possible to exercise a system by controlling its inputs. This process is called experimentation:

  An experiment is the process of extracting information from a system by exercising its inputs.

To perform an experiment on a system, it must be both controllable and observable. We apply a set of external conditions to the accessible inputs and observe the reaction of the system by measuring the accessible outputs.
Introduction

Disadvantages of the experimental method:

• For a large number of systems, many inputs are not accessible and controllable. Many outputs are not accessible for measurements.
• Experiments might be too expensive.
• Experiments might be too dangerous.
• The system needed for the experiment might not yet exist.
• The time scale of the dynamics of the system could be a problem. For example, climate change.

This drawbacks led us to the model concept.
Introduction

Model concept

A model can be used to answer questions about a system without doing experiments on the real system:

A model of a system is anything an experiment can be applied to in order to answer questions about that system.

Is a model of a system valid? → meaningless question:

• A model of a system might be valid for one experiment on the model and invalid for another.

Models:

• Mental models
• Verbal models
• Physical models
• Mathematical models
Introduction

Physical models

- Mimics some properties of a real system. Examples: During the design of artifacts (airplanes, buildings, etc.) it is common to construct physical models with the same shape and appearance as the real objects.

Mathematical models

- A description of a system where the relationships between variables of the system are expressed in mathematical form. Most laws of nature are mathematical models (Ohms’s law, Newton’s law, etc.).

The scope of this course is “Physical modelling”:

*Building mathematical models of physical systems in the computer. The structuring and synthesis process is the same as when building real physical models.*
Simulation

A simulation is an experiment performed on a model:

- Simulation of an industrial process. Goal: learn about the behavior under different operating conditions in order to improve the process
- Simulation of an energy system. Goal: improve the overall system efficiency
- Simulation of the vehicle behavior. Goal: realistic training

Advantages of Simulation:

- Cheap, save, time-scale of the dynamics (f.i. development of the universe), all variables can be studied, easy manipulation of models, etc.

Disadvantages of Simulation:

- The modeller forget that most models are simplified, weak, wrong, not valid for most experiments, etc. 😊
Modelling approaches

Physical Modeling vs. Block-Oriented Modeling
(acausal vs. causal modelling)

Acausal modelling
- Declarative modelling style: modelling based on equations instead of assignments
- Equation relationship: no input/output definitions
- The causality becomes fixed when the system is solved
- Physical modelling reflects the fact that acausal modelling is very well suited for representing the physical structure of systems
- The corresponding mathematical formalism is that of implicit differential-algebraic equations

Causal modelling
- Cause-effect relationship; input/output variables
- The corresponding mathematical formalism is that of explicit ordinary differential equations
## Introduction

### Visual component level

#### Acausal model

![Acausal model](image1)

#### Causal model using explicit signal flow

![Causal model](image2)

### Text level

#### Equation

\[ R \times I = U \]

#### Causal possibilities

\[
\begin{align*}
R & := \frac{U}{I} \\
U & := R \times I \\
I & := \frac{U}{R}
\end{align*}
\]
Introduction

Causal approaches

Advantages
- The model approach is close to the solution of the algorithm

Disadvantages
- Difficult to read code/models (physical topology is lost)
- Limited reusability / hard to maintain: even small changes in the physical structure may result in large changes to the corresponding block-oriented model.
- Model knowledge is stored in books which computer cannot access since programming languages usually do not allow equations.

Acausal approaches

Advantages
- Simple to read code/models
- Reusability, extensibility, adaptability of the models
- Model knowledge is stored in books which computer cannot access since programming languages usually do not allow equations.
- Reuse model knowledge

Disadvantage
- The model approach is further away from the solution of the algorithm – additional steps
Modelica
Modelica is a free, object-oriented, equation based language for modelling of complex systems.

Initiated in 1996 by Hilding Elmqvist and now used in academia and industry all over the world.

