Lecture 5:

Two different classes of modeling tools

Modeling of complex systems – loudspeaker example (F4)

Introduction to simulation of continuous-time models (E14.2, E14.3.1)

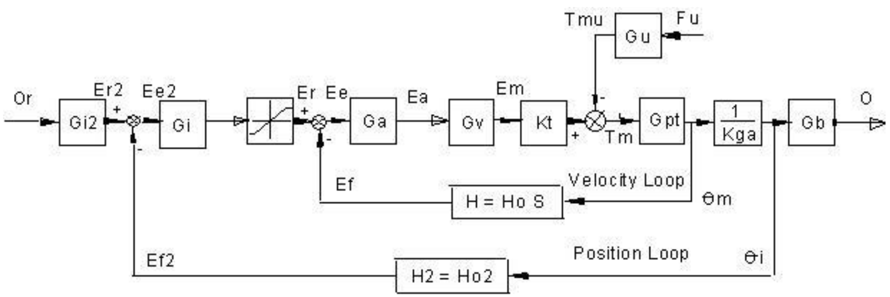
- Notation, computation errors
- Test system and stability function
- Euler's method

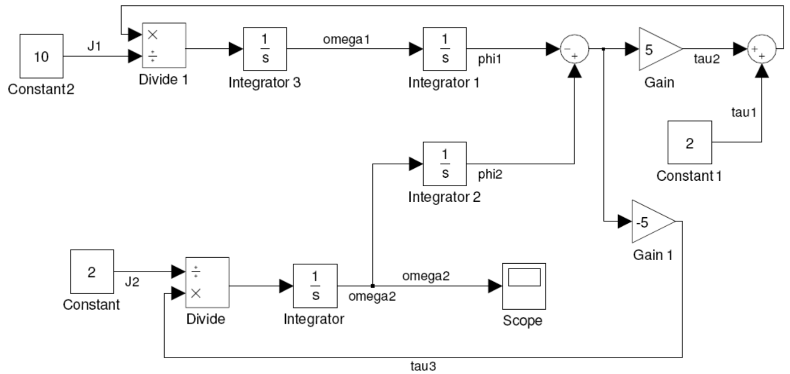
Two different classes of modeling tools

Signal-flow modelling

In cybernetics, we are often using *signal-flow* models

- Sub-models are 'blocks' with inputs and outputs
- Communication between blocks are signals with direction (arrows)
- Examples:
  - Block diagrams (signals are inputs/controls, outputs/measurements, references/set-points, etc.)
  - Simulink



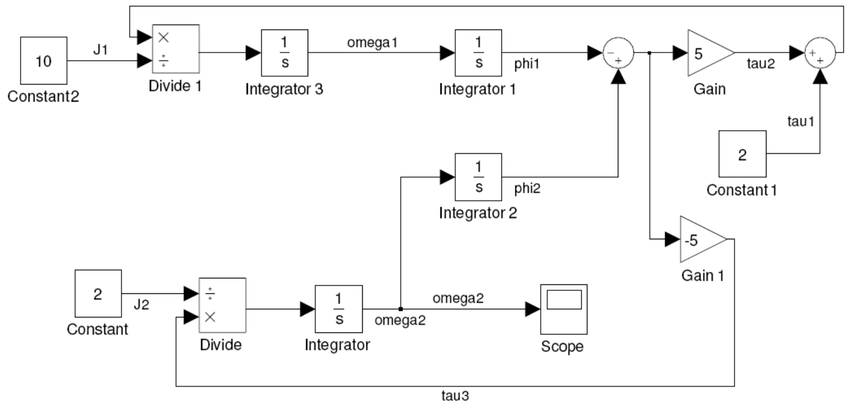


- Other name: Causal modelling

Signal-flow modelling

- Signal-flow models are well suited for control design and dynamic analysis
  - Intuitive(?) to implement, easy to simulate
  - Similarities with control methods taught in basic control courses
    - Transfer functions, Bode plots, Nyquist plots, etc.
    - In some respects: Passivity analysis, energy-based methods, ...

- Signal-flow models are not well suited for building ”large”/”complex” simulation-models
  - Constructing large models can become complicated due to little physical structure
  - It can be difficult to make changes
  - It can be difficult to re-use models



The other alternative: Object-oriented modelling

- Variants (also known as):
  - Equation-based modelling
  - Network description (from electrical circuit theory)
  - Energy-flow modeling (used in the book E)
  - Component based modelling, Acausal modelling, ...

- Technologies
  - Bond graphs
  - Modeling languages & tools
    - Modelica (Dymola, OpenModelica, MapleSim, ...), gPROMS, Ascend, ...

