TTK4130 Modeling and Simulation

To learn:

- Formulate mathematical models from first principles
- Simulate models using computers

Main purpose: Control system design/testing/validation, hence *dynamic* modeling.

Instructors:

- Lecturer: Leif Erik Andersson
- Teaching assistant: Mikkel Sørensen (D351 A)

Itslearning – Forum & calendar

- Use the forum for questions etc.
- Four categories available:
  - Exam
  - Exercises
  - Lectures
  - Other questions
  - (No category)
- Advantageous:
  - Probably not only you have the questions, so it helps others and we only have to answer the questions once.
  - Also other students can answer, for example to questions about the exercises. Therefore, you might get faster an answer.
- There calendar available where the exercise hours and lectures can be downloaded.

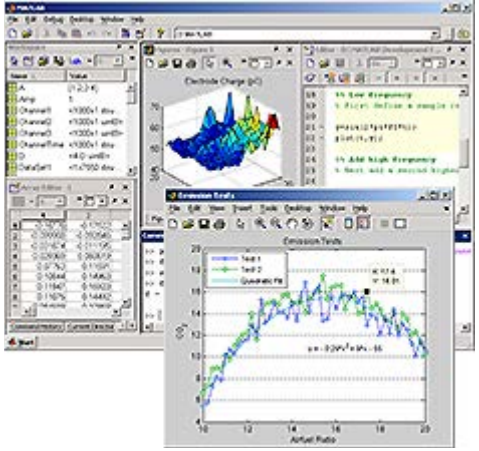
Modelica for Mac

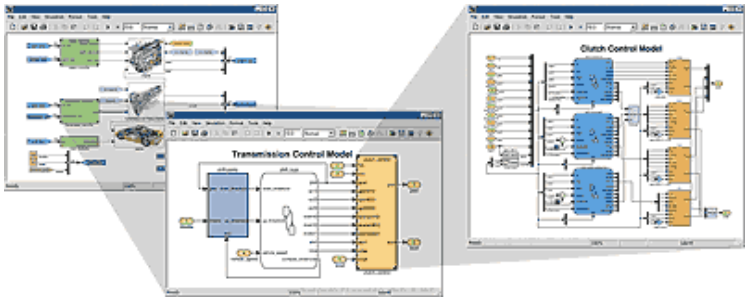
1. Install Windows on a parallel machine like:
   - Parallels (http://www.parallels.com/eu/products/desktop/)
   - VirtualBox (https://www.virtualbox.org/)

2. Install OpenModelica which has a Mac version
   - https://www.openmodelica.org/download/download-mac

3. There is also the option to download Virtual Machine on the Open Modelica webpage, which explains installation steps etc.
   - https://www.youtube.com/watch?v=11OkQs8VUrU (Youtube video on the installation with VirtualBox + OpenModelica)

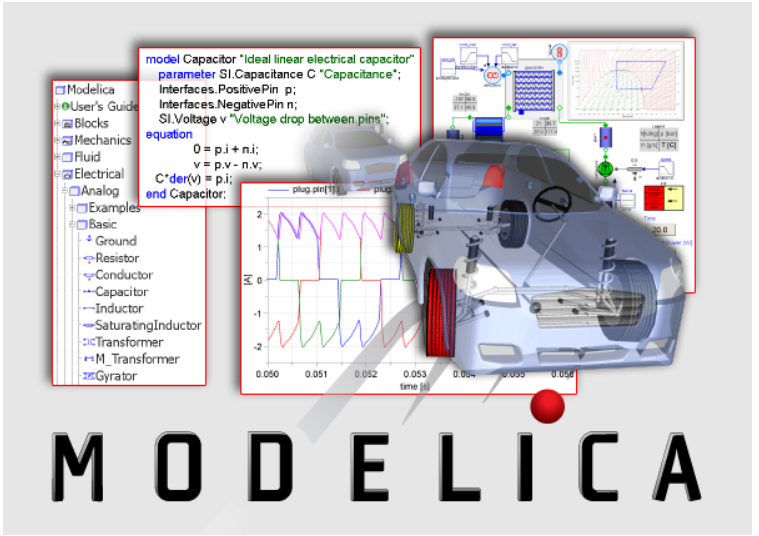
Software



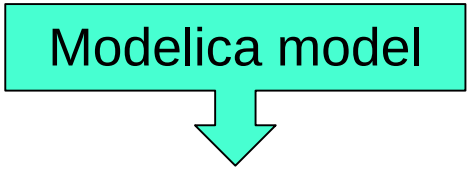


- Matlab/Simulink
  - Computer labs
  - Your own computer
  - Some familiarity with Matlab/Simulink is assumed

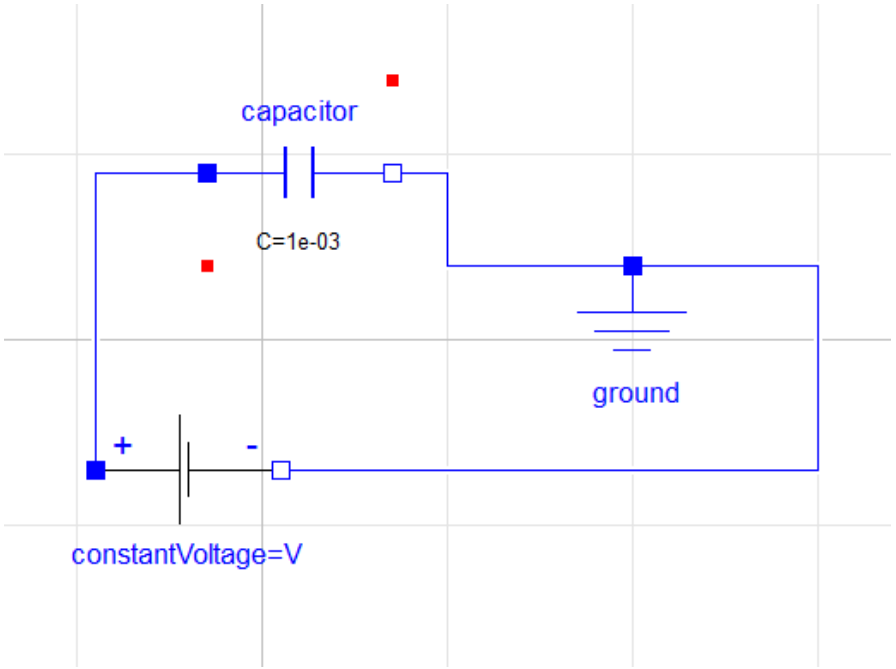
- Dymola
  - Based on the Modelica modelling language
  - Installed on computer labs, but you can also use your PCs (Windows and Linux)
  - License:
    - License server (via VPN), but limited number of licenses (50)
    - Much can be done with demo license