Universities: Aachen, Berkeley, Dresden, Miami, Companies: Audi, BMW, Daimler, Ford, Toyota, VW, ABB, Siemens
Modelica

Modelica Versions

- 1.0 released September 1997
- 1.1 released December 1998: Language elements to model discrete systems
- 2.0 released March 2002: Initialization of models, standardization of graphical appearance
- 2.1 released March 2004: Overdetermined connector to model 3-dim. mechanical systems
- 2.2 released March 2005
- 3.0 released September 2007
- 3.1 released May 2009: Stream connector to handle bi-directional flow of fluid,
- 3.2 released May 2010
- 3.3 released May 2012

Modelica Association established 2000

- Open, non-profit organization

International Modelica Conference

- Since 2000. This year 12\textsuperscript{th} in Prague (May)
Modelica

[...]

is a language for modelling of complex physical systems:

- Aircraft
- Automotive
- Control
- Energy systems
- Robotics
- Satellites
- Systems biology
- etc.
Modelica

Declarative language
- Allow acausal modelling
- The order of the equations does not matter.

Multi-domain modelling
- Modelica is a multi-domain language, not geared towards any specific domain. Easily couple models containing for example, mechanical and electrical components.

Object-oriented
- Models are classes and thus can easily be extended using ordinary object-oriented features.

Visual component programming
- Hierarchical system architecture capabilities.

The language elements are mapped to differential, algebraic and discrete equations. **No support for partial differential equations.**
Multi-domain modelling

- Example: 3 domains (i) electric, (ii) mechanics, and (iii) control

Source [1]
Visual Acausal Component Modeling

Acausal model in Modelica

Causal model in Simulink

Keeps the physical structure

Source [1]
Modelica

- Hierarchical system modelling
• Hybrid modelling

Discrete-time variable $z$ changes value only at event instants, whereas continuous-time variables like $y$ may change value both between and at events.
Modelica

Faster Development, Lower Maintenance than with Traditional Tools

Proprietary code (Fortran, C)
Block Diagram (Simulink, TRNSYS)

Source [2]
# Modelica

## Language vs. Simulation/Optimization Environments

Modelica → Compiler → Simulation Environment

### Language

<table>
<thead>
<tr>
<th>Commercial</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dymola [Dassault Systemes]</td>
<td>JModelica.org [supported by Modelon]</td>
</tr>
<tr>
<td>SimulationX [ITI GmbH]</td>
<td>OpenModelica [s. b. OSMC]</td>
</tr>
<tr>
<td>MapleSim [Maplesoft]</td>
<td>Modelica [part of Scilab]</td>
</tr>
<tr>
<td>Mosilab [Fraunhofer FIRST]</td>
<td></td>
</tr>
<tr>
<td>Wolfram SystemModeler [Wolfram]</td>
<td></td>
</tr>
</tbody>
</table>

### Tool
Modelica is a textual language. However, most simulation environments supports textual and graphical modelling:

- The equations in each model can always be viewed and the structure of the physical system is kept.

```model Pendulum
  constant Real g=9.81 "Gravity";
  parameter Real L=1 "Length";
  parameter Real m=1.0 "Mass";
  Real x(start=0.1);
  Real y(start=-0.9);
  Real vx;
  Real vy;
  Real F;

  equation
    der(x) = vx;
    der(y) = vy;
    x^2 + y^2 = L^2;
    m*der(vx) = (F/L)*x;
    m*der(vy) = (F/L)*y - m*g;

end Pendulum;
```
Modelica Standard Library (MSL)

Modelica Standard Library is a standardized predefined package developed by Modelica Association.

It can be used freely for noncommercial and commercial purposes.

Available online including documentation [https://www.modelica.org/libraries](https://www.modelica.org/libraries)

Free library consisting of over 1200 models and about 900 functions in many engineering domains.

- Blocks: Library for basic input/output control blocks
- Electrical: Library for electrical models (analog, digital and multiphase circuits)
- Fluid. 1-dim Flow in networks (pipes, fluid machines, vessels, etc.)
- Mechanics: Library for mechanical models (Rotational, Translational and MultiBody)
- Thermal: Library for thermal systems
- Etc...
A collection of free and commercial libraries. [https://www.modelica.org/libraries](https://www.modelica.org/libraries)

### Free libraries


- **MotorcycleDynamics**: dynamic simulation of a motorcycle. [https://github.com/modelica-3rdparty/MotorcycleDynamics](https://github.com/modelica-3rdparty/MotorcycleDynamics)

- **PowerSystems**: model electrical power systems. [https://github.com/modelica/PowerSystems](https://github.com/modelica/PowerSystems)

- **SystemDynamics**: graphical library for modeling mass and information flows in a continuous-time system. [https://build.openmodelica.org/Documentation/SystemDynamics.html](https://build.openmodelica.org/Documentation/SystemDynamics.html)

Modelica Basics
“Modelica programs are built from classes, also called models. From a class definition, it is possible to create any number of objects that are known as instances of that class. Think of a class as a collection of blueprints and instructions used by a factory to create objects. In this case the Modelica compiler and run-time system is the factory.” [1]
Modelica Basics

The long tradition of “Hello world”: Modelica is an equation-based modelling language, printing a string does not make much sense 😊.

