# Systems Laboratory, Spring 2025

Damiano Varagnolo – CC-BY-4.0



notes

- welcome to the course!
- on this side of this document you will find notes that accompany the text typically visualized in class
- these notes are meant to convey the messages that are not displayed in the text on the side, and basically constitute what the teacher intends to say in class

## Table of Contents I





notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

## Table of Contents II





## Table of Contents III





## Table of Contents IV





# Modelling in Continuous Time

notes

# Intended Learning Outcomes (ILOs)



notes

## By the end of module "Modelling in Continuous Time", be able to: I

- **Decide** whether a given function is a solution to a specified ODE by direct verification.
- **Classify** an ODE as linear or nonlinear, autonomous or non-autonomous, time-invariant or time-varying, based on its structural properties.
- **Compute** the equilibria of an ODE by solving for stationary points.
- **Construct** and interpret phase portraits of first- and second-order autonomous ODEs using qualitative analysis techniques.
- **Interpret** automatic control as an opportune operation on the dynamics of a system.
- **Linearize** a nonlinear ODE around an equilibrium point and assess the validity of the approximation.
- **Evaluate** the meaning and applicability of linearization in different contexts, discussing when it provides a reasonable approximation and when it does not.

notes

- go back to these all the times you revise this module

## By the end of module "Modelling in Continuous Time", be able to: II

- **Determine** and analyze the impulse response of an LTI system via software.
- **Compute** the free evolution and forced response of an LTI system via software, and also via the (continuous) convolution operator.
- **Explain** the significance of linearity and time-invariance in ODEs and how these properties affect solution methods and system behavior.
- **Apply** the superposition principle to solve and analyze LTI systems, demonstrating its implications in different scenarios.
- **Compute** free evolutions and forced responses of LTI systems using Laplace-based formulas (but only as procedural tools).
- **Define** the meaning of "state space representation" in the context of linear and non-linear dynamical systems.
- **Determine** the state space structure of an LTI system starting from an ARMA ODE.



## By the end of module "Modelling in Continuous Time", be able to: III

- **Analyse** the structure of the free evolution of the state variables by means of the eigendecomposition of the system matrix.
- **Give examples** of practical cases / real life systems to which one can apply the various concepts / procedures mentioned above.



## Is this function a solution of this ODE?

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| derivative | u1, e1 |



notes

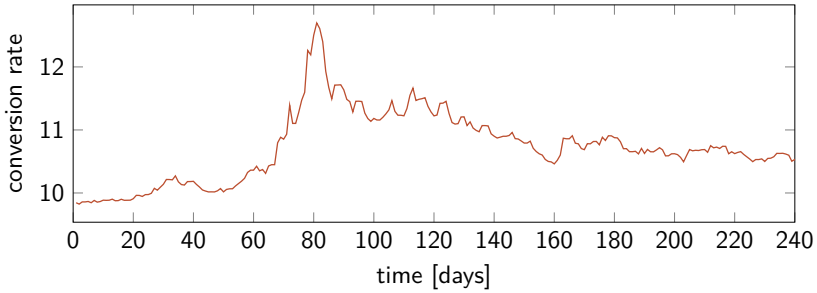
## Main ILO of sub-module "Is this function a solution of this ODE?"

**Decide** whether a given function is a solution to a specified ODE by direct verification

notes

- by the end of this module you shall be able to do this

## What is a signal?



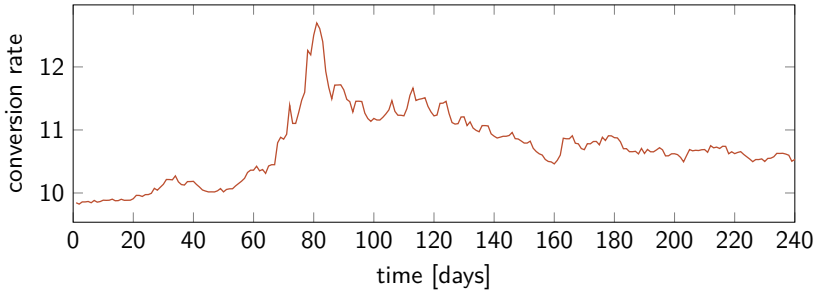

$y(t)$ (assuming $t$ continuous in this module)



notes

- a signal is any time-dependent (or space-dependent) variation of a physical quantity that conveys information. In engineering and science, signals can take many forms, such as electrical voltages, sound waves, images, or sequences of numbers in a digital system

## What is the derivative of this signal?

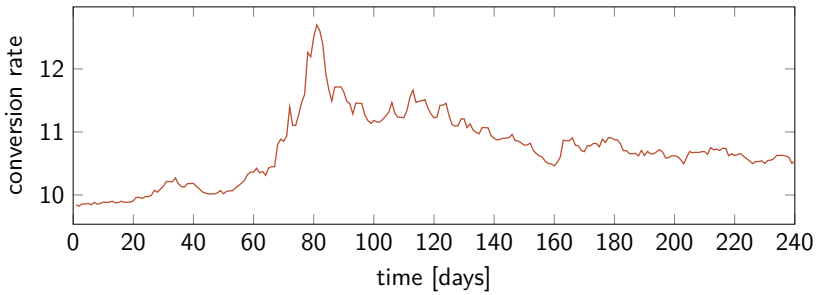



$$y(t) \quad \mapsto \quad \frac{dy(t)}{dt} = \dot{y}(t)$$

notes

- we will extensively use this notation in the course; basically never use the 'd / dt'

## Would you say that $y(t) = \dot{y}(t)$, in this case?





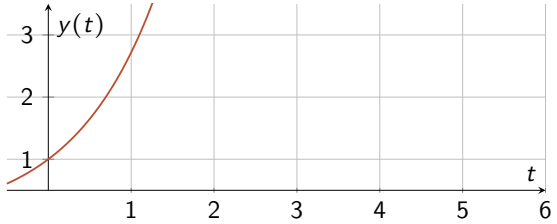

notes

- definitely not; the two signals are not the same at all

## *"uhm, where are we going with all this stuff?"*

↦ be able to do forecasts

would you be able to compute $y(5)$ from this graph, if you knew that $\dot{y}(t) = y(t)$?



notes

- we will see how an ODE (Ordinary Differential Equation) describes how a system's state evolves over time based on its current state and inputs. By the end of the course it will be obvious how solving the ODE with given initial conditions, we can predict future states. This is used in control to enable forecasting, that is essentially by propagating system dynamics forward in time. We will also see though that the accuracy of forecasts depends on how well the ODE models the real-world system (something that may be more or less good)

an ODE is a tool to produce forecasts

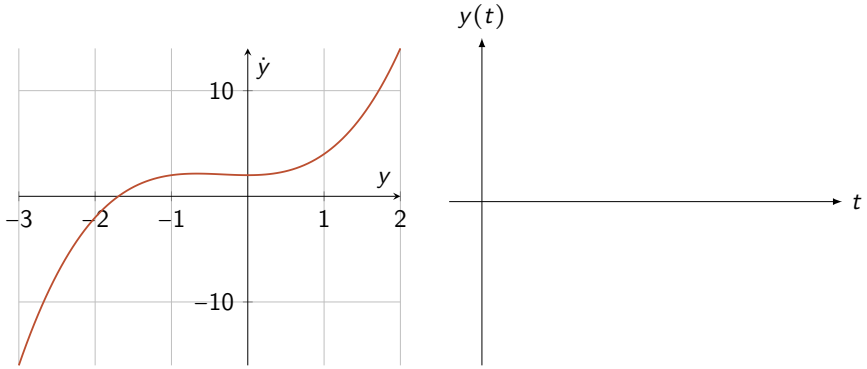
notation: instead of $\dot{y}(t)$ or $y(t)$ we will write $\dot{y}$ or $y$



notes

- so this is an important message
- watch out at the notation; we will always mean signals, and not scalars

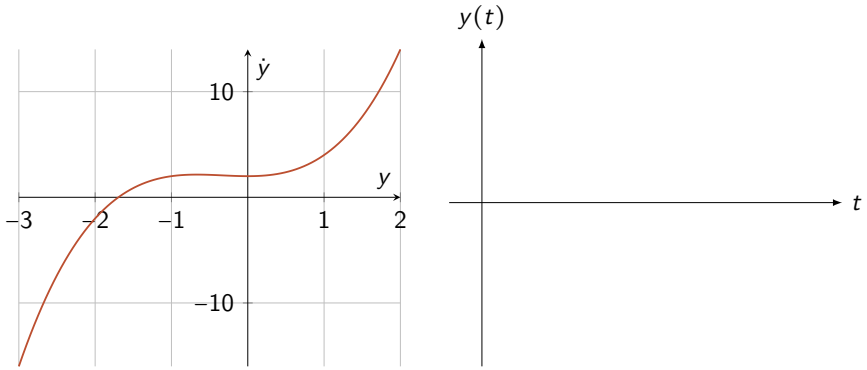
## But what does it mean to solve an ODE?



notes

- before continuing we need though to reflect on what it means to solve a differential equation. Let's take this example to investigate this concept
- remember that all the symbols here actually mean signals in time, i.e., they should be intended as $\dot{y}(t) = y(t) + u(t)$ ideally with $t$ in a certain domain (typically $[0, +\infty)$)
- for example this combination of signals does not satisfy the equation
- this instead does

## Is knowing the ODE enough to be able to generate a trajectory?

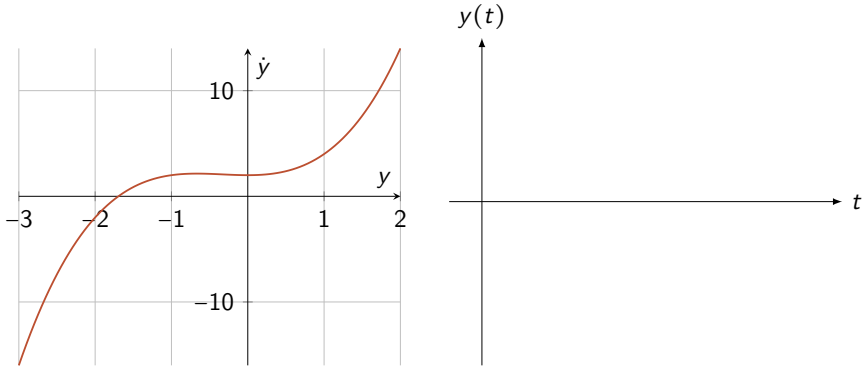





notes

- no, there is the need for defining an initial condition to be able to draw the trajectory

## Does $\{y(t) = \cos(t), y(0) = 1\}$ solve this ODE?



notes

- no, because that specific signal does not satisfy the ODE

## Are we done with this?

**Decide** whether a given function is a solution to a specified ODE by direct verification

→ no, there are still a lot of cases we shall cover



notes

- by the end of this module you shall be able to do this

## Notation time!

In control, modelling a dynamical system = defining

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

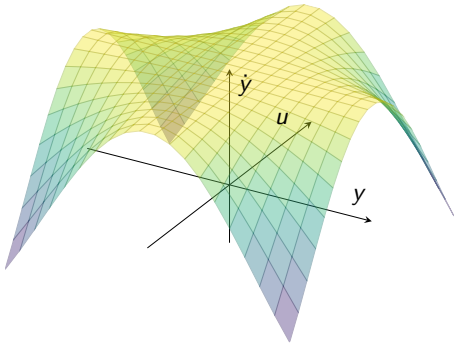
thus defining:

- the variables
  - $\boldsymbol{u}$ = inputs *(i.e., what we can steer)*
  - $\boldsymbol{d}$ = disturbances *(i.e., what we cannot steer but that still influences the system)*
  - $\boldsymbol{y}$ = outputs *(i.e., what we are interested into)*
- the shape of $\boldsymbol{f}$
- the value of its parameters $\boldsymbol{\theta}$
- bold font = vector

notes

- let's now generalize the ODE before to something that can be applied to more cases
- let's do this definition, where the names are given in this way for historical reasons
- for example the Lotka Volterra model that we will see below is a specific example of this way of writing things, where there is no $u$ and $d$ by the way
- once again $f$ has the meaning of indicating "where" the system is going towards, in time - also this will be more clear soon

## A graphical example of $\dot{y} = f(y, u)$




*https://www.geogebra.org/classic/mmppe6hs*

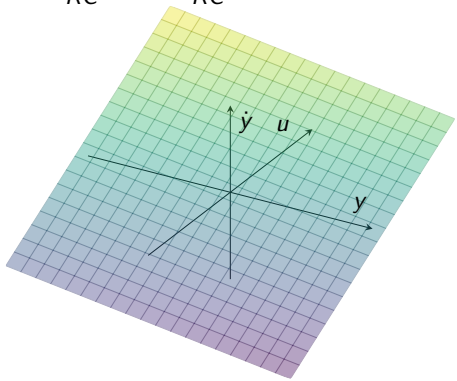


notes

- the simplest trajectories are that ones that seem simple constant numbers

## A couple of ODEs that you may have already seen, of the type $\dot{y} = ay + bu$

- velocity of a cart: $\dot{v}(t) = -\frac{k}{m}v(t) + \frac{k}{m}F(t)$
- RC-circuits: $\dot{v}_C(t) = -\frac{1}{RC}v_C(t) + \frac{1}{RC}V(t)$



notes

- you should have already seen these ODEs (or be able to derive them) from physics and electronics

## Some more details about the first example

$$\text{notation: } F(t) = ma(t) = m\dot{v}(t) \quad \mapsto \quad F = ma = m\dot{v} \tag{1}$$

. . . but $v = \dot{p}$, thus:

$$\begin{cases} \dot{p} = v \\ \dot{v} = \dfrac{F}{m} \end{cases} \tag{2}$$

this is a system of ODEs



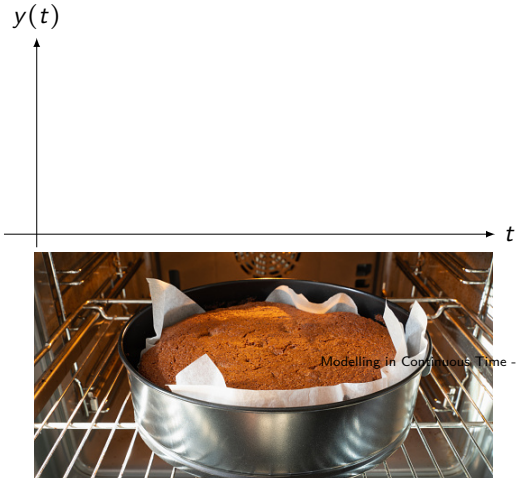
notes

- first of all we will virtually always drop the '$t$' because we know we are working with time signals
- moreover we also know that velocity is a time derivative of position, so we actually obtain a system of ODEs from Newton laws
- in the next slide we see what is the geometric meaning of an ordinary differential equation or systems of ODEs

## Another practical example

temperature of the center of a cake in an oven whose temperature is 200 degrees:

$$\dot{T} = -0.5\big(T - 200\big)$$



notes

- let's do a practical example of a very simple ODE
- here we can see that starting from any $T$ we ideally tend to go to $T_a$
- but how? Let's see




notes

- https://en.wikipedia.org/wiki/The_Treachery_of_Images

## Important point: model ≠ real world

Ceci n'est pas un gâteau.

$$\dot{T} = -0.5\big(T - 200\big)$$

notes

- we will re-consider this point later on, but for now it has to be obvious that this is an approximated description of reality
- why we use them and what are the usages we'll do about this, we'll see later on in the course

## Another example: a Lotka-Volterra model (≠ real world):

$$\begin{cases} \dot{y}_{\text{rabbits}} &= 0.4 \cdot y_{\text{rabbits}} - 0.5 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \\ \dot{y}_{\text{foxes}} &= -3 \cdot y_{\text{foxes}} + 0.7 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \end{cases}$$



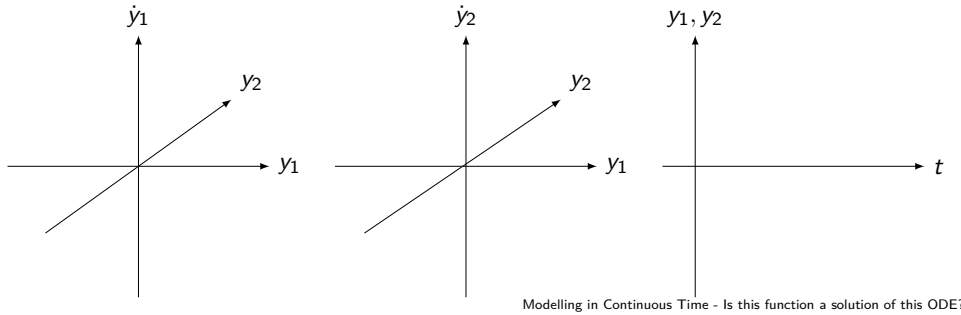

notes

- we may have though more complicated models
- for example this one is a model that may accurately describe the evolution of the populations of preys and predators in a closed environment (we'll discuss this one better later on)

## What do we mean with "dynamics"? More geometrically

(example: 2D system, autonomous)

example: two dimensional $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$ in the sense of $\begin{cases} \dot{y}_1 &= f_1(y_1, y_2) \\ \dot{y}_2 &= f_2(y_1, y_2) \end{cases}$

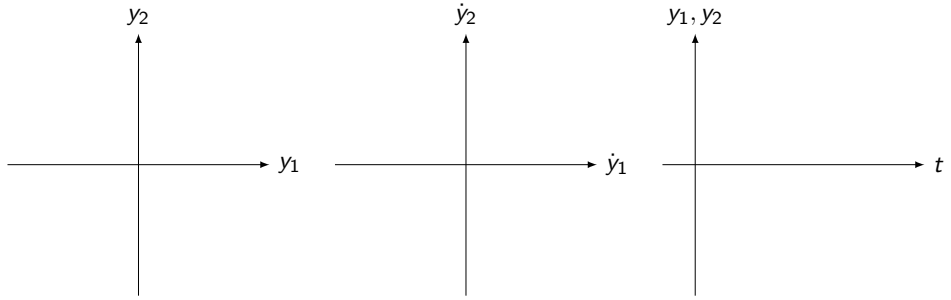


notes

- graphically we have different objects
- one typically looks at trajectories in time
- but one may also think at the phase portrait of the outputs (or states, if one is working with state space models)
- the important is to realize that for any given couple $\boldsymbol{y}, \boldsymbol{u}$ one is able to compute $f$, that means being able to compute the vector $\dot{\boldsymbol{y}}$, that can be seen in the first set of cartesian axes
- then this $\dot{\boldsymbol{y}}$ can also be superposed in the axes for $\boldsymbol{y}$, and we can give the interpretation that this is where the system is moving towards
- intuitively, changing $u$ means changing $\dot{\boldsymbol{y}}$, that means also steering where the system is heading towards
- note that the fact that they system has some dynamics means that if I want to reach a specific $\boldsymbol{y}$ starting from a given initial condition $\boldsymbol{y}_0$, there is the need for waiting a bit of time before we can reach the goal. We cannot go 'instantaneously' to the goal

## Same example, alternative viewpoint

$$\begin{cases} \dot{y}_1 &= f_1(y_1, y_2) \\ \dot{y}_2 &= f_2(y_1, y_2) \end{cases}$$






notes

- we may though see this with an alternative viewpoint, that will lead us to linear algebra formulations

## Coding the Lotka-Volterra example

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

`./LotkaVolterraSimulator.ipynb`

*(we'll see later on how this continuous thing is actually implemented in our discrete computers)*

notes

- to solve numerically the trajectories let's explore and run the corresponding python notebook
- for now we will do things numerically; later on we will do it also analytically

