TTK4225 - Systems Theory, Autumn 2020

Damiano Varagnolo

NTNU

Table of Contents I

Course Overview

welcome to TTK4225!

Instructors

- main teacher: Damiano Varagnolo, `damiano.varagnolo@ntnu.no`, room D233
- theaching assistant: Iver Andreas Ugelvik, `iverau@stud.ntnu.no`

Where is the information?

- Blackboard
- study guide
- GitHub repository
- mediasite.ntnu.no

Reference group – VERY IMPORTANT

- at least 3 students
- will do 4 meetings (1 after the exam)
- shall represent the whole class $\Longrightarrow$ you will have meetings among yourselves too
- shall lead to a referansegrupperapport containing suggestions for improvements

Expectations, teacher side

- be proactive
- be respectful
- be collaborative
- eventually, learn stuff

Prerequisites

- basic physics
- basic linear algebra
- basic analysis

- better if you can Matlab and Python

this course teaches the foundations of automatic control. Not mastering this course $\Longrightarrow$ failing to understand all the next courses

Expectations, learners side

?

Implementation leitmotif

maximize the interactions among us

$\Longrightarrow$ things that you can do by yourself you do by yourself

Recommendations

- read the study guide
- do the assignments in the suggested order & times
- signal bugs / imprecisions in the material
- charge the phone + bring pen and paper + bring your laptop

Structure of each lesson

1. frontal guidance
2. peer instructions
3. Jupyter coding

Peer instructions

- purpose = be active
- algorithm = for each question:
  1. first time, answer individually
  2. then form groups, discuss, try to convince each other, but eventually answer individually
  3. then I show the solution
  4. after that you pose questions

Peer instructions - test

`https://play.kahoot.it/v2/lobby?quizId=bef23121-b1a6-4cce-8565-b9631387bbbf`

Jupyter notebooks

`http://localhost:8888/notebooks/GitHub/TTK4225/trunk/Jupyter/odeint-example.ipynb`

Jupyter notebooks - instructions for the next lesson

1. download the notebooks in `https://github.com/damianovar/TTK4225-2020`
2. install *Jupyter notebooks* or *Anaconda*
3. launch and test `TTK4225/trunk/Jupyter/odeint-example.ipynb`

"At-home" self-assessment activities

1. contents mapping
2. Blackboard questionnaires

Contents mapping

why: enable you visually assess if you got the logical structure of the course contents

what: gamified mapping of the content units in the course

how: python script downloadable from Blackboard TODO

when: whenever you prefer – this is a self assessment activity!

More info in the study guide!

Blackboard questionnaires

why: enable the continuous quantitative collection of how much the class is learning / adapt the course pace

what: private Blackboard questionnaires

how: set of questions indexed in terms of "what is asked" and "how difficult the question is"

when: whenever you prefer – this is a self assessment activity!

More info in the study guide!

leitmotif: we want to help you, but we need you to help us

What about the CoVid-19?

Cornerstones:

- we may need to switch to fully digital teaching as in this past spring
- independently of this, who must be at home in quarantine is entitled to good teaching

self-assessments, peer-instructions, Jupiter notebooks, videotaping, etc = all elements to account for this

tips are most welcome

Course overview

Aim of the course

learn a language for describing phenomena
in an abstract and formal way,
that will be used later on
to design feedback and feedforward control schemes

What is control?

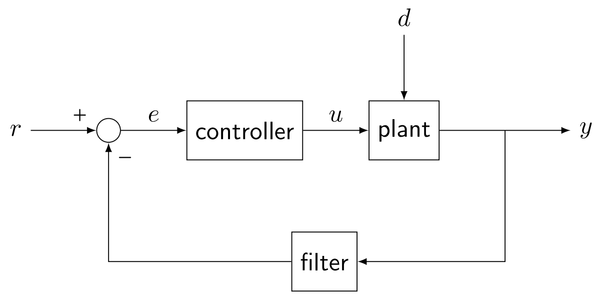
- adjusting the temperature in the shower
- riding a bike

What is automatic control?

applying mechanisms so to operate processes automatically,
so they don't need continuous direct human intervention

"remove the man from the loop"

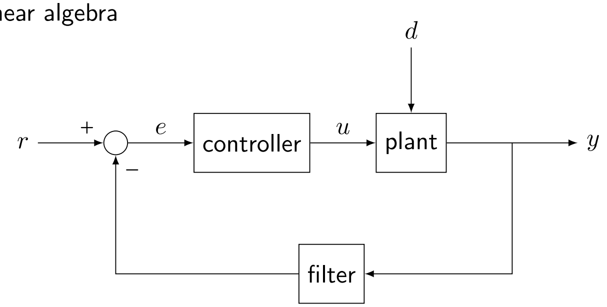
What is (feedback) control?



PS: this is not the only type of automatic control!

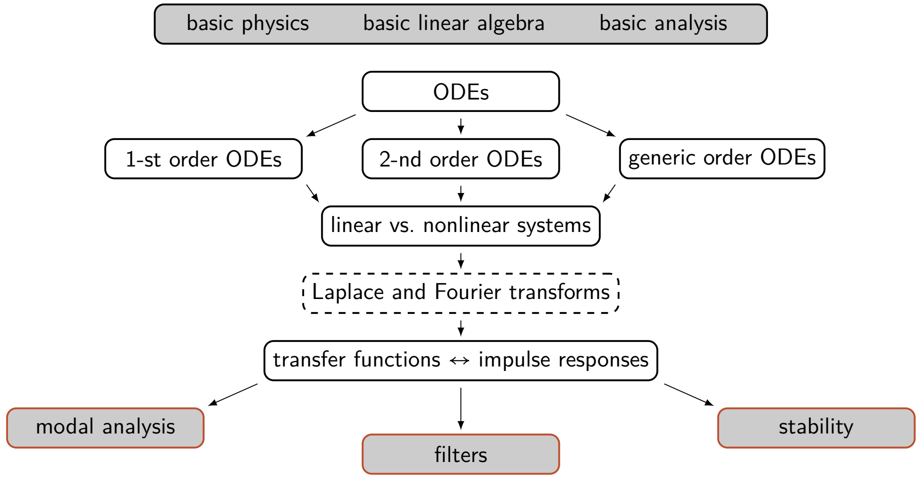
Intended Learning Outcomes, in brief:

- use differential equations to model continuous-time plants
- analyse the models in time and frequency domains
- learn to characterise important structural properties of these models
- learn a little bit about filters
- refresh linear algebra



why? need all these ingredients to do advanced control!

Summary / Roadmap of TTK4225



Tentative time line

GitHub repository

↓

`ContentsMap` folder

↓

`plan-and-content-units.xlsx` file

Introduction to modelling and dynamics

Roadmap

- what do we mean with
  - “dynamics”?
  - “modelling a dynamical system”?
  - “control-oriented modelling”?