Modelica’s “Hello world” is a trivial differential equation:

\[ \dot{x} = -a \cdot x, \quad x(t_0) = 1 \]

Initialized to a start value of 1. All variables have a start attribute (default = 0). A different start value is accomplished by providing a so-called modifier within parentheses.

```model HelloWorld
Real x(start=1);
parameter Real a=1;

equation
  der(x) = -x;

end HelloWorld;
```
Modelica Basics

Structure of Modelica models

Model body
- Declaration of constants / parameters / states / inputs

Equation section
- Declaration of model equations

Example

```model
Example
...
...
end Example;
```

Type of the object (f.i. a model)

Name of the model

Start of equations
Remark: Modelica is equation based, consider:

\[ x = y + 1 \text{ is identical with } 0 = y + 1 - x \]

The compiler takes care of manipulating the equations.

“Hello World #2”

```modelica
model Basic
  Real x(start=1);
  equation
    der(x) = -x;
end Basic;
```

```modelica
model Basic
  Real x(start=1);
  equation
    0 = - der(x) - x;
end Basic;
```
To specify derivatives the keyword `der()` is to be used.

\[ \text{der}(x) \rightarrow \frac{dx}{dt} \]

Only first order derivatives!
Pendulum equations in Modelica code?

\[
\begin{align*}
    m \frac{d^2 x}{dt^2} &= \frac{F}{L} x \\
    m \frac{d^2 y}{dt^2} &= \frac{F}{L} y - mg \\
    x^2 + y^2 &= L^2
\end{align*}
\]

- \( L \) is the length (parameter) of the Pendulum
- \( g \) is the gravity (constant)
- \( x \) and \( y \) are the coordinates
- \( F \) is the force

```model Pendulum
  constant Real g=9.81 "Gravity";

  parameter Real L=1 "Length";
  parameter Real m=1.0 "Mass";

  Real x(start=0.1);
  Real y(start=-0.9);
  Real vx;
  Real vy;
  Real F;

  equation
    der(x) = vx;
    der(y) = vy;
    x^2 + y^2 = L^2;
    m*der(vx) = (F/L)*x;
    m*der(vy) = (F/L)*y - m*g;

end Pendulum;
```
• Start attributes for time-varying variables may be set using the modifier \textit{start}.

\begin{verbatim}
Real x(start=1.0);
\end{verbatim}

\texttt{x} has (guess) start value 1.0!

\textbf{NOTE}, during initialization this value may change!

\begin{verbatim}
Real x(start=1.0, fixed=true);
\end{verbatim}

\texttt{x} must have start value 1.0!

Forces the initialization algorithm to use 1.0 as start value.
The Van der Pol oscillator

\[ \ddot{x} = (1 - x^2)\dot{x} - x + u \]

- Parameter declaration
- State declaration
- Equations declaration

```modelica
model VanDerPol
  // start values of states
  parameter Real x1_0 = 0;
  parameter Real x2_0 = 1;

  // The states
  Real x1(start = x1_0);
  Real x2(start = x2_0);

  // The control signal
  input Real u;

  equation
    der(x1) = (1 - x2^2) * x1 - x2 + u;
    der(x2) = x1;
end VanDerPol;
```
The important variable types are: Real, Integer, Boolean

Additionally: String, Complex

Variability: Constants, Parameters, Time-varyings, Inputs, Outputs

Constants
• Known by the compiler and do not change values.

Parameters
• Known before the simulation starts and remain constant throughout the simulation

```modelica
parameter Real p1 = 1;
constant Integer y = 10;
Real x;
```
Components and Connectors

In order to model large systems, (hierarchically) decomposing the system into smaller components which are then connected is essential.