## 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 ]
```



notes

- do you think that this code represents a model? yes, this contains all the information that is needed to compute the $f$, i.e., the direction where the system is pointing towards given the current situation of the system

## Discussion: do we need something more than just the model to simulate the system?

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

notes

- do you think we can simulate the model before just with this information? Or do we need something more? no, because we need an initial condition. Starting from different initial conditions we have completely different behaviors (think at no foxes vs there are some foxes)

## Remember: static $\neq$ dynamic

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



notes

- thus remember always that we are working with ODEs, that means that $f$ does not give what is the output directly, but rather where the system is going to. It gives a derivative!

## Summarizing

**Decide** whether a given function is a solution to a specified ODE by direct verification

- check $y$, compute $f(y)$, compute $\dot{y}$
- does $f(y) = \dot{y}$?
- same apply for higher orders / more complex ODES from notational perspectives

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

## Solving ODEs

`https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/chapter22.06-Python-ODE-Solvers.html`

notes

- check that page

# Self-assessment material



notes

## Question 1

Which of the following best describes what it means for a function $y(t)$ to be a solution of an ODE?

**Potential answers:**

I: **(wrong)** It satisfies the ODE for at least one value of $t$.
II: **(correct)** It satisfies the ODE for all values of $t$ in its domain.
III: **(wrong)** It approximately satisfies the ODE within a certain error margin.
IV: **(wrong)** It satisfies the ODE only at integer values of $t$.
V: **(wrong)** I do not know

**Solution 1:**

A function is a solution of an ODE if it satisfies the equation for all values of $t$ within its domain. A solution must be valid throughout the considered interval, not just at isolated points.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 2

What additional information is needed to uniquely determine a solution of an ODE?

**Potential answers:**

I: **(wrong)** The function $y(t)$ itself.
II: **(correct)** An initial condition specifying the value of $y$ at a given time.
III: **(wrong)** A boundary condition at two different points.
IV: **(wrong)** The highest-order derivative of $y$.
V: **(wrong)** I do not know

**Solution 1:**

An initial condition provides the necessary information to select a unique solution from the family of possible solutions to a differential equation. Without it, multiple solutions may exist.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 3

Given the ODE $\dot{y} = y$, which of the following functions is a solution?

**Potential answers:**

I: **(wrong)** $y(t) = t^2$
II: **(correct)** $y(t) = Ce^t$, where $C$ is a constant.
III: **(wrong)** $y(t) = \sin t$
IV: **(wrong)** $y(t) = \dfrac{1}{t+1}$
V: **(wrong)** I do not know

**Solution 1:**

The function $y(t) = Ce^t$ satisfies the equation since its derivative is also $Ce^t$, matching the right-hand side of the ODE.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 4

Which of the following differential equations is nonlinear?

**Potential answers:**

I: **(correct)** $\dot{y} + 2y = 3$
II: **(correct)** $\dot{y} = y^2$
III: **(correct)** $\dot{y} = 3y + 5$
IV: **(correct)** $\dot{y} + \sin y = t$
V: **(wrong)** I do not know

**Solution 1:**

An equation $\dot{y} = f(y)$ is linear only if $f(y) = \alpha(t)y$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 5

What is an equilibrium point of the ODE $\dot{y} = y(1 - y)$?

**Potential answers:**

I: **(wrong)** $y = 2$
II: **(correct)** $y = 0$ and $y = 1$
III: **(wrong)** $y = -1$
IV: **(wrong)** $y = \frac{1}{2}$
V: **(wrong)** I do not know

**Solution 1:**

Equilibrium points occur where $\dot{y} = 0$, meaning $y(1 - y) = 0$. This happens at $y = 0$ and $y = 1$.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "Is this function a solution of this ODE?"

- a function is a solution of an ODE if it satisfies the equation for all values in its domain
- initial conditions are necessary to uniquely determine a solution



notes

- the most important remarks from this sub-module are these ones

which type of ODE is this one?

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| linear | u1, e1 |
| time invariant | u1, e1 |
| autonomous | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |



notes

## Main ILO of sub-module "which type of ODE is this one?"

**Classify** an ODE as linear or nonlinear, autonomous or non-autonomous, time-invariant or time-varying, based on its structural properties

notes

- by the end of this module you shall be able to do this

# linear vs. nonlinear



notes

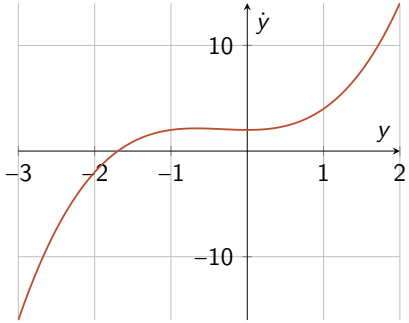
## Definition: $f(\cdot)$ is linear iff

- $f(x+y) = f(x) + f(y)$
- $f(\alpha y) = \alpha f(y)$

notes

- this is the classical definition of linearity

## Is this function linear?

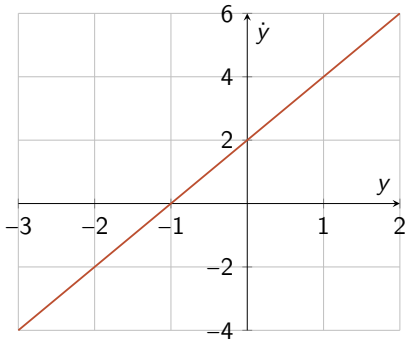





notes

- definitely not

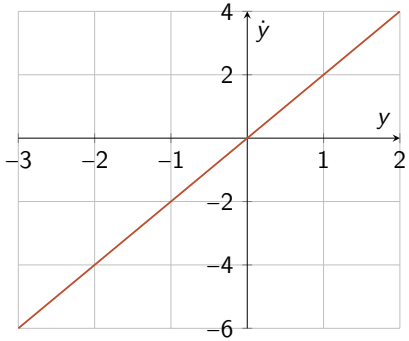
## Is this function linear?



notes

- definitely not too, it has to pass through zero

## Is this function linear?

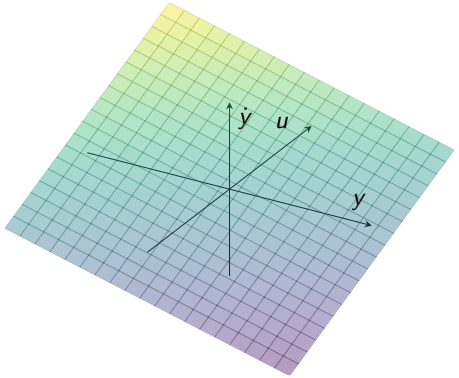





notes

- looking at this graph, we note that these two properties hold
- and this holds only because of linearity. If we are having an affine map, for example, the second would not hold

## Is this function linear?




*https://www.geogebra.org/classic/mmppe6hs*

notes

- this is linear too

## autonomous vs. non-autonomous



notes

## Definitions

- autonomous = the dynamics evolve based solely on the system state, e.g.:
  - $\dot{v} = -0.3v$
  - $\dot{T} = -4(T - 20)$
  - Lotka-Volterra (as saw before)
- non-autonomous = otherwise (i.e., some other variables matter too), e.g.:
  - $\dot{v} = -0.3v + F$
  - $\dot{T} = -4(T - u)$
  - Lotka-Volterra with human intervention

notes

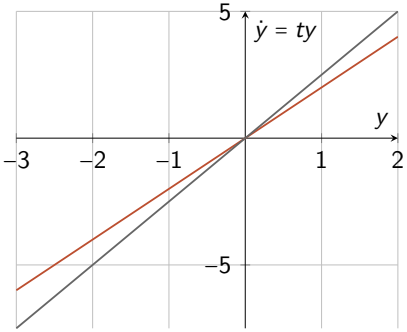
- the autonomous versions do not have external inputs
- the non-autonomous versions do have some instead

## time-invariant vs. time-varying



notes

## Is this function time invariant?



notes

- this function changes with time, so no

## *"but all the systems are time-varying. . . "*

True, but we care only about time horizons meaningful for the decision process.
Examples:

- the engine of the car will eventually break, but during one single trip its properties don't change too much
- the building structure degrades over decades, but for a single night its thermal properties remain practically constant



notes

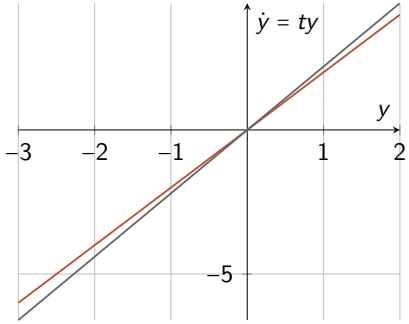
- we care only for a time horizon related to a planning phase

## Here no, here you shall consider it time-varying

notes

- the amount of fuel is consumed changes the mass very much during the mission

## Is this function linear?






notes

- for every fixed $t$ the function is actually linear. It is thus LTV

## LTI ARMA models

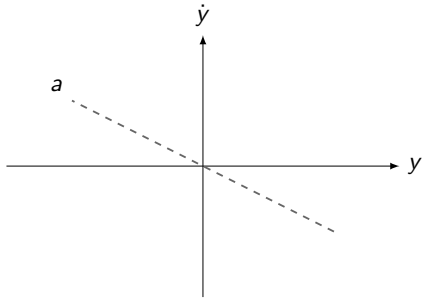
Is this LTI (linear + time invariant)?

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

notes

- as before, only a bit more complex

## What if we interpret a linear time invariant function as an ODE?




Example:

- $y'(0) = 10 \mapsto y'(t) = 10\exp(at)\mathbb{1}(t)$
- $y''(0) = 7 \mapsto y''(t) = 7\exp(at)\mathbb{1}(t)$

Thus $y'(0) + y''(0) \mapsto y'(t) + y''(t)$



notes

- looking at what we found in the previous module, this holds because the solutions to linear ODEs are exponentials passing by the initial conditions and whose exponent is always $a$
- and this holds only because of linearity

## Summarizing

**Classify** an ODE as linear or nonlinear, autonomous or non-autonomous, time-invariant or time-varying, based on its structural properties

- check the function, and answer accordingly

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

## Plotting functions

- `https://matplotlib.org/`
- `https://bokeh.org/`

notes

- the first one you shall know, the second one is a big plus

# Self-assessment material



notes

## Question 6

Which of the following autonomous systems is nonlinear?

**Potential answers:**

I: **(correct)** $\dot{y} = 3y + 5$
II: **(correct)** $\dot{y} = y^2 + 3y$
III: **(correct)** $\dot{y} = 2y - 4$
IV: **(wrong)** $\dot{y} = -0.5y$
V: **(wrong)** I do not know

**Solution 1:**

A differential equation is nonlinear if it includes terms such as $y^2$, $\sin(y)$, or $e^y$. The equation $\dot{y} = y^2 + 3y$ contains a quadratic term, making it nonlinear. The solutions $3y + 5$ and $2y - 4$ are affine, and thus nonlinear.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 7

Which of the following differential equations is autonomous?

**Potential answers:**

I: **(correct)** $\dot{y} = -2y + 5$
II: **(wrong)** $\dot{y} = 3y + \sin(t)$
III: **(wrong)** $\dot{y} = ty - 4$
IV: **(wrong)** $\dot{y} = e^t - y$
V: **(wrong)** I do not know

**Solution 1:**

A system is autonomous if its dynamics do not explicitly depend on time $t$. The equation $\dot{y} = -2y + 5$ only depends on $y$ and is therefore autonomous. The other options include explicit dependence on $t$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 8

Which of the following equations represents a time-invariant system?

**Potential answers:**

I: **(correct)** $\dot{y} = 4y + u$
II: **(wrong)** $\dot{y} = 2ty$
III: **(wrong)** $\dot{y} = \sin(t)y$
IV: **(wrong)** $\dot{y} = y + 3t$
V: **(wrong)** I do not know

**Solution 1:**

A system is time-invariant if its coefficients do not explicitly depend on time. The equation $\dot{y} = 4y + u$ meets this criterion, whereas the other options contain explicit time dependencies.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 9

Consider the equation $\dot{y} = -0.3y + (2t)u$. How should this system be classified?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | Linear, autonomous, time-invariant |
| II: | **(wrong)** | Linear, autonomous, time-varying |
| III: | **(correct)** | Linear, non-autonomous, time-varying |
| IV: | **(wrong)** | Nonlinear, non-autonomous, time-varying |
| V: | **(wrong)** | I do not know |

**Solution 1:**

The system is linear because it contains no nonlinear terms in $y$ or in $u$. It is non-autonomous due to the explicit dependence on $u$, and it is time-varying since the coefficient of $u$ depends on $t$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 10

Which function represents a linear system?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | $\dot{y} = y^3 + 2y$ |
| II: | **(correct)** | $\dot{y} = 5y + 3u$ |
| III: | **(wrong)** | $\dot{y} = \sin(y) + u$ |
| IV: | **(wrong)** | $\dot{y} = e^y - u$ |
| V: | **(wrong)** | I do not know |

**Solution 1:**

A system is linear if it satisfies the superposition principle. The equation $\dot{y} = 5y + 3u$ is linear because it only includes first-degree terms in $y$ and $u$. The other options contain nonlinear terms such as $y^3$, $\sin(y)$, or $e^y$.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "which type of ODE is this one?"

- an ODE can be classified based on its structural properties (linearity, autonomy, time-invariance)
- linearity requires both additivity and homogeneity
- autonomous systems evolve solely based on their state, while non-autonomous systems depend on external inputs
- time-invariant systems have dynamics that do not explicitly depend on time, while time-varying systems do
- graphical representations help in identifying these properties visually



notes

- the most important remarks from this sub-module are these ones

## compute the equilibria of the system

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| equilibrium | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |



notes

## Main ILO of sub-module "compute the equilibria of the system"

**Compute** the equilibria of an ODE by solving for stationary points

notes

- by the end of this module you shall be able to do this

## Is this in equilibrium?



notes

- yes, intuitively

## Are these in equilibrium, while falling?

notes

- no, intuitively

## Equilibrium = a trajectory that is constant in time

$$\dot{y}(t) = 0$$



notes

- the simplest trajectories are that ones that seem simple constant numbers

## Example

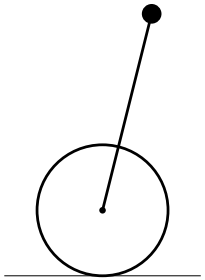
temperature of a small brick in a very large room whose temperature is 20 degrees:

$$\dot{T} = -0.5\big(T - 20\big)$$

notes

- let's do a practical example of a simple system
- here we can see that starting from any $T$ we ideally tend to go to $T_a$
- and if we start there we have the derivative equal to zero, thus an equilibrium

## What does it mean that this system is in equilibrium from an intuitive point of view?





notes

- let's start with the simplest concept, the one of equilibrium, through a practical example: a segway is in equilibrium if it is perfectly upright or if it is laying on the floor
- the term equilibrium means, from an intuitive point of view, 'things do not move', and thus the trajectories stay constant

## What does it mean that this system is in equilibrium from a mathematical point of view?

equilibrium means $\dot{\boldsymbol{y}} = \mathbf{0}$

this implies

$$\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}} \text{ is an equilibrium point iff } \dot{\boldsymbol{y}} = \boldsymbol{f}\left(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}\right) = \mathbf{0}$$

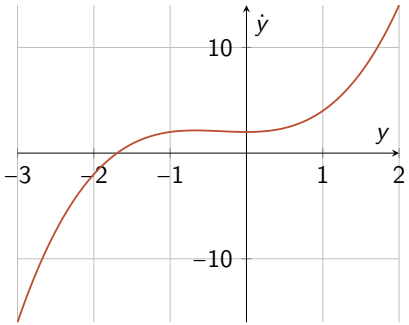
i.e., the equilibria of a system are the zeros of $\boldsymbol{f}\left(\boldsymbol{y}, \boldsymbol{u}\right)$

notes

- 'things do not move' then mathematically translate into $\dot{\boldsymbol{y}} = \mathbf{0}$
- $\dot{\boldsymbol{y}} = \mathbf{0}$ means that we are looking for that points that make $\boldsymbol{f}$ zero
- note that "iff" means if and only if

## Equilibra as the zeros of $\boldsymbol{f}$, graphically

Exemplified situation of *autonomous* single output systems:



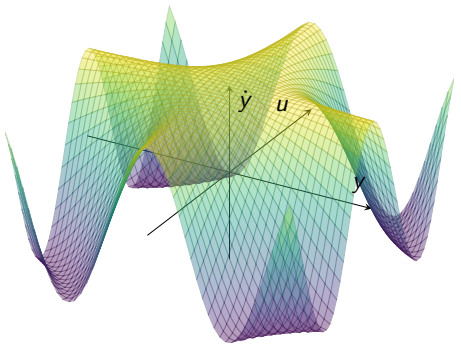



notes

- so one can see the equilibria from the crossings of $\boldsymbol{f}$ with the 'x axis' (note that this is a scalar case, and the "autonomous" case where there is no input $u$, so a simplified case)

## Equilibra as the zeros of $\boldsymbol{f}$, graphically

Exemplified situation of SISO (single input single output) systems:



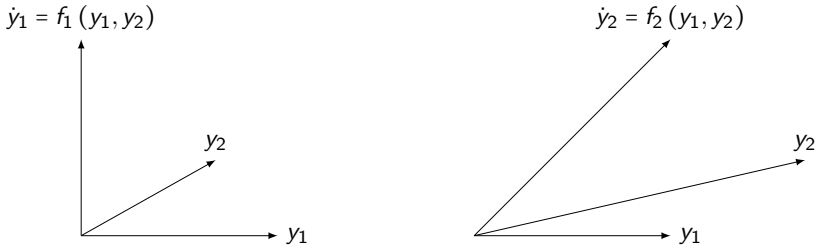

*https://www.geogebra.org/classic/mmppe6hs*

notes

- one can see the equilibria from the crossings of $\boldsymbol{f}$ with the 'yu plane' in case we have more dimensions

## Equilibra as the zeros of $\boldsymbol{f}$, graphically

Exemplified situation of automonous multiple output systems:




(remember: $\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}$ equilibrium iff $\boldsymbol{f}(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) = \boldsymbol{0}$, i.e., all the components simultaneously!)



notes

- note that if one has more than one $y$ then it must be $\dot{y}_i = 0$ for all the $i$'s simultaneously at the same point

equilibria = extremely important topic in automatic control!

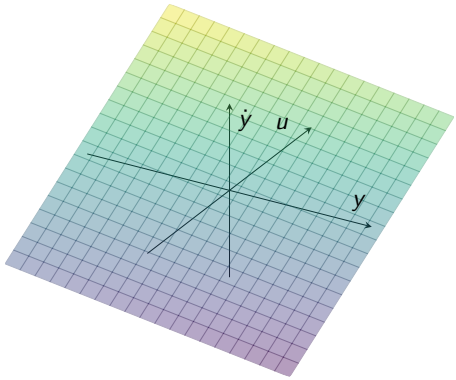
will be analysed in more details later on in this course
and much more extensively in others
*(feat. Lyapunov, Krasovskii, La-Salle among others)*

notes

- this module is on ODEs and not on equilibria; however better clarifying that this is an extremely important thing

## Discussion: what are the equilibria in this case?

$$\dot{y} = ay + bu \tag{3}$$




$(y_0 = 0, u = 0)$ is always an equilibrium for these systems!

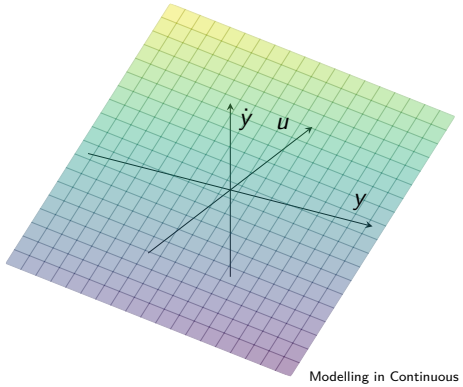


notes

- 
- $\dot{y} = 0$, so if $a = 0$ then any $y$ but it has to be $u = 0$, if $a \neq 0$ then it has to be so that $ay + bu = 0$
- it is also interesting to note this

## Discussion: can we have for this specific ODE an equilibrium if $u \neq$ constant?

$$\dot{y} = ay + bu \tag{4}$$



notes

- no

## Another example: a Lotka-Volterra model (≠ real world):

$$\begin{cases} \dot{y}_{\text{rabbits}} &= 0.4 \cdot y_{\text{rabbits}} - 0.5 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \\ \dot{y}_{\text{foxes}} &= -3 \cdot y_{\text{foxes}} + 0.7 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \end{cases}$$




notes

- this you should be immediately able to find an equilibrium, that is 0,0. The other one there is a bit of algebra to do but it comes natually

## Summarizing

**Compute** the equilibria of an ODE by solving for stationary points

- put $f = 0$ and compute the corresponding points. It may be that there is the need to put all the various inputs $u = \text{constant}$ (or disturbances $d = \text{constant}$)

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

## Root finding in python

`https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/chapter19.05-Root-Finding-in-Python.html`

notes

- this page explains how to find the zeros

Self-assessment material



notes

## Question 11

What is the mathematical definition of an equilibrium point for a dynamical system $\dot{y} = f(y, u)$?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | A point where $f(y, u)$ is maximized |
| II: | **(correct)** | A point where $f(y, u) = 0$ |
| III: | **(wrong)** | A point where $y$ is always increasing |
| IV: | **(wrong)** | A point where $y$ is always decreasing |
| V: | **(wrong)** | I do not know |

**Solution 1:**

An equilibrium point is defined as a point where the system's derivative $\dot{y}$ is zero, meaning the state remains constant over time.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 12

Which of the following statements about equilibrium points is correct?

**Potential answers:**

I: **(wrong)** An equilibrium point is always stable
II: **(correct)** An equilibrium point is where the system's state does not change over time
III: **(wrong)** An equilibrium point is a location where external inputs are irrelevant
IV: **(wrong)** An equilibrium point always corresponds to $y = 0$
V: **(wrong)** I do not know

**Solution 1:**

An equilibrium point is defined as a state where the time derivative of the system is zero, meaning the system does not evolve from that state unless disturbed. Stability depends on additional conditions.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 13

Graphically, how can equilibrium points be identified for an autonomous system $\dot{y} = f(y)$?

**Potential answers:**

I: **(wrong)** By finding where $y = 0$
II: **(correct)** By finding the points where $f(y) = 0$ on the phase plot
III: **(wrong)** By locating the steepest points of the function $f(y)$
IV: **(wrong)** By identifying the points where $y$ reaches its maximum or minimum values
V: **(wrong)** I do not know

**Solution 1:**

Equilibrium points occur where $\dot{y} = f(y) = 0$, which correspond to the intersections of $f(y)$ with the horizontal axis in a phase plot.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 14

Consider the system $\dot{T} = -0.5(T - 20)$. What is the equilibrium temperature?

**Potential answers:**

I: **(wrong)** $T = 0$
II: **(correct)** $T = 20$
III: **(wrong)** $T = -20$
IV: **(wrong)** $T = 40$
V: **(wrong)** I do not know

**Solution 1:**

Setting $\dot{T} = 0$ gives $-0.5(T - 20) = 0$, which simplifies to $T = 20$. This is the equilibrium point of the system.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 15

For the linear system $\dot{y} = ay + bu$, under what condition is $(y_0, u_0)$ an equilibrium?

**Potential answers:**

I: **(wrong)** When $a = 0$ only
II: **(correct)** When $ay_0 + bu_0 = 0$
III: **(wrong)** When $y_0 = 0$ and $u_0 = 0$ always
IV: **(wrong)** When $u_0$ is arbitrary
V: **(wrong)** I do not know

**Solution 1:**

To find an equilibrium, set $\dot{y} = 0$, leading to $ay + bu = 0$. Solving for $y_0$ and $u_0$ gives the equilibrium condition.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 16

If we have an autonomous time-varying ODE, can we have equilibria?

**Potential answers:**

I: **(wrong)** No, time-variation always prevents equilibria.
II: **(correct)** Yes, equilibria can exist if the system allows constant solutions.
III: **(wrong)** Only if the system is also linear.
IV: **(wrong)** Yes, but only if the system is also periodic.
V: **(wrong)** I do not know

**Solution 1:**

Equilibria exist when the derivative of the state variables is zero. Even though the system is time-varying, there can still be points where the vector field is zero, leading to equilibrium. For example, we may have $\dot{y} = ty$. When $y = 0$ we have an equilibrium anyway.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 17

Can we have dynamical systems that do not have any equilibria?

**Potential answers:**

I: **(wrong)** No, every system has at least one equilibrium.
II: **(correct)** Yes, with no fixed points may lack equilibria.
III: **(wrong)** Only non-autonomous systems can lack equilibria.
IV: **(wrong)** No, because every system must have at least a trivial equilibrium.
V: **(wrong)** I do not know

**Solution 1:**

Very simple example: $\dot{y} = 1$.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 18

If we have a non-autonomous ODE, can we have equilibria if the input is always changing, e.g., $u = \sin(t)$?

**Potential answers:**

I: **(wrong)** Yes, the input does not affect equilibrium conditions.
II: **(wrong)** No, because a changing input continuously affects system states.
III: **(wrong)** Only if the input has a zero mean.
IV: **(wrong)** Yes, but only if the system is linear.
V: **(correct)** Yes, and the system does not need to be linear.
VI: **(wrong)** I do not know

**Solution 1:**

A simple example: $\dot{y} = yu$. For $y = 0$ we have for sure an equilibrium independently of $u$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 19

If we have a non-autonomous LTI ODE, can we have equilibria if the input is always changing, e.g., $u = \sin(t)$?

**Potential answers:**

I: **(wrong)** Yes, because LTI systems always have equilibria.
II: **(correct)** No, because the continuously varying input prevents a steady state.
III: **(wrong)** Only if the system has no damping.
IV: **(wrong)** Yes, but only if the input is periodic.
V: **(wrong)** I do not know

**Solution 1:**

For an LTI system, equilibrium requires all derivatives to be zero. A continuously varying input like $u = \sin(t)$ forces the system to evolve dynamically, preventing equilibrium.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "compute the equilibria of the system"

- Equilibria in dynamical systems correspond to points where the system's state does not change over time.
- Autonomous time-varying ODEs can have equilibria, but their location may vary with time.
- Some dynamical systems may not have equilibria, particularly if they involve unbounded growth.
- Non-autonomous LTI ODEs can have equilibria only if the input $u(t)$ remains constant over time.



notes

- the most important remarks from this sub-module are these ones

# building and interpreting phase portraits

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| phase portrait | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |



notes

## Main ILO of sub-module "building and interpreting phase portraits"

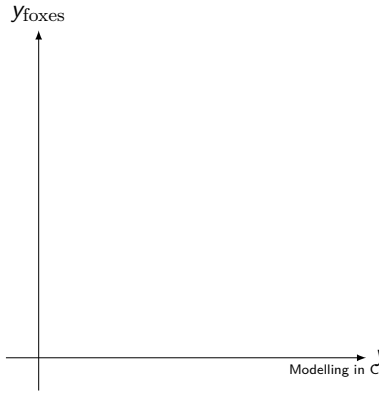
**Construct** and interpret phase portraits of first- and second-order autonomous ODEs using qualitative analysis techniques

notes

- by the end of this module you shall be able to do this

## Starting with an example: a Lotka-Volterra model (≠ real world):

$$\begin{cases} \dot{y}_{\text{rabbits}} &= 0.4 \cdot y_{\text{rabbits}} - 0.5 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \\ \dot{y}_{\text{foxes}} &= -3 \cdot y_{\text{foxes}} + 0.7 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \end{cases}$$



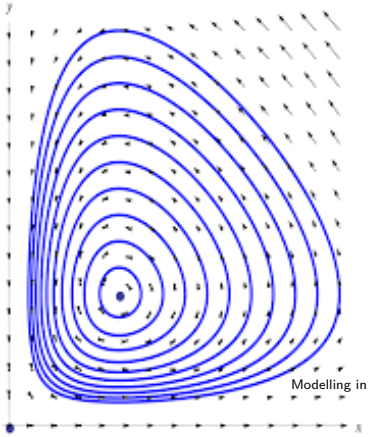



notes

- with this example we build things graphically, best to watch the video of the module

## The result, if we were plotting everything

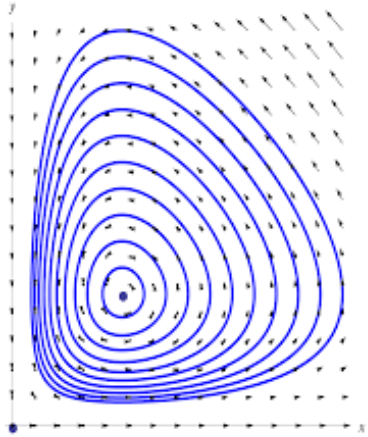
$$\begin{cases} \dot{y}_{\text{rabbits}} &= 0.4 \cdot y_{\text{rabbits}} - 0.5 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \\ \dot{y}_{\text{foxes}} &= -3 \cdot y_{\text{foxes}} + 0.7 \cdot y_{\text{rabbits}} \cdot y_{\text{foxes}} \end{cases}$$



notes

- if we were doing things graphically by hand we would eventually get this

## Phase Portrait = a graphical representation of the trajectories of a dynamical system in the state space

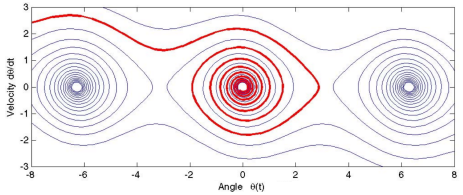




notes

- A phase portrait provides insight into the qualitative behavior of a system without requiring explicit solutions.
- It helps visualize equilibria, stability, and the general flow of solutions in state space.

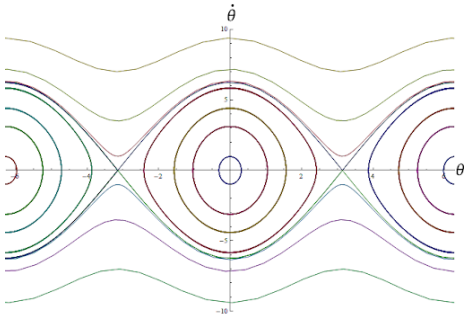
## Which system is this one?



notes

- pendulum with friction

## And this one?

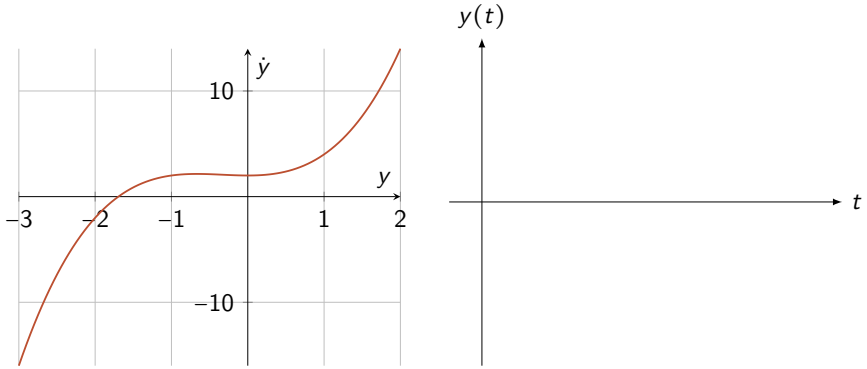





notes

- pendulum without friction

## Phase Portraits for first-order ODEs

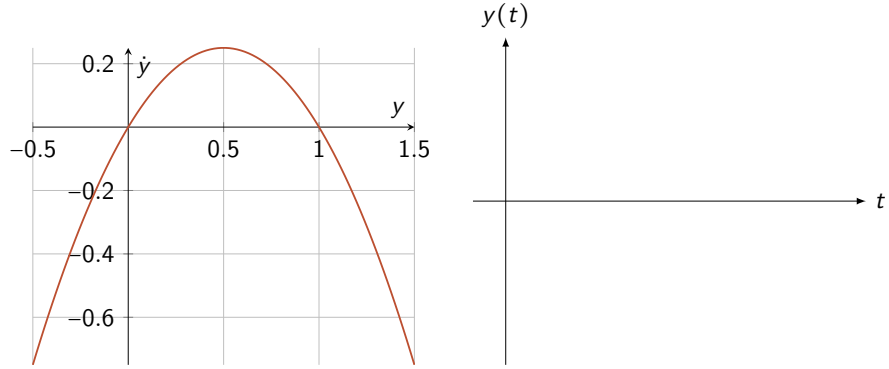



drawing the phase portrait as a 2-D thing in this case is a big error

notes

- The phase portrait consists of a one-dimensional state space (the $y$-axis).
- Equilibrium points are found by setting $f(y) = 0$.
- The sign of $f(y)$ determines the direction of motion.

## Another example: $\dot{y} = y(1 - y)$



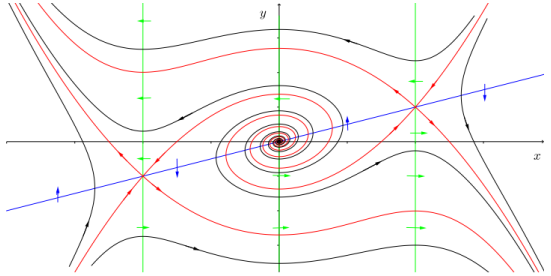

- equilibria: $y = 0$ and $y = 1$
- if $y < 0$ or $y > 1$, $\dot{y}$ is negative (flow left)
- if $0 < y < 1$, $\dot{y}$ is positive (flow right)



notes

- The stability of equilibrium points can be determined by the sign of $f(x)$.
- $x = 0$ is unstable, $x = 1$ is stable.

## Interpreting Phase Portraits

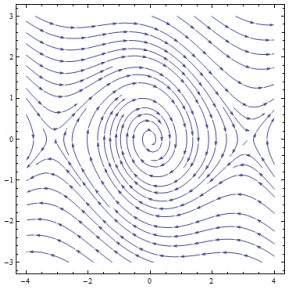



- equilibria: where trajectories do not move
- limit cycles: closed trajectories indicating periodic behavior

notes

- Understanding phase portraits helps predict long-term system behavior.
- Nonlinear systems may exhibit complex structures like attractors or chaotic dynamics.

## Interpreting Phase Portraits



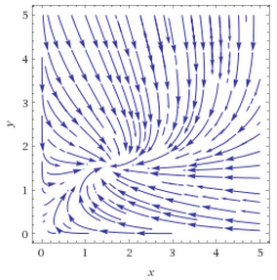
- equilibria: where trajectories do not move
- limit cycles: closed trajectories indicating periodic behavior



notes

- Understanding phase portraits helps predict long-term system behavior.
- Nonlinear systems may exhibit complex structures like attractors or chaotic dynamics.

## Interpreting Phase Portraits

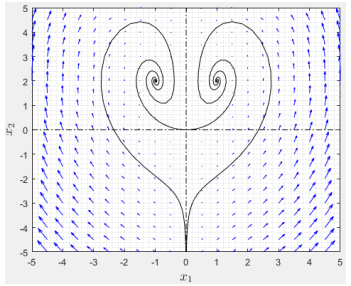



- equilibria: where trajectories do not move
- limit cycles: closed trajectories indicating periodic behavior

notes

- Understanding phase portraits helps predict long-term system behavior.
- Nonlinear systems may exhibit complex structures like attractors or chaotic dynamics.

## Interpreting Phase Portraits



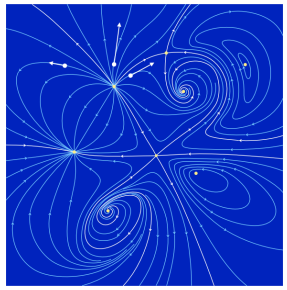

- equilibria: where trajectories do not move
- limit cycles: closed trajectories indicating periodic behavior



notes

- Understanding phase portraits helps predict long-term system behavior.
- Nonlinear systems may exhibit complex structures like attractors or chaotic dynamics.

## Interpreting Phase Portraits



- equilibria: where trajectories do not move
- limit cycles: closed trajectories indicating periodic behavior

notes

- Understanding phase portraits helps predict long-term system behavior.
- Nonlinear systems may exhibit complex structures like attractors or chaotic dynamics.

## Summarizing

**Construct** and interpret phase portraits of first- and second-order autonomous ODEs using qualitative analysis techniques

- do the plots using some opportune code
- have the domain expertise to be able to interpret why the trajectories do what they do



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Tutorial on how to plot phase portraits

`https://aleksandarhaber.com/phase-portraits-of-state-space-models-and-differential-equations-in-python/`



notes

- that tutorial is very well made, check it

## Self-assessment material

notes

## Question 20

What is the primary purpose of a phase portrait?

**Potential answers:**

I: **(wrong)** To find the exact numerical solution of a system
II: **(correct)** To visualize the qualitative behavior of a dynamical system
III: **(wrong)** To approximate the integral of a function
IV: **(wrong)** To determine the frequency response of a system
V: **(wrong)** I do not know

**Solution 1:**

A phase portrait is a graphical representation of the trajectories of a system in state space, giving insight into equilibrium points, stability, and system behavior without solving the equations explicitly.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 21

How do you determine equilibrium points in a phase portrait of a first-order system $\dot{y} = f(y)$?

**Potential answers:**

I: **(wrong)** By solving $\dot{y} = 0$ for all values of $t$
II: **(correct)** By solving $f(y) = 0$ for $y$
III: **(wrong)** By integrating $f(y)$ over time
IV: **(wrong)** By setting $f(y)$ to a constant value
V: **(wrong)** I do not know

**Solution 1:**

Equilibrium points are the values of $y$ where $\dot{y} = f(y) = 0$. These are points where the system remains at rest if not perturbed.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 22

Which of the following best describes the phase portrait of the system $\dot{y} = y(1 - y)$?

**Potential answers:**

I: **(wrong)** It consists of a single trajectory with no equilibrium points
II: **(correct)** It has two equilibrium points at $y = 0$ and $y = 1$, with flow directions determined by the sign of $f(y)$
III: **(wrong)** It has infinitely many equilibrium points
IV: **(wrong)** It has no equilibrium points and exhibits oscillatory behavior
V: **(wrong)** I do not know

**Solution 1:**

The function $f(y) = y(1 - y)$ has two roots at $y = 0$ and $y = 1$, which are the equilibrium points. The direction of flow depends on the sign of $f(y)$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 23

What distinguishes the phase portrait of a second-order system from a first-order system?

**Potential answers:**

I: **(wrong)** Second-order phase portraits only have one equilibrium point
II: **(correct)** Second-order phase portraits require a two-dimensional state space (e.g., $x$ vs. $\dot{x}$)
III: **(wrong)** First-order systems can have limit cycles, while second-order systems cannot
IV: **(wrong)** Phase portraits for second-order systems do not contain information about stability
V: **(wrong)** I do not know

**Solution 1:**



Second-order ODEs require a two-dimensional phase space, where each point represents both a position variable and its derivative. This allows visualization of

notes

- see the associated solution(s), if compiled with that ones :)

## Question 24

Which of the following statements about phase portraits of nonlinear systems is correct?

**Potential answers:**

I: **(wrong)** Nonlinear systems always have a single equilibrium point
II: **(wrong)** Nonlinear phase portraits can be analyzed only by solving the system numerically
III: **(correct)** Nonlinear phase portraits may exhibit equilibrium points, limit cycles, and chaotic behavior
IV: **(wrong)** Nonlinear phase portraits always resemble those of linear systems for small perturbations
V: **(wrong)** I do not know

**Solution 1:**



Nonlinear systems can have equilibrium points, periodic limit cycles, or even chaotic behavior depending on the dynamics. Their phase portraits often exhibit richer and more complex structures than linear systems.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "building and interpreting phase portraits"

- A phase portrait is a graphical representation of a dynamical systems trajectories in state space.
- Phase portraits provide qualitative insight into system behavior without requiring explicit solutions.
- First-order systems have a one-dimensional state space, while second-order systems require two dimensions, etc.

notes

- the most important remarks from this sub-module are these ones

# what is control



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| feedforward | u1, e1 |
| feedback | u1, e1 |
| model based control | u1, e1 |
| model free control | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |

notes

## Main ILO of sub-module "what is control"

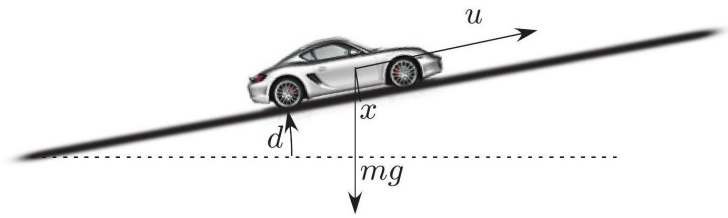