What do we mean with “dynamics”? Historical origins

From mechanics

dynamics = physics concerned with the effects of forces on the motion of bodies

Example:

$$F = ma = m\frac{dv}{dt} = m\dot{v} \qquad (1)$$

Even better:

$$\dot{v} = \frac{F}{m} \qquad (2)$$

this is an ODE

What do we mean with “dynamics”? More in general

Our acception

dynamics = the study of the motion (i.e., temporal evolution) of the variables that characterize a system

Example: Lotka-Volterra:

- y_{prey} := prey
- y_{pred} := predator

$$\begin{cases} \dot{y}_{\mathrm{prey}} &= \alpha y_{\mathrm{prey}} - \beta y_{\mathrm{prey}} y_{\mathrm{pred}} \\ \dot{y}_{\mathrm{pred}} &= -\gamma y_{\mathrm{pred}} + \delta y_{\mathrm{prey}} y_{\mathrm{pred}} \end{cases}$$

`GitHub/TTK4225/trunk/Jupyter/Lotka-Volterra-introduction.ipynb`

What do we mean with “modelling a dynamical system”?

Example of a system from before:

$$\begin{cases} \dot{y}_{\mathrm{prey}} &= \alpha y_{\mathrm{prey}} - \beta y_{\mathrm{prey}} y_{\mathrm{pred}} \\ \dot{y}_{\mathrm{pred}} &= -\gamma y_{\mathrm{pred}} + \delta y_{\mathrm{prey}} y_{\mathrm{pred}} \end{cases}$$

More generic definition: modelling a dynamical system = defining

$$\dot{\boldsymbol{y}} = \boldsymbol{f}\left(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\theta}\right),$$

thus defining:

- the variables
  - $\boldsymbol{u}$ = inputs
  - $\boldsymbol{y}$ = outputs
- the structure of the function $\boldsymbol{f}$
- the value of the parameters $\boldsymbol{\theta}$

What do we mean with "modelling a dynamical system"?

Even more generic definition: defining a *state-space system*:

$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}\right) \\ \boldsymbol{y} &= \boldsymbol{g}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}\right) \end{cases}$$

thus defining:

- the variables
  - $\boldsymbol{u}$ = inputs
  - $\boldsymbol{x}$ = state
  - $\boldsymbol{y}$ = measured outputs
- the structure of the functions $\boldsymbol{f}$ and $\boldsymbol{g}$
- the value of the parameters $\boldsymbol{\theta}$

Discussion: did we model the Lotka-Volterra dynamical system here?

```
def myModel(y, t):
#
# parameters
alpha = 1.1
beta  = 0.4
gamma = 0.4
delta = 0.1
#
# get the individual variables - for readability
yPrey = y[0]
yPred = y[1]
#
# individual derivatives
dyPreydt  =   alpha * yPrey - beta  * yPrey * yPred
dyPreddt  = - gamma * yPred + delta * yPrey * yPred
#
return [ dyPreydt, dyPreddt ]
```


Discussion: do we need something else to simulate this system?

$$\begin{cases} \dot{y}_{\text{prey}} &= 1.2 y_{\text{prey}} - 0.1 y_{\text{prey}} y_{\text{pred}} \\ \dot{y}_{\text{pred}} &= -0.6 y_{\text{pred}} + 0.2 y_{\text{prey}} y_{\text{pred}} \end{cases}$$

Connecting with next courses

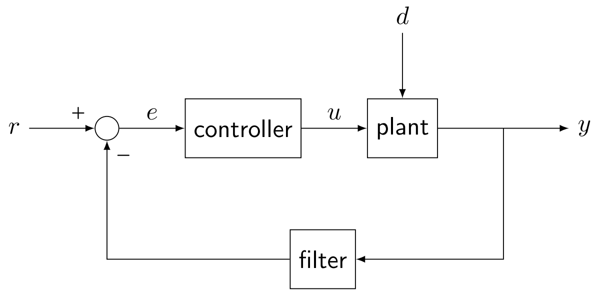
$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}\right) \\ \boldsymbol{y} &= \boldsymbol{g}\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}\right) \end{cases}$$

- how to estimate the structure of $\boldsymbol{f}$, $\boldsymbol{g}$ and the values of $\boldsymbol{\theta}$ = system identification
- how to estimate $\boldsymbol{x}(0)$ from $\boldsymbol{y}$ = state observers

Caveat: static ≠ dynamic

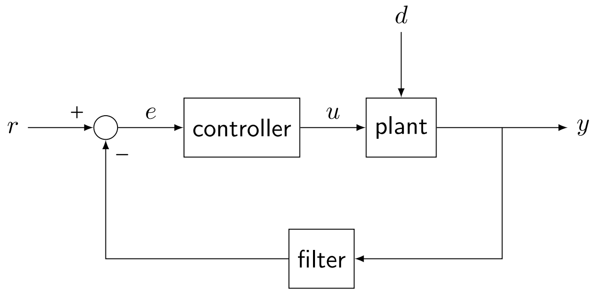
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{\theta}) \qquad \neq \qquad \dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\theta})$$

The workflow to control a dynamical system



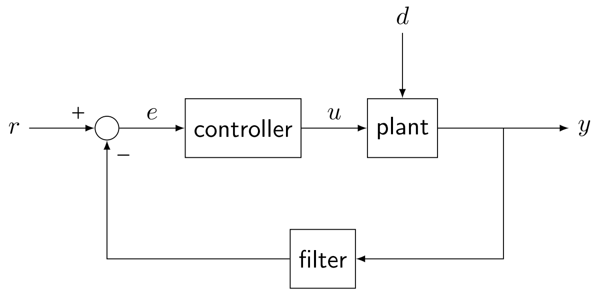
1. model the process
2. identify it *(sometimes not strictly necessary)*
3. design the controller

What does it mean to design a controller?



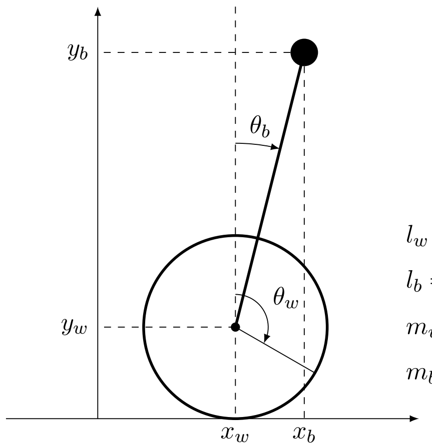
task: complete the following sentence: "In this case, the controller is $u = h(\_\_\_\_)$, and it shall be designed so that ____ is as desired, independently of ____"

What does it mean to control a dynamical system?