Implementation and Execution of Modelica

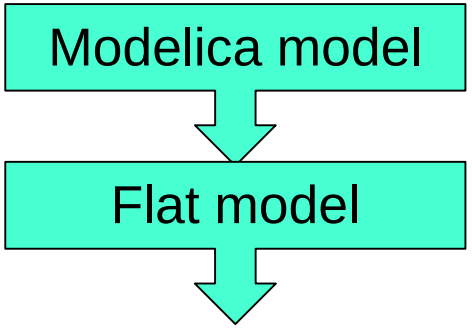


Modelica model




```
model Test_Capacitor

//   parameter Real i=1;
//   parameter Real v=20;
//
//   Pin p;

  Modelica.Electrical.Analog.Basic.Capacitor capacitor(C=1e-03);
  Modelica.Electrical.Analog.Sources.ConstantVoltage constantVoltage;
  Modelica.Electrical.Analog.Basic.Ground ground;
equation
  connect(constantVoltage.p, capacitor.p);
  connect(capacitor.n, ground.p);
  connect(ground.p, constantVoltage.n);

end Test_Capacitor;

partial package Modelica.Icons.Package "Icon for standard packages"

end Package;

model Modelica.Electrical.Analog.Basic.Capacitor
  "Ideal linear electrical capacitor"
  extends Interfaces.OnePort;
  parameter SI.Capacitance C(start=1) "Capacitance";

equation
  i = C*der(v);
end Capacitor;

partial package Modelica.Icons.InterfacesPackage
  "Icon for packages containing interfaces"
//extends Modelica.Icons.Package;
end InterfacesPackage;

partial model Modelica.Electrical.Analog.Interfaces.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)";
  NegativePin n "Negative pin";
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;

connector Modelica.Electrical.Analog.Interfaces.PositivePin
  "Positive pin of an electric component"
  Modelica.SIunits.Voltage v "Potential at the pin";
  flow Modelica.SIunits.Current i "Current flowing into the pin";
```

Implementation and Execution of Modelica

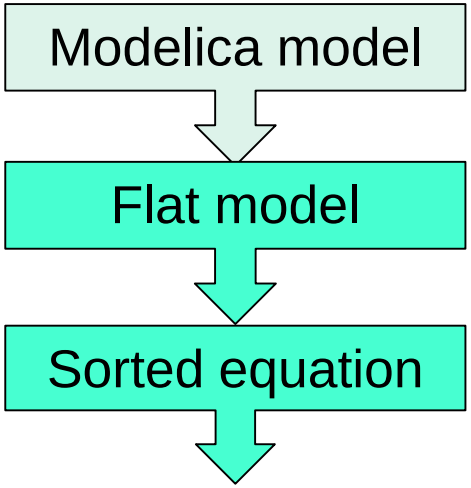


Parsed, preprocessing, flattening

Flat model

```
model Test_Capacitor
parameter Modelica.SIunits.Capacitance capacitor.C(start = 1) = 0.001
  "Capacitance";
parameter Modelica.SIunits.Voltage constantVoltage.V = 1 "Value of constant voltage";

Modelica.SIunits.Voltage capacitor.v "Voltage drop between the two pins (= p.v - n.v)";
Modelica.SIunits.Current capacitor.i "Current flowing from pin p to pin n";
Modelica.SIunits.Voltage capacitor.p.v "Potential at the pin";
Modelica.SIunits.Current capacitor.p.i "Current flowing into the pin";
Modelica.SIunits.Voltage capacitor.n.v "Potential at the pin";
Modelica.SIunits.Current capacitor.n.i "Current flowing into the pin";
Modelica.SIunits.Voltage constantVoltage.v "Voltage drop between the two pins (= p.v - n.v)";
Modelica.SIunits.Current constantVoltage.i "Current flowing from pin p to pin n";
Modelica.SIunits.Voltage constantVoltage.p.v "Potential at the pin";
Modelica.SIunits.Current constantVoltage.p.i "Current flowing into the pin";
Modelica.SIunits.Voltage constantVoltage.n.v "Potential at the pin";
Modelica.SIunits.Current constantVoltage.n.i "Current flowing into the pin";
Modelica.SIunits.Voltage ground.p.v "Potential at the pin";
Modelica.SIunits.Current ground.p.i "Current flowing into the pin";

// Equations and algorithms

  // Component capacitor
  // class Modelica.Electrical.Analog.Basic.Capacitor
    // extends Modelica.Electrical.Analog.Interfaces.OnePort
    equation
      capacitor.v = capacitor.p.v-capacitor.n.v;
      0 = capacitor.p.i+capacitor.n.i;
      capacitor.i = capacitor.p.i;
    // end of extends
  equation
    capacitor.i = capacitor.C*der(capacitor.v);

  // Component constantVoltage
  // class Modelica.Electrical.Analog.Sources.ConstantVoltage
    // extends Modelica.Electrical.Analog.Interfaces.OnePort
    equation
      constantVoltage.v = constantVoltage.p.v-constantVoltage.n.v;
      0 = constantVoltage.p.i+constantVoltage.n.i;
      constantVoltage.i = constantVoltage.p.i;
    // end of extends
  equation
    constantVoltage.v = constantVoltage.V;

  // Component ground
  // class Modelica.Electrical.Analog.Basic.Ground
  equation
    ground.p.v = 0;

  // Component
  // class Test_Capacitor
  equation
    capacitor.n.i+constantVoltage.n.i+ground.p.i = 0.0;
    constantVoltage.n.v = capacitor.n.v;
    ground.p.v = capacitor.n.v;
    capacitor.p.i+constantVoltage.p.i = 0.0;
    constantVoltage.p.v = capacitor.p.v;

end Test_Capacitor;
```