- Sub-systems interconnected via physically motivated interfaces
  - Easy to replace sub-systems
  - Easy to re-use models (build model libraries)
  - Graphical user interfaces
    - Can build models by drag-and-drop
    - Models are (to some extent) self-documenting

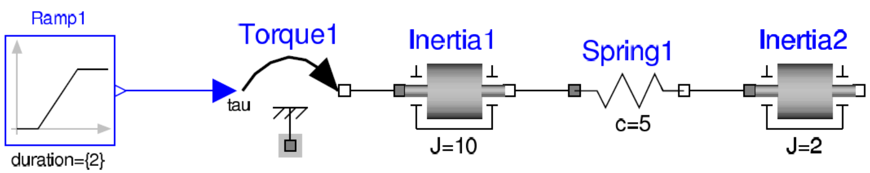
Tools for simulation

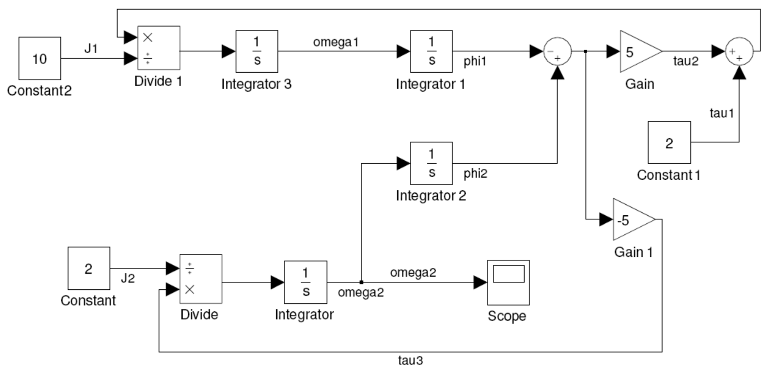
Signal-flow modelling

- Simulink
- Simulink ”clones”
  - e.g. SciCos, Xcos
- SystemBuild
  - (NI/Labview, MATRIXx)
- …

Object-oriented modelling

- Modelica (language)
  - Tools: Dymola, MapleSim, SimulationX, OpenModelica, Wolfram SystemModeler, …

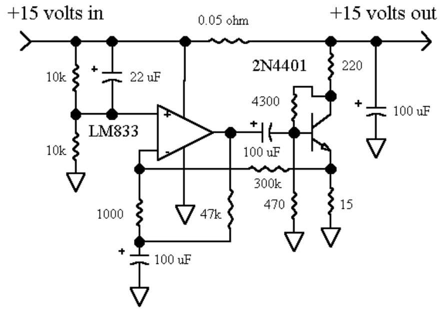


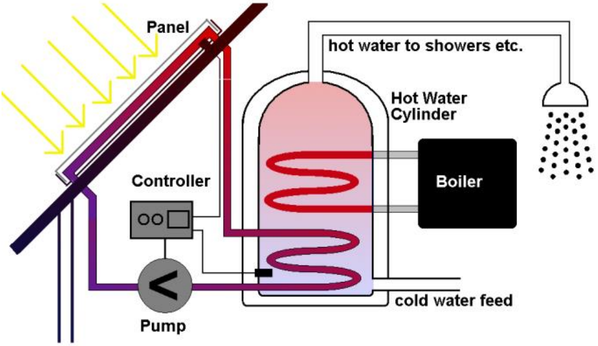


- Others
  - gPROMS
  - ASCEND
  - Aspen HYSYS Dynamics
  - Matlab Simscape
  - …

How should sub-systems be connected?

- How does physical sub-systems interact?
  - DC motor and load
  - Resistor and a capacitor
  - Heating water
  - Hydraulics on an excavator





- Change requires energy/power!
- Need information about two variables to decide energy/power transfer:
  - Force and velocity, or torque and rotational velocity (mechanical systems)
  - Voltage and current (electrical systems)
  - Enthalpy/temperature and mass flow (thermal systems)
  - Pressure and volume flow (hydraulic/flow systems)

Potential and flow variables

- Two variables needed to model **flow of energy** between two sub-systems
  - Potential variables: Variables that should be equal at interconnection (ex: voltage)
  - Flow variables: Variables that should sum to zero at interconnection (ex: current)

Domain	Potential	Flow
Translational mechanics	Velocity [m/s]	Force [N]
Rotational mechanics	Angular velocity [1/s]	Torque [Nm]
Electrical	Voltage [V]	Current [A]
Magnetical	Magnetomotive Force [A-turn]	Magnetic flux rate [Wb/s]
Hydraulical	Pressure [Pa]	Volume flow rate [m^3/s]
Thermal	Temperature [K]	Heat flow rate [J/Ks]
Chemical	Chemical potential [J/mol]	Molar flow rate [mol/s]

- Product of Potential and Flow variables is power transfer!
  - Other choices of potential and flow variables are possible, as long as power can be inferred

- Note: Book E uses other terminology and call the variables effort and flow (from bond graph-theory). But essentially the same concept.