= *designing a controller* = *designing a function*

Another example: balancing robot; what are u, x and y in this case?



l_w = radius of the wheel

l_b = body-wheel's center distance

m_w = mass of the wheel

m_b = mass of the body

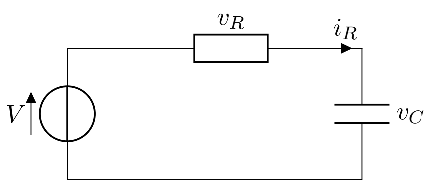
Modelling - simplest example: speed of a cart

- speed $v(t)$
- friction $F_f(t) = -kv(t)$
- force from the engine $F(t)$ *(this is our u!)*
- Newton's second law: $\sum$ forces $= ma(t)$

$$ma(t) = F_t(t) + F(t)$$
$$m\dot{v} = -kv(t) + F(t)$$
$$\Longrightarrow \dot{v} = -\frac{k}{m}v(t) + \frac{1}{m}F(t) \qquad (3)$$

Modelling - another example: RC-circuit

Example 2.2 in Balchen's book



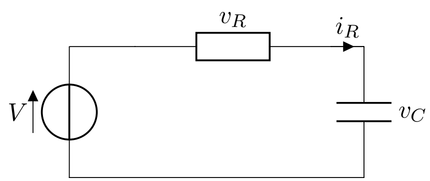
Question: what do we mean with dynamics in this case?

Remember our acception

dynamics = the study of the motion (i.e., temporal evolution) of the variables that characterize a system $\Longrightarrow$ modelling how tensions and currents evolve in time

Modelling - another example: RC-circuit

Example 2.2 in Balchen's book

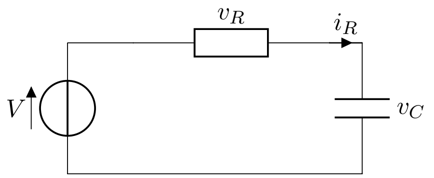


Question: what do we mean with modelling the dynamics in this case? We mean defining:

- the variables
  - $\boldsymbol{u}$ = inputs
  - $\boldsymbol{x}$ = state
  - $\boldsymbol{y}$ = measured outputs
- the structure of the functions $\boldsymbol{f}$ and $\boldsymbol{g}$
- the value of the parameters $\boldsymbol{\theta}$

Modelling - another example: RC-circuit

Example 2.2 in Balchen's book



How to obtain a dynamical model = use Kirchhoff's laws (the sum of the voltages in a circuit is zero)

$$\frac{v_R}{R} = i_R = i_C = C\frac{dv_C}{dt} \tag{4}$$

$$V(t) = v_R + v_C \Rightarrow v_R = V(t) - v_C \quad \Longrightarrow \quad C\dot{v}_C = \frac{V(t) - v_C}{R} \tag{5}$$

$$\Longrightarrow \dot{v}_C = -\frac{1}{RC}v_C + \frac{1}{RC}V(t) \tag{6}$$

Standing examples

Roadmap

Summary of the standing examples in this course:

- exponential growth
- RC-circuits
- RCL-circuits
- spring-mass system
- Lotka-Volterra
- Van-der-Pol oscillator
- balancing robot
- insuline concentration

Exponential growth

Generalization of "speed of a cart" and "RC circuit"

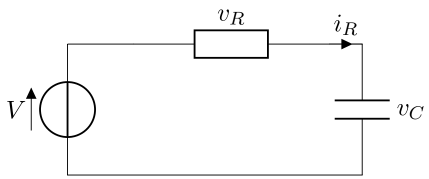
Scalar version:

$$\dot{y} = \alpha y + \beta u \tag{7}$$

Matricial version:

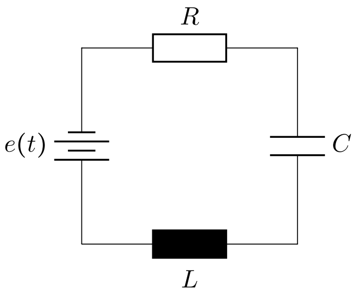
$$\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u} \tag{8}$$

RC-circuit



$$\dot{v}_C = -\frac{1}{RC}v_C + \frac{1}{RC}V(t) \tag{9}$$

RCL-circuit



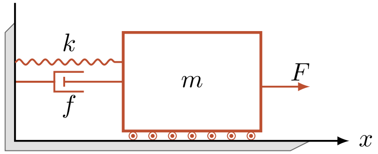
EOM: Kirkhoff laws

$$v_L(t) = L\frac{di(t)}{dt} \qquad v_i(t) = Ri(t) \qquad v_C(t) = \frac{1}{C}\int_0^t i(\tau)d\tau$$

$$e(t) = v_L(t) + v_R(t) + v_C(t) \quad \mapsto \quad e(t) = L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau \tag{10}$$

Spring-mass systems

E.g., position of a cart fastened with a spring to a wall and subject to friction



EOM:

- force from the spring: $F_x(t) = -kx(t)$
- friction: $F_f(t) = -f\dot{x}(t)$
- applied force: $F(t)$
- Newton's second law: $\sum F = m\ddot{x}(t)$

$$m\ddot{x}(t) = F_x(t) + F_f(t) + F(t) \quad \mapsto \quad m\ddot{x}(t) = -kx(t) - f\dot{x}(t) + F(t)$$

Lotka-Volterra

- y_{prey} := prey
- y_{pred} := predator

$$\begin{cases} \dot{y}_{\text{prey}} & = \alpha y_{\text{prey}} - \beta y_{\text{prey}} y_{\text{pred}} \\ \dot{y}_{\text{pred}} & = -\gamma y_{\text{pred}} + \delta y_{\text{prey}} y_{\text{pred}} \end{cases}$$

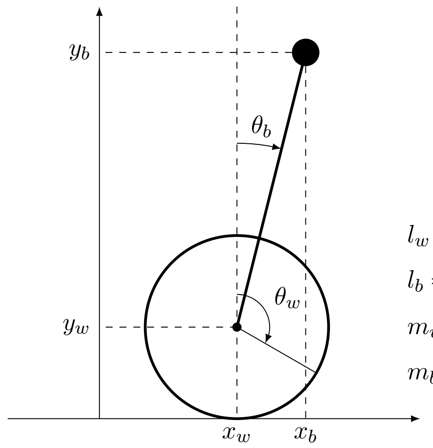
`GitHub/TTK4225/trunk/Jupyter/Lotka-Volterra-introduction.ipynb`

Van-der-Pol oscillator

$$\begin{cases} \dot{y}_1 &= \mu\left(y_1 - \dfrac{y_1^3}{3} - y_2\right) \\ \dot{y}_2 &= \dfrac{y_1}{\mu} \end{cases} \tag{11}$$