Implementation and Execution of Modelica



Parsed, preprocessing, flattening

“make equations causal”, perform BLT transformation

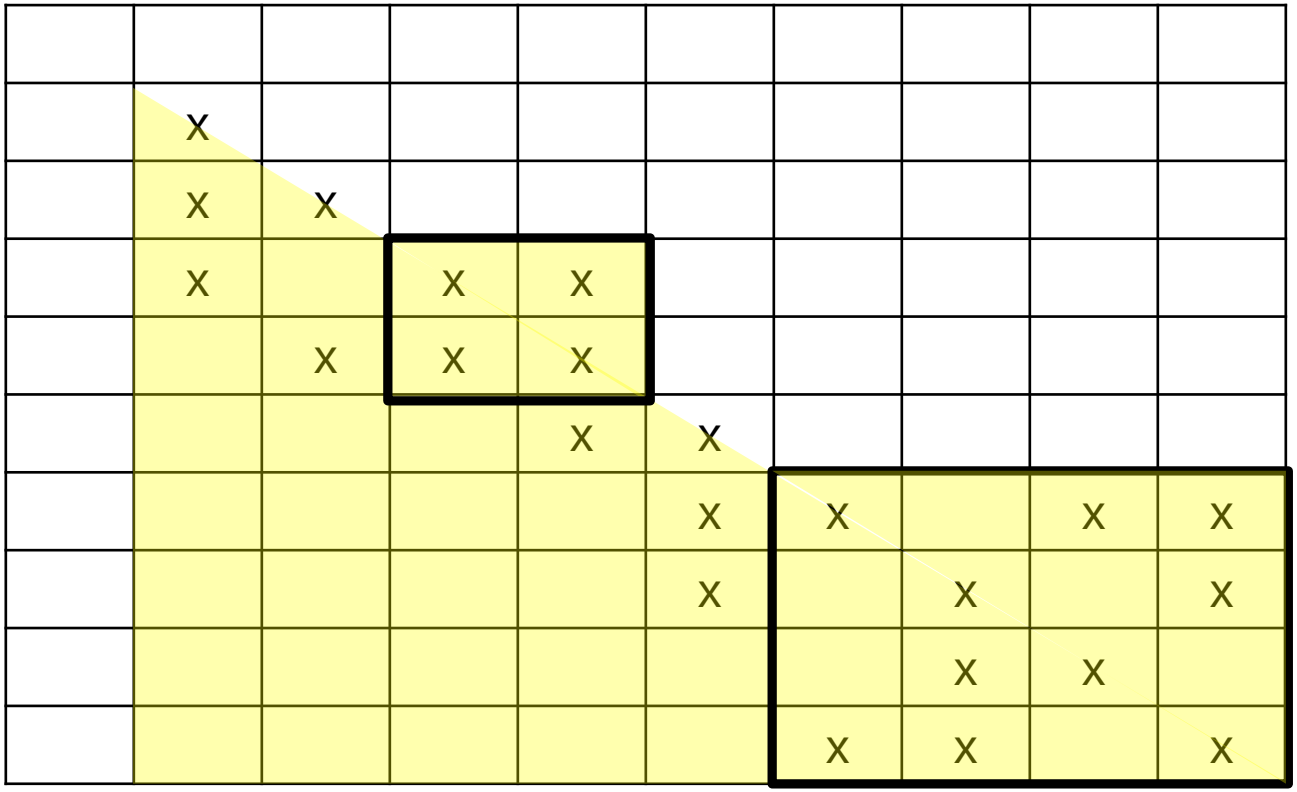
Lower Triangular Matrix (Example)

	v_G	v_{C1}	v_{R1}	u_R	i_{R1}	i_{S1}	i_{C1}	du_c	i_G
1)	x								
6)	x	x							
2)	x		x						
4)		x	x	x					
3)				x	x				
7)					x	x			
8)					x		x		
5)							x	x	
9)						x	x		x

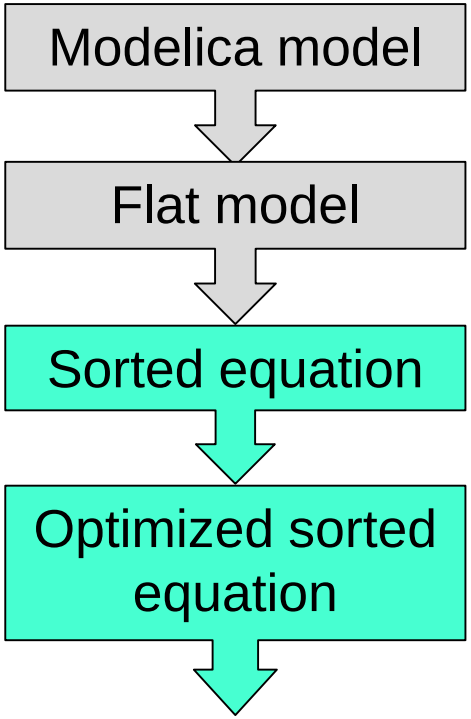
Causal List:

1) $v_G := 0$
6) $v_{C1} := -u_c + v_G$
2) $v_{R1} := v_G + 10V$
4) $u_R := v_{C1} - v_{R1}$
3) $i_{R1} := u_R/R$
7) $i_{S1} := i_{R1}$
8) $i_{C1} := i_{R1}$
5) $\frac{du_c}{dt} := i_{C1}/C$
9) $i_G := i_{C1} - i_{S1}$

Block Lower Triangular Matrix (Example)