**Interpret** automatic control as an opportune operation on the dynamics of a system



notes

- by the end of this module you shall be able to do this

## Example: speed control



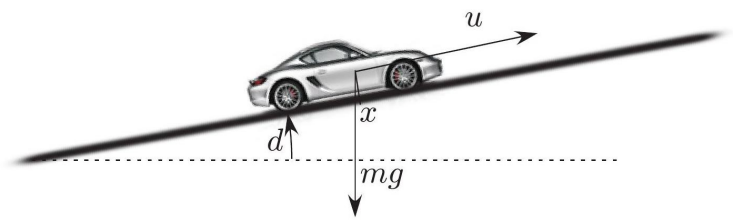

main parameters:

- $m$ mass of the car
- $b$ friction coefficient
- $g$ gravity coefficient

notes

- in this example we shall make the car keep a target speed independently of the inclination of the road

## A sufficiently accurate model for the purpose




from Newton laws: $m\ddot{x} = -b\dot{x} + u - mg\sin(d)$

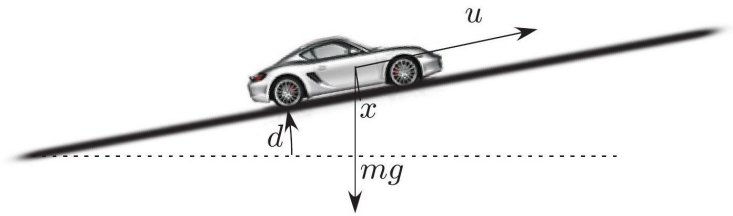
control objective: minimize $|y(t) - r(t)|$ with $r(t) =$ wished speed

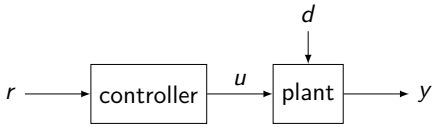


notes

- and (trust me) this is a good enough model for the purpose of designing a control law

## Feedforward control: "I think I know which $d(t)$ will happen, and I compensate for that"




$u(t)$ = something that compensates that $\widehat{d}(t)$

notes

- assuming we have an estimate / forecast $\widehat{d}(t)$, knowing the model we know how big we need $u$ to compensate that disturbance

## Open loop / feedforward control: "I think I know which $d(t)$ will happen, and I compensate for that"

$$u(t) = \text{something that compensates that } \widehat{d}(t)$$

problem: if $\widehat{d} - d$ is big, then we expect $y - r$ to be big too, and we won't be able to note this!

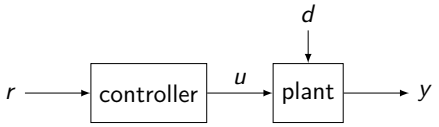


notes

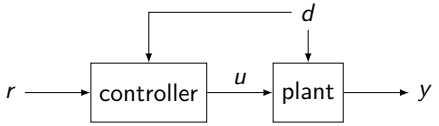
- of course pure feedforward / open loop control may be problematic, especially for the fact that there won't be ways to detect if there are big errors and react to them

## Note that open loop ≠ feedforward

Open loop:




Feedforward:



notes

- in summary, open-loop control works without feedback or adjustment based on the system's state, while feedforward control adjusts proactively based on expected disturbances or changes. Both are reactive in the sense that they dont correct in real-time based on the output, but feedforward aims to address known disturbances before they affect the system
- the differences will be more clear as we go on with the course

## Question 25

Open loop / feedforward control is so simple and naïve that no system in the world uses it

**Potential answers:**

I: **(wrong)** true
II: **(correct)** false
III: **(wrong)** I do not know

**Solution 1:**

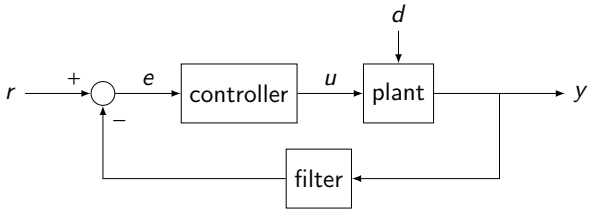
Absolutely false! Open-loop controllers are simple, cost-effective, and require no feedback, making them easy to design and implement. They are fast since they dont need to process sensor data, making them suitable for time-sensitive applications. However, they are less accurate and cannot correct for disturbances or system variations. Examples include a washing machine running a fixed cycle, a microwave heating for a set time, a traffic light operating on a fixed schedule, an irrigation system with a timer, and an electric kettle that shuts off based on time rather than temperature.



notes

- see the associated solution(s), if compiled with that ones :)

## Feedback control: "I measure something, and depending on what I measure I take a decision"




*(= designing a controller, i.e., in this case designing a function that maps the signal **e** into the signal **u**)*

notes

- note that this is again the block scheme of feedback control, that is NOT the unique way of doing control (there is also 'feedforward', for example)

## Main dichotomy on how to build a feedback controller

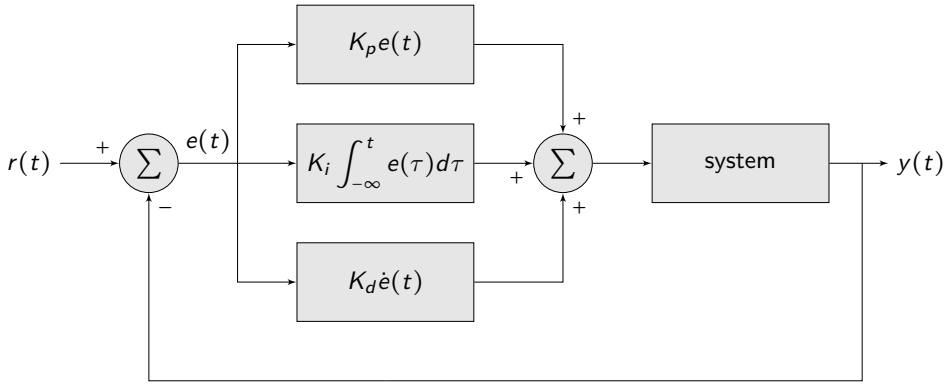
- model free *(e.g., PIDs)*
- model based *(e.g., MPCs)*



notes

- actually there are more details here you will discover while studying control systems. For now consider though just this dicothomy

## Crash-slide on PIDs

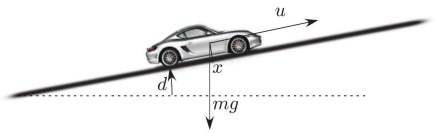



*implicit assumption: we can measure* $y(t)$*!* (see also `https://www.youtube.com/watch?v=UR0hOmjaHp0`!)

notes

- you will see this controller in big details in the next courses, for now let's only get some intuitions
- important thing: we need sensors and processing units, to be able to implement this. This means that we need to allocate money for buying and installing this piece of hardware - may be more expensive than open loop control

## PID for the speed control task




$u(t) =$ $K_p$ error right now
$+K_i$ sum of all past errors
$+K_d$ current tendency of the error




notes

- so from an intuitive perspective it is good to keep in mind that each component has a certain meaning
- tuning PIDs is an art. A nice book on this is this: `https://link.springer.com/book/10.1007/1-84628-586-0`

## Question 26

A PID is guaranteed to work well

**Potential answers:**

I: **(wrong)** yes, always
II: **(wrong)** no, never
III: **(correct)** no, it depends on how well tuned it is
IV: **(wrong)** I do not know

**Solution 1:**

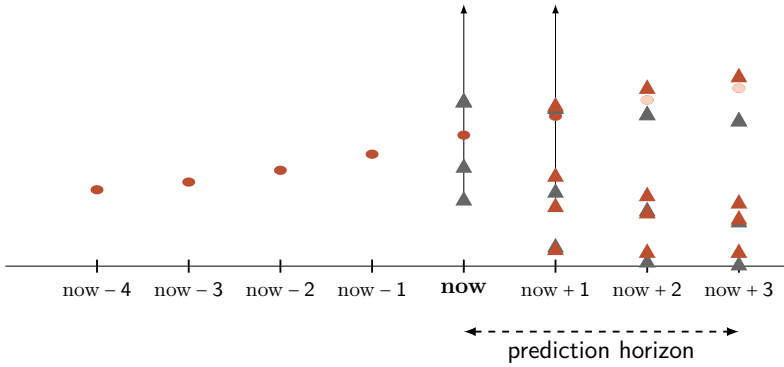
A PID controller is not guaranteed to work well in all cases; its performance depends on proper tuning. Poorly tuned PID controllers can lead to instability, slow response, or excessive oscillations. The three gains (proportional ($K_p$), integral ($K_i$), and derivative ($K_d$)) must be adjusted based on the system dynamics. For example, if $K_p$ is too high, the system may oscillate or become unstable. If $K_i$ is too high, the system may suffer from overshoot and integral windup. If $K_d$ is too high, the system may become too sensitive to noise.

notes

- see the associated solution(s), if compiled with that ones :)

## Crash-slide on MPCs

signals measured past $y$ wished future $y$ potential $u$, and corresponding forecasted $y$ best $u$ = "measure, and repeat"






notes

- see also https://www.youtube.com/watch?v=UR0h0mjaHp0!

## Question 27

A MPC is guaranteed to work well

**Potential answers:**

I: **(wrong)** yes, always
II: **(wrong)** no, never
III: **(correct)** no, it depends on how good the model is
IV: **(wrong)** I do not know

**Solution 1:**

Model Predictive Control (MPC) is not guaranteed to work well in all cases; its performance depends on how accurately the model represents the real system. Since MPC relies on predicting future system behavior using a model, inaccuracies in the model can lead to poor performance or instability.



Key factors affecting MPC performance:

- **Model Accuracy:** If the model does not capture the system dynamics well, predictions will be incorrect, leading to suboptimal control actions.

notes

- see the associated solution(s), if compiled with that ones :)

notes

- see the associated solution(s), if compiled with that ones :)

## Question 28

Is MPC guaranteed to work better than PID?

**Potential answers:**

I: **(wrong)** yes, always
II: **(wrong)** no, never
III: **(correct)** no, it actually depends on the situation
IV: **(wrong)** I do not know

**Solution 1:**

MPC is not always guaranteed to work better than PID; its effectiveness depends on the specific system and control objectives. While MPC offers advantages such as constraint handling, predictive capabilities, and optimization-based control, it also has drawbacks compared to PID.



Key factors influencing the choice between MPC and PID:

- **System Complexity:** PID works well for simple, well-modeled systems, while MPC is better suited for multivariable or highly constrained systems.
- **Computational Resources:** PID is computationally inexpensive and easy to implement, whereas MPC requires solving an optimization problem at each step, making it more demanding.
- **Tuning Effort:** While PID requires gain tuning, MPC requires model identification and tuning of multiple parameters, which can be complex.
- **Disturbances and Uncertainty:** MPC can anticipate and compensate for

## Question 29

Is closed loop control guaranteed to work better than open loop control?

**Potential answers:**

I: **(wrong)** yes, always
II: **(wrong)** no, never
III: **(correct)** no, it actually depends on the situation
IV: **(wrong)** I do not know

**Solution 1:**

Closed-loop control is not always guaranteed to work better than open-loop control; the effectiveness of each approach depends on the specific application and system characteristics. While closed-loop control provides feedback and can correct errors, open-loop control can be sufficient or even preferable in certain scenarios.



Key factors influencing the choice between open-loop and closed-loop control:

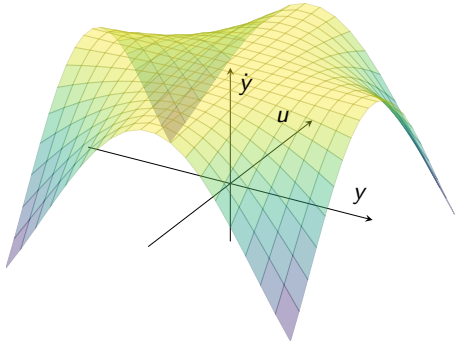
- **System Variability:** Closed-loop control is beneficial when the system is

notes

- see the associated solution(s), if compiled with that ones :)

## But eventually, what is control?

an algorithm to compute $u(t)$ starting from the available information

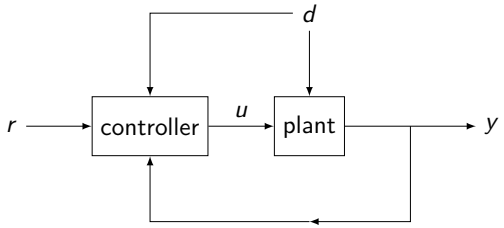





notes

- and there are many different algorithms to do so. A goal that you may put to yourself in these years is to arrive at knowing a sufficient number of them, and in details enough to have a feeling of the pros and cons, so to choose the right approach depending on the needs

## A final note: in practice it is a good choice to combine both feedback and feedforward actions



notes

- this block scheme shows that there is both a feedback and a feedforward action

## Summarizing

**Interpret** automatic control as an opportune operation on the dynamics of a system

- think at what feedforward and feedback mean
- think at the fact that essentially they are ways of computing $u$, and that that $u$ enters the dynamics of the system



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

Note: going through everything here would take months - just be aware of their existence and start playing with them

- `https://python-control.readthedocs.io/en/0.10.1/`
- `https://pypi.org/project/simple-pid/`
- `https://www.do-mpc.com/en/latest/`



notes

- these libraries all relate to what we saw in this module; watch out though that

Self-assessment material

notes

## Question 30

PID control requires a model of the system to function correctly.

**Potential answers:**

I: **(wrong)** yes, always
II: **(correct)** no, it works without a model
III: **(wrong)** I do not know

**Solution 1:**

A PID controller works without requiring a model of the system. Instead, it uses feedback from the systems output to adjust the control input. The three parameterproportional ($K_p$), integral ($K_i$), and derivative ($K_d$)are tuned based on system behavior, but no explicit system model is necessary. This makes PID controllers simple and widely applicable, even in systems where modeling is difficult or not feasible.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 31

Model Predictive Control (MPC) can only be applied when the model is perfect.

**Potential answers:**

I: **(wrong)** yes, the model must be perfect
II: **(correct)** no, it works with approximate models
III: **(wrong)** I do not know

**Solution 1:**

MPC does not require a perfect model, though its performance depends on how accurately the model represents the system. If the model is approximate, the controller may still work well, but the performance may degrade if the model is too far from reality. In practice, methods like robust or adaptive MPC are used to handle model inaccuracies and disturbances.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 32

Feedforward control is generally better than feedback control for handling disturbances.

**Potential answers:**

I: **(wrong)** yes, feedforward is always better
II: **(correct)** no, feedback control is better for disturbances
III: **(wrong)** I do not know

**Solution 1:**

Feedforward control can be effective when disturbances are predictable, as it compensates for them proactively. However, feedback control is generally better for handling unexpected disturbances or system changes because it can adjust in real-time based on the system's output. Feedback ensures that the system can respond to unmeasured or unforeseen variations, making it more robust in dynamic environments.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 33

Open-loop control is more reliable than closed-loop control in all situations.

**Potential answers:**

I: **(wrong)** yes, open-loop is always more reliable
II: **(correct)** no, it depends on the system and application
III: **(wrong)** I do not know

**Solution 1:**

Open-loop control is simpler and can be more reliable in cases where the system is predictable and not subject to disturbances. However, closed-loop control is more reliable when disturbances or system variations are present, as it adjusts based on feedback. The choice between open-loop and closed-loop control depends on the specific system dynamics, complexity, and performance requirements.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 34

PID controllers are always preferable to MPC in terms of performance.

**Potential answers:**

I: **(wrong)** yes, PID always outperforms MPC
II: **(correct)** no, it depends on the system and objectives
III: **(wrong)** I do not know

**Solution 1:**

PID controllers are well-suited for simple systems with few inputs and outputs, especially when the system is well-understood and not subject to significant disturbances. However, MPC can be more powerful in handling complex, multivariable systems with constraints. MPC is capable of optimizing system behavior over a time horizon, making it suitable for systems where PID controllers might not be able to effectively manage multiple interacting variables or constraints. Therefore, the choice depends on the system's complexity and control objectives.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "what is control"

- designing a controller means designing an algorithm that transforms information into decision
- there are several types of controllers, each with pros and cons
- taking decisions (i.e., actuating $u$) means modifying the dynamics of the system

notes

- the most important remarks from this sub-module are these ones

# how to linearize an ODE



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| linearization | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |

notes

## Main ILO of sub-module "how to linearize an ODE"

**Linearize** a nonlinear ODE around an equilibrium point



notes

- by the end of this module you shall be able to do this

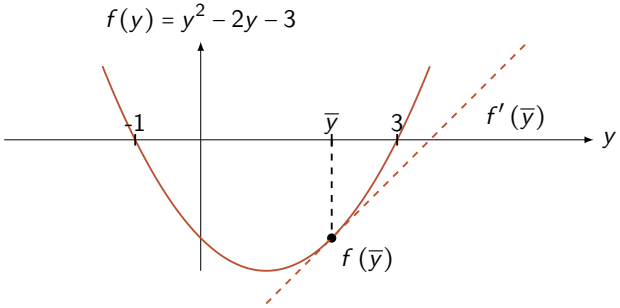
## The path towards linearizing a model

- what does linearizing a function mean?
- what does linearizing a model mean?
- how shall we linearize a model?

notes

- so now we will enter into the new content from this module
- we will proceed following this mental order in answering questions

## What does linearizing a scalar function mean?




$$f(y) \approx f(\overline{y}) + \left.\frac{\partial f}{\partial y}\right|_{\overline{y}} (y - \overline{y})$$

(but the approximation is valid only close to the linearization point)



notes

- graphically, it means substituting the function with an approximation that is a valid one in the neighborhood of a specific point
- in formulas we get this, that should be well known
- try to do by yourself the example of linearizing this function around $\overline{y} = 1$. You should get $f(y) = -4$ (constant)

## Obvious requirement

(but sometimes people forget about it . . . )

to compute the approximation

$$f(y) \approx f(\overline{y}) + \left.\frac{\partial f}{\partial y}\right|_{\overline{y}} (y - \overline{y})$$

the derivative of $f$ at $\overline{y}$ must be defined. *(notation: $f \in C^n$ means that $f$ has all its derivatives up to order $n$ defined in $\mathbb{R}$. $f \in C^n(\mathcal{X})$ means defined in $\mathcal{X} \subseteq \mathbb{R}$)*

notes

- in other words, you cannot linearize a function on a discontinuity point
- note: if you did not see this notation keep it in mind

## What does linearizing a vectorial function mean?

$$\boldsymbol{f}: \mathbb{R}^n \mapsto \mathbb{R}^m, \quad \boldsymbol{f} \in C^1 \quad \text{enables computing} \quad \boldsymbol{f}(\boldsymbol{y}) \approx \boldsymbol{f}(\boldsymbol{y}_0) + \nabla_{\boldsymbol{y}} \boldsymbol{f}\big|_{\boldsymbol{y}_0} (\boldsymbol{y} - \boldsymbol{y}_0)$$

*linearize* $\Longrightarrow$ *approximate each component!*

*Discussion:* then $\nabla_{\boldsymbol{y}} \boldsymbol{f}\big|_{\boldsymbol{y}_0}$ must be a matrix. Of which dimensions?



notes

- if we have a vectorial function we can think that this means linearizing each component
- $\boldsymbol{f}(\boldsymbol{y}_0)$ transforms a $n$-dimensional vector to a $m$-dimensional one. $(\boldsymbol{y} - \boldsymbol{y}_0)$ is a $n$-dimensional vector, thus that matrix must be $m$ rows and $n$ columns

## Example: linearize $\boldsymbol{f}$ around $\boldsymbol{y}_0$

$$\boldsymbol{f}(\boldsymbol{y}(t)) = \begin{bmatrix} \sin\left(y_1(t)\right) + \cos\left(y_2(t)\right) \\ \exp\left(y_1(t) y_2(t)\right) \end{bmatrix} \qquad \boldsymbol{y}_0 = \boldsymbol{y}(0) = [0, \pi]$$

notes

- if you do not feel able of doing this linearization then you should definitely refresh how to do derivatives
- The linearization involves computing the Jacobian matrix of $\boldsymbol{f}$ at $\boldsymbol{y}_0$. The Jacobian matrix is given by:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial}{\partial y_1}(\sin(y_1)+\cos(y_2)) & \frac{\partial}{\partial y_2}(\sin(y_1)+\cos(y_2)) \\ \frac{\partial}{\partial y_1}\exp(y_1 y_2) & \frac{\partial}{\partial y_2}\exp(y_1 y_2) \end{bmatrix}.$$

Evaluating the Jacobian at $\boldsymbol{y}_0 = [0, \pi]$ gives

$$\boldsymbol{J}(\boldsymbol{y}_0) = \begin{bmatrix} \cos(0) & -\sin(\pi) \\ \pi \cdot \exp(0) & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \pi & 0 \end{bmatrix}.$$

Thus, the linearized system around $\boldsymbol{y}_0$ is

$$\boldsymbol{J}(\boldsymbol{y}_0)\Delta\boldsymbol{y}$$

## And what if the vectorial function depends on more than one variable?

Assuming $\boldsymbol{f}$ differentiable in $\boldsymbol{y}_0, \boldsymbol{u}_0$,

$$\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \approx \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{u}_0) + \nabla_{\boldsymbol{y}} \boldsymbol{f}\big|_{\boldsymbol{y}_0, \boldsymbol{u}_0} (\boldsymbol{y} - \boldsymbol{y}_0) + \nabla_{\boldsymbol{u}} \boldsymbol{f}\big|_{\boldsymbol{y}_0, \boldsymbol{u}_0} (\boldsymbol{u} - \boldsymbol{u}_0)$$

with both $\nabla_{\boldsymbol{y}} \boldsymbol{f}\big|_{\boldsymbol{y}_0, \boldsymbol{u}_0}$ and $\nabla_{\boldsymbol{u}} \boldsymbol{f}\big|_{\boldsymbol{y}_0, \boldsymbol{u}_0}$ matrices of opportune size. Alternative notation:

$$\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \approx \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{u}_0) + \nabla \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_0, \boldsymbol{u}_0} \begin{bmatrix} \Delta \boldsymbol{y} \\ \Delta \boldsymbol{u} \end{bmatrix}$$

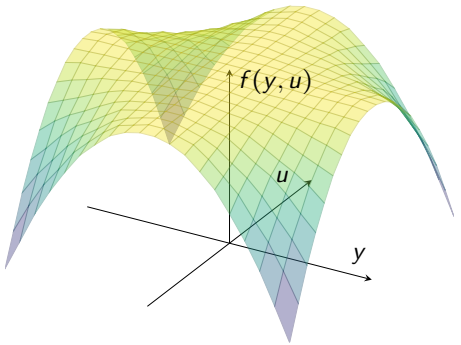


notes

- if we have more variables the same rules apply, just that we have more elements in the stack
- to save space it is convenient to write things in this way. This means defining

## Graphical example with a $\mathbb{R}^2 \mapsto \mathbb{R}$ function

$$f(y, u) \approx f(y_0, u_0) + \nabla f(y, u)\Big|_{y_0, u_0} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}$$




*if* $\boldsymbol{f} = [f_1, f_2]$ *then have two distinct plots, but the concept is the same*

notes

- see the plotting in the video of the course
- and if you got the point then this last sentence is obvious. If it does not seem obvious then this is a sign that actually better for you if you re-go through the material

## Thus, linearization = stopping the Taylor series at order one

$$f \in C^M(\mathbb{R}) \qquad \Longrightarrow \qquad f(y) \approx \sum_{m=0}^{M} \frac{f^{(m)}(y_0)}{m!} (y - y_0)^m$$

multivariable extension = less neat formulas, but the concept is the same. The most important case for our purposes:

$$\boldsymbol{f} \in C^1(\mathbb{R}^n, \mathbb{R}^m) \qquad \Longrightarrow \qquad \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \approx \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{u}_0) + \nabla_{\boldsymbol{y}} \boldsymbol{f}\big|_{\boldsymbol{y}_0} (\boldsymbol{y} - \boldsymbol{y}_0) + \nabla_{\boldsymbol{u}} \boldsymbol{f}\big|_{\boldsymbol{u}_0} (\boldsymbol{u} - \boldsymbol{u}_0)$$



notes

- actually what we are doing is using a Taylor expansion, and stopping at order one
- and what we eventually have is this thing

## What does linearizing an ODE mean?

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \qquad \approx \qquad \dot{\widetilde{\boldsymbol{y}}} = A\widetilde{\boldsymbol{y}} + B\widetilde{\boldsymbol{u}}$$

*linearize* $\Longrightarrow$ *approximate the dynamics!*

notes

- eventually if we think that a state-space model is a couple of functions, linearizing it means linearizing these functions
- as for the solution to the question, the fact is that $\boldsymbol{u}$ is the input, and in a simulation we may think that this variable is fixed. However if $\boldsymbol{u}$ is computed in closed loop, then the model itself will affect this variable, if the controller is model based. In this case we should write $\widetilde{\boldsymbol{u}}$ in the approximated model

## Discussion: what is the simplest way to make this linear?

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

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



notes

- note that sometimes one may do some simple tricks to do a linearization that is actuall a non-linearization
- for example here we may say that $\overline{u}(t) = u(t)^{2/3}$, and get a linear model in $\overline{u}$
- this basically would say just putting a nonlinear function in the block scheme, and with this "renaming" we may use linear control theory for this nonlinear system
- this trick though does not work always. It works only if we have the autonomous version of the system that is linear, and then the inputs appear as a sum of independent inputs each with its own nonlinearity

*Discussion:* why do we linearize nonlinear systems?

notes

- remember: we linearize a model because it may be more meaningful to do linear control than nonlinear control
- moreover in any case the linear approximation may be a good description, if the curvature of $\boldsymbol{f}$ is not too big and if we consider a sufficiently small neighborhood of the linearization point

*Discussion:* where do we linearize nonlinear systems?



notes

- it has no sense to linearize in a point that is not an equilibrium, because we would then consider a system for which the trajectories by default go away from the neighborhood where the approximation is meaningful
- moreover we want to do controllers typically around plant operation points, and operation points tend to be equilibria
- formally thus one may linearize everywhere (assuming that the maps are differentiable in that points) but in practice one does linearizations only around equilibria

## Linearization procedure - continuous time systems

$$(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) \text{ equilibrium} \implies \boldsymbol{f}(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) = 0$$

Procedure (assuming that the Taylor expansion exists):

- consider $\boldsymbol{y} = \boldsymbol{y}_{\text{eq}} + \Delta\boldsymbol{y}$, and $\boldsymbol{u} = \boldsymbol{u}_{\text{eq}} + \Delta\boldsymbol{u}$
- compute

$$\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \approx \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{u}_0) + \nabla_{\boldsymbol{y}}\boldsymbol{f}\big|_{\boldsymbol{y}_0}(\boldsymbol{y} - \boldsymbol{y}_0) + \nabla_{\boldsymbol{u}}\boldsymbol{f}\big|_{\boldsymbol{u}_0}(\boldsymbol{u} - \boldsymbol{u}_0)$$

setting though $\boldsymbol{y}_0 = \boldsymbol{y}_{\text{eq}}$

$$\implies \quad \frac{\partial(\boldsymbol{y}_{\text{eq}} + \Delta\boldsymbol{y})}{\partial t} \approx \boldsymbol{f}(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) + \nabla\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}} \begin{bmatrix} \Delta\boldsymbol{y} \\ \Delta\boldsymbol{u} \end{bmatrix}$$

note then that $\dfrac{\partial(\boldsymbol{y}_{\text{eq}} + \Delta\boldsymbol{y})}{\partial t} = \Delta\dot{\boldsymbol{y}}$ and that $\boldsymbol{f}(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) = \boldsymbol{0}$

notes

- to see the whole procedure, let's start considering that by definition this happens
- then one may compute this
- since we are working around the equilibrium, this happens
- but then given that $\boldsymbol{y}_{\text{eq}}$ is constant and given the fact that we are on an equilibrium, this follows

## Linearization procedure - continuous time systems

$$(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}) \text{ equilibrium} \implies$$

$$\Delta\dot{\boldsymbol{y}} \approx \nabla_{\boldsymbol{y}} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}} \Delta\boldsymbol{y} + \nabla_{\boldsymbol{u}} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}}} \Delta\boldsymbol{u}$$

and, since

- the two $\nabla$'s are matrices, and
- this is an approximate dynamics,

it follows that the approximated system is

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

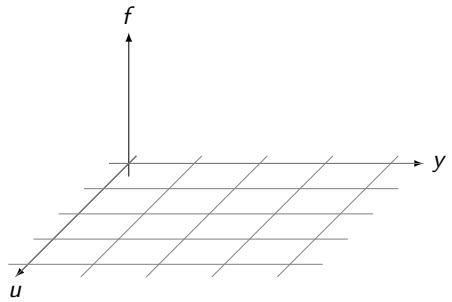


notes

- summarizing, we get this
- note that we are thus getting some equations that refer to an approximated system, and thus we should use different letters to indicate the different variables

## What does this mean graphically?

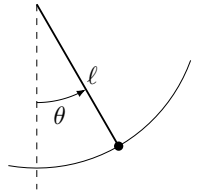
$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \qquad \text{vs.} \qquad \Delta\dot{\widetilde{\boldsymbol{y}}} = A\Delta\widetilde{\boldsymbol{y}} + B\Delta\widetilde{\boldsymbol{u}}$$



notes

- see the drawings made during the video of the lesson

## A from-start-to-end example: the pendulum




First step: equations of motion:

- gravity: $F_g = -mg\sin(\theta)$
- friction: $F_f = -f\ell\dot{\theta}$
- input torque: $F_u = u/\ell$

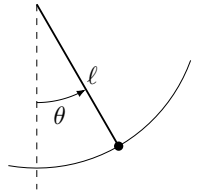
resulting dynamics: $m\ell\ddot{\theta} = -mg\sin(\theta) - f\ell\dot{\theta} + \frac{u}{\ell}$



notes

- let's consider a whole example, from start to end
- let's consider the pendulum example
- the system dynamics are these ones
- we can see that the system is nonlinear

## First step: transforming this in a state space system




thus from

$$m\ell\ddot{\theta} = -mg\sin(\theta) - f\ell\dot{\theta} + \frac{u}{\ell}$$

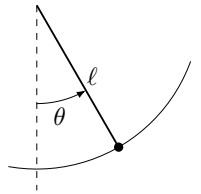
into

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -\frac{g}{\ell}\sin(y_1) - \frac{f}{m}y_2 + \frac{1}{m\ell^2}u$$

notes

- then we can rewrite the system into its state space form
- obviously here the states will be angular position and angular velocity

## Second step: finding the equilibria (assuming $u = 0$)




thus from

$$\begin{aligned}\dot{y}_1 &= y_2\\ \dot{y}_2 &= -\frac{g}{\ell}\sin(y_1) - \frac{f}{m}y_2 + \frac{1}{m\ell^2}u\end{aligned}$$

to

$$\begin{cases} 0 = y_2 \\ 0 = -\frac{g}{\ell}\sin(y_1) - \frac{f}{m}y_2 \end{cases} \quad \Longrightarrow \quad \boldsymbol{y}_{\text{eq.inst}} = \begin{bmatrix} \pi + 2k\pi \\ 0 \end{bmatrix}, \qquad \boldsymbol{y}_{\text{eq.st}} = \begin{bmatrix} 0 + 2k\pi \\ 0 \end{bmatrix}$$



notes

- then the equilibria follow immediately
- as intuition drives, we have several equilibria (all the upwards, and all the downwards). From practical perspectives we may just consider these two ones: $y_{\text{eq.st}} = [0, 0]^T$ and $y_{\text{eq.inst}} = \pi$

## Linearizing around the first equilibrium

$$\begin{aligned}\dot{y}_1 &= y_2\\ \dot{y}_2 &= -\frac{g}{\ell}\sin(y_1) - \frac{f}{m}y_2 + \frac{1}{m\ell^2}u\end{aligned}$$

linearizing around $\boldsymbol{y}_{\text{eq.st}} = [0, 0]^T$, $u = 0$ implies

$$A = \left[\begin{matrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{matrix}\right]\Bigg|_{\boldsymbol{y}_{\text{eq}\alpha}} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell} & -\frac{f}{m} \end{bmatrix}$$

$$B = \left[\begin{matrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{matrix}\right]\Bigg|_{\boldsymbol{y}_{\text{eq}\alpha}} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$$

notes

- then doing the computations we get these equivalences

## Linearizing around the second equilibrium

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -\frac{g}{\ell}\sin(y_1) - \frac{f}{m}y_2 + \frac{1}{m\ell^2}u$$

linearizing around $\boldsymbol{y}_{\text{eq}\beta} = [\pi, 0]^T$, $u = 0$ implies

$$A = \left[\begin{matrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{matrix}\right]\Bigg|_{\boldsymbol{y}_{\text{eq}\beta}} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{f}{m} \end{bmatrix}$$

$$B = \left[\begin{matrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{matrix}\right]\Bigg|_{\boldsymbol{y}_{\text{eq}\beta}} = \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}$$



notes

- and for the second case we get this

## The two linearized systems

Around the stable equilibrium: $\begin{bmatrix} \Delta\dot{y}_1 \\ \Delta\dot{y}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{\ell} & -\frac{f}{m} \end{bmatrix}\begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}u$

Around the unstable equilibrium: $\begin{bmatrix} \Delta\dot{y}_1 \\ \Delta\dot{y}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{f}{m} \end{bmatrix}\begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix}u$

the trajectories starting close to the stable equilibrium "stay around there", while the trajectories starting close to the unstable equilibrium "run away". This is because of the inner structure of the two state update matrices – another reason why we shall study linear algebra

notes

- summarizing, we get this
- and note then how just changing one sign in the $A$ matrices we have to get two systems that behave completely differently
- we definitely have to learn linear algebra

## Summarizing the procedure

- linearizing $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})$ is meaningful only around an equilibrium $(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}})$
- to find the equilibria of a system we need to solve $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) = \boldsymbol{0}$
- each equilibrium will lead to its "own" corresponding linear model $\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u}$, where $A$ and $B$ thus depend on $(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}})$ and $\boldsymbol{y}$, $\boldsymbol{u}$ in $\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u}$ have actually the meaning of $\Delta\boldsymbol{y}$, $\Delta\boldsymbol{u}$ with respect to the equilibrium
- each linearized model $\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u}$ is more or less valid only in a neighborhood of $(\boldsymbol{y}_{\text{eq}}, \boldsymbol{u}_{\text{eq}})$. Moreover the size of this neighborhood depends on the curvature of $\boldsymbol{f}$ around that specific equilibrium point



notes

- so, the first step to linearize a system is remembering that we do linearizations around equilibria
- then we should also remember how to compute equilibria
- as shown in the pendulum example above, each equilibrium leads to its own dynamics
- and we also have that the approximation is more or less valid, depending on different factors

## Recapping the rationale behind linearization

- linear systems are easier to analyze than nonlinear systems
- modal analysis and rational Laplace-transforms call for linear systems
- many advanced control techniques are based on linear systems

linearization = a very useful tool to do
analysis and design of control systems

notes

- and let's also spend a minute on saying again why we do this
- first of all, this is true
- moreover we will see soon that linear systems enable doing a lot of very useful analyses
- and you can get controllers performing very well with linear systems
- so, remember this

## Linearization - Another example

electrostatic microphone:

- $q$ = capacitor charge
- $h$ = distance of armature from its natural equilibrium
- $\boldsymbol{y} = \left[q, h, \dot{h}\right]$
- $R$ = circuit resistance
- $E$ = voltage generated by the generator (constant)
- $C$ = capacity of the capacitor
- $m$ = mass of the diaphragm + moved air
- $k$ = mechanical spring coefficient
- $\beta$ = mechanical dumping coefficient
- $u_1$ = incoming acoustic signal



notes

- this is another example that you may try by yourself

## Linearization - Another example

a physics-driven model:

$$\begin{cases} \dot{y}_1 &= -\dfrac{1}{Ra} y_1 \left(L + y_2\right) + \dfrac{E}{R} \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= -\dfrac{\beta}{m} y_3 - \dfrac{k}{m} y_2 - \dfrac{y_1^2}{2am} + \dfrac{1}{m} u_1 \end{cases}$$

notes

- without entering too much into details (you can also take a look at the corresponding example in the Fornasini Marchesini), this is a meaningful model of this system

## Linearization - Example

1-st step: compute the equilibria

$$\begin{cases} \dot{y}_1 &= -\dfrac{1}{Ra} y_1 (L + y_2) + \dfrac{E}{R} \\ \dot{y}_2 &= y_3 \\ \dot{y}_3 &= -\dfrac{\beta}{m} y_3 - \dfrac{k}{m} y_2 - \dfrac{y_1^2}{2am} + \dfrac{1}{m} u_1 \end{cases}$$

2-nd step: compute the matrices

$$A = \nabla_{\boldsymbol{y}} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{eq}, \boldsymbol{u}_{eq}} \quad B = \nabla_{\boldsymbol{u}} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{eq}, \boldsymbol{u}_{eq}} \quad C = \nabla_{\boldsymbol{y}} \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{eq}, \boldsymbol{u}_{eq}} \quad D = \nabla_{\boldsymbol{u}} \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{u})\Big|_{\boldsymbol{y}_{eq}, \boldsymbol{u}_{eq}}$$



notes

- try to do this by yourself!

## Summarizing

**Linearize** a nonlinear ODE around an equilibrium point

- find the equilibria
- select an equilibrium
- compute the derivatives around that equilibrium
- use the formulas
- don't forget that you are also changing the coordinate system!

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

This will do everything for you

`https://python-control.readthedocs.io/en/latest/generated/control.linearize.html`

though it is dangerous to use tools without knowing how they work

notes

- be always wary of using code without knowing the effects and meaning of the operations they do

Self-assessment material



notes

## Question 35

What does it mean to linearize a nonlinear ordinary differential equation (ODE)?

**Potential answers:**

I: **(correct)** It means approximating the nonlinear ODE with a linear model around an equilibrium point.

II: **(wrong)** It means replacing the ODE with a completely unrelated linear system.

III: **(wrong)** It means integrating the ODE analytically to find a closed-form solution.

IV: **(wrong)** It means ignoring all nonlinear terms in the system dynamics.

V: **(wrong)** I do not know

**Solution 1:**



Linearizing an ODE means approximating it with a linear model around an equilibrium point using a first-order Taylor series expansion. This allows for easier analysis and control design.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 36

What is the primary requirement for performing a valid linearization of a function?

**Potential answers:**

I: **(wrong)** The function must be polynomial.
II: **(correct)** The function must be differentiable at the point of linearization.
III: **(wrong)** The function must be bounded over the entire real line.
IV: **(wrong)** The function must have a second derivative at all points.
V: **(wrong)** I do not know

**Solution 1:**

A function must be differentiable at the point of linearization to compute its first-order Taylor series expansion, which is the basis for linearization.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 37

Why do we typically linearize a nonlinear system around an equilibrium point?

**Potential answers:**

I: **(wrong)** Because equilibrium points always yield globally valid linear models.
II: **(wrong)** Because nonlinear systems have no real solutions.
III: **(correct)** Because an equilibrium point ensures the validity of the local linear approximation.
IV: **(wrong)** Because linearization eliminates all system dynamics.
V: **(wrong)** I do not know

**Solution 1:**

Linearizing around an equilibrium point ensures that the approximation is meaningful in a small neighborhood, as the system is at rest or has steady-state behavior there.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 38

In a state-space representation of an ODE, what do the matrices $A$ and $B$ represent in the linearized system?

**Potential answers:**

I: **(wrong)** $A$ and $B$ are arbitrary matrices chosen for stability.
II: **(wrong)** $A$ represents the second derivative of the state, and $B$ represents the system's damping.
III: **(wrong)** $A$ and $B$ are obtained by solving the system for eigenvalues and eigenvectors.
IV: **(correct)** $A$ is the Jacobian of the system dynamics with respect to the state, and $B$ is the Jacobian with respect to the input.
V: **(wrong)** I do not know

**Solution 1:**



The matrices $A$ and $B$ in a linearized state-space model come from computing the Jacobian matrices of the system dynamics with respect to the state and input, respectively, at the equilibrium point.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 39

Which of the following is a common limitation of linearizing a nonlinear system?

**Potential answers:**

I: **(correct)** The linearized model is only valid in a small neighborhood around the linearization point.
II: **(wrong)** The linearized model has no practical applications in control.
III: **(wrong)** Linearization makes the system unstable.
IV: **(wrong)** Linearization eliminates all dynamic behavior of the system.
V: **(wrong)** I do not know

**Solution 1:**

A linearized model is typically valid only in a small neighborhood around the equilibrium point where it was derived. If the system deviates significantly from this region, the approximation may no longer be accurate.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "how to linearize an ODE"

- linearization requires following a series of steps (see the summary above)
- the model that one gets in this way is an approximation of the original model
- having a graphical understanding of what means what is essential to remember how to do things
- better testing a linear controller before a nonlinear one



notes

- the most important remarks from this sub-module are these ones

## when is linearizing meaningful

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| linearization | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |



notes

## Main ILO of sub-module "when is linearizing meaningful"

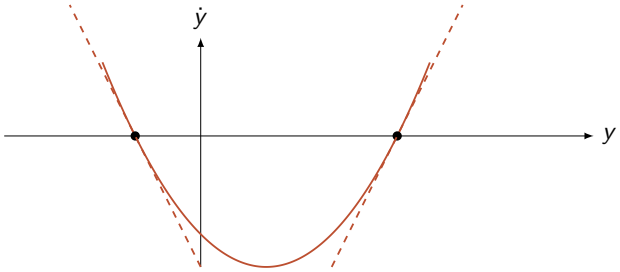
**Assess** the validity of the approximation introduced when linearizing a nonlinear ODE around an equilibrium point

**Evaluate** the meaning and applicability of linearization in different contexts, discussing when it provides a reasonable approximation and when it does not

notes

- by the end of this module you shall be able to do this

## Discussion: around which equilibrium may we consider this model approximation a "good one"?



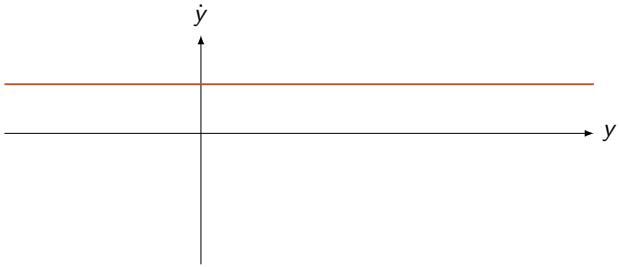

also for the 'unstable' equilibrium the approximation may be a good one - depends on the time horizon under consideration and how close $y_0$ is to the equilibrium



notes

- the fact that $A$ and $\boldsymbol{f}$ are different (even if one is the approximation of the other one) means that, if we think at their physical meaning, starting from the same point the two models give different indications towards where $\boldsymbol{y}$ should go
- this means that the trajectories will be different
- how much different, though? Depends of course on some sort of distance between $A$ and $\boldsymbol{f}$
- for the asymptotically stable equilibrium the approximation will get better and better in time; for the unstable equilibrium worse and worse in time
- recall though that one may consider an arbitrarily small neighborhood of the approximation point. In this way one may think that the linearized version may be an arbitrarily good approximation, if one focuses in a sufficiently small neighborhood
- in this course we will not see how to compute bounds of the error between these two trajectories; you will do it in later on courses

## Discussion: is it always meaningful to linearize?

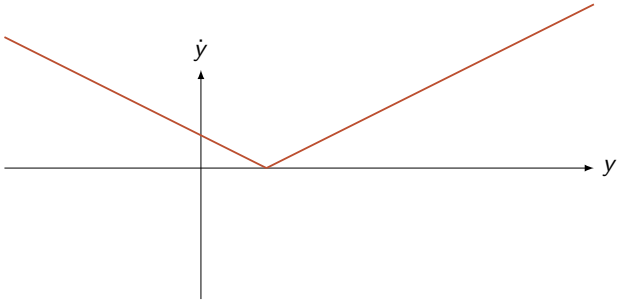



in this case we do not have equilibria

notes

- a simple case that shows that if we do not have equilibria actually we can't linearize

## Discussion: and here, can we linearize?



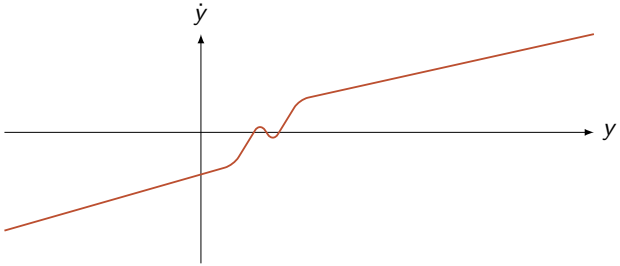

in this case we cannot compute the first derivative



notes

- a simple case that shows that if we do not have continuity for the derivative actually we can't linearize

## Discussion: can we trust the stable linearized system for this case?




in this case the basin of attraction is very small

notes

- a simple case that shows that if the basin of attractions are very small, the linearized system may be trusted in a very small region

## Summarizing

**Assess** the validity of the approximation introduced when linearizing a nonlinear ODE around an equilibrium point

**Evaluate** the meaning and applicability of linearization in different contexts, discussing when it provides a reasonable approximation and when it does not

- if we have an asymptotically stable equilibrium, the approximation improves in time
- if we have an unstable equilibrium, the approximation degrades in time
- the closer we start from the equilibrium, the better
- the bigger the curvature of the ODE, the more "local" the results will be



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## This will do everything for you

`https://python-control.readthedocs.io/en/latest/generated/control.linearize.html`

though it is dangerous to use tools without knowing how they work



notes

- be always wary of using code without knowing the effects and meaning of the operations they do

# Self-assessment material

notes

## Question 40

When linearizing a nonlinear ODE around an equilibrium point, which of the following conditions ensures that the approximation improves over time?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The equilibrium point is unstable. |
| II: | **(correct)** | The equilibrium point is asymptotically stable. |
| III: | **(wrong)** | The ODE has a high curvature near the equilibrium point. |
| IV: | **(wrong)** | The initial point is far from the equilibrium. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

The approximation improves over time when the equilibrium point is asymptotically stable. This is because trajectories near such equilibria converge toward the equilibrium, making the linearized model increasingly accurate.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 41

In which of the following cases is it NOT meaningful to linearize a nonlinear ODE?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The ODE has multiple equilibrium points. |
| II: | **(correct)** | The ODE does not have any equilibrium points. |
| III: | **(wrong)** | The ODE has a small basin of attraction. |
| IV: | **(wrong)** | The ODE is highly nonlinear. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

Linearization is not meaningful when the ODE does not have any equilibrium points, as the process of linearization relies on approximating the system near an equilibrium.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 42

Which of the following factors limits the validity of a linearized ODE approximation?

**Potential answers:**

I: **(wrong)** The linearized system has a stable equilibrium.
II: **(correct)** The basin of attraction of the equilibrium is very small.
III: **(wrong)** The ODE is continuous and differentiable.
IV: **(wrong)** The initial point is close to the equilibrium.
V: **(wrong)** I do not know.

**Solution 1:**

A very small basin of attraction limits the validity of the linearized approximation, as the region where the approximation holds becomes very restricted.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 43

What happens to the accuracy of a linearized ODE approximation near an unstable equilibrium point over time?

**Potential answers:**

I: **(correct)** The approximation degrades over time.
II: **(wrong)** The approximation improves over time.
III: **(wrong)** The accuracy remains constant.
IV: **(wrong)** The accuracy depends on the curvature of the ODE.
V: **(wrong)** I do not know.

**Solution 1:**

Near an unstable equilibrium, the approximation degrades over time because trajectories diverge from the equilibrium, making the linearized model less accurate.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 44

Which of the following statements about linearization is true?

**Potential answers:**

I: **(wrong)** Linearization is always a good approximation for any nonlinear ODE.

II: **(correct)** Linearization provides a better approximation when the initial point is closer to the equilibrium.

III: **(wrong)** Linearization is only valid for ODEs with high curvature.

IV: **(wrong)** Linearization cannot be applied to stable systems.

V: **(wrong)** I do not know.

**Solution 1:**

Linearization provides a better approximation when the initial point is closer to the equilibrium, as the linearized model is most accurate in a small neighborhood around the equilibrium.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "when is linearizing meaningful"

- be careful when using a linearized system - be always aware of where it comes from

notes

- the most important remarks from this sub-module are these ones

what is the superposition principle, and what does it imply



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| superposition principle | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| LTI ODE | u1, e1 |

notes

## Main ILO of sub-module

## "what is the superposition principle, and what does it imply"

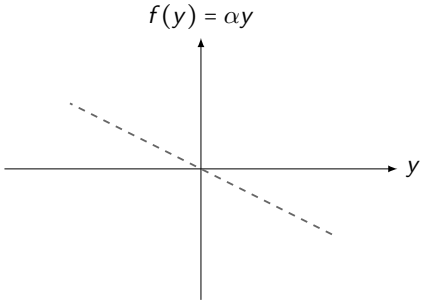
**Describe** the importance of the superposition principle to analyze LTI systems



notes

- by the end of this module you shall be able to do this

## Starting with graphs



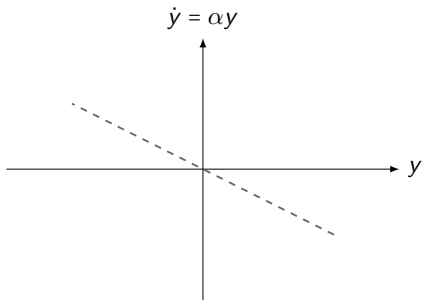

implications/definition of linearity:

- $f(x+y) = f(x) + f(y)$
- $f(\alpha y) = \alpha f(y)$

notes

- looking at this graph, we note that these two properties hold
- and this holds only because of linearity. If we are having an affine map, for example, the second would not hold

## What if we interpret this as an ODE?




$\Longrightarrow$ an LTI system, for which

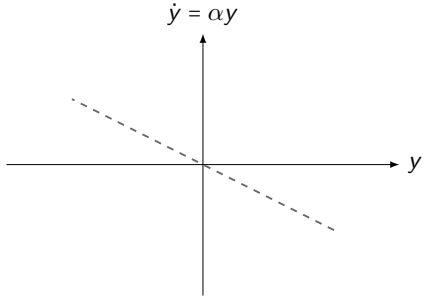
$$\dot{y} = \alpha y \quad \text{is solved by} \quad y(t) = y(0)\exp(\alpha t) \quad \forall y(0), \alpha, t$$



notes

- you may verify this solution by doing the direct verification

## And can we build on top of this?



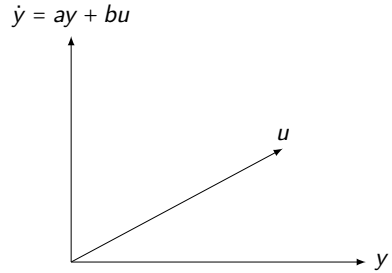

- $y'(0) = 2 \mapsto y'(t) = 2\exp(\alpha t)$
- $y''(0) = 3 \mapsto y''(t) = 3\exp(\alpha t)$
- $y'''(0) = 3 + 2 \mapsto y'''(t) = (3 + 2)\exp(\alpha t)$

$$y'(0) \; + \; y''(0) \quad \mapsto \quad y'(t) \; + \; y''(t)$$

notes

- looking at what we found in the previous module, this holds because the solutions to linear ODEs are exponentials passing by the initial conditions and whose exponent is always $a$
- and this holds only because of linearity

## Further generalization




- $\{y'(0), u'\} \mapsto y'(t)$
- $\{y''(0), u''\} \mapsto y''(t)$
- $\{y'(0) + y''(0), u' + u''\} \mapsto y'(t) + y''(t)$



notes

- we can then generalize the previous result in this way, when we have linearity

## Aiding intuitions with math

Linearity implies that if $\{y', u', y'(0)\}$ and $\{y'', u'', y''(0)\}$ satisfy

$$\begin{cases} \dfrac{dy'(t)}{dt} & = ay'(t) + bu'(t) \\ y'(0) & = y_0' \\ \dfrac{dy''(t)}{dt} & = ay''(t) + bu''(t) \\ y''(0) & = y_0'' \end{cases} \tag{5}$$

then their sum also satisfies

$$\begin{cases} \dfrac{d\big(\alpha' y'(t) + \alpha'' y''(t)\big)}{dt} & = a\big(\alpha' y'(t) + \alpha'' y''(t)\big) + b\big(\alpha' u'(t) + \alpha'' u''(t)\big) \\ \alpha' y'(0) + \alpha'' y''(0) & = \alpha' y_0' + \alpha'' y_0'' \end{cases} \tag{6}$$

notes

- let's start realizing that this holds because derivatives hold

## Rephrasing

Linearity implies that if $\{y', u', y'(0)\}$ and $\{y'', u'', y''(0)\}$ satisfy the ODE then also their sum $\{y' + y'', u' + u'', y'(0) + y''(0)\}$ satisfies the ODE.

### The superposition principle in words

*in LTI systems*
*combining inputs and initial conditions*
*produces a total effect*
*that is the linear combination*
*of that effects*
*one would get with the individual causes*
*each acting separately*



notes

- this is the same thing in the previous slide, written in words
- then the previous math basically says this

## Important: the superposition principle works with any LTI

Will be repeated and stated again precisely later on

the proof holds for every system that generalizes $\dot{y} = ay + bu$,
i.e., every "linear combination of dots of $y$ = linear combination of dots of $u$"

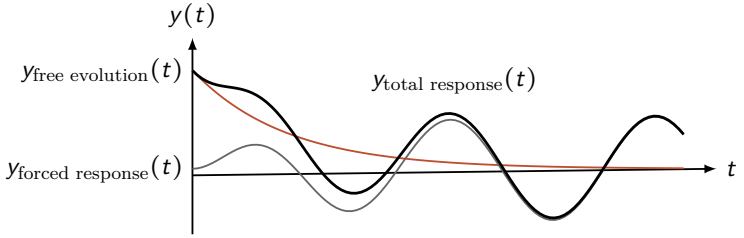
notes

- we have moreover this generalization, that we will see again and again, since derivatives are linear

## Superposition principle $\Longrightarrow$ response of LTIs = free evolution + forced response

assume:

- $\dot{y} = ay + bu$
- $\{u(t) = 0(t), \quad y(0) \neq 0\}$ causes $y_{\text{free evolution}}(t)$
- $\{u(t) \neq 0(t), \quad y(0) = 0\}$ causes $y_{\text{forced response}}(t)$




then $\{u(t) \neq 0(t), y(0) \neq 0\}$ causes $y_{\text{free evolution}}(t) + y_{\text{forced response}}(t)$



notes

- the generalization in the previous slide can immediately be useful to show something very important, i.e., that we can decompose the general solution in to main parts, as here

## A mnemonic scheme

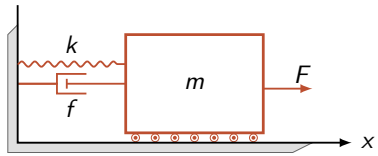
(only for LTI systems!!)

$$(u, y_0) = (0, y_0) + (u, 0)$$

*total response = free evolution + forced response*

notes

## Continuing with some intuitions



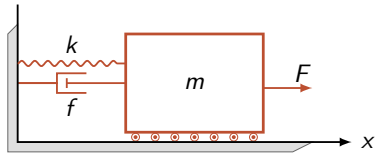

Discussion: how will the cart move if I use $u(t) = \sin(\omega t)$ starting from a resting state? (only intuitively, assuming everything ideal)
And what about if $u(t) = 2\sin(\omega t)$?
And what about $u(t) = \sin(\omega' t) + \sin(\omega'' t)$?
And what about $u(t) = \alpha' \sin(\omega' t) + \alpha'' \sin(\omega'' t)$?



notes

- we will see later on precisely; for now let's say that it moves somehow
- here intuition may not help: anyway it makes the same movement as before but with twice the amplitude
- here intuition may again not help: it makes the sum of the two movements
- here intuition may again not help: it makes the sum of the two movements scaled

## Refining the intuitions




Assume to have measured

$$\boldsymbol{y}'(0), u'(t) \mapsto y'(t) \qquad \boldsymbol{y}''(0), u''(t) \mapsto y''(t)$$

Saying "this system is linear" means assuming $\forall \alpha', \alpha'' \in \mathbb{R}$

$$\alpha' \boldsymbol{y}'(0) \alpha' \boldsymbol{y}''(0), u'(t) \mapsto y'(t) \qquad \boldsymbol{y}''(0), u''(t) \mapsto y''(t)$$

thus assuming that from a resting state the input $u(t) = \alpha \sin(\omega_\alpha t) + \beta \sin(\omega_\beta t)$ causes $y(t) = \alpha y_{\omega\alpha}(t) + \beta y_{\omega\beta}(t)$

notes

- so this is the solution to the previous question

## Summarizing

**Describe** the importance of the superposition principle to analyze LTI systems

- it makes us able to say "total = free + forced"



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Suggestion

part of the SciPy library (`scipy.signal`) provides tools for working with LTI systems, including creating transfer functions, state-space representations, and analyzing system responses (stuff that will be seen in the next modules)



notes

- check this library, you will use it

# Self-assessment material

notes

## Question 45

What does the superposition principle imply for LTI systems?

**Potential answers:**

I: **(wrong)** The total response is the product of the free evolution and forced response.
II: **(correct)** The total response is the sum of the free evolution and forced response.
III: **(wrong)** The total response is independent of the initial conditions.
IV: **(wrong)** The total response is only determined by the input.
V: **(wrong)** I do not know.

**Solution 1:**

The superposition principle implies that the total response of an LTI system is the sum of the free evolution (response due to initial conditions) and the forced response (response due to the input). This is a fundamental property of linear systems.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 46

Which of the following is a necessary condition for the superposition principle to hold in a system?

**Potential answers:**

I: **(wrong)** The system must be nonlinear.
II: **(correct)** The system must be linear and time-invariant.
III: **(wrong)** The system must have time-varying parameters.
IV: **(wrong)** The system must be unstable.
V: **(wrong)** I do not know.

**Solution 1:**

The superposition principle holds only for Linear Time-Invariant (LTI) systems. Nonlinear or time-varying systems do not satisfy the superposition principle.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 47

What is the free evolution of an LTI system?

**Potential answers:**

I: **(wrong)** The response of the system to a nonzero input with zero initial conditions.
II: **(correct)** The response of the system to zero input with nonzero initial conditions.
III: **(wrong)** The steady-state response of the system.
IV: **(wrong)** The transient response of the system.
V: **(wrong)** I do not know.

**Solution 1:**

The free evolution of an LTI system is the response of the system when the input is zero, but the initial conditions are nonzero. It represents the system's natural behavior without external forcing.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 48

If an LTI system has an input $u(t) = \alpha' u'(t) + \alpha'' u''(t)$ and initial conditions $y(0) = \alpha' y'(0) + \alpha'' y''(0)$, what is the total response $y(t)$?

**Potential answers:**

I: **(wrong)** $y(t) = \alpha' y'(t) \cdot \alpha'' y''(t)$
II: **(correct)** $y(t) = \alpha' y'(t) + \alpha'' y''(t)$
III: **(wrong)** $y(t) = \alpha' y'(t) - \alpha'' y''(t)$
IV: **(wrong)** $y(t) = \alpha' y'(t) / \alpha'' y''(t)$
V: **(wrong)** I do not know.

**Solution 1:**

For an LTI system, the total response $y(t)$ is the linear combination of the individual responses $y'(t)$ and $y''(t)$, scaled by $\alpha'$ and $\alpha''$, respectively. This is a direct consequence of the superposition principle.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 49

What is the forced response of an LTI system?

**Potential answers:**

I: **(correct)** The response of the system to a nonzero input with zero initial conditions.

II: **(wrong)** The response of the system to zero input with nonzero initial conditions.

III: **(wrong)** The response of the system to a step input.

IV: **(wrong)** The response of the system to a sinusoidal input.

V: **(wrong)** I do not know.

**Solution 1:**

The forced response of an LTI system is the response of the system when the input is nonzero, but the initial conditions are zero. It represents the system's behavior due to external forcing.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module

## "what is the superposition principle, and what does it imply"

- superposition principle helps logically separating specific causes into specific effects
- linear ODEs $\Longrightarrow$ superposition principle
- superposition principle $\Longrightarrow$ "whole = free + forced"
- nonlinear systems WON'T satisfy this principle!

notes

- the most important remarks from this sub-module are these ones

# what is an impulse response



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| Dirac delta | u1, e1 |
| impulse response | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| superposition principle | u1, e1 |
| LTI ODE | u1, e1 |

notes

## Main ILO of sub-module "what is an impulse response"

**Describe** what the impulse response of an LTI system is in practice



notes

- by the end of this module you shall be able to do this

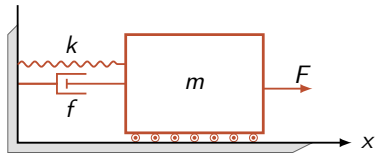
## Impulse response ↔ superposition principle ↔ LTI system

talking about the impulse response of a nonlinear system is such a big mistake that may make you fail the exam on the spot

notes

- not joking here; we will see in this module that the concept is a direct consequence, and not having understood that an impulse response is meaningful only for LTI systems indicates a complete misunderstanding of the basis of the course

## Practical example: spring-mass system



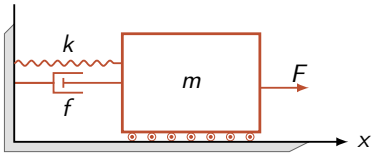

- output = position
- input = force (in Newtons)

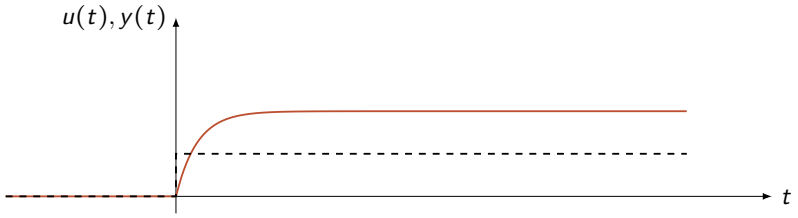


notes

- In the mass-spring-damper system the impulse is a quick push (force times time) applied to the mass. The impulse response is the displacement of the mass over time after the push. The units of the impulse response are m/Ns.

## What if I push the cart with a force of 1 Newton?

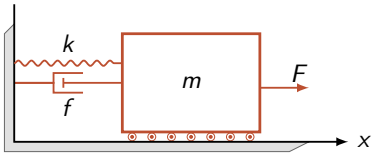


notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 0.5 Newtons?



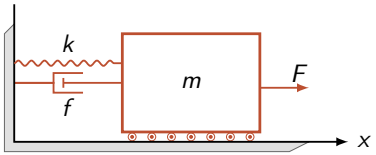


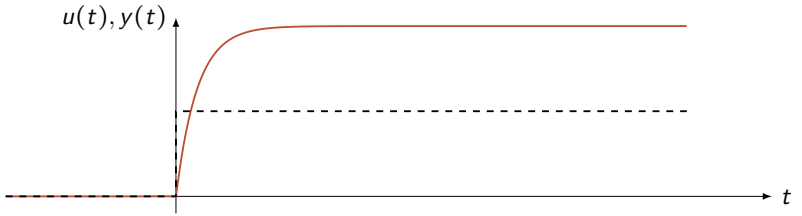


notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 2 Newtons?

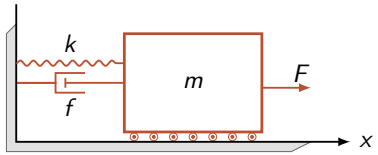


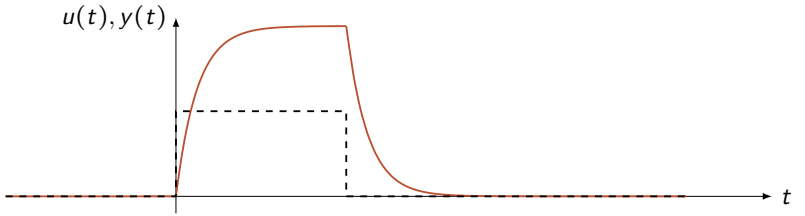



notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 2 Newtons for 2 seconds?



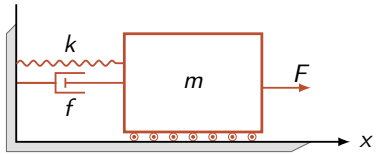



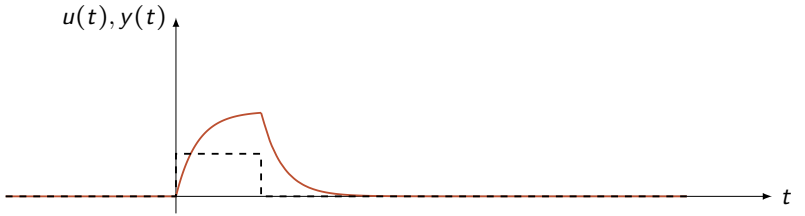



notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 1 Newton for 1 second?

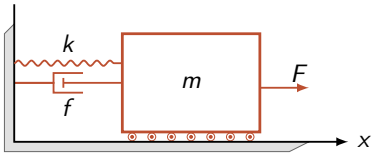


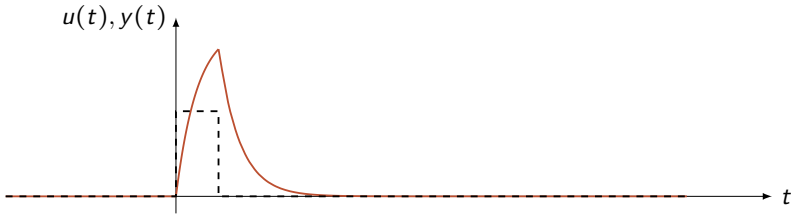



notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 2 Newtons for 0.5 seconds?



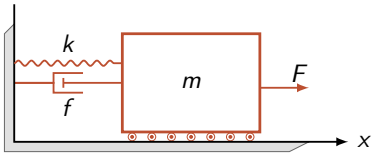



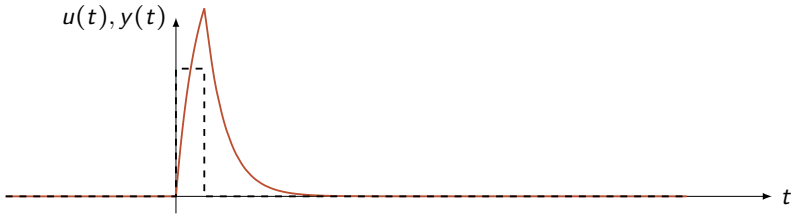



notes

- note that here the position is $x$, but also $y = x$

## What if I push the cart with a force of 3 Newtons for 1/3 of a second?

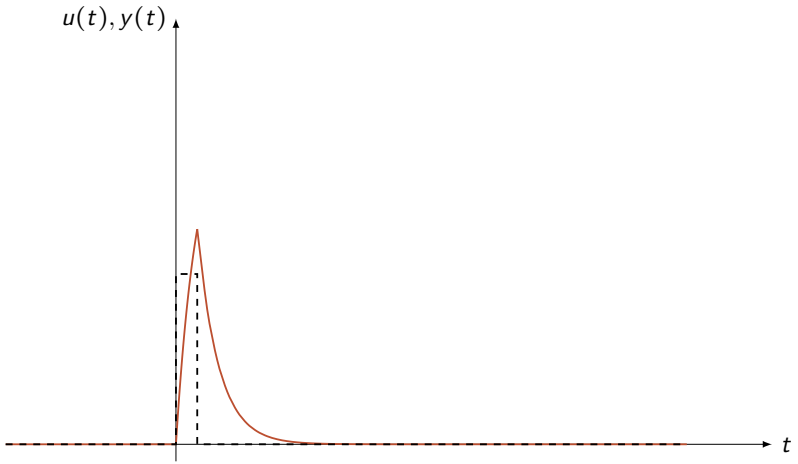


notes

- note that here the position is $x$, but also $y = x$

## 4 Newtons for 1/4 of a second?

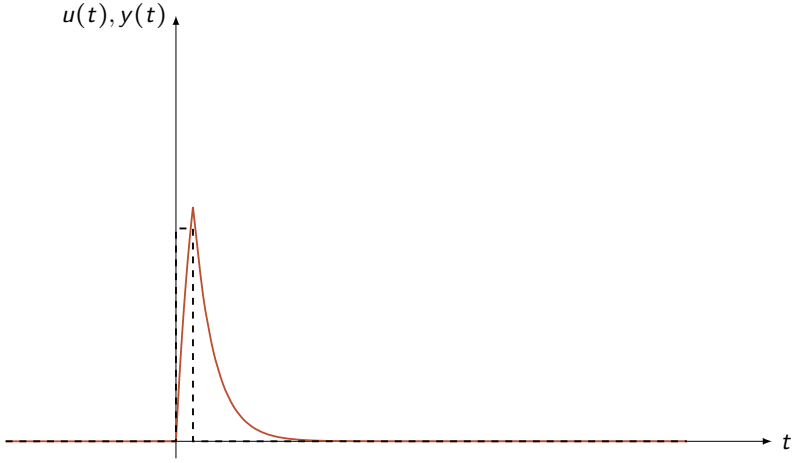





notes

- note that here the position is $x$, but also $y = x$

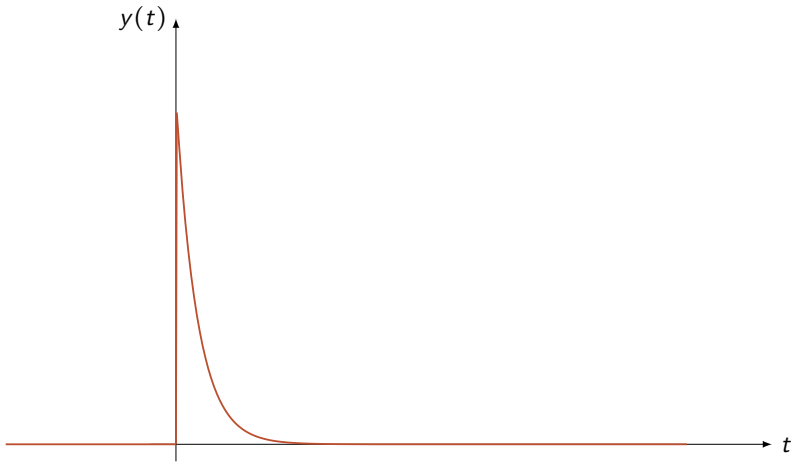
## 5 Newtons for 1/5 of a second?



notes

- note that here the position is $x$, but also $y = x$

## 100 Newtons for 1/100 of a second?

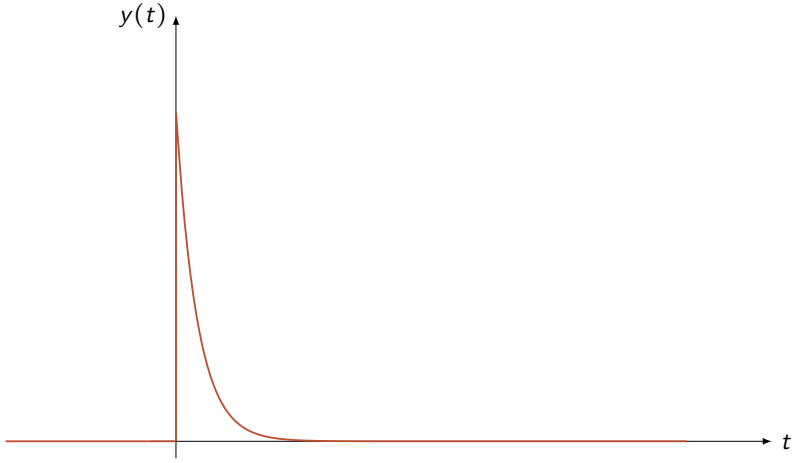





notes

- here we do not plot $u$, since it is out of scale

## 1000 Newtons for 1/1000 of a second?

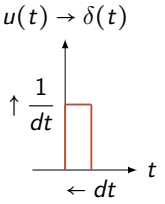


notes

- here we do not plot $u$, since it is out of scale

## Where are we going with the input signal?

↦ Dirac's delta, i.e., pushing an unitary mass within an infinitesimal space






notes

- let's introduce this signal here
- note how for $dt$ that goes to zero the signal becomes a sort of spike
- see also `https://en.wikipedia.org/wiki/Dirac_delta_function`

## ESSENTIAL POINT

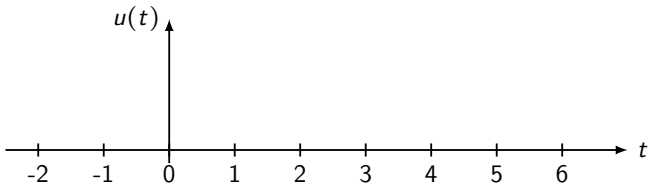
this game has sense only because the system is assumed to be LTI

notes

- otherwise we would not be guaranteed that we were having this sort of well defined limit behavior that we may explicitly compute by opportunely modifying the step response. If the system is not LTI, the step response and the impulse response are not connected, and this (as will be clear later on) implies we cannot use the response to the impulse as a template for computing all the potential responses that the system may give for any generic input

## Discussion

Where would you draw $\delta(t-4)$?



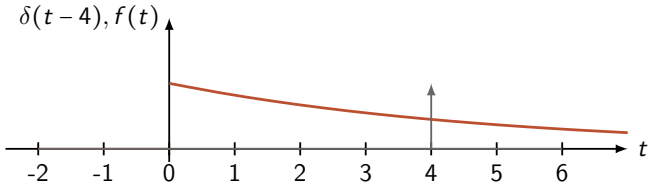



notes

- for this the spike is in $t = 10$ (because the spike is where the argument is zero, and in this case the argument is $t - 10$)

## Discussion

$$\int_{-\infty}^{+\infty} f(\tau)\delta(\tau-4)d\tau = ?$$



notes

- it will be $f(10)$ because the $\delta$ has infinite mass in 10 and thus the product of the two functions is everywhere 0 but in $f(10)$
- when within an integral, $\delta$ thus somehow reveals that function within the integral in a specific point

## Summarizing

**Describe** what the impulse response of an LTI system is in practice

- an opportune limit behavior of a transformation of the step response of a LTI system
- it is though a transformation that makes sense only if the system is LTI



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Important libraries / methods

- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.impulse.html`
- `https://python-control.readthedocs.io/en/latest/generated/control.impulse_response.html`



notes

- with these methods you can compute the impulse response of a LTI system

Self-assessment material

notes

## Question 50

What is the impulse response of an LTI system?

**Potential answers:**

I: **(wrong)** The output of the system when the input is a sinusoidal function.
II: **(wrong)** The output of the system when the input is a ramp function.
III: **(wrong)** The output of the system when the input is a step function.
IV: **(correct)** The output of the system when the input is a Dirac delta function.
V: **(wrong)** I do not know.

**Solution 1:**

The impulse response of an LTI system is the output of the system when the input is a Dirac delta function. This response characterizes the system's behavior and can be used to predict its output for any arbitrary input.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 51

Why is the impulse response meaningful only for LTI systems?

**Potential answers:**

I: **(correct)** Because the impulse response is a direct consequence of the superposition principle, which applies only to LTI systems.
II: **(wrong)** Because nonlinear systems do not respond to impulses.
III: **(wrong)** Because the impulse response is too complex to compute for nonlinear systems.
IV: **(wrong)** Because nonlinear systems have infinite impulse responses.
V: **(wrong)** I do not know.

**Solution 1:**

The impulse response is meaningful only for LTI systems because it relies on the superposition principle, which is a fundamental property of LTI systems. Nonlinear systems do not satisfy this principle, making the concept of an impulse response invalid for them.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 52

What happens to the mass-spring-damper system when the input force is a Dirac delta function?

**Potential answers:**

I: **(wrong)** The mass oscillates indefinitely without damping.
II: **(correct)** The mass exhibits a transient response that decays over time due to damping.
III: **(wrong)** The mass remains stationary because the impulse is too short to affect it.
IV: **(wrong)** The mass moves with constant velocity.
V: **(wrong)** I do not know.

**Solution 1:**



When the input force is a Dirac delta function, the mass-spring-damper system exhibits a transient response that decays over time due to the damping effect. This response is the impulse response of the system.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 53

What is the integral of $f(\tau)\delta(\tau - 4)$ from $-\infty$ to $+\infty$?

**Potential answers:**

I: **(wrong)** $\int_{-\infty}^{+\infty} f(\tau)d\tau$
II: **(wrong)** 0
III: **(correct)** $f(4)$
IV: **(wrong)** $\delta(4)$
V: **(wrong)** I do not know.

**Solution 1:**

The integral of $f(\tau)\delta(\tau - 4)$ from $-\infty$ to $+\infty$ is $f(4)$. This is because the Dirac delta function "samples" the function $f(\tau)$ at $\tau = 4$.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "what is an impulse response"

- impulse responses are directly connected to step responses
- actually this connection is valid only if the system is LTI



notes

- the most important remarks from this sub-module are these ones

# 1D convolution in continuous time

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| convolution | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| signal | u1, e1 |



notes

## Main ILO of sub-module "1D convolution in continuous time"

**Compute** the convolution between two single dimensional continuous time signals

notes

- by the end of this module you shall be able to do this

## Why convolution?

because for a LTI system with impulse response $h(t)$
it follows that $y_{\text{forced}}(t) = u * h(t)$



notes

- and indeed formally we get this result, that we do not prove for now

extremely important result for LTI systems:

$$y_{\text{forced}}(t) = h * u(t) = u * h(t) :=$$

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

. . . and this module = what that formula actually means from graphical perspectives

notes

## Additional material

Videos:

- https://www.youtube.com/watch?v=KuXjwB4LzSA
- https://www.youtube.com/watch?v=acAw5WGtzuk
- https://www.youtube.com/watch?v=IaSGqQa5O-M (for connections with probability)
- https://www.youtube.com/playlist?list=PL4iThgVpN7hmbIhHnCa7SDO0gLMoNwED_
- https://www.youtube.com/playlist?list=PL4mJLdGEHNvhCuPXsKFrnD7AaQBlMEB6a

Animations:

- https://lpsa.swarthmore.edu/Convolution/CI.html
- https://phiresky.github.io/convolution-demo/



notes

- the world is full of material on convolution - check also this stuff and not only what we do in class

## Towards decomposing this formula in pieces

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

better focus on

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

or on

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

*in automatic control typically better the second*

notes

- in the first option one has to "flip" $u$
- in the second option one has to "flip" $h$
- the second gives us more intuitions about the system, makes the impulse response directly interpretable
- they though give the same results

## Towards decomposing this formula in pieces, small change of notation

$$y_{\text{forced}}(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau \quad \mapsto \quad y_{\text{forced}}(\text{now}) = \int_{-\infty}^{+\infty} h(\tau)u(\text{now}-\tau)d\tau$$



notes

- better doing this change of notation, so to avoid some confusion

## Decomposing this formula in pieces

$$y_{\text{forced}}(\text{now}) = \int_{-\infty}^{+\infty} h(\tau)u(\text{now}-\tau)d\tau$$

$\Longrightarrow$ constituent pieces $=$

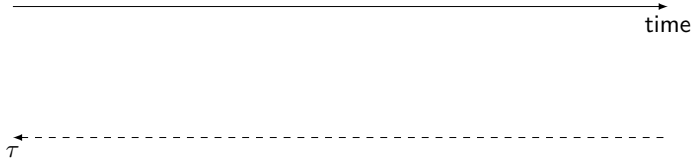
- $u(\text{now}-\tau)$
- $h(\tau)$
- $u(\text{now}-\tau)h(\tau)$
- $\int u(\text{now}-\tau)h(\tau)d\tau$

notes

- the main pieces / steps of computing a convolution are these ones

## Visualizing the various pieces

$u(\text{now} - \tau)$ $h(\tau)$ $u(\text{now} - \tau)h(\tau)$ $\int_{-\infty}^{+\infty} h(\tau)u(\text{now} - \tau)d\tau$



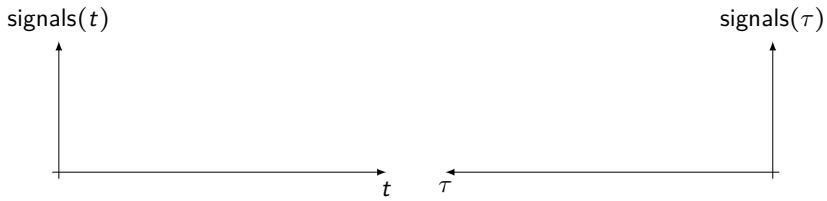



notes

- likely better watch the video in class for this animation, or try to refer to `https://lpsa.swarthmore.edu/Convolution/CI.html`

## Example

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



notes

- see the video of the lesson for the example
- note though that with the second option the meaning of $h$ is "how important the past is to create the current output"

## Another Example:

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



notes

- see the video of the lesson

## Paramount message

$h$ in $y_{\text{forced}}(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$ represents how much the past $u$'s contribute to the current $y_{\text{forced}}$:

notes

- this is a very important interpretation, thus better to fix it
- for example, $h(7)$ says how much $u(t-7)$ will enter into $y(t)$ for any $t$
- so somehow $h(\tau)$ has an important graphical meaning
- if $h(\tau)$ decays fast, then somehow the system "forgets" the past inputs fast, and viceversa

## Refreshing what we are doing and why

Dynamics of a cart: $\dot{v}(t) = -\frac{k}{m}v(t) + \frac{k}{m}F(t)$ with:

- **control input:** $u(t)$ (actuation from the motor, in this case = $F(t)$)
- **system output:** $y(t)$ (cart velocity, in this case = $v(t)$)
- **impulse response:** $h(t)$ (output corresponding to the input $\delta(t)$ assuming $y(0) = 0$)
- **free evolution:** $y_{\text{free}}(t)$ (output in time corresponding to no input, i.e., $u(t) = 0$, and initial condition $y(0)$ whatever it is)
- **forced response:** $y_{\text{forced}}(t) = u * h(t)$ (output in time corresponding to null initial condition, i.e., $y(0) = 0$, and input $u(t)$ whatever it is)
- **total response:** $y(t) = y_{\text{free}}(t) + y_{\text{forced}}(t)$



notes

- just to be sure: what we are doing here is essential
- we want to understand how to map the input into the output
- if we want to do model predictive control indeed we need to know what a certain input is going to cause to the output

## Quiz time!

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

- is $h * u(t) = u * h(t)$?
- is $(\alpha h_1 + \beta h_2) * u(t) = \alpha\,(h_1 * u(t)) + \beta\,(h_2 * u(t))$?
- if both $h(\tau) = 0$ and $u(t) = 0$ if $t < 0$, how can we simplify $y(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$?

notes

- 
- obviously yes, since we can do the change of variables
- 
- yes because the integrals are linear. This implies that the convolution operator is linear
- 
- yes, it becomes $y(t) = \int_0^t h(\tau)u(t-\tau)d\tau$

## Summarizing

**Compute** the convolution between two single dimensional continuous time signals

- take one of the two signals
- translate it to the "current $t$"
- flip it
- multiply the two signals in a pointwise fashion
- compute the integral of the result



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Methods implementing (discrete) convolutions

- `https://numpy.org/doc/2.1/reference/generated/numpy.convolve.html`
- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve.html`



notes

- note that these methods take discrete arrays, also multidimensional

Self-assessment material

notes

## Question 54

What does the convolution integral $y_{\text{forced}}(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$ represent in the context of LTI systems?

**Potential answers:**

I: **(wrong)** The free evolution of the system output.
II: **(correct)** The forced response of the system output due to the input $u(t)$.
III: **(wrong)** The total response of the system, including initial conditions.
IV: **(wrong)** The impulse response of the system.
V: **(wrong)** I do not know.

**Solution 1:**

The convolution integral $y_{\text{forced}}(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$ represents the forced response of the system output due to the input $u(t)$. It describes how the system responds to the input when initial conditions are zero.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 55

Which of the following is true about the convolution operation $h * u(t)$?

**Potential answers:**

I: **(wrong)** It is only defined for periodic signals.
II: **(wrong)** It is only applicable to discrete-time systems.
III: **(correct)** It is commutative, i.e., $h * u(t) = u * h(t)$.
IV: **(wrong)** It requires both signals to be symmetric.
V: **(wrong)** I do not know.

**Solution 1:**

The convolution operation is commutative, meaning $h * u(t) = u * h(t)$. This property holds for continuous-time signals in LTI systems.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 56

What does the impulse response $h(t)$ of an LTI system represent?

**Potential answers:**

I: **(wrong)** The input signal $u(t)$ applied to the system.
II: **(wrong)** The free evolution of the system output.
III: **(wrong)** The total response of the system, including initial conditions.
IV: **(correct)** The output of the system when the input is a Dirac delta function $\delta(t)$.
V: **(wrong)** I do not know.

**Solution 1:**

The impulse response $h(t)$ represents the output of the system when the input is a Dirac delta function $\delta(t)$. It characterizes the system's behavior.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 57

If $h(\tau) = 0$ for $\tau < 0$ and $u(t) = 0$ for $t < 0$, how can the convolution integral $y(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$ be simplified?

**Potential answers:**

I: **(correct)** $y(t) = \int_{0}^{t} h(\tau)u(t-\tau)d\tau$
II: **(wrong)** $y(t) = \int_{0}^{+\infty} h(\tau)u(t-\tau)d\tau$
III: **(wrong)** $y(t) = \int_{-\infty}^{+\infty} h(\tau)u(\tau)d\tau$
IV: **(wrong)** $y(t) = \int_{-\infty}^{0} h(\tau)u(t-\tau)d\tau$
V: **(wrong)** I do not know.

**Solution 1:**



If $h(\tau) = 0$ for $\tau < 0$ and $u(t) = 0$ for $t < 0$, the convolution integral simplifies to $y(t) = \int_{0}^{t} h(\tau)u(t-\tau)d\tau$, as the integrand is zero outside this interval.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 58

What is the graphical interpretation of $h(\tau)$ in the convolution integral $y_{\text{forced}}(t) = \int_{-\infty}^{+\infty} h(\tau)u(t-\tau)d\tau$?

**Potential answers:**

I: **(wrong)** It represents the future inputs of the system.
II: **(correct)** It represents how much past inputs contribute to the current output.
III: **(wrong)** It represents the free evolution of the system.
IV: **(wrong)** It represents the total energy of the system.
V: **(wrong)** I do not know.

**Solution 1:**

The term $h(\tau)$ in the convolution integral represents how much past inputs $u(t-\tau)$ contribute to the current output $y(t)$. This is a key interpretation in LTI systems.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "1D convolution in continuous time"

- convolution is an essential operator, since it can be used for LTI systems to compute forced responses
- its graphical interpretation aids interpreting impulse responses as how the past inputs contribute to current outputs

notes

- the most important remarks from this sub-module are these ones

# computing free evolutions and forced responses of LTI systems



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| free evolution | u1, e1 |
| forced response | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| LTI ODE | u1, e1 |
| convolution | u1, e1 |
| partial fraction decomposition | u1, e1 |

notes

## Main ILO of sub-module
## "computing free evolutions and forced responses of LTI systems"

**Compute** free evolutions and forced responses of LTI systems
using Laplace-based formulas (but only as procedural tools)



notes

- by the end of this module you shall be able to do this

## Disclaimer

the formulas introduced in this module shall be taken as "ex machina"

notes

- in other words, they are given and to be assumed as true
- in other courses or modules they will be derived from other principles

## Focus in this module = on ARMA models

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

with $^{(i)}$ meaning the $i$-th time derivative. *Discussion:* why is the LHS $y^{(n)}$ and not $a_n y^{(n)}$? *Discussion:* and which initial conditions shall we consider?



notes

- generalizing the LTIs we saw until now, we can arrive at these models, and in this module we will treat only these models (there may be other generalizations but you will see them in other modules)
- the $a_{n-1}y^{(n-1)} + \ldots + a_0 y$ part is called Auto-Regressive
- the $b_m u^{(m)} + \ldots + b_0 u$ part is called Moving-Average
- these names make more sense in discrete time systems of the type $y(k+n) = a_{n-1}y(k+n-1) + \ldots + a_0 y(k) + b_m u^(k+m) + \ldots + b_0 u(k)$ and $k$ a discrete time index. Here we see that the $a$'s correspond to an autoregression, and the $b$'s to the coefficients of a moving average. In any case we use ARMA for both continuous and discrete dynamics of these types
- note that in mechanical systems like motors, the derivatives of $u$ are meaningful because they capture the system's response to changes in the input signal, accounting for physical constraints like inertia
- this is because if we were having $a_n y^{(n)}$ on the left hand side then we could divide all the $a$'s and $b$'s on the right hand side and get the same dynamics
- so we prefer to work with monic polynomials (i.e., in which the leading coefficient, that is the nonzero coefficient of highest degree, is equal to 1) because we have less numbers to carry around (plus it will be convenient for other purposes that we will see later on in the course)
- as for the initial conditions that one shall consider, we typically assume all the conditions on the $u$ equal to zero, while on the $y$ they may be different from zero

## Laplace transforms - links for who would like to get more info about them

Laplace transforms = extension of Fourier transforms; interesting material:

- `https://www.youtube.com/watch?v=r6sGWTCMz2k` (Fourier series)
- `https://www.youtube.com/watch?v=spUNpyF58BY` (Fourier transforms)
- `https://www.youtube.com/watch?v=nmgFG7PUHfo` (on the historical importance of Fast Fourier Transforms)
- `https://www.youtube.com/watch?v=7UvtU75NXTg` (Laplace Transforms, in math)
- `https://www.youtube.com/watch?v=n2y7n6jw5d0` (Laplace Transforms, graphically)

notes

- note that this module treats the formulas as "given", so who wants to look at these links shall do only for self-interest, not for preparing oneself for exercises at the exam related to this module

## Main usefulness: convolution in time transforms into multiplication in Laplace-domain, and viceversa

$$\begin{cases} H(s) = \mathcal{L}\{h(t)\} \\ U(s) = \mathcal{L}\{u(t)\} \end{cases} \implies \mathcal{L}\{h * u(t)\} = H(s)U(s)$$

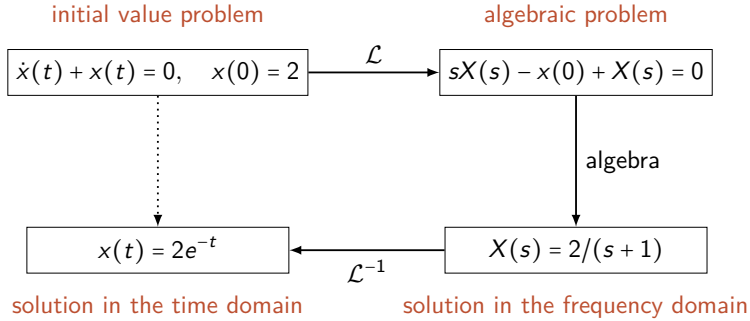
Noticeable name: *transfer function* ($= H(s) = \mathcal{L}\{\text{impulse response}\}$)



notes

- this is by far one of the most important properties of Laplace transforms for our purposes: convolution in one of the domains will be multiplication in the other
- this implies that instead of computing $u * y$, if computing $H$ and $U$ is fast, and if inverting $HU$ is fast, that way is preferrable
- the name "transfer function" is an important one and you will hear about it quite often

## An intuitive explanation of the usefulness of the Laplace transform in automatic control



notes

- this means that we can follow this scheme
- in other words, for complicated differential equations Laplace transform allow us to solve the problem algebraically. This is often much easier than solving the ODE directly

## First set of formulas to memorize: Laplace-transforming derivatives

(these will be motivated in other courses)

$$\mathcal{L}\{\dot{x}\} = sX(s) - x(0)$$

$$\mathcal{L}\{\ddot{x}\} = s^2X(s) - sx(0) - \dot{x}(0)$$

$$\mathcal{L}\{\dddot{x}\} = s^3X(s) - s^2x(0) - s\dot{x}(0) - \ddot{x}(0)$$

$$\mathcal{L}\{x^m\} = \dots$$



notes

- these formulas shall be remembered by heart

## Example: spring mass system

$$\ddot{y} = -\frac{f}{m}\dot{y} - \frac{k}{m}y + u$$

$$\Downarrow$$

$$s^2Y(s) - sy_0 - \dot{y}_0 = -\frac{f}{m}\Big(sY(s) - y_0\Big) - \frac{k}{m}Y(s) + U(s)$$

$$\Downarrow$$

$$s^2Y(s) + \frac{f}{m}sY(s) + \frac{k}{m}Y(s) = +sy_0 + \dot{y}_0 + \frac{f}{m}y_0 + U(s)$$

$$\Downarrow$$

$$Y(s) = \frac{y_0\left(\frac{f}{m} + s\right) + \dot{y}_0}{s^2 + \frac{f}{m}s + \frac{k}{m}} + \frac{1}{s^2 + \frac{f}{m}s + \frac{k}{m}}U(s)$$

notes

- let's then start this path building on top of previous results
- more precisely, from the fact that using Laplace transforms we were able to characterize the free evolution of second order LTI systems
- and $Y(s) \neq 0$ happens when the initial conditions of the system are not null

## And what shall we do once we get this?

generalizing the previous slide: $Y(s) = \frac{M(s)}{A(s)} + \frac{B(s)}{A(s)}U(s)$

with

- $\frac{M(s)}{A(s)}$ = Laplace transform of the free evolution
- $\frac{B(s)}{A(s)}U(s)$ = Laplace transform of the forced response

$\Longrightarrow$ we shall anti-transform; how? Main 2 cases:

- either $U(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s}$
- or $U(s)$ = something else



notes

- now we have this first result, where we note that the total signal is the sum of the two individual signals "free evolution" plus "forced response", but in the Laplace domain
- for the sake of this module we consider that $U(s)$ may be rational or not

## Question 59

Is the Laplace transform of the signal

$$h(t) = \begin{cases} \frac{1}{t+1} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

a rational Laplace transform?

**Potential answers:**

I: **(wrong)** yes
II: **(correct)** no
III: **(wrong)** it depends
IV: **(wrong)** I don't know

**Solution 1:**



A Laplace transform is rational if and only if it can be expressed as

$$H(s) = \frac{N(s)}{D(s)}$$

notes

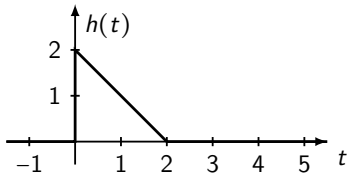
- see the associated solution(s), if compiled with that ones :)

notes

- see the associated solution(s), if compiled with that ones :)

## Question 60

Is the Laplace transform of the signal $h(t)$ below a rational Laplace transform?




**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | yes |
| II: | **(correct)** | no |
| III: | **(wrong)** | it depends |
| IV: | **(wrong)** | I don't know |



**Solution 1:**

A Laplace transform is rational if and only if it can be expressed as

$$H(s) = \frac{N(s)}{D(s)}$$

with both the numerator and the denominator finite order polynomials in $s$. If it is rational, then doing a partial fraction decomposition in the Laplace's domain will translate into a finite number of terms like $\frac{c}{(s-\lambda_i)^{\mu_i}}$ for opportune values of $\lambda$ and $\mu$. Taking the inverse-Laplace transform of these terms means eventually that the associated signal, in the time domain, is a finite sum of terms like $\alpha t^{\mu} \exp^{\gamma t} \cos(\beta t)$. However, the considered $h(t) = \frac{1}{t+1}$ can be represented only with an infinite number of terms $\alpha t^{\mu} \exp^{\gamma t} \cos(\beta t)$. In other words, to obtain this signal there is the need for an infinite number of elementary modes, and this means an infinitely long partial fraction decomposition in Laplace. So the

first case: rational $U(s)$

associated transform is not rational, since its denominator will be a polynomial with infinite order.

notes

## How to do if $U(s) = \dfrac{\text{polynomial in } s}{\text{polynomial in } s}$

$$Y(s) = \frac{M(s)}{A(s)} + \frac{B(s)}{A(s)}U(s) \quad \mapsto \quad Y(s) = \frac{M(s)}{A(s)} + \frac{C(s)}{D(s)}$$

write each of the two parts of the signal as

$$\frac{N(s)}{(s-\lambda_1)(s-\lambda_2)(s-\lambda_3)\cdots}$$



notes

- in this case we have a situation for which we can write both elements as polynomial over polynomial

## Next step: partial fraction decomposition

- **case single poles:** if $\dfrac{N(s)}{(s-\lambda_1)(s-\lambda_2)(s-\lambda_3)\cdots}$ is s.t. $\lambda_1 \neq \lambda_2 \neq \lambda_3 \neq \cdots$ then there exist $\alpha_1$, $\alpha_2$, $\alpha_3$, ... s.t.

$$\frac{N(s)}{(s-\lambda_1)(s-\lambda_2)(s-\lambda_3)\cdots} = \frac{\alpha_1}{s-\lambda_1} + \frac{\alpha_2}{s-\lambda_2} + \frac{\alpha_3}{s-\lambda_3} + \cdots \tag{7}$$

- **case repeated poles:** if some poles are repeated, then there exist $\alpha_{1,1}$, ..., $\alpha_{1,n1}$, $\alpha_{2,1}$, ..., $\alpha_{2,n2}$, ..., s.t.

$$\frac{N(s)}{(s-\lambda_1)^{n1}(s-\lambda_2)^{n2}\cdots} = \frac{\alpha_{1,1}}{s-\lambda_1} + \ldots + \frac{\alpha_{1,n1}}{(s-\lambda_1)^{n1}} + \frac{\alpha_{2,1}}{s-\lambda_2} + \ldots + \frac{\alpha_{2,n2}}{(s-\lambda_2)^{n2}} + \ldots \tag{8}$$

"But how do I compute $\alpha_1$, $\alpha_2$, etc.?" $\mapsto$
`en.wikipedia.org/wiki/Partial_fraction_decomposition`
(tip: start from `en.wikipedia.org/wiki/Heaviside_cover-up_method`)

notes

- let's remember that the partial fraction decomposition concept helps us factorizing ratios of polynomials in a sum of simpler ratios
- and let's also remember that there is the possibility of having multiple poles (something that, as we will see very soon, connects with the concept of non-trivial Jordan structure of the $A$ expressing this LTI system)
- in case somebody does not remember how to do it, there is a couple of resources that may help re-gaining knowledge on this tool

## Anti-transforming in the rational $U(s)$ case

if $Y(s) = \frac{\alpha_{1,1}}{s-\lambda_1} + \ldots + \frac{\alpha_{1,n1}}{(s-\lambda_1)^{n1}} + \frac{\alpha_{2,1}}{s-\lambda_2} + \ldots + \frac{\alpha_{2,n2}}{(s-\lambda_2)^{n2}} + \ldots$ then use

$$\mathcal{L}\left\{t^n e^{\lambda t}\right\} = \frac{n!}{(s-\lambda)^{n+1}} \quad \leftrightarrow \quad \mathcal{L}^{-1}\left\{\frac{n!}{(s-\lambda)^{n+1}}\right\} = t^n e^{\lambda t}$$



notes

- given this transform, $y(t)$ is then immediately a sum of terms of the type $t^n e^{\lambda t}$ for opportune $n$'s that depend on the specific $\lambda$
- we see that this must connect with the structure of the Jordan form of the $A$ expressing this LTI
- we will reinforce this connection later on – the important for now is to realize that it exists
- now either all the terms are simple, or there are some repeated lambda's

## Numerical Example: Inverse Laplace Transform of a Rational Function

$$Y(s) = \frac{3}{s-2} + \frac{4}{(s-2)^2} + \frac{5}{s+1}$$

goal = compute the inverse Laplace transform $y(t) = \mathcal{L}^{-1}\{Y(s)\}$

notes

- now let's do this exercise. Let's assume we started from an opportune $u(t)$ and ARMA model and initial conditions such that the general formula

$$Y(s) = \frac{M(s)}{A(s)} + \frac{B(s)}{A(s)}U(s)$$

has brought us to this specific $Y(s)$

## Step 1: Identify the terms

$$Y(s) = \frac{3}{s-2} + \frac{4}{(s-2)^2} + \frac{5}{s+1}$$

Here:

- $\lambda_1 = 2$, with coefficients $\alpha_{1,1} = 3$ and $\alpha_{1,2} = 4$
- $\lambda_2 = -1$, with coefficient $\alpha_{2,1} = 5$



notes

- we get immediately this, by applying what we saw in the previous slides

## Step 2: Apply the inverse Laplace transform formula

by means of

$$\mathcal{L}^{-1}\left\{\frac{n!}{(s-\lambda)^{n+1}}\right\} = t^n e^{\lambda t}$$

we compute the inverse Laplace transform of each term:

- $\mathcal{L}^{-1}\left\{\frac{3}{s-2}\right\} = 3e^{2t}$
- $\mathcal{L}^{-1}\left\{\frac{4}{(s-2)^2}\right\} = 4te^{2t}$
- $\mathcal{L}^{-1}\left\{\frac{5}{s+1}\right\} = 5e^{-t}$

notes

- then we get this

## Step 3: Combine the results

then we have that the inverse Laplace transform $y(t)$ is the sum of the individual transforms, i.e.,

$$y(t) = 3e^{2t} + 4te^{2t} + 5e^{-t}$$



notes

- then we get this

## Another Example: Inverse Laplace Transform with Complex Conjugate Terms

let

$$Y(s) = \frac{2s+3}{s^2+2s+5}$$

and the goal to be to compute the inverse Laplace transform $y(t) = \mathcal{L}^{-1}\{Y(s)\}$

notes

- assume this situation

## Step 1: Factor the denominator

note: $s^2 + 2s + 5$ has complex conjugate roots, indeed

$$s^2 + 2s + 5 = (s+1)^2 + 4$$

and thus

$$Y(s) = \frac{2s+3}{(s+1)^2+4}$$



notes

- note that we have complex conjugate pairs

## Step 2: Express in terms of standard forms

rewrite $Y(s)$ to match the standard forms for inverse Laplace transforms involving complex conjugates, i.e.,

$$Y(s) = \frac{2(s+1)+1}{(s+1)^2+4} = 2 \cdot \frac{s+1}{(s+1)^2+4} + \frac{1}{(s+1)^2+4}.$$

notes

- note that we have complex conjugate pairs

## Step 3: Apply the inverse Laplace transform formula

since

$$\mathcal{L}^{-1}\left\{\frac{s+a}{(s+a)^2+b^2}\right\} = e^{-at}\cos(bt),$$

$$\mathcal{L}^{-1}\left\{\frac{b}{(s+a)^2+b^2}\right\} = e^{-at}\sin(bt),$$

we have, for the various terms:

- $\mathcal{L}^{-1}\left\{2\cdot\frac{s+1}{(s+1)^2+4}\right\} = 2e^{-t}\cos(2t)$
- $\mathcal{L}^{-1}\left\{\frac{1}{(s+1)^2+4}\right\} = \frac{1}{2}e^{-t}\sin(2t)$



notes

- and here we are just using formulas

## Step 4: Combine the results

$$y(t) = 2e^{-t}\cos(2t) + \frac{1}{2}e^{-t}\sin(2t)$$

notes

- and here we are just using formulas

## Extremely important result

a LTI in free evolution behaves as a combination
of terms $e^{\lambda t}$, $te^{\lambda t}$, $t^2e^{\lambda t}$, etc. for a set of different
$\lambda$'s and powers of $t$, called the *modes* of the system

*Discussion:* assuming that we have two modes, $e^{-0.3t}$ and $e^{-1.6t}$, so that

$$y(t) = \alpha_1 e^{-0.3t} + \alpha_2 e^{-1.6t}.$$

What determines $\alpha_1$ and $\alpha_2$?



notes

- these signals are thus somehow describing the natural way a free evolution evolves
- we already saw them with Jordan forms, and we did not give them a name then
- but they have a specific name: they are the modes of a LTI
- these numbers are given by the initial conditions of the system

second case: irrational $U(s)$

notes

## In this case we cannot use partial fractions decompositions as before

from $Y(s) = \frac{M(s)}{A(s)} + \frac{B(s)}{A(s)} U(s)$ we follow the algorithm

- find $y_{\text{free}}(t)$ from PFDs of $\frac{M(s)}{A(s)}$ as before
- find the impulse response $h(t)$ from PFDs of $\frac{B(s)}{A(s)}$ as before
- find $y_{\text{forced}}(t)$ as $h * u(t)$



notes

- in this case we need to find the various terms independently

## Summarizing

**Compute** free evolutions and forced responses of LTI systems using Laplace-based formulas (but only as procedural tools)

- Laplace the ARMA
- if $u(t)$ admits a rational $U(s)$ then write $Y(s) = \frac{\text{polynomial}}{\text{polynomial}}$, do PFD, and do inverse-Laplaces
- if $u(t)$ does not admit a rational $U(s)$, do similarly as before but do PFD only for the free evolution and impulse response, and find the forced response by means of convolution

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module



notes

## Two essential libraries

- `https://python-control.readthedocs.io/en/0.10.1/generated/control.modal_form.html`
- `https://docs.sympy.org/latest/modules/physics/control/lti.html`

notes

- these libraries provide you the necessary tools to perform modal analysis as here

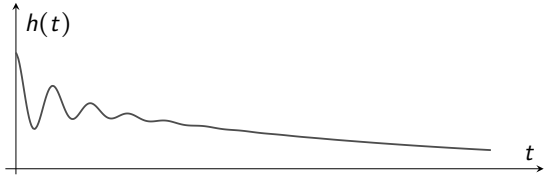
Self-assessment material



notes

## Question 61

Which type of LTI system may produce the impulse response $h(t)$ represented in the picture?




**Potential answers:**

I: **(wrong)** first order
II: **(wrong)** second order
III: **(correct)** at least third order
IV: **(wrong)** I do not know



**Solution 1:**

notes

- see the associated solution(s), if compiled with that ones :)

notes

- see the associated solution(s), if compiled with that ones :)

## Question 62

Which type of LTI system may produce the impulse response $h(t)$ represented in the picture?


**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | first order |
| II: | **(wrong)** | second order |
| III: | **(wrong)** | third order |
| IV: | **(correct)** | at least fourth order |
| V: | **(wrong)** | I do not know |

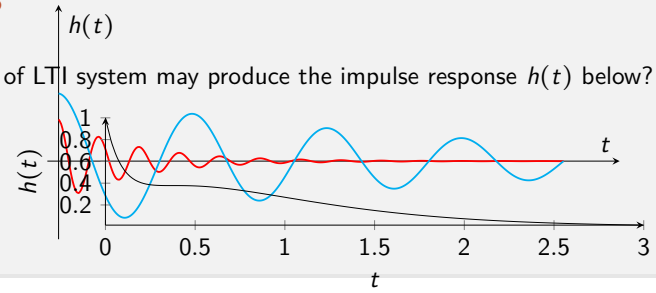
**Solution 1:**



The impulse response may be decomposed as a sum of two decaying oscillatory behaviors, i.e., as $h(t) = e^{\alpha t}\cos(\omega_1 t) + e^{\beta t}\cos(\omega_2 t)$, as in the figure below. The first part $e^{\alpha t}\cos(\omega_1 t)$ decays faster than the second part $e^{\beta t}\cos(\omega_2 t)$, and is also associated to a cosine oscillating at a higher frequency than the second. Hence this impulse response associates with two modes, both relating to a second order subsystem. Thus the correct answer is a system whose order is at least four.

## Question 63

Which type of LTI system may produce the impulse response $h(t)$ below?




**Solution 2:**
**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | first order |
| II: | **(wrong)** | second order |
| III: | **(correct)** | at least third order |
| IV: | **(wrong)** | I do not know |



indicate a transfer function with two complex conjugates stable poles on the left

**Solution 1:**

Looking at the graph of $h(t)$, we decompose it in the sum of two different modes:

notes

- see the associated solution(s), if compiled with that ones :)

## Question 64

What is the primary purpose of using Laplace transforms in solving LTI systems?

**Potential answers:**

I: **(wrong)** To convert differential equations into algebraic equations for easier solving.
II: **(correct)** To transform convolution in the time domain into multiplication in the Laplace domain.
III: **(wrong)** To directly compute the eigenvalues of the system matrix.
IV: **(wrong)** To eliminate the need for initial conditions in solving differential equations.
V: **(wrong)** I do not know.

**Solution 1:**



The primary purpose of using Laplace transforms is to transform convolution in the time domain into multiplication in the Laplace domain, simplifying the solution of differential equations.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 65

What is the correct form of the inverse Laplace transform of $\frac{1}{(s-\lambda)^2}$?

**Potential answers:**

I: **(wrong)** $e^{\lambda t}$
II: **(wrong)** $te^{\lambda t}$
III: **(correct)** $te^{\lambda t}$
IV: **(wrong)** $\frac{1}{2}t^2e^{\lambda t}$
V: **(wrong)** I do not know.

**Solution 1:**

The inverse Laplace transform of $\frac{1}{(s-\lambda)^2}$ is $te^{\lambda t}$.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 66

What is the inverse Laplace transform of $\frac{s+1}{(s+1)^2+4}$?

**Potential answers:**

I: **(wrong)** $e^{-t}\sin(2t)$
II: **(correct)** $e^{-t}\cos(2t)$
III: **(wrong)** $e^{-t}\cos(t)$
IV: **(wrong)** $e^{-t}\sin(t)$
V: **(wrong)** I do not know.

**Solution 1:**

The inverse Laplace transform of $\frac{s+1}{(s+1)^2+4}$ is $e^{-t}\cos(2t)$.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 67

In the ARMA model $y^{(n)} = a_{n-1}y^{(n-1)} + \ldots + a_0 y + b_m u^{(m)} + \ldots + b_0 u$, why is the leading coefficient of $y^{(n)}$ typically set to 1?

**Potential answers:**

I: **(wrong)** To ensure the system is stable.
II: **(wrong)** To simplify the computation of eigenvalues.
III: **(correct)** To reduce the number of parameters and work with monic polynomials.
IV: **(wrong)** To make the system linear time-invariant.
V: **(wrong)** I do not know.

**Solution 1:**

The leading coefficient of $y^{(n)}$ is typically set to 1 to reduce the number of parameters and work with monic polynomials, which simplifies calculations.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 68

What determines the coefficients $\alpha_1$ and $\alpha_2$ in the free evolution response $y(t) = \alpha_1 e^{-0.3t} + \alpha_2 e^{-1.6t}$?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The eigenvalues of the system matrix. |
| II: | **(wrong)** | The input signal $u(t)$. |
| III: | **(correct)** | The initial conditions of the system. |
| IV: | **(wrong)** | The poles of the transfer function. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

The coefficients $\alpha_1$ and $\alpha_2$ are determined by the initial conditions of the system.



notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "computing free evolutions and forced responses of LTI systems"

- finding such signals require knowing a couple of formulas by heart
- partial fraction decomposition is king here, one needs to know how to do that

notes

- the most important remarks from this sub-module are these ones

# state space representations



notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| state of a system | u1, e1 |
| separation principle | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ODE | u1, e1 |

notes

## Main ILO of sub-module "state space representations"

**Define** the meaning of "state space representation" in the context of linear and non-linear dynamical systems



notes

- by the end of this module you shall be able to do this

## Discussion: which information do you need to forecast accurately how long you may use your cellphone before its battery hits 0%?

notes

- think at which factors are important for you

## Summarizing

these pieces of information contain all I need to forecast the future evolution of the battery level:

- current level of charge of the battery
- how much I will use the phone in the future
- how healthy the battery of my phone is
- which environmental factors may induce additional effects (too warm, too cold)



notes

- the state condenses somehow the past
- from a control point of view this is important, because the state somehow works as a "memory": to decide wich $u$ is best right now, I just need to check what is the current state – I do not care about what state the system was experiencing before
- so this is a concept that is very instrumental for control

## A simple model of the battery charge as a dynamical system

$$\text{Time Remaining} = \frac{\text{Remaining Capacity}}{\text{Discharge Rate}} \qquad \text{example: } \frac{2000\text{mAh}}{500\text{mA}} = 4\text{hours}$$

rewriting as an ODE:

- $y(t) = Q(t) =$ remaining battery capacity at time $t$ (mAh)
- $u(t) = I(t) =$ current discharge rate at time $t$ (mA)

$$\Longrightarrow \qquad \dot{y} = -u$$

notes

- let's make a physical model of this
- and if we make it as a LTI ODE, given that the set of phenomena that we are assuming as constituting the models do not comprise self-discharge, we shall have something that in free evolution looks like $\dot{y} = 0 y = 0$
- moreover if I draw positive current I decrease the charge, thus we get a 'minus' sign associated to the $u$

## What is a state?

$$\begin{cases} \dot{x} &= -u \\ y &= x \end{cases}$$