In exercise 2:

- Attempt to use physically motivated interfaces in Simulink!
  - Advantage: You get Simulink blocks that are easy (easier) to reuse
  - Disadvantage (with Simulink):
    - No ”language constructs” forcing/helping you to use “right” port variables
    - Signal flow direction must be decided beforehand

- Then do the same with Modelica/Dymola

MODELING OF COMPLEX SYSTEMS – LOUDSPEAKER EXAMPLE

Modeling of complex systems

- Many (modern) technical systems are
  - complex (composed of many components)
  - multi-domain (electrical, mechanical, . . .)
- Often non-trivial to go from physical process to mathematical model
  - what to model and what to not
  - appropriate abstractions & simplifications, relevant physics
- Need a modeling methodology!

Multidomain modeling example: Loudspeaker

Task: Model the relationship between applied voltage and resulting sound pressure

(Example taken from H. Hjalmarsson, KTH)

Modeling methodology

- Structure the problem
  - Understand how the system works!
  - What is the purpose of the model?
  - Decompose the model hierarchically into sub-systems (if the system is «large»)

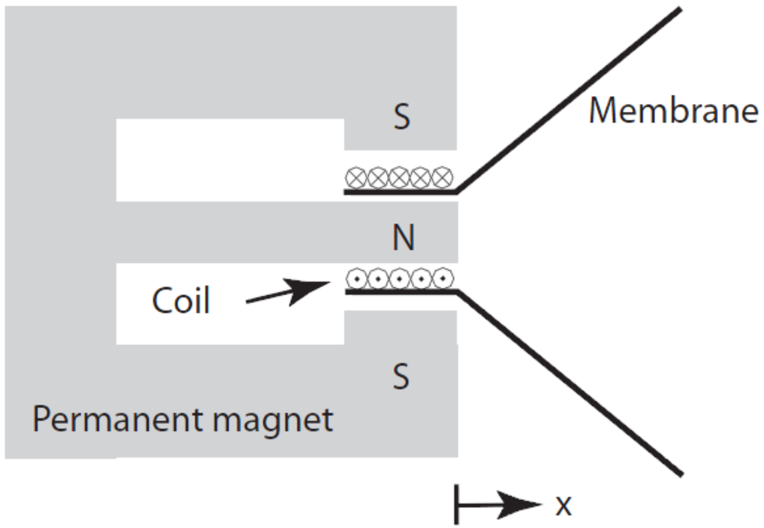
- Define models for sub-systems, and interactions
  - Based on physics and domain knowledge

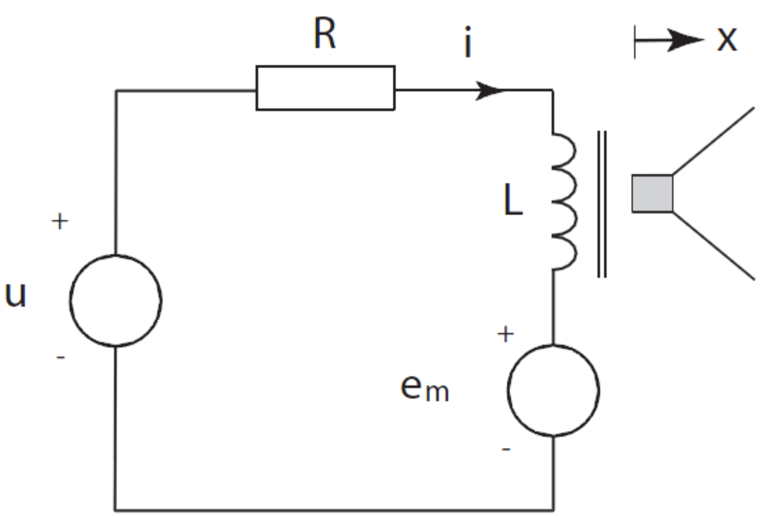
- Organize equations
  - State-space form (ODE, DAE or PDE)