`GitHub/TTK4225/trunk/Jupyter/Van-der-Pol-introduction.ipynb`

Balancing robot



l_w = radius of the wheel

l_b = body-wheel's center distance

m_w = mass of the wheel

m_b = mass of the body

Balancing robot

$$\begin{aligned} \left(I_b + m_b l_b^2\right)\ddot{\theta}_b &= +m_b l_b g \sin(\theta_b) - m_b l_b \ddot{x}_w \cos(\theta_b) - \frac{K_t}{R_m} v_m + \left(\frac{K_e K_t}{R_m} + b_f\right)\left(\frac{\dot{x}_w}{l_w} - \dot{\theta}_b\right) \\ \left(\frac{I_w}{l_w} + l_w m_b + l_w m_w\right)\ddot{x}_w &= -m_b l_b l_w \ddot{\theta}_b \cos(\theta_b) + m_b l_b l_w \dot{\theta}_b^2 \sin(\theta_b) + \frac{K_t}{R_m} v_m - \left(\frac{K_e K_t}{R_m} + b_f\right)\left(\frac{\dot{x}_w}{l_w} - \dot{\theta}_b\right) \end{aligned} \tag{12}$$

Insulin concentration

- x_1 := sugar concentration
- x_2 := insulin concentration
- u_1 := food intake
- u_2 := insulin intake
- c := sugar concentration in fasting

$$\begin{cases} \dot{x}_2 = a_{21}(x_1 - c) - a_{22} x_2 + b_2 u_2 & x_1 \geq c \\ \dot{x}_2 = -a_{22} x_2 + b_2 u_2 & x_1 < c \end{cases}$$

$$\begin{cases} \dot{x}_1 = -a_{11} x_1 x_2 - a_{12}(x_1 - c) + b_1 u_1 & x_1 \geq c \\ \dot{x}_1 = -a_{11} x_1 x_2 + b_1 u_1 & x_1 < c \end{cases}$$

Seeing this course as a first step towards model predictive control

Roadmap

- how can we do control?
- what does it mean to do model predictive control?
- why are we doing this course?

control = to take out humans from the loop

(and do not waste time in doing things that machines could do)

Remember: control = design a function

$$\dot{\boldsymbol{y}} = \boldsymbol{f}\left(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\theta}\right)$$

where

- $\boldsymbol{f}$ and $\boldsymbol{\theta}$ = the model
- $\boldsymbol{y}$ the outputs
- $\boldsymbol{u}$ the inputs

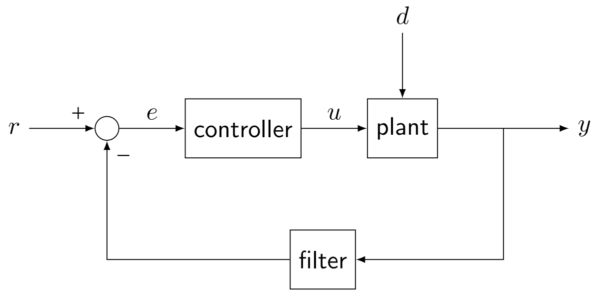
Two important paradigms

$$\dot{\boldsymbol{y}} = \boldsymbol{f}\left(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{\theta}\right)$$

model free control: $\boldsymbol{u}$ is built without knowing $\boldsymbol{f}$ and $\boldsymbol{\theta}$

model based control: $\boldsymbol{u}$ is built through knowing $\boldsymbol{f}$ and/or $\boldsymbol{\theta}$

The simplest controller: the P control



$$u(t) = \kappa e(t)$$

Discussion: we want to automate the control of the steering wheel while driving in a curve with a P controller.

1. What is u? What is r? What is e?
2. What does it imply to make κ bigger? And smaller?

P controlling a steering wheel

`GitHub/TTK4225/trunk/Jupyter/`
`first-order-system-introduction-with-P-controller.ipynb`

Discussion: what do you expect may happen if there is a spring somewhere?

important message: we can do 'model free' control

(and actually something like 90% of the controllers in the world are model-free...)

efficient, but kind of "dumb". We don't actually know what is happening!

Discussion: can we do something better if we know the model?

How can we do something better if we know the model?

1. if I know $\boldsymbol{f}$ and my current initial condition, I can plug in a tentative $\widehat{\boldsymbol{u}}$, and see what would be the corresponding $\widehat{\boldsymbol{y}}$
2. if I have a $\boldsymbol{y}_{\text{desired}}$, I may look for that tentative $\widehat{\boldsymbol{u}}$ that will make $\widehat{\boldsymbol{y}}$ as close as possible to $\boldsymbol{y}_{\text{desired}}$

$$\boldsymbol{u}^{\star} = \arg \min_{\boldsymbol{u} \in \mathcal{U}, \boldsymbol{f}(\boldsymbol{u}) \in \mathcal{F}} \text{Cost}\left(\boldsymbol{f}\left(\boldsymbol{u}\right), \boldsymbol{u}\right)$$

What does this require?

$$\boldsymbol{u}^{\star} = \arg \min_{\boldsymbol{u} \in \mathcal{U}, \boldsymbol{f}(\boldsymbol{u}) \in \mathcal{F}} \text{Cost}\left(\boldsymbol{f}\left(\boldsymbol{u}\right), \boldsymbol{u}\right)$$

- define "Cost"
- be able to compute $\boldsymbol{f}\left(\boldsymbol{u}\right)$ rapidly
- be sure that the model does not have "nasty" properties

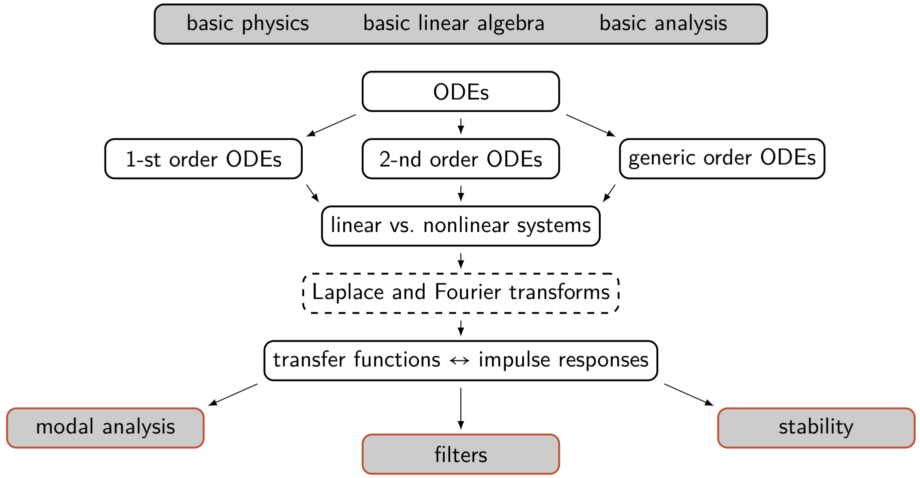
the most important point: to do model-based control we need to be able to simulate

this course = understanding the properties of the system,
& understanding how to simulate it with the purpose of doing advanced control

The control courses at NTNU ITK

- how do I compute explicitly $\boldsymbol{u}^\star$ in special cases?
- how do I compute numerically $\boldsymbol{u}^\star$ in general?
- how do I estimate $\boldsymbol{f}$ from data?
- how do I handle uncertainties?
- which properties can I guarantee?
- ...