"the current value of the state $x(t)$ contains all the information necessary to forecast the future evolution of the output $y(t)$ and of the state $x(t)$, assuming to know the future $u(t)$. I.e., to compute the future values $y(t+\tau)$ and $x(t+\tau)$ it is enough to know the current $x(t)$ and the current and future inputs $u(t:t+\tau)$"



notes

- the state condenses somehow the past
- from a control point of view this is important, because the state somehow works as a "memory": to decide wich $u$ is best right now, I just need to check what is the current state – I do not care about what state the system was experiencing before
- so this is a concept that is very instrumental for control

## Question 69

In a spring-mass system, which of the following is a valid state variable?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The temperature of the spring. |
| II: | **(correct)** | The displacement of the mass from its equilibrium position. |
| III: | **(wrong)** | The color of the mass. |
| IV: | **(wrong)** | The external force applied to the system. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

The displacement of the mass from its equilibrium position is a valid state variable because it describes the system's configuration and is essential for predicting its future behavior. Temperature and color are irrelevant, and the external force is an input, not a state.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 70

Which of the following pairs of variables can fully describe the state of a spring-mass system?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The mass of the spring and the stiffness of the mass. |
| II: | **(wrong)** | The external force and the displacement of the mass. |
| III: | **(correct)** | The displacement of the mass and the velocity of the mass. |
| IV: | **(wrong)** | The acceleration of the mass and the color of the spring. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

The displacement of the mass and the velocity of the mass fully describe the state of a spring-mass system because they capture the system's current configuration (displacement) and its rate of change (velocity). Mass, stiffness, external force, and color are not state variables.



notes

- see the associated solution(s), if compiled with that ones :)

## What do we mean with "modelling a state-space dynamical system"?

Defining

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

and

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

notes

- these are called state space representations
- take home message: the input-output maps saw in other modules are not the unique ways of representing systems

## State space model - definition

Ingredients:

- the number of inputs, outputs and state variables must be finite
- the differential equations must be first order
- the separation principle *(the current value of the state contains all the information necessary to forecast the future evolution of the outputs and of the state)* shall be satisfied

state space model = finite set of first-order differential equations that connect a finite set of inputs, outputs and state variables so that they satisfy the separation principle



notes

- so, if we recall what we did in some modules ago, this was the formal definition of a state space system
- remember that, first of all, it is a finite representation: for example a metal bar that is heating up, we may describe it with partial differential equations. But this would mean considering the temperature in every point, and this means an infinite number of points - no good
- we work with computers, and somehow we need always to consider a discrete and finite number of objects. Thus we consider finite number of states

## State space representations - Notation

- $u_1, \ldots, u_m$ = inputs
- $x_1, \ldots, x_n$ = states
- $y_1, \ldots, y_p$ = outputs
- $d_1, \ldots, d_q$ = disturbances

notes

- remember also that this is the standard notation (but the $q$ in $d_q$, for which there is no standard notation)
- if you will use something different in your job you will look like a fool

## State space representations - Notation

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

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



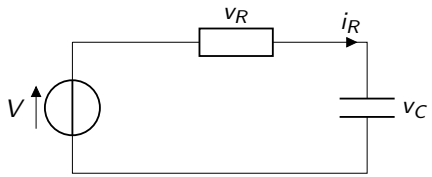
notes

- finally, we use this notation

examples

notes

## RC-circuit




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

or, using control-oriented names,

$$\dot{x} = -\frac{1}{RC} x + \frac{1}{RC} u \qquad y = x \tag{10}$$



notes

- we start with the simplest dynamic system possible, that is a scalar first order system where the dynamics are implied by the presence of the capacitor
- note that if we were not having the capacitor the system would have been a static one
- here we can change the names so to highlight what is the output and the input, i.e., what we can steer
- here note that how much $y$ grows depends on both $y$ and $u$, and this dependence is "fixed" by $R$ and $C$
- qualitatively what happens if we put $u = 0$ but we have some initial tension?

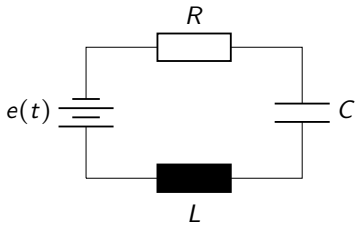
## Generalization: exponential growth, scalar version

$$\dot{x} = \alpha x + \beta u \qquad y = x \tag{11}$$

notes

- the previous example can be generalized in this way
- we will see better later on that "exponentials" play a big role here, since if we neglect $u$ you see that we have that we must have that the derivative of $y$ must be proportional to $y$ itself. Exponentials have this property (also sinusoids, but we know that sinusoids are complex exponentials, because of Euler's identities)
- we will see this better later on though
- here note that how much $y$ grows depends on both $y$ and $u$, and this dependence is "fixed" by $\alpha$ and $\beta$

## Generalizing in an other way: RCL-circuits




EOM: Kirchhoff laws $\implies$ $v_L(t) = L\frac{di(t)}{dt}$ $v_i(t) = Ri(t)$ $v_C(t) = \frac{1}{C}\int_0^t i(\tau)d\tau$

$$e(t) = v_L(t) + v_R(t) + v_C(t) \implies e(t) = L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau \quad (12)$$



notes

- what happens if we add an inductor?
- the equations of motion can be derived from Kirchhoff's laws, that can be summarized in this way
- and then we can state that the tension in the generator must equal to the sum of the tensions along the various components

## Generalizing in an other way: RCL-circuits part two

$$e(t) = L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int_0^t i(\tau)d\tau \quad (13)$$

can be rewritten as

$$\begin{cases} \left(\int_0^t \dot{i}(\tau)d\tau\right) &= i(t) \\ \dot{i}(t) &= \frac{1}{L}e(t) - \frac{R}{L}i(t) - \frac{1}{LC}\int_0^t i(\tau)d\tau \end{cases} \quad (14)$$

that can be rewritten as

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{L}u(t) - \frac{R}{L}x_2 - \frac{1}{LC}x_1 \end{cases} \qquad y = x_2 \quad (15)$$

notes

- for the purposes of the course it is convenient to do this rewriting
- and then this second rewriting, where we express the variables as states
- somehow it may have been more convenient to write $x$ instead of $y$, but this is a sort of nuisance that will not matter at all when you understood the messages from this course

## Exponential growth, matricial version

Generalization of all linear systems, thus also of "RCL circuits"

$$\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} \end{cases}$$

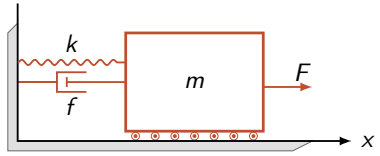


notes

- this example generalizes the one saw before. Better to recall the geometrical interpretation of a matrix times a vector, that highlights each column of $A$ to be a direction in the space where $\dot{\boldsymbol{x}}$ lives, and every component of $\boldsymbol{x}$ being thus how much that direction of that column should be followed
- the same interpretation of columns times scalars follows for the term $B\boldsymbol{u}$. Here each term of $\boldsymbol{u}$
- also for this type of ODE we will have that exponentials play a big role

## 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: $\sum F = m\ddot{x}(t)$

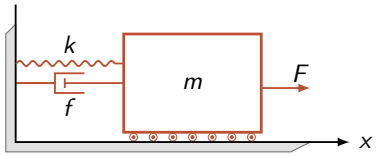
$$m\ddot{x}(t) = F_x(t) + F_f(t) + F(t) \quad \mapsto \quad m\ddot{x}(t) = -kx(t) - f\dot{x}(t) + F(t)$$

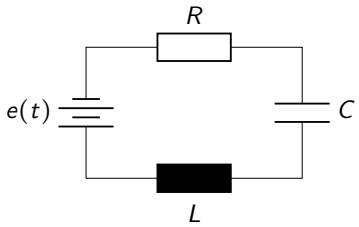
(rewritable again as $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$)

notes

- next example: a cart
- Newton's laws of motion tell us the following
- and we can rewrite things again in this way
- note that this is thus another example of the linear system that we used to represent RCL circuits

## Important message: these two systems are "the same"




thus studying $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ in means studying both systems simultaneously!



notes

- so we can conclude that essentially these two systems share the same structure, only with different parameters
- so this means that studying the equations in general makes us "save time", because we study all these systems (and many more) in one shot

## Lotka-Volterra

- $y_{\text{prey}}$ := prey
- $y_{\text{pred}}$ := predator

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

`./LotkaVolterraSimulator.ipynb`

notes

- of course not all the systems are of the type $\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u}$. This one is for example nonlinear, since there is a product among the $y$'s that cannot be captured by the linear relation above
- this example was seen before. More information and history behind it in `https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations`
- go through the python notebook for more information

## Van-der-Pol oscillator

$$\begin{cases} \dot{x}_1 &= \mu\left(x_1 - \frac{x_1^3}{3} - x_2\right) \\ \dot{x}_2 &= \frac{x_1}{\mu} \end{cases} \tag{16}$$

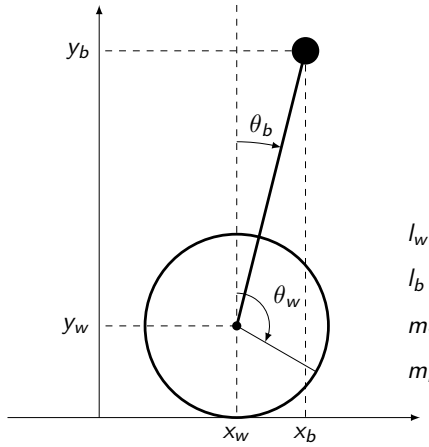
`./VanDerPolSimulator.ipynb`



notes

- another interesting example is the oscillator here, that we will see in more details later on
- the nice property of this system is that it has an orbit that attracts all the remaining ones
- we will discuss this example a few times. If you are already now interested in reading about it, check `https://en.wikipedia.org/wiki/Van_der_Pol_oscillator`
- go through the python notebook for more information

## 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

notes

- another example is a model of a segway
- here we will use this notation

## Balancing robot

$$\begin{aligned}\left(I_b + m_b l_b^2\right)\ddot{\theta}_b &= +m_b l_b g \sin\left(\theta_b\right) - m_b l_b \ddot{x}_w \cos\left(\theta_b\right) - \frac{K_t}{R_m} v_m + \left(\frac{K_e K_t}{R_m} + b_f\right)\left(\frac{\dot{x}_w}{l_w} - \dot{\theta}_b\right) \\ \left(\frac{I_w}{l_w} + l_w m_b + l_w m_w\right)\ddot{x}_w &= -m_b l_b l_w \ddot{\theta}_b \cos\left(\theta_b\right) + m_b l_b l_w \dot{\theta}_b^2 \sin\left(\theta_b\right) + \frac{K_t}{R_m} v_m - \left(\frac{K_e K_t}{R_m} + b_f\right)\left(\frac{\dot{x}_w}{l_w} - \dot{\theta}_b\right)\end{aligned} \tag{17}$$



notes

- the EOM can be found again with Newton's laws
- this is a nonlinear system, since it contains some trigonometric transformations of the variables

## Insulin concentration

- $x_1$ := sugar concentration
- $x_2$ := insulin concentration
- $u_1$ := food intake
- $u_2$ := insulin intake
- $c$ := sugar concentration in fasting *(person-specific)*

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

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

notes

- this is a switched system that represents in a very simplified way what happens to the body when eating or taking artificial insuline
- depending on whether there is more or less sugar in the blood than what is the person specific parameter $c$, then the body answers in different ways
- the main take home message for this model is that it tries to mimick biological phenomena that are quite understood
- the model can then be used to design when / how much to eat and to inject insuline
- for more information towards biology see for example `https://en.wikipedia.org/wiki/Insulin`, while for control-oriented explanations see for example "Model individualization for artificial pancreas", in Computer Methods and Programs in Biomedicine, Volume 171, April 2019, Pages 133-140, Messori et al.

## Summarizing

**Define** the meaning of "state space representation" in the context of linear and non-linear dynamical systems

- recall the definition of state space model
- be sure to have interiorized the separation principle with some practical examples



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Important library

`https://python-control.readthedocs.io/en/0.10.1/conventions.html#state-space-systems`



notes

- this is more or less a reference library

# Self-assessment material

notes

## Question 71

What is the primary purpose of the separation principle in state space representations?

**Potential answers:**

I: **(wrong)** To ensure that the system has an infinite number of states.
II: **(wrong)** To eliminate the need for inputs in the system model.
III: **(correct)** To ensure that the current state contains all information needed to predict future behavior.
IV: **(wrong)** To simplify the computation of system eigenvalues.
V: **(wrong)** I do not know.

**Solution 1:**

The separation principle ensures that the current state contains all the information necessary to predict the future evolution of the system, given the future inputs. This is a fundamental property of state space representations.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 72

Which of the following is a valid state variable in a state space representation of a dynamical system?

**Potential answers:**

I: **(wrong)** The external force applied to the system.
II: **(correct)** The displacement of a mass in a spring-mass system.
III: **(wrong)** The color of the system components.
IV: **(wrong)** The temperature of the environment.
V: **(wrong)** I do not know.

**Solution 1:**

The displacement of a mass in a spring-mass system is a valid state variable because it describes the system's configuration and is essential for predicting its future behavior. External forces, color, and environmental temperature are not state variables.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 73

What does the state transition map $\boldsymbol{f}$ in a state space representation describe?

**Potential answers:**

I: **(wrong)** The relationship between inputs and outputs.
II: **(correct)** The evolution of the state variables over time.
III: **(wrong)** The effect of disturbances on the system.
IV: **(wrong)** The stability of the system.
V: **(wrong)** I do not know.

**Solution 1:**

The state transition map $\boldsymbol{f}$ describes how the state variables evolve over time based on the current state and inputs. It is a key component of state space representations.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 74

What is the role of the output map $\boldsymbol{g}$ in a state space representation?

**Potential answers:**

I: **(wrong)** To define the system's stability.
II: **(wrong)** To describe the evolution of the state variables.
III: **(correct)** To relate the state variables and inputs to the measured outputs.
IV: **(wrong)** To eliminate the need for disturbances in the model.
V: **(wrong)** I do not know.

**Solution 1:**

The output map $\boldsymbol{g}$ relates the state variables and inputs to the measured outputs. It defines how the system's internal state is reflected in the observable outputs.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 75

Which of the following pairs of variables is sufficient to describe the state of a simple pendulum system?

**Potential answers:**

| | | |
|---|---|---|
| I: | **(wrong)** | The mass of the pendulum and the length of the string. |
| II: | **(wrong)** | The external torque and the angular displacement. |
| III: | **(correct)** | The angular displacement and the angular velocity. |
| IV: | **(wrong)** | The color of the pendulum and the gravitational constant. |
| V: | **(wrong)** | I do not know. |

**Solution 1:**

The angular displacement and the angular velocity are sufficient to describe the state of a simple pendulum system because they capture the system's current configuration (displacement) and its rate of change (velocity). Mass, length, external torque, and color are not state variables.



notes

- see the associated solution(s), if compiled with that ones :)

## Exercise: find which parts of these paragraphs are correct and which ones are wrong

The RCL circuit can be modeled by a second-order linear differential equation where the inductance, resistance, and capacitance determine the system's resonance frequency. Interestingly, in an underdamped RCL circuit, the system will always return to equilibrium without oscillating, which reflects the energy dissipation in the resistor.

notes

- the solution is:
- RCL Circuit Misconception: The statement "the system will always return to equilibrium without oscillating" is incorrect. An underdamped RCL circuit does oscillate before eventually returning to equilibrium due to the resistance.

## Exercise: find which parts of these paragraphs are correct and which ones are wrong

The Lotka-Volterra model is a non-linear system that describes interactions between two species: one as a predator and the other as prey. The model assumes that the growth rate of the prey population is proportional to the current population size, which would mean that the population would grow indefinitely in the absence of predators. Similarly, the predator population is dependent solely on the availability of prey, implying that predators could not survive without prey even if there were other food sources available.



notes

- the solution is:
- Lotka-Volterra Misconception: The claim that "predators could not survive without prey even if there were other food sources available" oversimplifies the model. The model assumes that the prey is the only food source, but in reality, predators might have alternative food sources.

## Exercise: find which parts of these paragraphs are correct and which ones are wrong

The Van der Pol oscillator is an example of a non-linear system that exhibits limit cycle behavior. This behavior is critical as it shows how the system can maintain a stable oscillation regardless of initial conditions, which is a feature not present in linear oscillators. It's important to note that the Van der Pol oscillator can only have a single limit cycle, and any perturbations will lead to a quick return to this cycle, indicating that the system is highly stable.

notes

- the solution is:
- Van der Pol Oscillator Misconception: The statement "the Van der Pol oscillator can only have a single limit cycle" is correct, but saying that "any perturbations will lead to a quick return to this cycle, indicating that the system is highly stable" is misleading. The Van der Pol oscillator returns to its limit cycle, but the speed and nature of this return depend on the systems parameters, and calling it "highly stable" is misleading and pushes persons into thinking it's more stable than it actually is.

## Recap of sub-module "state space representations"

- a set of variables is a state vector if it satisfies for that model the separation principle, i.e., the current state vector "decouples" the past with the future
- state space models are finite, and first order vectorial models



notes

- the most important remarks from this sub-module are these ones

# state space from ARMA (and viceversa)

notes

## Contents map

| developed content units | taxonomy levels |
|---|---|
| realization | u1, e1 |

| prerequisite content units | taxonomy levels |
|---|---|
| ARMA model | u1, e1 |
| state space model model | u1, e1 |
| matrix inversion | u1, e1 |
| Laplace transforms | u1, e1 |



notes

## Main ILO of sub-module "state space from ARMA (and viceversa)"

**Determine** the state space structure of an
LTI system starting from an ARMA ODE

notes

- by the end of this module you shall be able to do this

## ARMA models

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



notes

- the $a_{n-1}y^{(n-1)} + \ldots + a_0 y$ part is called Auto-Regressive
- the $b_m u^{(m)} + \ldots + b_0 u$ part is called Moving-Average
- these names make more sense in discrete time systems of the type $y(k+n) = a_{n-1}y(k+n-1) + \ldots + a_0 y(k) + b_m u^(k+m) + \ldots + b_0 u(k)$ and $k$ a discrete time index. Here we see that the $a$'s correspond to an autoregression, and the $b$'s to the coefficients of a moving average. In any case we use ARMA for both continuous and discrete dynamics of these types

## State space representations - Notation

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

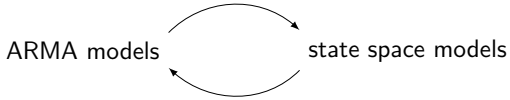
$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right) \\
\boldsymbol{y} &= \boldsymbol{g}\left(\boldsymbol{x}, \boldsymbol{u}\right)
\end{aligned}$$

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

notes

- notation wide, remember that state space means first order ODEs
- they will thus look like these ones, in general
- we can also compress the notation in this way
- remember that bold non-capital fonts mean vectors in this course
- and we give to the various things these names

## This module:




*But why do we study this?*

because from physical laws we get ARMA,
but with state space we get more explainable models



notes

- we will learn how to do two simple operations
- we will only scratch the surface though, there is a lot of material to cover here and you will do it much better in other modules / courses
- and often one does the "ARMA to SS" operation

# From state space to ARMA

notes

## SS to ARMA

Tacit assumption: $\boldsymbol{x}(0) = \mathbf{0}$

$$\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases} \quad \rightarrow \quad \mathcal{L}\left( \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases} \right)$$

$$\rightarrow \quad \begin{cases} sX &= AX + BU \\ Y &= CX + DU \end{cases}$$

$$\rightarrow \quad \begin{cases} (sI - A)X &= BU \\ Y &= CX + DU \end{cases}$$

$$\rightarrow \quad \begin{cases} X &= (sI - A)^{-1}BU \quad (*) \\ Y &= CX + DU \end{cases}$$

$$\Rightarrow \quad Y = \Big(C(sI - A)^{-1}B + D\Big)U$$

$$\Rightarrow \quad Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s}U(s)$$



notes

- This slide shows the step-by-step derivation of the transfer function from the state-space representation using Laplace transforms.
- the assumption $\boldsymbol{x}(0) = \mathbf{0}$ simplifies the Laplace transform of the derivative.
- the key step where $X = (sI - A)^{-1}BU$ is derived is the foundation for the transfer function.
- the final result, $Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s}U(s)$, is the ARMA representation of the system.
- For computations, I recommend using tools like `simpy` for symbolic algebra, but you should be able to handle 2x2 systems by hand.

## A note on the last formula

$$Y(s) = \frac{\text{polynomial in } s}{\text{polynomial in } s}U(s) \quad \mapsto \quad \text{ARMA:}$$

$$Y(s) = \frac{s+3}{2s^3 + 3s}U(s) \quad \mapsto \quad 2\dddot{y} + 3\dot{y} = \dot{u} + 3u$$

notes

- This slide connects the transfer function to the ARMA model in the time domain.
- the numerator and denominator polynomials in $s$ directly translate to differential equations in the time domain.
- the example shows how the transfer function $Y(s) = \frac{s+3}{2s^3+3s}U(s)$ corresponds to the differential equation $2\dddot{y} + 3\dot{y} = \dot{u} + 3u$.
- this is a key step in understanding the relationship between the Laplace domain and time domain.

## A note on the second to last formula

$$Y = \Big(C(sI - A)^{-1}B + D\Big)U$$

DISCLAIMER: in this course we consider SISO systems, thus $C$ and $B$ = vectors, and $D$ = scalar (if present)



notes

- that this course focuses on Single Input Single Output (SISO) systems, which simplifies the matrices $C$, $B$, and $D$.
- $C$ and $B$ are vectors, and $D$ is a scalar (often zero in many systems).
- this simplification is important for understanding the structure of the transfer function.
- MIMO (Multiple Input Multiple Output) systems in other courses!

## Numerical Example: $2 \times 2$ State-Space to ARMA

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

notes

- this numerical example is used to illustrate the conversion from state-space to ARMA.
- this is a 2x2 system, which is manageable by hand and helps students understand the process.
- the matrices $A$, $B$, $C$, and $D$ are chosen for simplicity, but the method is general.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 1: State-Space Equations

$$\begin{cases} \dot{x}_1 = x_1 + 2x_2 + u \\ \dot{x}_2 = 3x_1 + 4x_2 \\ y = x_1 \end{cases}$$



notes

- the state-space equations explicitly using the given matrices.
- $\dot{x}_1$ and $\dot{x}_2$ are linear combinations of the states and the input $u$.
- the output $y$ is simply the first state variable $x_1$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 2: Laplace Transform

$$\begin{cases} sX_1(s) = X_1(s) + 2X_2(s) + U(s) \\ sX_2(s) = 3X_1(s) + 4X_2(s) \\ Y(s) = X_1(s) \end{cases}$$

notes

- Apply the Laplace transform to the state-space equations, assuming zero initial conditions.
- the Laplace transform converts differential equations into algebraic equations in $s$.
- $Y(s) = X_1(s)$, which connects the output directly to the first state variable.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 3: Rearrange in Matrix Form

$$\begin{cases} (sI - A)X(s) = BU(s) \\ Y(s) = CX(s) + DU(s) \end{cases}$$

implies

$$\begin{cases} \begin{bmatrix} s-1 & -2 \\ -3 & s-4 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} U(s) \\ Y(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} \end{cases}$$



notes

- Rearrange the Laplace-transformed equations into matrix form.
- the output equation $Y(s) = CX(s)$ remains simple due to the choice of $C$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 4: Solve for $X(s)$

$$X(s) = (sI - A)^{-1}BU(s)$$

$$(sI - A) = \begin{bmatrix} s-1 & -2 \\ -3 & s-4 \end{bmatrix}$$

$$(sI - A)^{-1} = \frac{1}{(s-1)(s-4) - (-2)(-3)} \begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix}$$

$$\det(sI - A) = (s-1)(s-4) - 6 = s^2 - 5s - 2$$

$$(sI - A)^{-1} = \frac{1}{s^2 - 5s - 2} \begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix}$$

notes

- Solve for $X(s)$ by computing $(sI - A)^{-1}$.
- the determinant $\det(sI - A)$, which appears in the denominator of the transfer function, is key.
- the step-by-step computation of the inverse matrix is assumed as a given skill.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 5: Multiply by $B$

Now, multiply by $B$:

$$X(s) = \frac{1}{s^2 - 5s - 2}\begin{bmatrix} s-4 & 2 \\ 3 & s-1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} U(s) = \frac{1}{s^2 - 5s - 2}\begin{bmatrix} s-4 \\ 3 \end{bmatrix} U(s)$$



notes

- Multiply $(sI - A)^{-1}$ by $B$ to obtain $X(s)$.
- this step simplifies the expression for $X(s)$.
- $X_1(s)$ and $X_2(s)$ are now expressed in terms of $U(s)$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 6: Solve for $Y(s)$

Substitute $X(s)$ into the output equation:

$$Y(s) = CX(s) + DU(s) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = X_1(s)$$

Thus:

$$Y(s) = \frac{s-4}{s^2 - 5s - 2} U(s)$$

notes

- Substitute $X(s)$ into the output equation to find $Y(s)$.
- $Y(s)$ is directly proportional to $X_1(s)$.

## Numerical Example: $2 \times 2$ State-Space to ARMA

### Step 7: Final Result

Transfer function $H(s)$:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s-4}{s^2-5s-2}$$

and from this we get the ARMA representation of the system as before



notes

- this is the ARMA representation of the system.

## From ARMA to SS

notes

## Starting point (blending Laplace notation with time notation)

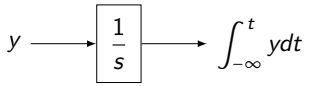
$$y(t) = \frac{b(s)}{a(s)}u(t) = \frac{b_1 s^{n-1} + \ldots + b_n}{s^n + a_1 s^{n-1} + \ldots + a_n}u(t)$$

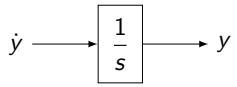


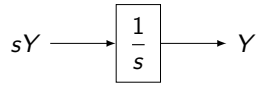
notes

- our goal is now that of converting an ARMA model to a state-space representation.
- the starting point is the transfer function in the Laplace domain.
- the numerator and denominator polynomials define the ARMA model.

## Building block = the integrator (block)



notes

- the integrator block is a fundamental building block for state-space representations.
- the integrator relates to differentiation and integration in the time domain.
- the integrator is key to constructing state variables.

## How do we use integrators?

$$\dddot{y} + a_1\ddot{y} + a_2\dot{y} + a_3 y = b_1 u$$
$$\downarrow$$
$$\dddot{y} = -a_1\ddot{y} - a_2\dot{y} - a_3 y + b_1 u$$



notes

- This shows how to rearrange a higher-order differential equation into a form suitable for state-space representation.
- the highest derivative is expressed as a function of lower derivatives and the input.
- this step is crucial for defining the state variables.

## Towards SS with a useful trick

$$y(t) = \frac{b(s)}{a(s)}u(t) = \frac{b_1 s^{n-1} + \ldots + b_n}{s^n + a_1 s^{n-1} + \ldots + a_n}u(t) \rightarrow \begin{cases} x_n(t) = \dfrac{1}{a(s)}u(t) \\ y(t) = b(s)x_n(t) \end{cases}$$

notes

- We then use the trick of defining an intermediate variable $x_n(t)$ to simplify the conversion process.
- $x_n(t)$ is the output of the denominator dynamics driven by the input $u(t)$.
- this trick separates the AR (denominator) and MA (numerator) parts of the system.

## This is an AR model on $x_n$

$$x_n(t) = \frac{1}{a(s)}u(t) \qquad \Longrightarrow \qquad a(s)x_n(t) = u(t)$$

implies

$$x_n = \frac{1}{s}x_{n-1} \qquad \rightarrow \qquad x_{n-1} = sx_n \qquad \rightarrow \qquad \begin{cases} x_{n-1} & = sx_n \\ x_{n-2} & = s^2 x_n \\ \vdots & \\ x_2 & = s^{n-2}x_n \\ x_1 & = s^{n-1}x_n \end{cases}$$



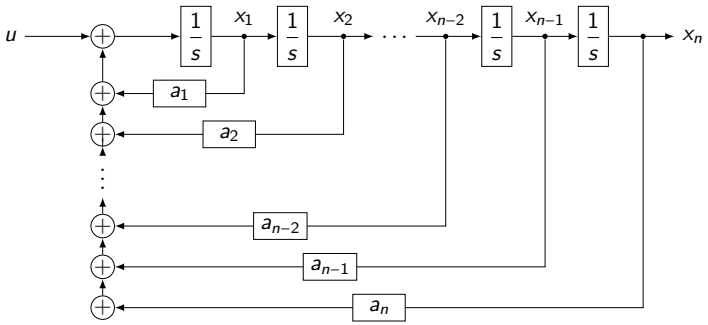
notes

- The intermediate variable $x_n(t)$ leads to the definition of state variables $x_1, x_2, \ldots, x_n$.
- each state variable is a scaled version of the next, with the scaling factor being $s$.
- this step defines the state vector $\boldsymbol{x}$.

## This is an AR model on $x_n$

$$x_n(t) = \frac{1}{a(s)}u(t) \qquad \Longrightarrow \qquad a(s)x_n(t) = u(t)$$

implies

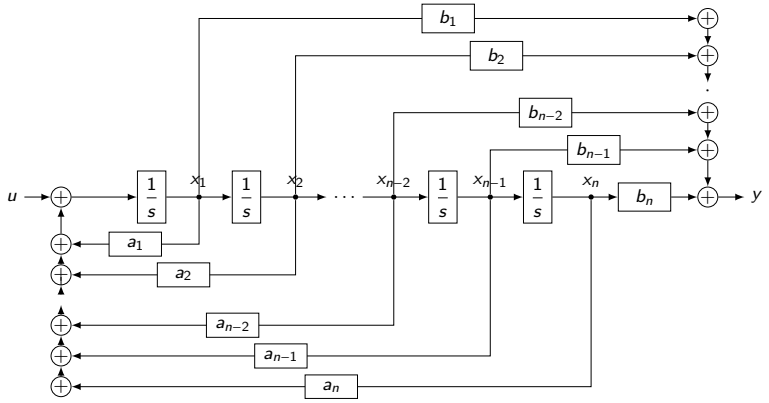


notes

- We now use a block diagram to illustrate the relationship between the state variables.
- the state variables are interconnected through integrators.
- this structure is the foundation of the state-space representation.

## Completing the picture (a MA from $x_n$ to $y$)

$$y(t) = b(s)x_n(t) = b_1x_1(t) + \ldots + b_nx_n(t)$$






notes

- the output $y(t)$ is constructed as a linear combination of the state variables.
- the coefficients of the linear combination are the numerator coefficients $b_1, b_2, \ldots, b_n$.
- this step completes the state-space representation.

## From concepts to formulas

$$\begin{cases} y(t) = b_1x_1(t) + \ldots + b_nx_n(t) \\ \dot{x}_1(t) = -a_1x_1(t) - \ldots - a_nx_n(t) + u(t) \\ \dot{x}_i(t) = x_{i-1}(t) \end{cases} \rightarrow \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases}$$

$$\dot{\boldsymbol{x}} := \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u$$

notes

- This presents the final state-space equations in matrix form.
- the structure of the $A$ matrix is then in control canonical form.
- the $B$ vector has a single non-zero entry, corresponding to the input $u(t)$.

## And $y$?

$$\begin{cases} y(t) = b_1 x_1(t) + \ldots + b_n x_n(t) \\ \dot{x}_1(t) = -a_1 x_1(t) - \ldots - a_n x_n(t) + u(t) \\ \dot{x}_i(t) = x_{i-1}(t) \end{cases} \rightarrow \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} + Du \end{cases}$$

$$y = \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



notes

- The output equation is constructed from the state variables.
- the $C$ matrix contains the numerator coefficients $b_1, b_2, \ldots, b_n$.
- this step completes the state-space representation.

## From ARMA to state space (in Control Canonical Form)

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \ldots & \ldots & -a_n \\ 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u \\ \\ y = \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \end{cases}$$

notes

- The state-space representation in control canonical form.
- the structure of the $A$ matrix becomes upper Hessenberg with a diagonal of ones.
- this form is particularly useful for control design and analysis, you will see it very often.

## Matlab / Python implementation