Implementation and Execution of Modelica

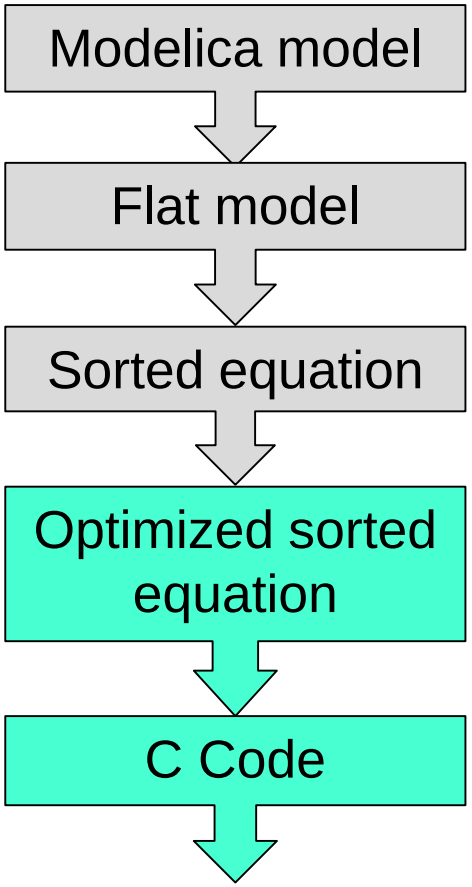


Parsed, preprocessing, flattening

“make equations causal”, perform BLT transformation

Optimize and eliminate equation

Implementation and Execution of Modelica



Parsed, preprocessing, flattening

“make equations causal”, perform BLT transformation

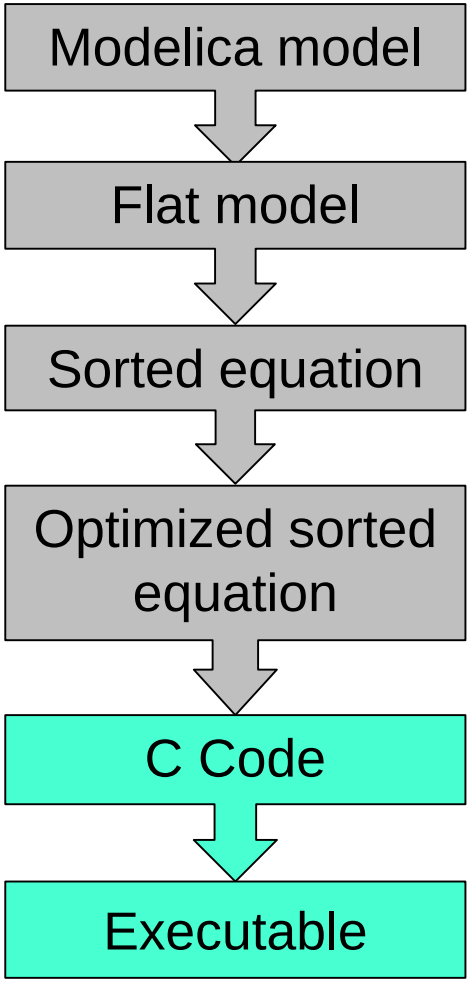
Optimize and eliminate equation

Generate C code

C Code

```
/* DSblock model generated by Dymola from Modelica model Test_Capacitor
 Dymola Version 2014 FD01 (32-bit), 2013-10-17 translated this at Mon Jan 11 08:49:59 2016

   */

#include <matrixop.h>
/* Declaration of C-structs */
/* Prototypes for functions used in model */
/* Codes used in model */
/* DSblock C-code: */

#include <moutil.c>
PreNonAliasDef(0)
PreNonAliasDef(1)
DYMOLA_STATIC const char*modelName="Test_Capacitor";
DYMOLA_STATIC const char*usedLibraries[]={0};
DYMOLA_STATIC const char*dllLibraryPath[]={0};
DYMOLA_STATIC const char*default_dymosim_license_filename=
 "c:/users/leifea/appdata/roaming/dynasim/dymola.lic";
#include <dsblock1.c>

/* Define variable names. */

#define Sections_

TranslatedEquations

InitialSection
W_[3] = 0;
W_[0] = 0.0;
W_[1] = 0.0;
W_[10] = 0.0;
W_[4] = 0.0;
W_[2] = 0.0;
W_[5] = 0.0;
W_[8] = 0.0;
W_[7] = 0.0;
W_[6] = 0.0;
W_[9] = 0.0;
BoundParameterSection
InitialSection
InitialSection
InitialStartSection
InitialSection
DefaultSection
InitializeData(0)
InitialSection
InitialSection
Init=false;InitializeData(2);Init=true;
EndInitialSection

OutputSection

DynamicsSection

AcceptedSection1

AcceptedSection2

DefaultSection
InitializeData(1)
EndTranslatedEquations
```

Implementation and Execution of Modelica



Parsed, preprocessing, flattening

“make equations causal”, perform BLT transformation

Optimize and eliminate equation

Generate C code

Lecture 2:

- About modeling and simulation (F1)
- Why&what do control engineers need to model?
  - Examples (mostly examples I have worked with)
- Model types (E1.1-1.3,E2.1-2.2)
  - State space models, transfer functions
  - Linear models, nonlinear models

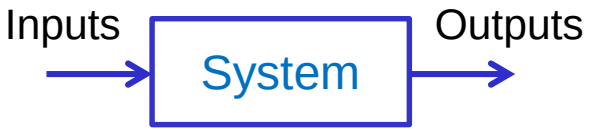
We will *model* and *simulate systems*

What is a system, what is a model and why simulate?

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

"A system is what we distinguish as a system"

- Note that we are usually interested in *specific properties* of these objects – that is, there is usually an *abstraction* implied
  - Ex.: Our system is a pendulum, but we are only interested in (say) pendulum motion, not pendumlum temperature, electrical conductivity, bacterial contamination, etc.
- Our systems have inputs and outputs
  - Inputs: Variables of the environment that influences the behavior of the system
  - Outputs: Variables determined by the system that may influence the environment



Systems and experiments

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

- Challenges:
  - Not all inputs may be manipulated
    - Manipulated vs. disturbance inputs
  - Not all outputs may be measured
  - Experiments may be
    - too expensive
    - too dangerous
    - too slow
  - Our system may not exist (yet)

The Model concept

- "A *model* of a system is anything an *experiment* can be applied to in order to answer questions about that *system*."
  - Mental model
  - Verbal model
  - Physical model
  - Mathematical model
- Mathematical models are often implemented in computers (virtual prototypes)