- Validate and simplify

Structuring the problem

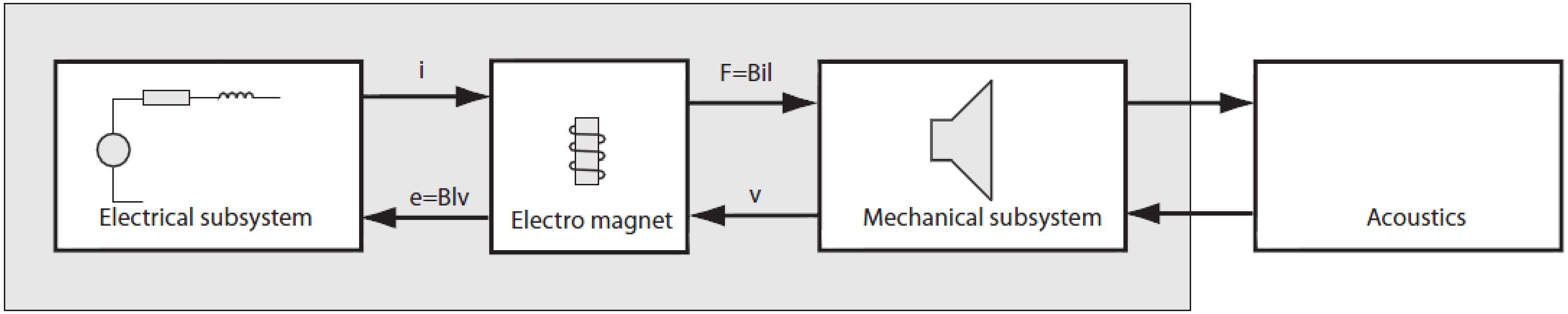
- How does a loudspeaker work?





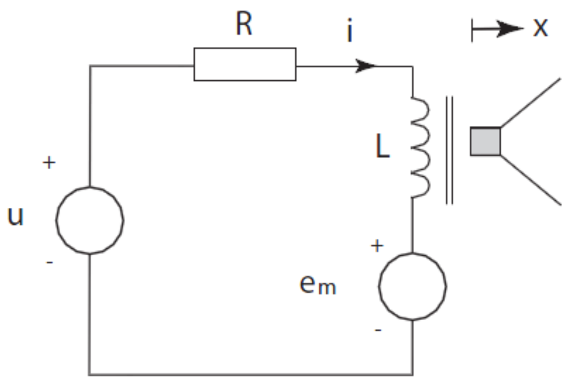
Structuring the problem

- Draw simple pictures and diagrams



- Try to reflect logical relations, physical principles, cause-and-effect

Modeling of subsystems



Electrical subsystem:

$$u - Ri - L\frac{\mathrm{d}i}{\mathrm{d}t} - e_m = 0$$

Electromagnetic subsystem:

$$e_m = Blv$$
$$F_m = Bil$$

Mechanical subsystem:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = F_m - dv - kx$$

Organizing the equations

Goal: Have the model equations in useful form (for analysis, simulation, ...)

…typically, in state space form

- Can be tedious and time-consuming, but good software support exists!
  - For example, Dymola does this automatically for overall system

- Transforming model into state-space form
  1. Choose state variables
  2. Express time derivatives of states in terms of states and inputs
  3. Express outputs in terms of states and inputs

Implement model in computer

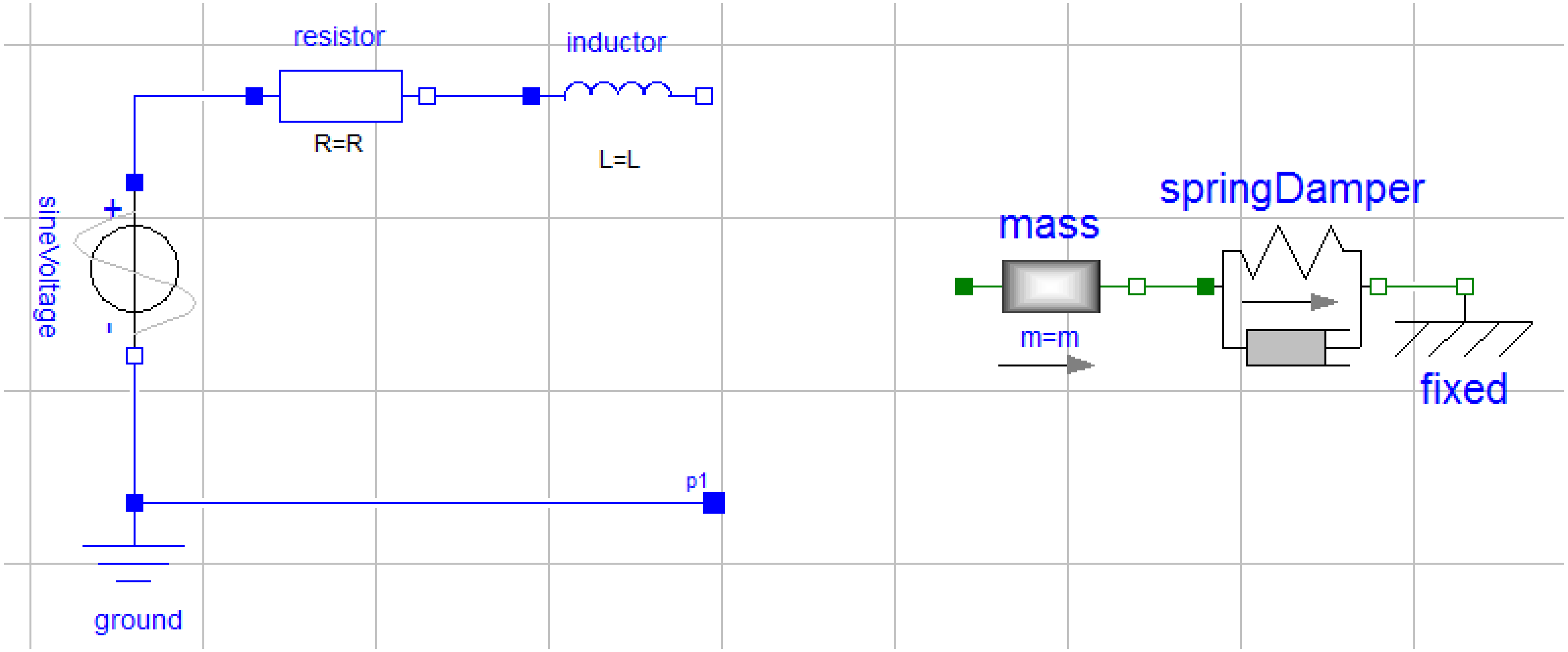
- Choose software
  - Low-level, high-level: C, Fortran, Matlab, Simulink, Modelica, …
  - Higher-level software increases reusability, correctness, implementability

