Simulation, Analysis, and Validation of Computational Models

— Modelling —

Lecturer: Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Overview

- Systems-level modeling
- Modelling
- Modelica
- Outlook

Bridging between equations and high-level systems?

Equations describe

- Different forms of energy: Acceleration, potential, friction
- Motion of matter: Air pressure and velocity
- Expectations and value: Option prices
- Input and output

RW systems are described by

- Graphs, schemes, diagrams (boxology)
- Stories, experience, knowledge
- Mathematical constructs
- Modelling languages

Complexity

- Increase of complexity requires tools to handle complexity
- Hierarchical, flexible, extensible, expressive, standardised
- Part count + source lines of code (Aerospace, automobile, electronic circuits)
  - 1960: 10^3 - 10^4
  - 1990: 10^5 - 10^6
  - 2010: 10^8 - 10^9
- Merging of modeling and programming

Modelling languages vs. programming languages

- Programming languages are executable. They share many features with modelling: expressivity, heterogeneity, mechanisms of import and reuse, libraries, frameworks, hierarchies
- Modelling languages: Not directly executable (but are often translated into an executable programming language)
- Computational aspect are usually not considered in modelling, e.g. parallel and distributed computing
- Characteristic for modelling languages is "sketchiness" incl.
  - abstraction
  - underspecification/refinement
  - reduced precision and detailedness
  - compactness
  - views, relatedness to modelling domain

Gray & Rumpe (2022) Reflection on the differences between modeling and programming. *Software and Systems Modeling* 21:2097–2099

Model-based Systems Engineering

- MBSE replaces documents with (executable) models
- Need System Modeling Languages
  - Ontology, semantics, syntax
  - Object-Process Methodology (OPM) – Excellent for pre-Phase A
  - SysML – Widely used in some industries, 9 diagram types
  - Modelica – Declarative language, able to execute models in the time domain to simulate steady-state and transient behavior

Modelling languages

Main types

- Graphical modeling languages: Diagram techniques to describe structure of domain knowledge
- Textual modeling languages: keywords, parameters or natural language phrases

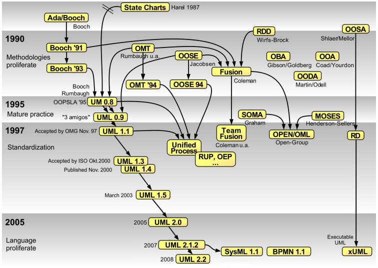
Computer-interpretable expressions can be generated from both

Many variants

- Behavioral languages (process calculus or process algebra)
- Information and knowledge modeling
- VR modelling
- ...
- Examples: UML, Petri nets, EXPRESS, pseudo-code, ...

Unified Modeling Language (UML)

- Modelling language for specification, construction, documentation and visualisation of software components, standardised by ISO/IEC 19505
- Specification of required and provided interfaces
- Independent of particular programming languages and development processes.
- Behavior diagrams, interaction diagrams, and structure diagrams: class, component, deployment, object, package, composite structure, profile, use case, activity, state machine, sequence, communication, interaction overview, timing



Guido Zockoll, Axel Scheithauer & Marcel Douwe Dekker (2009), CC BY-SA 3.0, curid=23052855

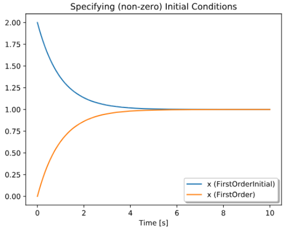
Modelica

- Object-oriented modelling language, strongly typed
- Language for Cyber-Physical Systems
- Acausal modeling possible by mathematical functions
- Modelica Association (1997)
- Implementations: AMESim, CATIA Systems, Dymola, JModelica.org, MapleSim, Wolfram SystemModeler, Scicos, SimulationX, Xcos;
  free and open source version: OpenModelica
- Alternatives: Simulink (Matlab), Scilab (numerics), GNU Octave (mathematical modelling), Inventor (3D CAD), SOLIDWORKS (finite elements), Autodesk (3D design)

Simple First Order System in Modelica

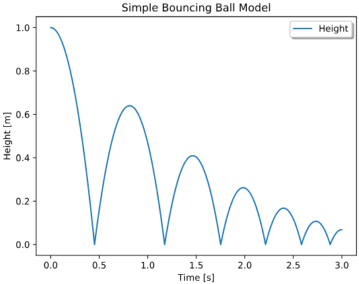
```
model FirstOrderDocumented "A first order differential equation"
  Real x "State variable";
equation
  der(x) = 1-x "Drives value of x toward 1.0";
end FirstOrderDocumented;
```

```
model FirstOrderInitial "A first order differential equation"
  Real x "State variable";
initial equation
  x=2; "Compute initial values";
equation
  der(x) = 1-x "x approaches 1";
end FirstOrderDocumented;
```



https://mbe.modelica.university/behavior/equations/first_order/

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



https://mbe.modelica.university/behavior/discrete/bouncing/

Modelica: Advection

```
model Advection "advection equation"
  parameter Real pi = Modelica.Constants.pi;
  parameter DomainLineSegment1D omega(L = 1, N = 100)
"domain";
  field Real u(domain = omega) "field";
initial equation
  u = sin(2*pi*omega.x) "IC";
equation
  der(u) + pder(u,x) = 0 indomain omega "PDE";
  u = 0 indomain omega.left "BC";
  u = extrapolateField(u) indomain omega.right "extrapolation";
end Advection;
```