Simulation

- "A *simulation* is an experiment performed on a model."
  - For us: Mathematical model implemented in computer

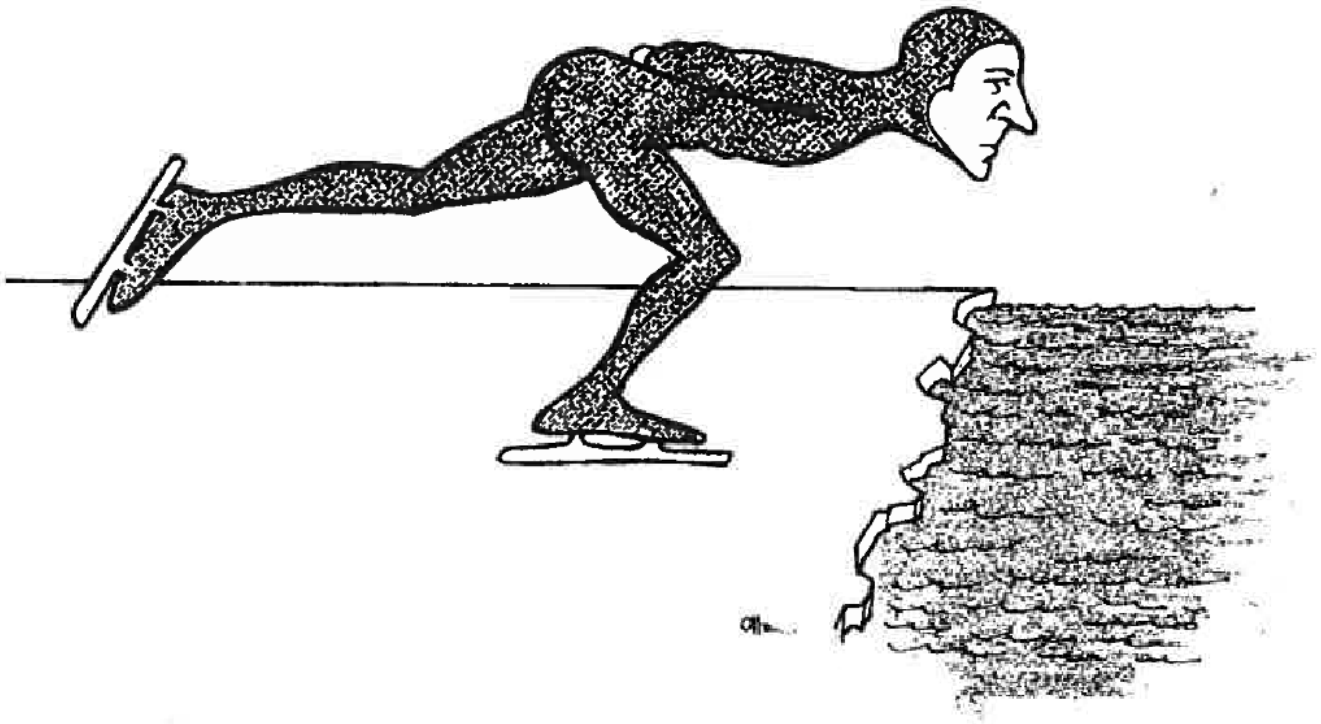
- Why simulate instead of experiments on system?
  - Experiments are too *expensive*, too *dangerous*, too *slow*, or the system to be investigated does *not yet exist*.
  - Variables may be *inaccessible.*
  - Easy *manipulation of models.*
  - Suppression of *disturbances* and *second-order effects.*

- Our (main) purpose for modeling and simulation: Design, testing and validation of control systems

Dangers of modeling and simulation

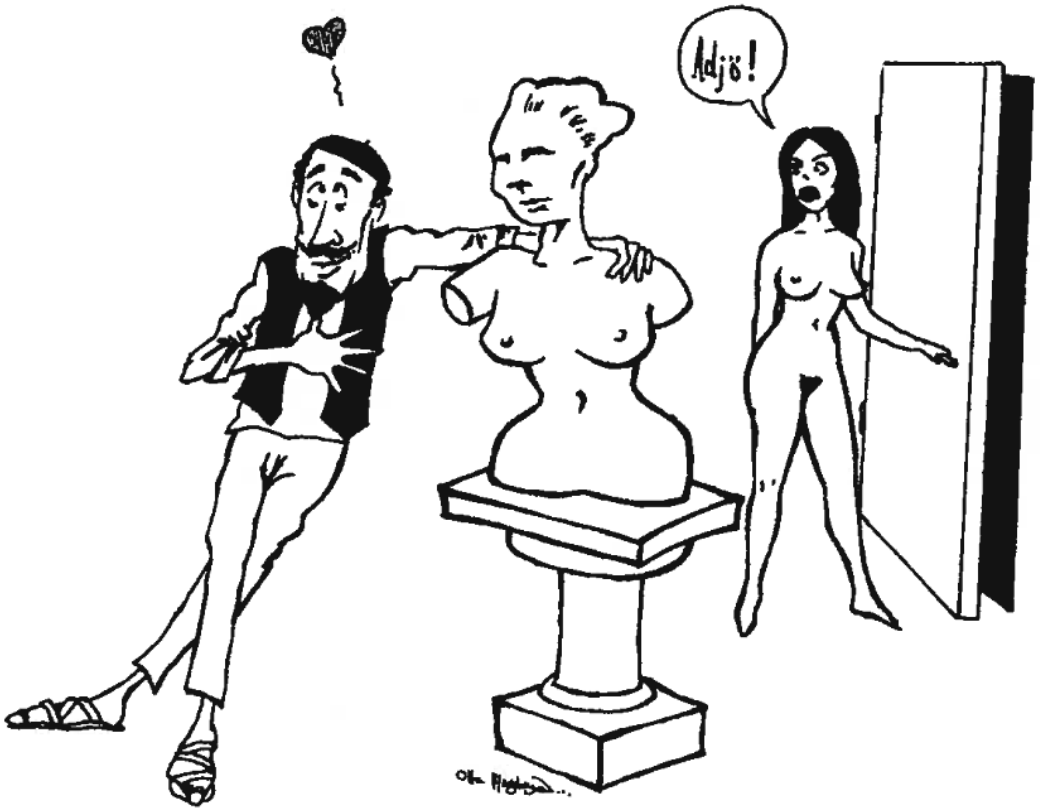
(from Ljung&Glad, ”Modelbygge og simulering”)