- Each rectangle in the diagram example represents a physical component (resistor, capacitor, mechanical gear, valve, pump, etc.)
- The connections represented by lines in the diagram correspond to real, physical connections (connections can be realized by electrical wires, the mechanical connections, pipes, heat exchange etc.)
- The connectors are shown as small square dots on the rectangle in the diagram. Variables at such interface points define the interaction between the component

[Diagram of components and connections]
Modelica Basics

Components and Connectors

Front wheels → Chassis → Rear wheels

Controller → Chassis
Connectors: In order to make the components interact with each other, interfaces needs to be defined. In the Modelica language they are called “Connectors”.

- Two kind of variables within connectors:
  - Non-flow variables (potential variables)
  - Flow variables: Represent some kind of flow in a certain domain. The value is positive when the flow is going into a component.

- Coupling
  - Equality coupling for non-flow variables (Kirchhoff’s first law)
  - Sum-to-zero coupling for flow variables (Kirchhoff’s current law)

- Connection equation in Modelica.
  - F.i. electrical domain: Connect(R1. p, R2. p)
Modelica Basics

- Connect(pin1, pin2) corresponds to

\[
\begin{align*}
\text{pin1.} \, v & = \text{pin2.} \, v \\
\text{pin1.} \, i + \text{pin2.} \, i & = 0
\end{align*}
\]

Multiple connections

\[
\begin{align*}
v_1 = v_2 = v_3 = \ldots v_n & \rightarrow n - 1 \text{ equations per connection} \\
i_1 + i_2 + i_3 + \ldots \, i_n & = 0 \rightarrow 1 \text{ equation per connection}
\end{align*}
\]
# Modelica Basics

Connectors in different domains

<table>
<thead>
<tr>
<th>Domain</th>
<th>Potential</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical</td>
<td>Voltage</td>
<td>Current</td>
</tr>
<tr>
<td>Translational</td>
<td>Position</td>
<td>Force</td>
</tr>
<tr>
<td>Rotational</td>
<td>Angle</td>
<td>Torque</td>
</tr>
<tr>
<td>Magnetic</td>
<td>Magnetic potential</td>
<td>Magnetic flux rate</td>
</tr>
<tr>
<td>Hydraulic</td>
<td>Pressure</td>
<td>Volume flow</td>
</tr>
<tr>
<td>Heat</td>
<td>Temperature</td>
<td>Heat flow</td>
</tr>
<tr>
<td>Chemical</td>
<td>Chemical potential</td>
<td>Particle flow</td>
</tr>
<tr>
<td>Pneumatic</td>
<td>Pressure</td>
<td>Mass flow</td>
</tr>
</tbody>
</table>
In Modelica you can additionally specify:

- Functions
- Algorithms
- External functions (calling C or FORTRAN)
- Hybrid constructs (if and when clauses)
Introduction to Dymola
Dymola

State of the art modelling and simulation tool based on the Modelica language.

“Dymola, Dynamic Modeling Laboratory, is a complete tool for modeling and simulation of integrated and complex systems for use within automotive, aerospace, robotics, process and other applications.”

Developed since 1996 and now used in industries such as, SCANIA, Toyota, BMW, Ford and Tetra Pak.
Dymola

Dymola (modelling layer)

- Modelica standard library
- Toggle between viewing textual or objects layer
- Modelling area
- Toggle between the modelling and simulation tab
Retrieving information after a simulation:

- Goto, Simulation
- Choose, Show Log

A log showing the simulation statistics will appear.
Integrators and Simulation Setup

Integrators are found under General in Simulation Setup.

- Switch integrator
  - Dassl
  - Euler
  - Radau
  - more...
- Change tolerance
In the Variable Browser:

- Right click on a variable
- Choose to use as independent variable


Acknowledgment

• Christian Andersson, Per-Ola Larsson and Stéphane Velut: Modelon AB, Sweden
• Claus Führer, Professor LTH Lund, Sweden
• David Broman: Associate Professor KTH Royal Institute of Technology, Sweden
• Andres Markus and Thomas Schmitt: 3ds, Germany

Modelica References

• Modelica Association: https://www.modelica.org/
• Modelica by Example, free book about Modelica: http://book.xogeny.com/