Summary / Roadmap of TTK4225



Caveat! Better doing MPC than data-driven control:
Benjamin Recht (UC Berkeley, USA)
Reflections on the Learning-to-Control Renaissance

(and also `https://www.youtube.com/watch?v=nF2-39a29Pw`)

Introduction to first order systems - the concept of free evolution

Roadmap

- generalization of our examples of first order systems
- the concept of free evolution
- introduction to stability
- free evolution $\neq$ forced response

First order dynamical models

- train velocity

$$\dot{v} = -\frac{k}{m} v(t) + \frac{k}{m} F(t) \tag{13}$$

- RC-circuit

$$\dot{v}_C = -\frac{1}{RC} v_C + \frac{1}{RC} V(t) \tag{14}$$

In general:

$$\dot{y} = ay + bu \tag{15}$$

`GitHub/TTK4225/trunk/Jupyter/`
`first-order-system-introduction-with-P-controller.ipynb`

Overarching question towards model-based control: how does y evolve?

$$\dot{y} = ay + bu \tag{16}$$

Qualitative discussions:

- what happens if $u = 0$, $y_0 = 0$?
- what happens if $u = 0$, $y_0 > 0$, and $a > 0$?
- what happens if $u = 0$, $y_0 < 0$, and $a > 0$?
- what happens if $u = 0$, $y_0 \neq 0$, and $a < 0$?
- what happens if $u = 0$, and $a = 0$?

stability = extremely important topic!

will be analysed in more details later on in this course
and much more extensively in others
(feat. Lyapunov, Krasovskii, La-Salle among others)

An important case: free evolution

$$\dot{y} = ay \cancel{+bu} \qquad y(0) = y_0 \neq 0 \tag{17}$$

Towards the solution of the free evolution starting from a simplified case

$$\dot{y} = y \qquad y(0) = y_0 \neq 0 \tag{18}$$

(in other words, which function is equal to its derivative?)

The solution of the free evolution in general

$$\dot{y} = ay \qquad y(0) = y_0 \neq 0 \tag{19}$$

Alternatively:

$$\dot{y} = \frac{dy}{dt} = ay(t) \tag{20}$$

$$\frac{dy}{y(t)} = a\ dt \tag{21}$$

$$\int_0^t \frac{dy}{y(\tau)} = \int_0^t a\ \mathrm{d}\tau \tag{22}$$

$$\Big[\ln y(\tau)\Big]_0^t = a\Big[\tau\Big]_0^t \tag{23}$$

$$\ln\big(v(t)\big) = \ln(y_0) + at \tag{24}$$

$$y(t) = e^{\ln(y_0)at} = e^{\ln(y_0)}e^{at} \tag{25}$$

$$y(t) = y_0 e^{at} \tag{26}$$

And what about the vectorial generalization?

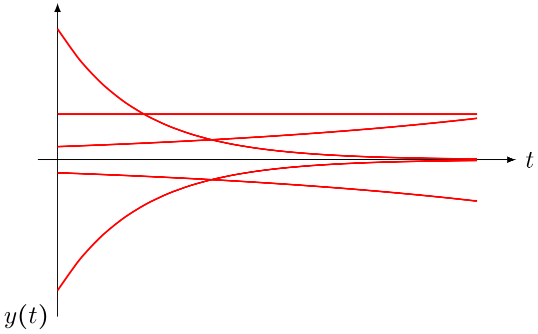
$$\dot{\boldsymbol{y}} = A\boldsymbol{y}$$

$\Longrightarrow$ later in the course, and extensively!

Analysis of first order systems in free evolution

$$\dot{y} = ay \qquad y(0) = y_0 \neq 0 \qquad \Longrightarrow \qquad y(t) = y_0 e^{at} \tag{27}$$

Discussion: which case is this one?



Introduction to stability

stable system: $y(t) \to 0$ for $t \to +\infty$

unstable system: $|y(t)| \to +\infty$ for $t \to +\infty$

marginally stable system: $y(t) = y_0$ for every t

much more on stability both later on & in later courses!

Are we done with studying the response of first order dynamical models?

$$\dot{y} = ay + bu \tag{28}$$

Forced response

$$\dot{y} = ay + bu \qquad y(0) = y_0 = 0 \tag{29}$$

(remember: free evolution = "$u = 0, y_0 \neq 0$")

Discussion: does $y(0) = y_0 = 0$ imply that we should study $\dot{y} = bu$?

Try it yourself at home!

`GitHub/TTK4225/trunk/Jupyter/`
`first-order-system-introduction-with-P-controller.ipynb`

Introduction to first order systems - the concept of forced response

Roadmap

- the concept of "forced response"
- the convolution operator
- some properties of convolution

First order dynamical models

$$\dot{y} = ay + bu \tag{30}$$

`GitHub/TTK4225/trunk/Jupyter/`
`first-order-system-introduction-with-P-controller.ipynb`

Overarching question towards model-based control: how does y evolve?

$$\dot{y} = ay + bu \tag{31}$$

Free evolution vs. forced response

$$\dot{y} = ay + bu \tag{32}$$

free evolution: $u = 0,\ y_0 \neq 0$

forced response: $u \neq 0,\ y_0 = 0$

Solution to the forced response

(actual solution through equations (2.11)–(2.17) in Balchen's book)

$$\dot{y} = ay + bu, \quad y(0) = y_0 = 0 \qquad \Longrightarrow \qquad y(t) = \int_0^t e^{a(\tau)} bu(t-\tau) d\tau \tag{33}$$

extremely important generalization:

$$h(t) = e^{at} \qquad \Longrightarrow \qquad y(t) = \int_0^{+\infty} h(\tau) u(t-\tau) d\tau$$

extremely important definition:

convolution := $h * u(t) := \int_{-\infty}^{+\infty} h(\tau) u(t-\tau) d\tau$

Visualizing $h * u(t) := \int_{-\infty}^{+\infty} h(\tau) u(t-\tau) d\tau$

Discussions

$$h * u(t) := \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$$

1. is $h * u(t) = u * h(t)$?
2. is $(\alpha h_1 + \beta h_2) * u(t) = \alpha\left(h_1 * u(t)\right) + \beta\left(h_2 * u(t)\right)$?
3. why is it that $\dot{y} = ay + bu, \quad y(0) = y_0 = 0$ implies

$$y(t) = \int_{0}^{+\infty} e^{a(\tau)} bu(t-\tau)d\tau$$

and *not*

$$y(t) = \int_{-\infty}^{+\infty} e^{a(\tau)} bu(t-\tau)d\tau \quad ?$$

Example

$$y(t) = \int_{0}^{+\infty} h(\tau)u(t-\tau)d\tau,$$

$$u(t) = \begin{cases} 1 & \text{for } t \in [1,2] \\ 0 & \text{otherwise} \end{cases} \qquad h(t) = \begin{cases} 2 & \text{for } t \in [0,1) \\ 1 & \text{for } t \in [1,2] \\ 0 & \text{otherwise} \end{cases}$$