Models are based on assumptions with limited validity



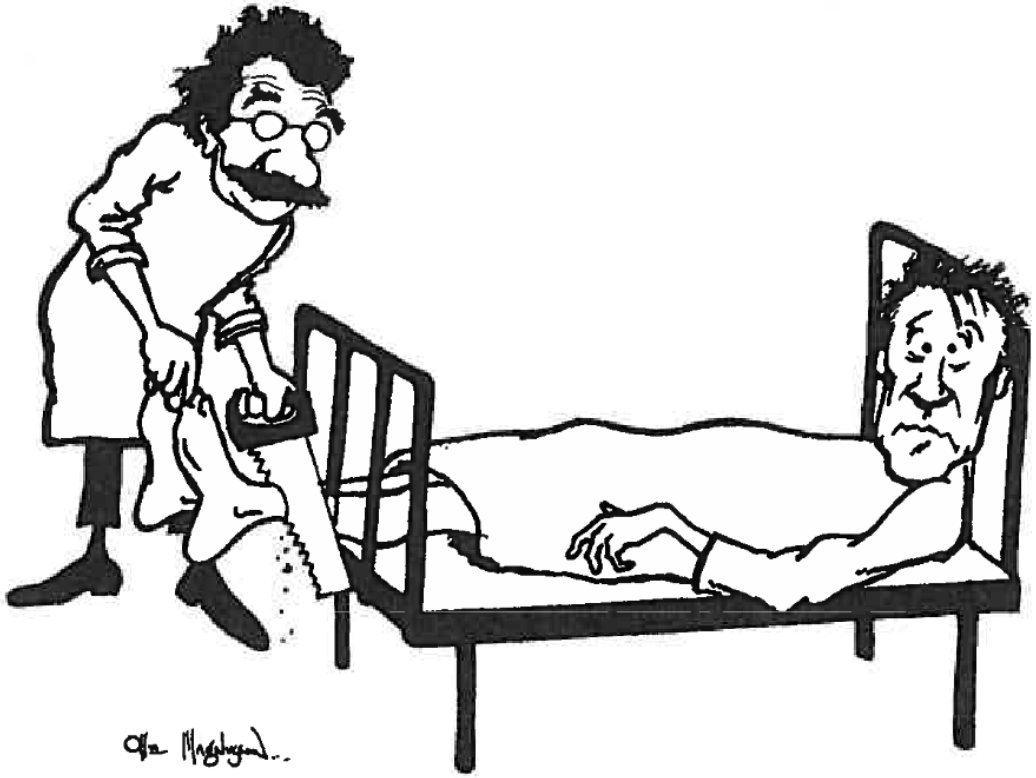
Figur 16.4: Faran att extrapolera modeller.

Models are not the 'real world' (The Pygmalion-effect)



Figur 16.5: Bli inte förälskad i modellen.

Do not force reality into the constraints of a model (The Procrustes effect)



Figur 16.6: Försök inte anpassa verkligheten till modellen.

Keep an engineering sense about your modeling process

Figur 16.7: Förkasta inte fakta som är i strid med modellen.

Figur 16.8: Använd gärna flera olika modeller.

Kinds of mathematical models

- Static vs dynamic
  - Does the model involve time?
- Continuous vs discrete

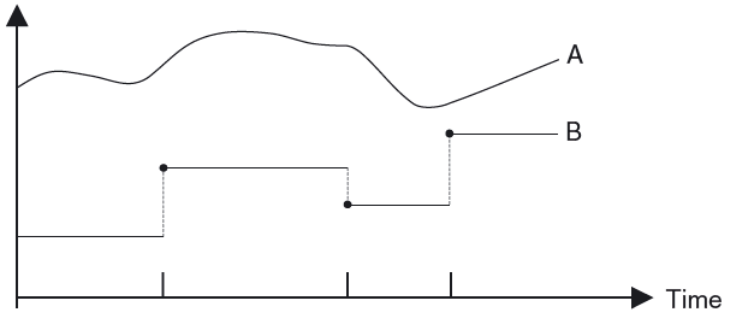


Figure 1.4 Discrete-time system B changes values only at certain points in time, whereas continuous-time systems like A evolve values continuously.