- To save time and avoid unnecessary errors: Always re-use models as much as you can
  - Use existing (verified) library model components, or
  - Modify and/or improve library model components

- Develop new model components only when this is not feasible
  - And consider making this a new library model (do the extra effort)

Loudspeaker model in Modelica

- Most of the model can be constructed using components from Modelica Standard Library



- We need only to construct a component for the electro magnet

Modelica model of electro magnet

- Need both electrical and mechanical (translational) connectors

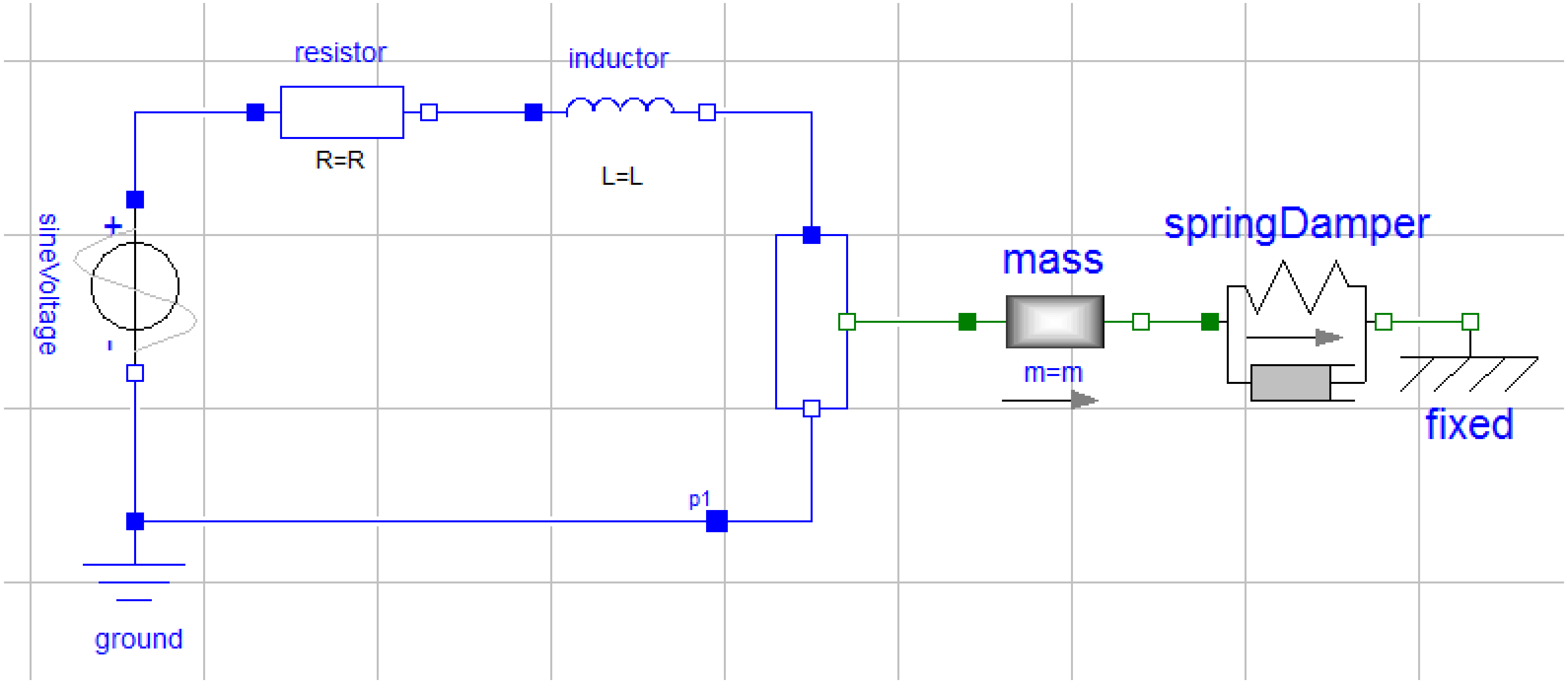
```
connector Pin
  Voltage      v;
  flow Current i;
end Pin;

connector Flange
  Position   s "Absolute position of flange";
  flow Force f "cut force directed into flange";
end Flange;
```