```
[A, B, C, D] = tf2ss([b1 b2 .. bn], [1 a1 a2 .. an])
```



notes

- the MATLAB/Python function `tf2ss` is used for converting transfer functions to state-space form.
- the input arguments are the numerator and denominator coefficients of the transfer function.
- this function automates the process of deriving the state-space matrices.
- you can use this function to verify your hand calculations only for small examples, at work don't do computations by hand

## Summarizing

**Determine** the state space structure of an
LTI system starting from an ARMA ODE

- there are some formulas, that you may simply know by heart, or that you may want to understand
- for understanding there is the need to get how the transformations work, and what is what
- likely the most important point is that to go from ARMA to SS the (likely) most simple strategy is to build the states as a chain of integrators, and ladder on top of that

notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

## Most important python code for this sub-module



notes

## These functions have also their opposite, i.e., tf2ss

- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.ss2tf.html`
- `https://python-control.readthedocs.io/en/latest/generated/control.ss2tf.html`

notes

- in the references you will see much more information than what is given in this module

Self-assessment material



notes

## Question 76

What is the role of $(sI - A)^{-1}$ in the derivation of the transfer function from a state-space model?

**Potential answers:**

I: **(wrong)** It represents the output matrix $C$.
II: **(correct)** It is used to solve for the state vector $X(s)$ in the Laplace domain.
III: **(wrong)** It defines the input matrix $B$.
IV: **(wrong)** It is the Laplace transform of the state transition matrix.
V: **(wrong)** I do not know

**Solution 1:**

$(sI - A)^{-1}$ is used to solve for the state vector $X(s)$ in the Laplace domain. It allows us to express $X(s)$ in terms of the input $U(s)$, which is then used to derive the transfer function.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 77

What is the structure of the $A$ matrix in the control canonical form of a state-space model?

**Potential answers:**

I: **(correct)** An upper Hessenberg matrix with a lower diagonal of ones and coefficients on the first row from the denominator polynomial.
II: **(wrong)** A diagonal matrix with the eigenvalues of the system.
III: **(wrong)** A lower triangular matrix with zeros on the diagonal.
IV: **(wrong)** A symmetric matrix with off-diagonal elements equal to zero.
V: **(wrong)** I do not know

**Solution 1:**

The $A$ matrix in control canonical form is an upper Hessenberg matrix with a lower diagonal of ones and coefficients from the denominator polynomial. This structure is particularly useful for control design and analysis.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 78

What is the purpose of the integrator block in the conversion from ARMA to state-space models?

**Potential answers:**

I: **(wrong)** To differentiate the input signal.
II: **(wrong)** To invert the Laplace transform of the output.
III: **(correct)** To construct the state variables as a chain of scaled integrators.
IV: **(wrong)** To compute the determinant of the state matrix.
V: **(wrong)** I do not know

**Solution 1:**

The integrator block is used to construct the state variables as a chain of scaled integrators. This allows us to define the state vector $\mathbf{x}$ and build the state-space representation.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 79

What does the transfer function $H(s) = \frac{Y(s)}{U(s)}$ represent in the context of state-space models?

**Potential answers:**

I: **(wrong)** The state transition matrix.
II: **(wrong)** The input matrix $B$.
III: **(wrong)** The determinant of the state matrix.
IV: **(correct)** The relationship between the input $U(s)$ and the output $Y(s)$ in the Laplace domain.
V: **(wrong)** I do not know

**Solution 1:**



The transfer function $H(s) = \frac{Y(s)}{U(s)}$ represents the relationship between the input $U(s)$ and the output $Y(s)$ in the Laplace domain. It is derived from the state-space model and encapsulates the system's dynamics.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 80

In the context of SISO systems, what are the dimensions of the matrices $C$ and $B$ in a state space representation?

**Potential answers:**

I: **(wrong)** $C$ is a scalar, and $B$ is a vector.
II: **(correct)** $C$ is a row vector, and $B$ is a column vector.
III: **(wrong)** $C$ is a square matrix, and $B$ is a scalar.
IV: **(wrong)** $C$ is a column vector, and $B$ is a row vector.
V: **(wrong)** I do not know

**Solution 1:**

In SISO systems, $C$ is a row vector ($1 \times n$), and $B$ is a column vector ($n \times 1$). This is because $C$ maps the state vector to the output, and $B$ maps the input to the state vector.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 81

Given the state-space matrices $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$, and $D = \begin{bmatrix} 0 \end{bmatrix}$, what is the transfer function $H(s)$?

**Potential answers:**

I: **(correct)** $H(s) = \dfrac{s-4}{s^2-5s-2}$

II: **(wrong)** $H(s) = \dfrac{s-1}{s^2-5s-2}$

III: **(wrong)** $H(s) = \dfrac{s+3}{s^2-5s-2}$

IV: **(wrong)** $H(s) = \dfrac{s-2}{s^2-5s-2}$

V: **(wrong)** I do not know

**Solution 1:**



The transfer function is $H(s) = \dfrac{s-4}{s^2-5s-2}$. This is derived by solving the state-space equations and computing $(sI - A)^{-1}B$, followed by multiplying by $C$.

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module "state space from ARMA (and viceversa)"

- one can go from ARMA to state space and viceversa
- we did not see this, but watch out that the two representations are not equivalent: there are systems that one can represent with state space and not with ARMA, and viceversa
- typically state space is more interpretable, and tends to be the structure used when doing model predictive control

notes

- the most important remarks from this sub-module are these ones

# Connections between eigendecompositions and free evolution in continuous time LTI state space systems



notes

## Contents map

| **developed content units** | **taxonomy levels** |
|---|---|
| modal analysis | u1, e1 |

| **prerequisite content units** | **taxonomy levels** |
|---|---|
| LTI ODE | u1, e1 |
| state space system | u1, e1 |
| eigenvalue | u1, e1 |
| eigenspace | u1, e1 |
| phase portrait | u1, e1 |

notes

## Main ILO of sub-module

"Connections between eigendecompositions and free evolution in continuous ti

**Analyse** the structure of the free evolution of the state variables by means of the eigendecomposition of the system matrix



notes

- by the end of this module you shall be able to do this

## Important initial remark

focus = LTI in state space and free evolution, meaning $u(t) = 0$, and thus

$$\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= C\boldsymbol{x} \end{cases} \quad \mapsto \quad \begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} \\ y &= C\boldsymbol{x} \end{cases}$$

notes

- the first simplification that we consider in this module is that we ignore the forced response

## . . . and then an important disclaimer

$$\begin{cases} \dot{\boldsymbol{x}} &= A\boldsymbol{x} \\ y &= C\boldsymbol{x} \end{cases}$$

*the module ignores what happens if $A$ is non-diagonalizable*



notes

- for the sake of aiding graphical intuition the module ignores what the visualizations should be in that cases where $A$ is not diagonalizable (and thus should involve generalized eigenspaces - a concept that the reader may ignore what they are, for the sake of this course)

## Roadmap

- set the focus just on $\boldsymbol{x}$, and not on $\boldsymbol{y}$
- get a graphical intuition of what $A\boldsymbol{x}$ means
- interpreting eigenspaces in the real of LTI continuous time systems
- adding the "superposition principle" ingredient to the mixture

notes

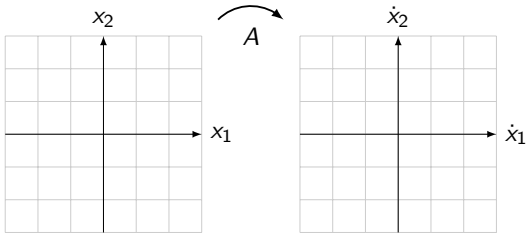
- we will follow a specific pattern to get to the final point of the module

## What does Ax mean, graphically?



notes

## The physical meaning of the operation $\dot{\boldsymbol{x}} = A\boldsymbol{x}$




$\Longrightarrow$ structure of $A$ determines how the time derivative $\dot{\boldsymbol{x}}$ is, and how the time derivative is determines the stability and time-evolution properties of the system. E.g.,

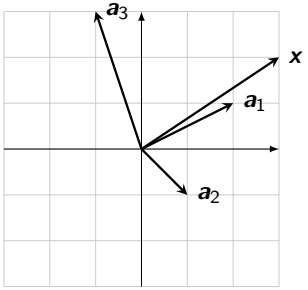
$$\text{span}\,(A) = \begin{bmatrix} +1 \\ -1 \end{bmatrix} \quad \Longrightarrow \quad \text{if } x_1 \text{ grows then } x_2 \text{ diminishes, and viceversa}$$

notes

- then if we see that the columns of $A$ generate this span, then we know that this must happen to the system
- recall that $x_1$ and $x_2$ are often physically interpretable variables, such as position and velocity. Being able to say sentences like this one means being able to describe in a qualitative way the mechanisms underlying the evolution of the system

## How may we represent vectors and matrices?

$$\boldsymbol{x} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$



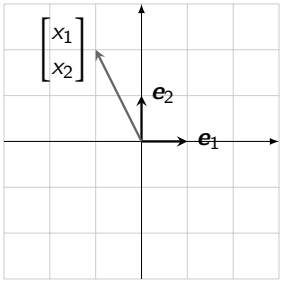



notes

- in the cartesian plane we can represent these objects as opportune (column) vectors
- note that due to our conventions we will draw a matrix $A$ as a set of column vectors
- $\boldsymbol{a}_1$ is the first column, and so on

## But what is a vector?

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_2 = \boldsymbol{e}_1 x_1 + \boldsymbol{e}_2 x_2$$

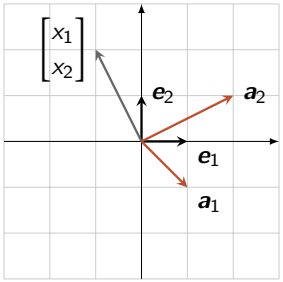


notes

- a vector is actually a combination of the elements of the canonical basis
- in a sense, the vector itself is defined by this basis
- also this concept will be expanded in later on courses ...
- consider this also a sort of superposition of the effects

## So, what is a matrix-vector product, geometrically?

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad A\boldsymbol{x} = ?$$





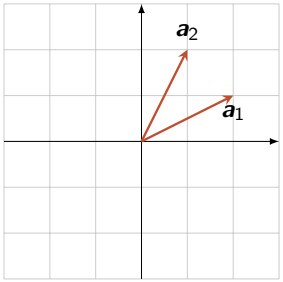

notes

- and this is a re-interpretation of the same thing we saw before
- matrix vector multiplication means multiplying each of the columns for the corresponding scalar, BUT this time we can see the whole operation
- the columns of the matrix are the transformed versions of the canonical basis, then the vector $[ab]^T$ goes expanding / compressing / flipping (depending on the values of its components) each of the transformed vectors independently

## The effect of eigenspaces

notes

## Eigenvectors of a square matrix




*are there some directions that get only stretched, i.e., that do not rotate?*

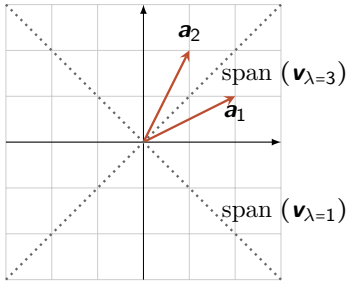
$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mapsto \quad \boldsymbol{v}_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \boldsymbol{v}_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



notes

- note that we are defining things in a way for which this concept relates only to the case of square matrices
- the concept of eigenvector relates to the concept of warping the fabric of space
- if you warp space in this way as this matrix says, are there some 'lines' that remain untouched by the transformation? I.e., that do not rotate, but only get compressed or stretched?
- in this specific case there are two: the one defined by $[\alpha, \alpha]^T$, and the one defined by $[-\alpha, \alpha]^T$
- formally the question can be formulated in this way, where both $\lambda$ and $\boldsymbol{x}$ are variables that shall be identified (i.e., read this as "for which $\boldsymbol{x}$ and $\alpha$ does this happen?")
- $\lambda$, the eigenvalues, should be interpreted as the "stretching factors", while $\boldsymbol{x}$ is any element within this "line that does not rotate"
- from the physical intuitions that we derive by looking at how the fabric of space warps, we get these two guesses
- putting these two guesses in the equation we see that they verify it, so they are actually the objects we were looking for

## Eigenspaces = subspaces spanned by the eigenvectors-eigenvalues pairs



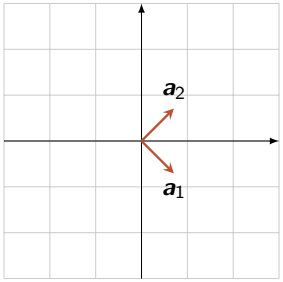

*eigenspaces = subspaces spanned by the eigenvectors*

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mapsto \quad \boldsymbol{v}_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \boldsymbol{v}_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

notes

- the eigenspaces are then that subspaces that are defined by the eigenvectors
- note that each eigenvalue has its own eigenvector – even if this is a bit imprecise; we will see the full picture when we discuss Jordan forms in a few units

## Eigenvectors: sometimes you may seem them from the transformation of the hypercube, sometimes you don't

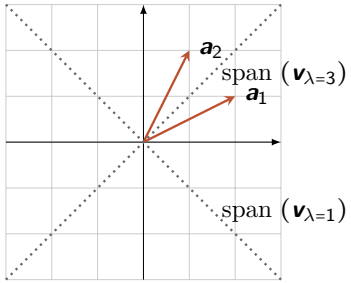





notes

- there is a problem with visualizing eigenvalues, though: sometimes eigenvalues may be complex (something that is associated to rotation, as we will see soon)
- thus for matrices that perform rotations of the fabric the graphical approach seen before cannot apply
- very instrumental to understand why is the video from 3Blue1Brown `https://www.youtube.com/watch?v=v0YEaeIClKY&ab_channel=3Blue1Brown`

## Why do we like eigenspaces?

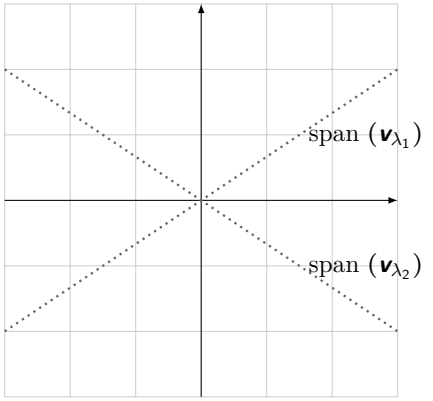



because $\dot{x} = \lambda x \implies$ "keep moving along that line"

notes

- the eigenspaces are so that if the initial condition of the system is on that subspace, then the direction of motion is aligned with that subspace, and this means that the system will stay there

## Why do we like eigenspaces? Take 2



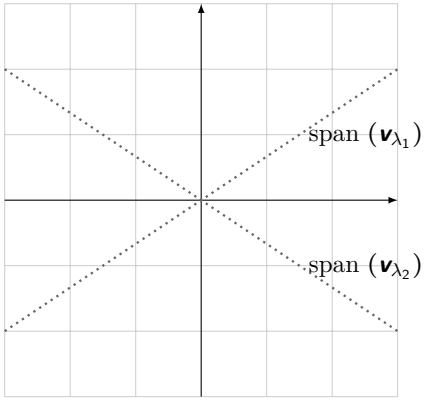

superposition principle $\Longrightarrow$ one can caracterize the whole phase portrait



notes

- moreover the eigenvectors may define also where the system will end up in free evolution starting from a generic point
- this concept is connected with the one of "mode of a system"

## Why do we like eigenspaces? Take 3



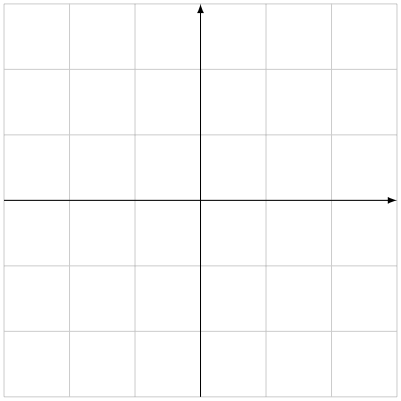

the trajectory along each eigenspace is driven by a first order differential equation

$\Longrightarrow$ if $\boldsymbol{x}_0 \in \mathrm{span}(\boldsymbol{v}_\lambda)$, then $\boldsymbol{x}(t) = e^{\lambda t}\boldsymbol{x}_0$

notes

- finally we get that the movement along each specific eigenspace is essentially like $\dot{y} = \alpha y$, and thus dominated by exponentials

## Examples





notes

- likely best to watch the video of the lecture here, that there is going to be some examples of how different eigenvalues and eigenspaces will imply different phase portraits

## How do we compute eigenvalues and eigenvectors numerically?