- Lumped vs distributed

$$\frac{\mathrm{d}}{\mathrm{d}t}T(t) = -aT(t)$$

$$\frac{\partial}{\partial t}T(t,x) = -\frac{\partial^2}{\partial x^2}T(t,x)$$

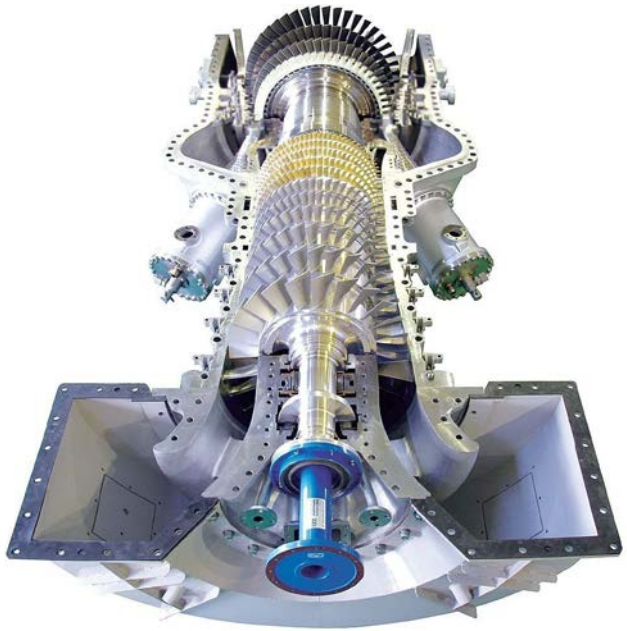
- Stochastic vs deterministic

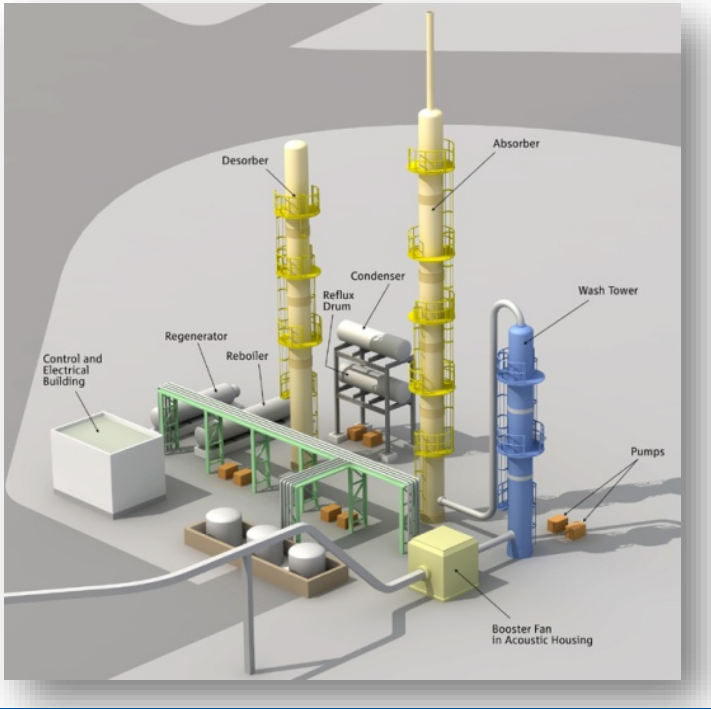
- Empirical vs 'first principles'
  - Build models from measurement data, or derive from laws of physics?
  - Or both: "Grey box"

Examples of modeling and simulation

Power

- Power productions
  - Gas turbines, hydro-electric power, bio, wind turbines, fuel cells, etc.
- Pollution&emissions
  - CO2 capture
- Smart grids





Gas turbine

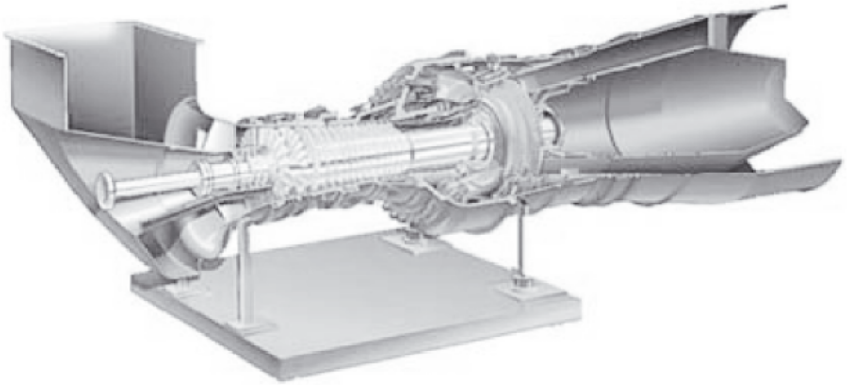


Figure 1.7 Schematic picture of the gas turbine GTX100. (Courtesy Siemens Industrial Turbomachinery AB, Finspång, Sweden.)