Modelica model of electro magnet

```
model ElectroMagnet
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
        // OnePort defines variables v and i
  Modelica.Mechanics.Translational.Interfaces.Flange_b flange;

  parameter Modelica.SIunits.MagneticFluxDensity B = 1;
  parameter Modelica.SIunits.Length l = 1;

  Modelica.SIunits.Velocity velocity;
  Modelica.SIunits.Force F;

equation
  // Model
  v = B*l*velocity;  // Voltage
  F = B*i*l;  // Force

  // Set variables in connectors
  velocity = der(flange.s);
  flange.f = F;
end ElectroMagnet;
```

Note use of Modelica.SIunits!

Complete loudspeaker model



Validation of model

- Best: validate against real data (experiment)

- Soft validation also useful:
  - Verify that equations are dimensionally correct

    Example Bernoullis law $v = \sqrt{2gh}$.

    $$[v/\sqrt{gh}] = [ms^{-1}(ms^{-2}m)^{-1/2}] = m^0 s^0$$

  - verify that equations describe qualitatively correct behavior
    - stationary points, static relations
    - linearized dynamics (time constants)

Modeling purpose and complexity

- The purpose of the model determines its complexity
  - simple models often enough for order-of-magnitude estimates or “adequate” control performance
  - more detailed models may be required for critical design decisions or high performance designs

- Remember:
  - all models are approximate (in practice)
  - a highly accurate sub-model may be of limited value if other sub-models are inaccurate
  - a good model is simple, yet captures the essentials!
    - Einstein: “…*as simple as possible*, *but not simpler”*

Simplification in modeling

Many approaches, for example

- Neglect minor effects
- Use idealized relationships
  - e.g., ideal gas law, non-compressible fluids, …
- Aggregation of state variables
  - Approximate infinite-dimensional variables (systems of PDEs) by one or more «lumped» variables (systems of ODEs)
  - Example
    - The water temperature in a tank described by a single (average) temperature
    - The pressure in a long pipeline described by a single pressure (or a few pressures along the line)
- **Separate time constants**
  - Perhaps especially important for control applications

Separation of time constants

Focus modeling effort on dynamics whose time constants are relevant for the intended purpose of the model

- subsystems with fast dynamics are approximated as static relations
- variables that vary slowly are approximated by constants

This gives models that are easier to manipulate and simulate

For example, in control applications:

- To verify stability (and performance) of a design, dynamics in bandwidth frequency range most important (remember Bode)
- Slow dynamics are not important for control *if its impact can be measured* (then integral control takes care of it anyway)

Lecture 5:

Two different classes of modeling tools

Modeling of complex systems – loudspeaker example (F4)

Introduction to simulation of continuous-time models (E14.2, E14.3.1)

- Notation, computation errors
- Test system and stability function
- Euler's method

Why learn about simulation methods?

1. You will need to implement your own solvers
   – What solver fits my problem, what time-step should I choose?
   – Primarily: Explicit solvers

2. You will need to make qualified choices of solvers when using advanced modeling software
   – What solver fits my problem, choice of accuracy?
   – Typically: Implicit solvers with varying time-steps

   – Examples:
     • Simulink: Three-body problem, satellite in combined moon and earth gravity field (orbit.mdl, ode45 vs ode1 (Euler))
     • Dymola

Initial value problem

Computational error

Method: One step method

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Order of a one step method:

A method is of order *p* if *p* is the smallest integer such that:

$$e_{n+1} = O(h^{p+1})$$

Order: Taylor series expansion

Linearization

(14.2.4)