Modelica playground

- https://playground.modelica.university

Modelica language

- Model: Unit in a hierarchical structure
- Variables, parameters, conditionals, predefined functions
- (differential) equations
- Connectors are ports that carry the value of a variable (not just a "class" but a connection to the RW), expresses meaning for variables and parameters
- Primarily equation-based
  - Assignment ($E = m\,c^2$): naming r.h.s. "E", causality: $\leftarrow$
  - Equality ($E == m\,c^2$): to be solved either way
    acausal modeling

Ontology

"An ontology encompasses a representation, formal naming, and definitions of the categories, properties, and relations between the concepts, data, or entities that pertain to one, many, or all domains of discourse."

A formal ontology shows the following properties:

- Indefinite expandability
- Remains consistent with increasing content.
- Content and context independence:
- Any kind of *concept* from the target domain finds its place.
- Accommodates different levels of granularity.

For Modelica this is still work in progress

https://eliseck.github.io/MO-x-IFC/TBox/MoOnt/index.html,
https://en.wikipedia.org/wiki/Formal_ontology

System-level simulation

- Study global behavior of large cyber-physical systems
- Application of the holistic principle to computer simulation. Consider e.g.
  - Feedback in control, adaptation, learning
  - Realistic noise
  - Failure of components
  - Requirement verification
- In the system, not all components will cover their full behavioural repertoire
- Compare: Co-simulation of the system sub-parts
- Compare: Digital twins

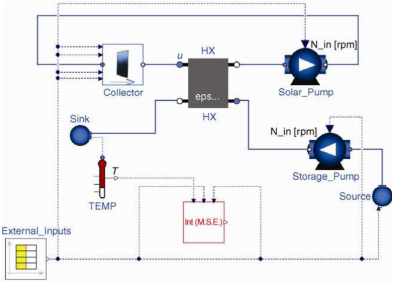
System-level simulation

- Modelling
  - Hybrid systems
  - Acausal modeling
  - Hardware-in-the-loop vs. software-in-the-loop
- Tasks (beyond testing
  - Dimensioning vs. efficiency
  - Refining specification vs. model order reduction
  - Optimization, calibration
- Modelling languages are not necessarily synchronous languages as have been developed for *reactive systems*

System-level simulation: Example

Schematic of the solar thermal system model

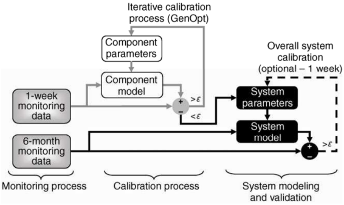
Component	Parameter	Catalog values
Solar collector field, consisting of 4 collectors	Aperture area	5.355 m^2
	Efficiency c_0	0.781
	Efficiency curve coefficient c_1	3.09 $W/(m^2 \cdot K)$
	Efficiency curve coefficient c_2	0.0096 $W/(m^2 \cdot K)$
	Incidence angle modifier	0.92
Primary pump	Maximum power	430 W
Secondary pump	Maximum power	80 W
Heat exchanger, overall values for 2 heat exchangers in series	Heat transfer coefficient	3408 $W/(m^2 \cdot K)$
	Surface area	10.56 m^2
	Nominal heat flow rate	180 kW



Fontanella, (2012) Calibration and validation of a solar thermal system model in Modelica. *Building simulation* 5, 293-300. Tsinghua Press.

System-level simulation: Example

Calibration of the solar thermal system model



ε: effectiveness

Adjust model parameters (such as rate parameters, incident angle modifier)
Optimise: Solar pumps and storage pump to maximise "heat flow effectiveness"

Fontanella, (2012) Calibration and validation of a solar thermal system model in Modelica. *Building simulation* 5, 293-300. Tsinghua Press.

Types of applications

- World models[1]: for climate, weather, migration, trade, transport, global diseases,
- Civil engineering (Infrastructure)
- (Mechanical and electrical) engineering: Automotive, aerial, space, production plants, cyber-physical systems
- Further engineering: Biological, chemical, interdisciplinary
- Fintech
- Scientific models
- Specialised models

[1] Differently used in biology and robotics for models of the environment.

Remaining topics

11 Case studies (week 8/1)

12 Modeling and simulation today (week 8/2)

B1 PINN (week 6/2)

B2 More on PINNs (week 9/1)

B3 Industry 4.0 (week 9/2)

B4 Digital twins (week 10/1)

R Revision (week 10/2)