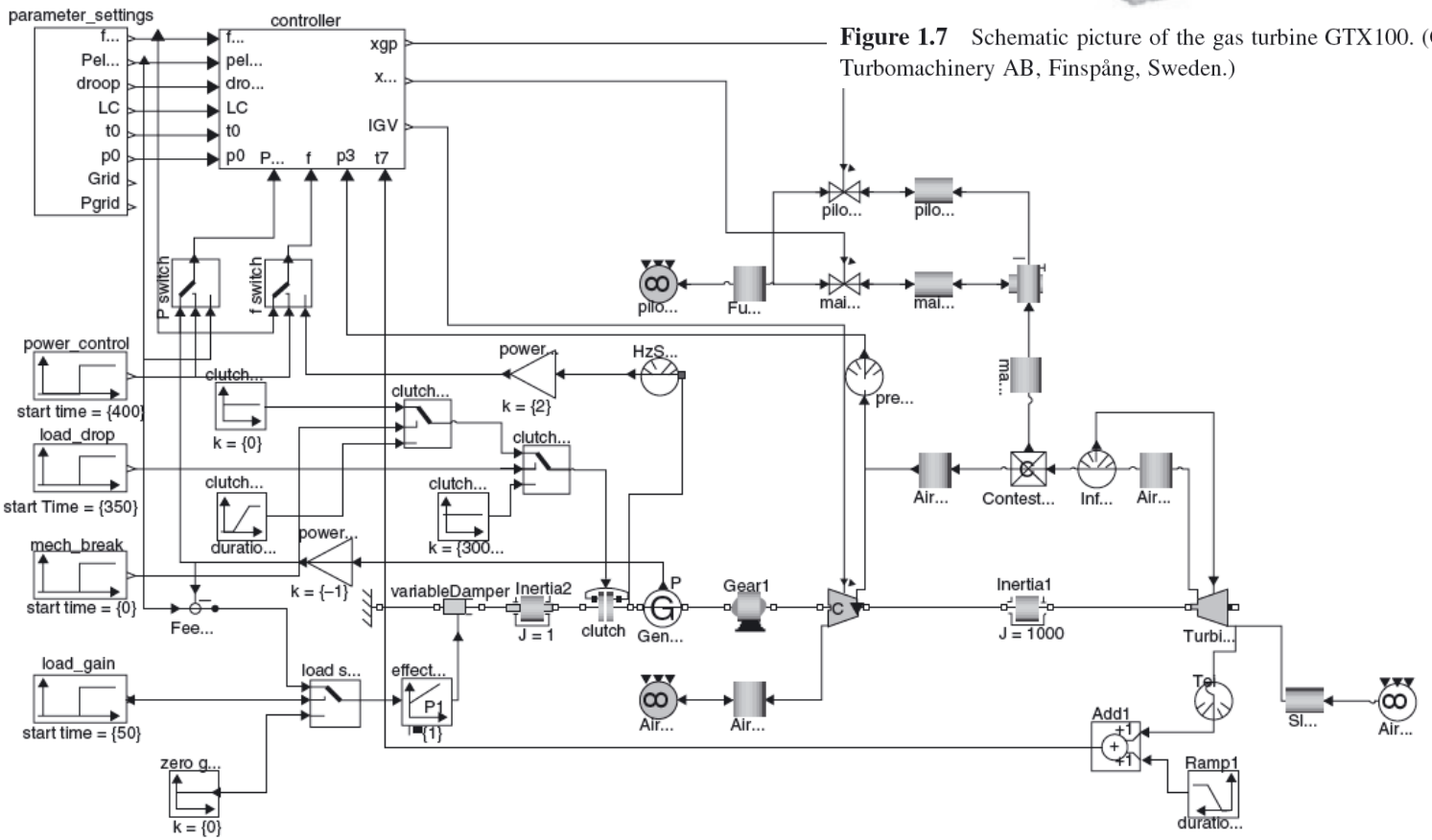
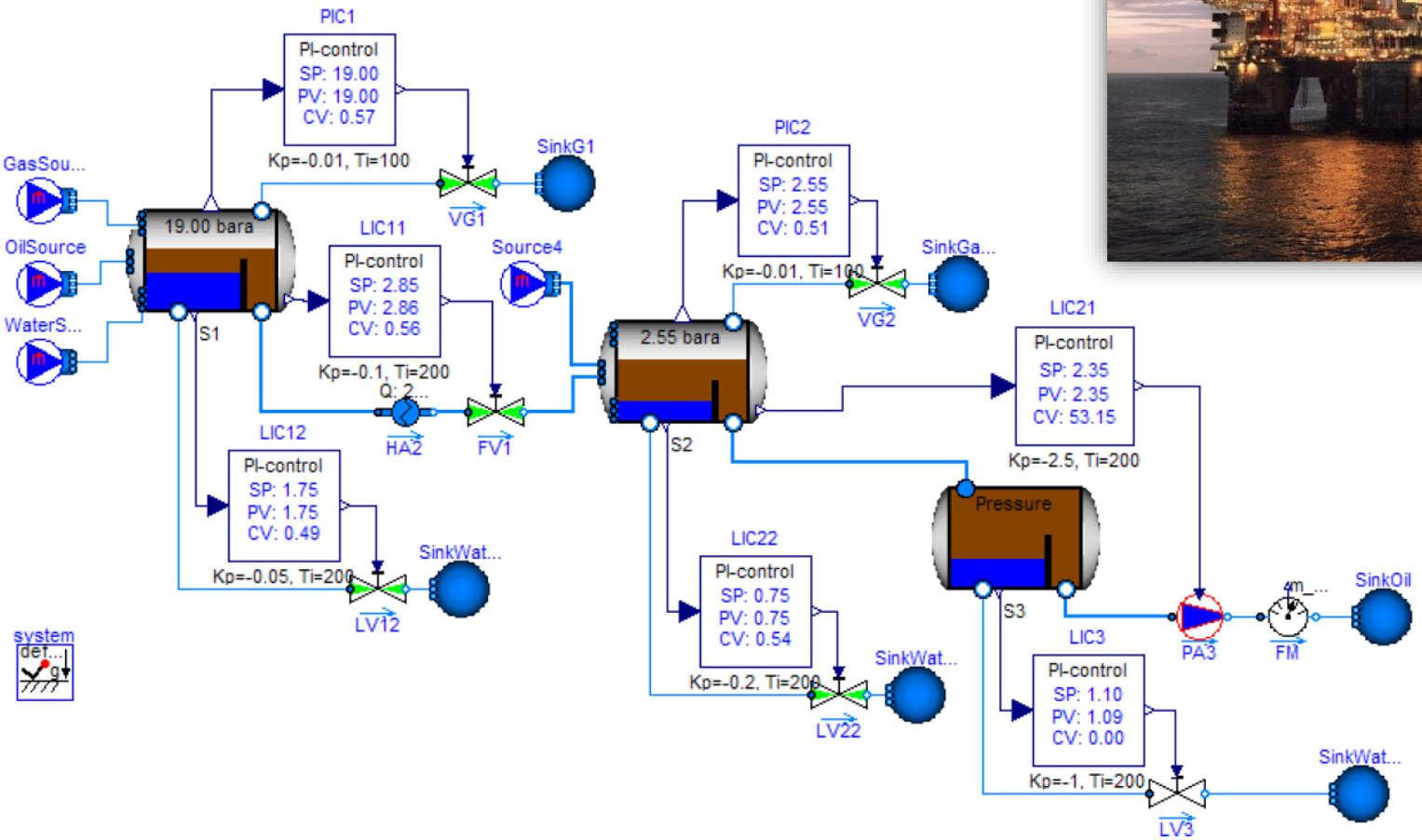


Figure 1.8 Detail of power cutoff mechanism in 40 MW GTX100 gas turbine model. (Courtesy Siemens Industrial Turbomachinery AB, Finspång, Sweden.)

Offshore production of oil and gas

- Production facilities
  - Separation, compression, routing, ...
- Subsea production, remote&arctic production
- Automation in offshore drilling

Offshore oil and gas processing (topside)



Courtesy: Cybernetica AS

Petrochemical industry

- Refineries, oil&gas processing
- Polymer production
- Methanol production

Production of metals

- Aluminium, ferrosilicon, ferromanganese, silicon, ...
- Si-wafers for solar cells

Navigation and control

- Underwater, ships, cars, air
- Unmanned vehicles
- Robots
- Mechatronics

Courtesy: Maritime Robotics

Courtesy: Prox Dynamics

Active safety in cars

- Control systems that prevent accidents
- Examples:
  - Anti-lock braking systems (ABS)
  - Anti-skid (ESC)
  - Collision avoidance

Uten ESC:

Med ESC:

Lecture 2:

- About modeling and simulation (F1)
- Why&what do control engineers need to model?
  - Examples (mostly examples I have worked with)
- Model types (E1.1-1.3,E2.1-2.2)
  - State space models, transfer functions
  - Linear models, nonlinear models