Discussion:

What is $y(t) = h * u(t)$ for the case

$$u(t) = \begin{cases} 1 & \text{for } t \in [1,2] \text{ and } t \in [2,3] \\ 0 & \text{otherwise} \end{cases} \qquad h(t) = \begin{cases} 2 & \text{for } t \in [0,2) \\ 0 & \text{otherwise} \end{cases} \quad ?$$

TODO

`GitHub/TTK4225/trunk/Jupyter/TODO.ipynb`

Summary

$$\dot{y} = ay + bu \tag{34}$$

free evolution: $u = 0$, $y_0 \neq 0$; implies $y(t) = y_{\text{free}}(t) = y_0 e^{at}$
forced response: $u \neq 0$, $y_0 = 0$; implies $y(t) = y_{\text{forced}}(t) = h * u(t)$ with $h(t) = e^{at}$

e^{at} = "impulse response"

yet another pillar concept for automatic control
→ *will be discussed better in a dedicated module*
*and re-discussed in **every** control-oriented course*

General response

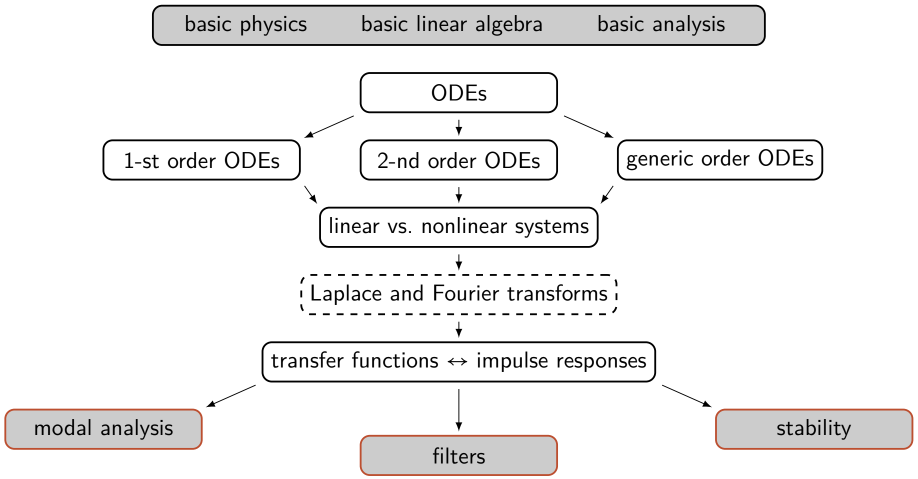
$$\dot{y} = ay + bu \tag{35}$$

free evolution: $u = 0$, $y_0 \neq 0$; implies $y(t) = y_{\text{free}}(t) = y_0 e^{at}$

forced response: $u \neq 0$, $y_0 = 0$; implies $y(t) = y_{\text{forced}}(t) = h * u(t)$ with $h(t) = e^{at}$

general response: $u \neq 0$, $y_0 \neq 0$; implies $y(t) = y_{\text{free}}(t) + y_{\text{forced}}(t)$ thanks to the fact that the model is *linear* *(will be seen in more details later on)*

Summary / Roadmap of TTK4225



Introduction to first order systems - visualizing the system through a block scheme

Roadmap

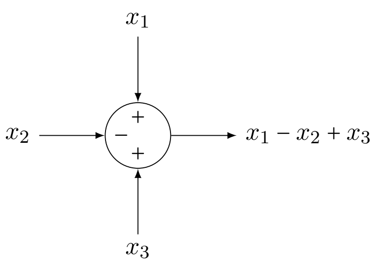
- the most common block schemes
- first order systems as block schemes

Block diagrams - why?

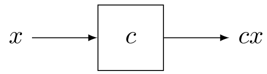
- used very often in companies
- aid visualization *(until a certain complexity is reached. . .)*
- enable “drag & drop” way of programming

- in this course, primarily used for interpretations

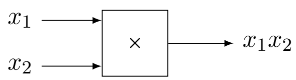
Most common block diagrams - sum of n signals



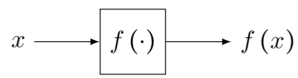
Most common block diagrams - multiplication for a constant



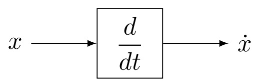
Most common block diagrams - multiplication of two signals



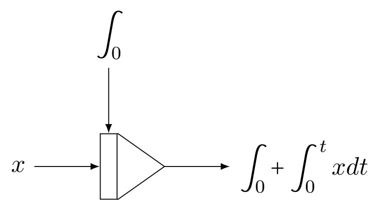
Most common block diagrams - generic functions



Most common block diagrams - derivatives



Most common block diagrams - integrals

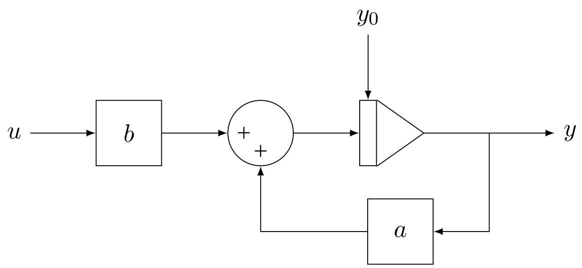


Discussion: how do we represent a first order DE with a block scheme?

$$\dot{y} = ay + bu$$

The solution

$$\dot{y} = ay + bu$$



Discussion: do you note the presence of a feedback loop?

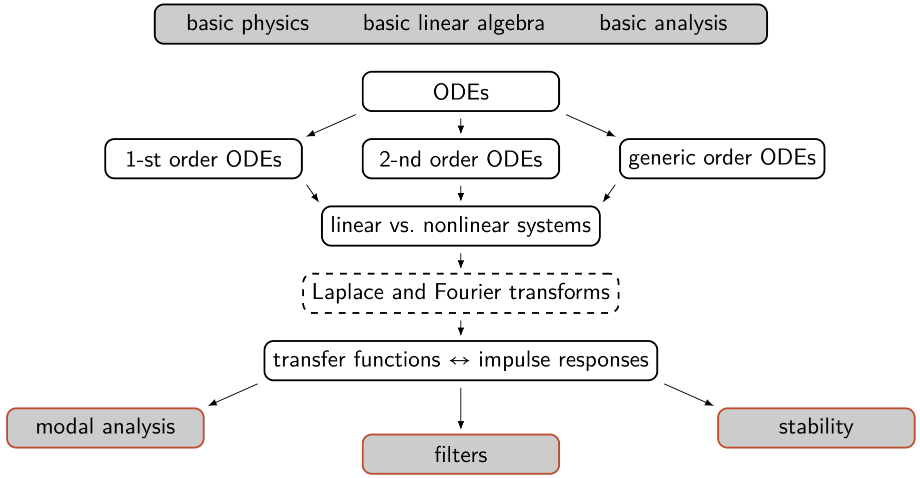
Introduction to Simulink

Introduction to generic order systems

Roadmap

- from first order to second order
- generic orders
- classification of the different types of ODEs