- System $\dot{y} = f(y,t)$, $y = (y_1, \ldots, y_d)^{\mathsf{T}}$
- Linearize around operating point y^*: $\Delta\dot{y} = J\Delta y, \quad J = \left.\frac{\partial f}{\partial y}\right|_{y=y^*}$
- Diagonalize: $Jm_i = \lambda_i m_i, \quad \text{where} \quad \begin{cases} m_i : \text{eigenvectors of } J \\ \lambda_i : \text{eigenvalues of } J \end{cases}$
- Define $q = M^{-1}\Delta y$:

$$\dot{q} = M^{-1}J\Delta y = M^{-1}JMq = \Lambda q, \qquad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix}$$

- That is, $\dot{q}_i = \lambda_i q_i$ from which we can find $\Delta y(t) = Mq = \sum_{i=1}^{d} q_i(t) m_i$

We can study properties of a method used to simulate the system $\Delta\dot{y} = J\Delta y$, by study properties of the method for the systems $\dot{q}_i = \lambda_i q_i, \quad i = 1, \ldots, d$.

Homework

- Try to implement the Loudspeaker in Modelica
- Read 14.2 (covers second part of lecture)
- Read 14.3 (Euler's method)

Next lecture ….

Example linearization

- System:

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -y_1^3 - c y_2$$

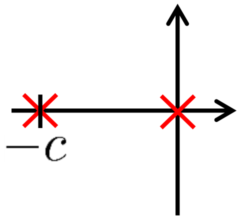
Linearization about $(y_1^*, y_2^*)^\mathsf{T}$:

$$\begin{pmatrix} \Delta \dot{y}_1 \\ \Delta \dot{y}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -3\left(y_1^*\right)^2 & -c \end{pmatrix} \begin{pmatrix} \Delta y_1 \\ \Delta y_2 \end{pmatrix}$$

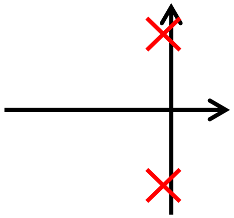
- Eigenvalues:

$$\lambda^2 + c\lambda + 3\left(y_1^*\right) = 0 \qquad \lambda_{1,2} = -\frac{c}{2} \pm \sqrt{\left(\frac{c}{2}\right)^2 - 3\left(y_1^*\right)^2}$$

$$y_1^* = 0: \quad \lambda_1 = 0,\ \lambda_2 = -c$$



$$y_1^* = \text{large}: \quad \lambda_{1,2} \to \pm j\omega_0$$



Test system, stability function

- One step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n)$$

- Apply it to scalar test system:

$$\dot{y} = \lambda y$$

- We get:

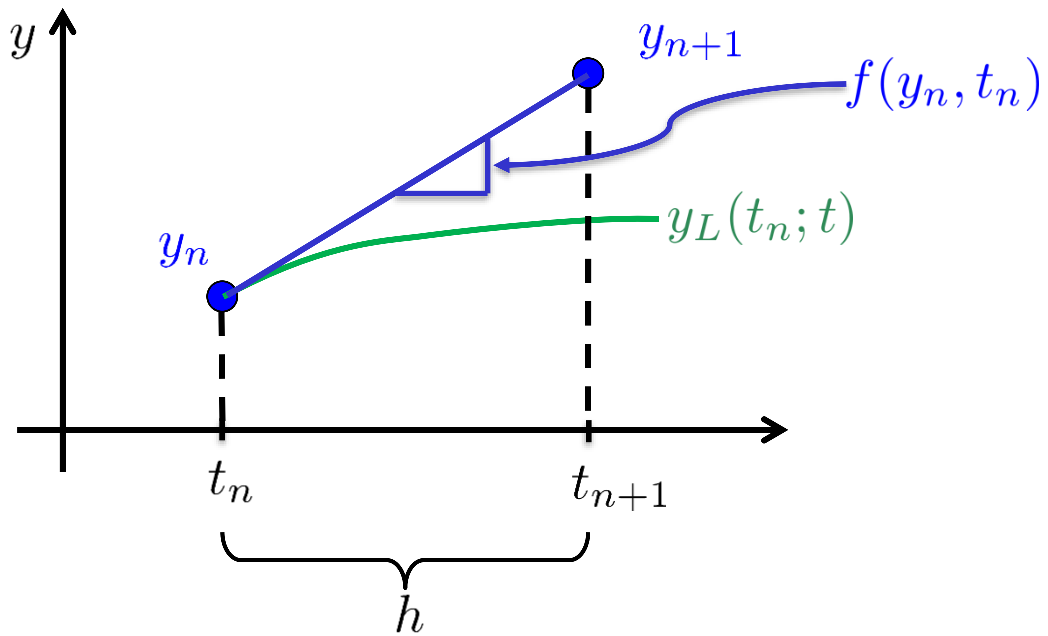
$$y_{n+1} = R(h\lambda)y_n$$

  where $R(h\lambda)$ is stability function

- The method is stable (for test system!) if $|y_{n+1}| \leq |y_n|$

$$|R(h\lambda)| \leq 1$$

Simplest method: Euler



- Slope:

$$\frac{y_{n+1} - y_n}{h} = f(y_n, t_n)$$

- Euler's method: $y_{n+1} = y_n + hf(y_n, t) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n, t)}{\mathrm{d}t} + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}f(y_n, t)}{\mathrm{d}t^{p-1}} + O(h^{p+1})$

$$y_{n+1} = y_n + hf(y_n, t_n)$$

Simplest method: Euler

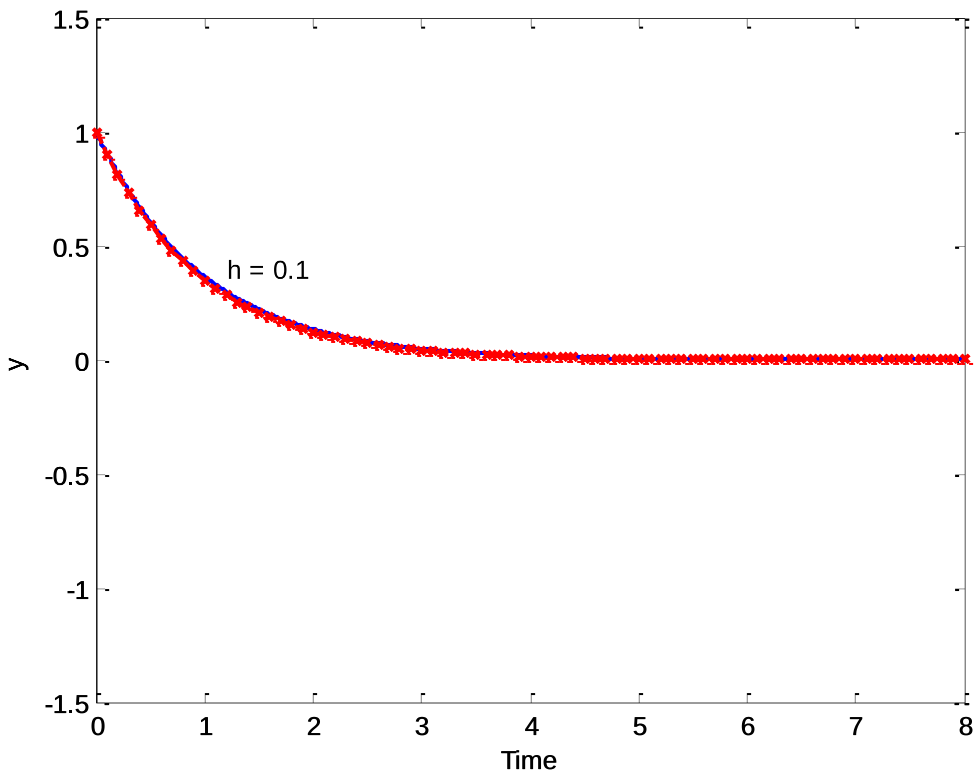
Example Euler's method

ODE: $\dot{y} = -y, \quad y(0) = 1$

Euler simulation: $y_{n+1} = y_n + h\left(-y_n\right), \quad y_0 = 1$

Example, $h = 0.1$:

n	t_n	y_n
0	0	1
1	0.1	
2	0.2	
3	0.3	
4	0.4	
…	…	…



Order (accuracy)

- Given IVP:

$$\dot{y} = f(y,t), \quad y(0) = y_0$$

- One-step method:

$$y_{n+1} = y_n + h\phi(y_n, t_n), \quad h = t_{n+1} - t_n$$

- If we can show that

$$y_{n+1} = y_n + hf(y_n, t) + \frac{h^2}{2}\frac{\mathrm{d}f(y_n,t)}{\mathrm{d}t} + \ldots + \frac{h^p}{p!}\frac{\mathrm{d}^{p-1}f(y_n,t)}{\mathrm{d}t^{p-1}} + O(h^{p+1})$$

- Then:
  - Local error is $O(h^{p+1})$
  - Method is order p

Example Euler's method

ODE: $$\dot{y} = -y, \quad y(0) = 1$$

Euler simulation: $$y_{n+1} = y_n + h\left(-y_n\right), \quad y_0 = 1$$

Stability: $$|R(h\lambda)| = |1 - h| \leq 1 \Rightarrow 0 \leq h \leq 2$$