Summary / Roadmap of TTK4225



From first order to second order, third order

Discussion: if we can write

$$\dot{y} = ay + bu \,,$$

can we also write

$$\ddot{y} = a_1 \dot{y} + a_0 y + bu \; ?$$

And at this point also

$$\dddot{y} = a_2 \ddot{y} + a_1 \dot{y} + a_0 y + bu \; ?$$

And why not

$$\dddot{y} = a_2 \ddot{y} + a_1 \dot{y} + a_0 y + b_1 \dot{u} + b_0 u \; ?$$

ARMA models

$$y^{(n)} = a_{n-1} y^{(n-1)} + \ldots + a_0 y + b_m u^{(m)} + \ldots b_0 u \; ?$$

Discussion: and why don't we have $a_n y^{(n)}$, instead of only $y^{(n)}$ on the LHS?

Discussion: can we generalize even more?

$$y^{(n)} = a_{n-1}y^{(n-1)} + \ldots + a_0 y + b_m u^{(m)} + \ldots b_0 u \ ?$$

non-homogeneous: $\dot{y} = a(t)y + b(t)u + c$

time-varying: $\dot{y} = a(t)y + b(t)u$

non-linear: $\dot{y} = a\sqrt{y} + bu^{2/3}$

Discussion: can we make this linear again?

$$\dot{y} = ay + bu^{2/3}$$

Another discussion: can we apply the same "linearization trick" to $\dot{y} = a\sqrt{y} + bu$?

main focus in this course:
linear, time invariant, homogeneous (LTI)

why?

Remember: predictive control of the type

$$\boldsymbol{u}^\star = \arg \min_{\boldsymbol{u}\in\mathcal{U}, \boldsymbol{f}(\boldsymbol{u})\in\mathcal{F}} \mathrm{Cost}\left(\boldsymbol{f}\left(\boldsymbol{u}\right), \boldsymbol{u}\right)$$

requires to be able to:

- define "Cost"
- compute $\boldsymbol{f}\left(\boldsymbol{u}\right)$ rapidly
- be sure that the model does not have "nasty" properties

from the previous lessons:
1-st order LTI models $\mapsto$ impulse responses;
as will be clear later generic order LTI models $\mapsto$ also impulse responses

The question we are trying to answer in this first part of the course

"What will be the solution $y(t)$ to a generic-order LTI ODE starting from a generic initial condition $y(0) = y_0$ and generic input signal $u(t)$?"

(our hope: the solution will be structurally easy, and fast to compute)

What is what?

(i.e., a small quiz to see if we are aligned)

	autonomous	linear	homogeneous	time constant
$\frac{\dot{x}}{x} = 1$				
$\pi = \dot{x}x$				
$2\dot{x} + \sin t = 0$				
$\sin(t\dot{x}) - x = 0$				
$t\dot{x} = x \sin t$				
$x - e^{-t} = \ddot{x} + \dot{x}$				

Introduction to generic order systems

Roadmap

- state space representations
- from n-th order to vectorial first order

State space representations - Definition

mathematical model (typically but not limited to of a physical system) as a finite set of inputs, outputs and state variables related by first-order differential equations satisfying the separation principle

Ingredients

- finite number of inputs, outputs and state variables
- first-order differential equations
- *satisfies the separation principle:* the current value of states contains all the information necessary to forecast the future evolution of the outputs and of the states given the inputs

State space representations - Facts

- the future output depends only on the current state and the future input
- the future output depends on the past input only through the current state
- the state summarizes the effect of past inputs on future output
  (sort of a "memory" of the system)

Example

Rechargeable flashlight:

- state = level of charge of the battery & on / off button
- output = how much light the device is producing

State space representations - Notation

$$\begin{aligned} u_1, \dots, u_m &= \text{inputs} \\ x_1, \dots, x_n &= \text{states} \\ y_1, \dots, y_p &= \text{outputs} \end{aligned}$$

State space representations - Notation

$$\begin{aligned} \dot{x}_1 &= f_1 \left(x_1, \dots, x_n, u_1, \dots, u_m \right) \\ &\vdots \\ \dot{x}_n &= f_n \left(x_1, \dots, x_n, u_1, \dots, u_m \right) \\ y_1 &= g_1 \left(x_1, \dots, x_n, u_1, \dots, u_m \right) \\ &\vdots \\ y_p &= g_n \left(x_1, \dots, x_n, u_1, \dots, u_m \right) \end{aligned}$$

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f} \left(\boldsymbol{x}, \boldsymbol{u} \right) \\ \boldsymbol{y} &= \boldsymbol{g} \left(\boldsymbol{x}, \boldsymbol{u} \right) \end{aligned}$$

- $\boldsymbol{f}$ = state transition map
- $\boldsymbol{g}$ = output map

Discussion: is any $\boldsymbol{f}$ and $\boldsymbol{g}$ ok?

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})\end{aligned}$$

No: for every given $\boldsymbol{x}_0$ and $\boldsymbol{u}$ the solution must be:

- unique
- guaranteed to exist for $t \in [0, \bar{t})$ for some $\bar{t} > 0$

(check "existence and uniqueness of solutions of ODEs")

State space representations - Example

$$\begin{cases} \dot{x} = -x^2 \\ y = x \end{cases} \cup \quad x_0 = -1 \qquad \Longrightarrow x(t) = \frac{1}{t-1}$$

Discussion: (a bit outside of the current flow of concepts on ODEs, but still relevant and very important) what happens as $t \to 1$?

there is a finite escape time!

Discussion: can every physical system be described with a state space representation?

Noticeable exceptions:

- non-lumped parameters model (i.e., PDEs)
- continuous-time systems with delays
  (will be discussed better later on)

Example: dynamics of sugar and insulin concentrations in blood

- $x_1 :=$ sugar concentration
- $x_2 :=$ insulin concentration
- $u_1 :=$ food intake
- $u_2 :=$ insulin intake
- $c :=$ sugar concentration in fasting

$$\begin{cases} \dot{x}_2 = a_{21}(x_1 - c) - a_{22}x_2 + b_2u_2 & x_1 \geq c \\ \dot{x}_2 = -a_{22}x_2 + b_2u_2 & x_1 < c \end{cases}$$

$$\begin{cases} \dot{x}_1 = -a_{11}x_1x_2 + a_{12}(c - x_1) + b_1u_1 & x_1 \geq c \\ \dot{x}_1 = -a_{11}x_1x_2 + b_1u_1 & x_1 < c \end{cases}$$

How can I transform a generic-order ODE into a state space representation (that requires first order DEs)?

$$\dddot{y} = \phi\left(\ddot{y}, \dot{y}, y, \ddot{u}, \dot{u}, u\right);$$

change of variables:

$$\boldsymbol{x} = \begin{bmatrix} \ddot{y} \\ \dot{y} \\ y \end{bmatrix}, \; \boldsymbol{u} = \begin{bmatrix} \ddot{u} \\ \dot{u} \\ u \end{bmatrix} \qquad \Rightarrow \qquad \dot{\boldsymbol{x}} = \begin{bmatrix} \dddot{y} \\ \ddot{y} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \phi\left(\boldsymbol{x}, \boldsymbol{u}\right) \\ [1\;0\;0]\,\boldsymbol{x} \\ [0\;1\;0]\,\boldsymbol{x} \end{bmatrix} = \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right)$$

Thus, how do I transform a generic order LTI system into a first order LTI system?

$$\ddot{y} = 2\dot{y} + 3y + 4\ddot{u} + 5\dot{u} + 6u$$

Transforming ARMA models

$$y^{(n)} = a_{n-1}y^{(n-1)} + \ldots + a_0 y + b_m u^{(m)} + \ldots b_0 u \;?$$

Discussion: but then any LTI system is a first order vectorial system?

$$\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u}$$

YES, AND THIS IS WHY
LINEAR ALGEBRA IS ESSENTIAL
FOR CONTROL!

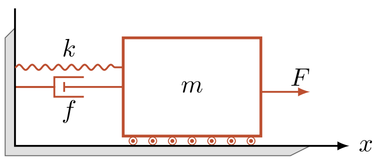
Second order systems

Roadmap

- interesting examples
- interesting responses
- differences between 1st order and 2nd order

Example 1: spring-mass systems

Position of a cart fastened with a spring to a wall and subject to friction:



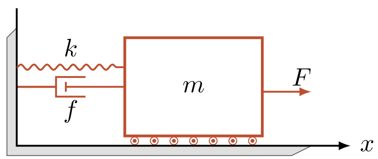
EOM:

- force from the spring: $F_x(t) = -kx(t)$
- friction: $F_f(t) = -f\dot{x}(t)$
- applied force: $F(t)$
- Newton's second law: $\sum F = m\ddot{x}(t)$

$$m\ddot{x}(t) = F_x(t) + F_f(t) + F(t) \qquad \mapsto \qquad m\ddot{x}(t) = -kx(t) - f\dot{x}(t) + F(t)$$

Example 1: spring-mass systems

Position of a cart fastened with a spring to a wall and subject to friction:



$$\ddot{x}(t) + \frac{f}{m}\dot{x}(t) + \frac{k}{m}x(t) = \frac{1}{m}F(t) \tag{36}$$

Generalizing:

$$\ddot{x} = a_1\dot{x} + a_0 x + b_0 u \tag{37}$$

Discussion: may we write this dynamics using a vectorial first order DE?
Discussion: may we write this dynamics using a scalar first order DE?

Example 1: spring-mass systems

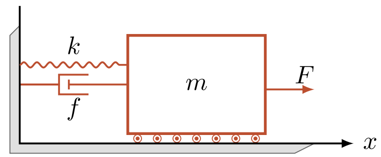
https://youtu.be/lZPtFDXYQRU?t=31

(here there is very little friction, though)

Example 1: spring-mass systems - stability analysis from intuitive perspectives

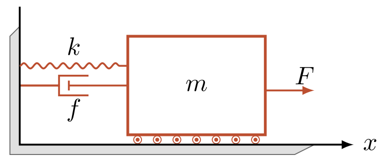
Discussion: $\dot{x} = a_0 x + b_0 u$ unstable if . . . ? And What about, from intuitive perspectives, when considering

$$\ddot{x} = a_1 \dot{x} + a_0 x + b_0 u \, ? \tag{38}$$



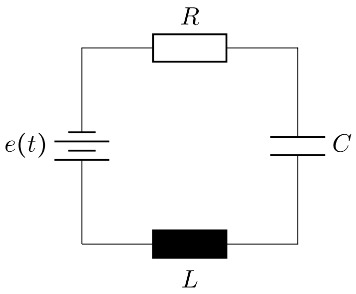
GitHub/TTK4225/trunk/Jupyter/spring-mass-systems-introduction.ipynb

Intuition: asymptotic stability ↔ dissipation of energy



will see this MUCH better in later on courses
when discussing about Lyapunov stability

Example 2: RCL-circuit

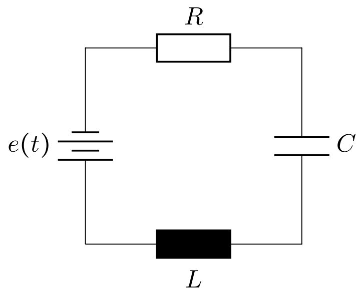


EOM: Kirkhoff laws

$$v_L(t) = L\frac{di(t)}{dt} \qquad v_i(t) = Ri(t) \qquad v_C(t) = \frac{1}{C}\int_0^t i(\tau)d\tau$$

$$e(t) = v_L(t) + v_R(t) + v_C(t) \quad \mapsto \quad e(t) = L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau \qquad (39)$$

Example 2: RCL-circuit



$$e(t) = L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau$$

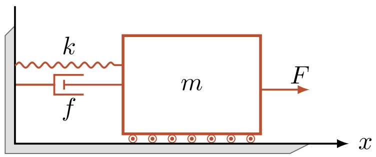
Discussion: how do we write it in terms of x, $\dot{x}$, ...?

$$x(t) = i(t), \quad u(t) = \dot{e}(t) \qquad \Longrightarrow \qquad \ddot{x} = -\frac{R}{L}\dot{x} - \frac{1}{LC}x + \frac{1}{L}u \qquad (40)$$

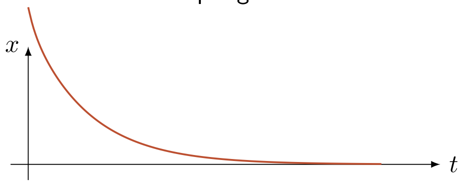
Generalizing:

$$\ddot{x} = a_1\dot{x} + a_0x + b_0u \qquad (41)$$

Discussion: what are the potential different ways the cart may move back to its equilibrium?



very big friction w.r.t. the spring stiffness $\Longrightarrow$ overdamping:



very small friction w.r.t. the spring stiffness $\Longrightarrow$ underdamping:

How do the free evolutions and forced responses look like numerically?

we will solve this problem with the aid of Laplace transforms

Summary / Roadmap of TTK4225