```
eigenvalues, eigenvectors = numpy.linalg.eig(A)
```

notes

- computationally speaking, all these things have been implemented in very robust and user friendly libraries

## Summarizing

**Analyse** the structure of the free evolution of the state variables by means of the eigendecomposition of the system matrix

- find the eigenspaces and the eigenvalues
- depending on the values of the eigenvalues, understand how the trajectories along the eigenspaces look like
- depending on the relative angle among the eigenspaces, infer the phase portrait
- if the system matrix is not diagonalizable, then this concept complicates due to the presence of generalized eigenspaces (not in this module)



notes

- you should now be able to do this, following the pseudo-algorithm in the itemized list

Most important python code for this sub-module

notes

## Linear algebra in general

`https://numpy.org/doc/2.1/reference/routines.linalg.html`



notes

- this library can be used to do much more than eigendecompositions

## Self-assessment material

notes

## Question 82

What does a positive eigenvalue imply about the system's behavior along its corresponding eigenspace?

**Potential answers:**

I: **(correct)** The state grows exponentially along that eigenspace.
II: **(wrong)** The state decays exponentially along that eigenspace.
III: **(wrong)** The state oscillates along that eigenspace.
IV: **(wrong)** The state remains constant along that eigenspace.
V: **(wrong)** I do not know.

**Solution 1:**

A positive eigenvalue implies that the state grows exponentially along the corresponding eigenspace. This is derived from the solution $\boldsymbol{x}(t) = e^{\lambda t}\boldsymbol{x}_0$, where $\lambda > 0$ leads to exponential growth.



notes

- see the associated solution(s), if compiled with that ones :)

## Question 83

In the context of free evolution of a linear time-invariant (LTI) system, what does the equation $\dot{\boldsymbol{x}} = A\boldsymbol{x}$ represent?

**Potential answers:**

I: **(wrong)** The evolution of the system's output over time.
II: **(correct)** The evolution of the state variables over time, influenced by the system matrix $A$.
III: **(wrong)** The relationship between input and output signals in the system.
IV: **(wrong)** The response of the system to external inputs.
V: **(wrong)** I do not know

**Solution 1:**

The equation $\dot{\boldsymbol{x}} = A\boldsymbol{x}$ describes the free evolution of the system's state variables over time, where the rate of change of the state vector $\boldsymbol{x}$ is determined by the system matrix $A$.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 84

Why is it useful to consider the eigendecomposition of the system matrix $A$ in analyzing the free evolution of state variables?

**Potential answers:**

I: **(wrong)** It simplifies calculating the system's forced response.
II: **(wrong)** It directly determines the output $y$ of the system.
III: **(correct)** It helps identify invariant directions (eigenvectors) and growth/decay rates (eigenvalues) that govern the system's behavior over time.
IV: **(wrong)** It only affects the graphical representation, not the actual system behavior.
V: **(wrong)** I do not know



**Solution 1:**

Eigendecomposition reveals the system's eigenvectors and eigenvalues, which represent invariant directions and the associated rates of exponential growth or decay. This insight simplifies the analysis of the system's dynamics.

notes

- see the associated solution(s), if compiled with that ones :)

## Question 85

In a graphical representation, what does the matrix-vector product $A\boldsymbol{x}$ illustrate in the context of system dynamics?

**Potential answers:**

I: **(wrong)** The projection of the state vector onto the output space.
II: **(wrong)** The response of the system to a unit impulse.
III: **(correct)** Where the trajectory of the system is going, starting from $\boldsymbol{x}$.
IV: **(wrong)** The change in the input signal over time.
V: **(wrong)** I do not know

**Solution 1:**

The product $A\boldsymbol{x}$ represents $\dot{\boldsymbol{x}}$, that indicates the system's dynamics on the state evolution.

notes

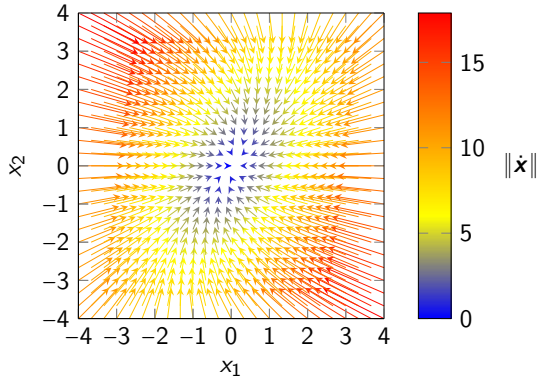
- see the associated solution(s), if compiled with that ones :)

notes

- see the associated solution(s), if compiled with that ones :)

## Question 86

Which eigenvalues and eigenspaces would you say characterize the system matrix $A$, looking just at this phase portrait?



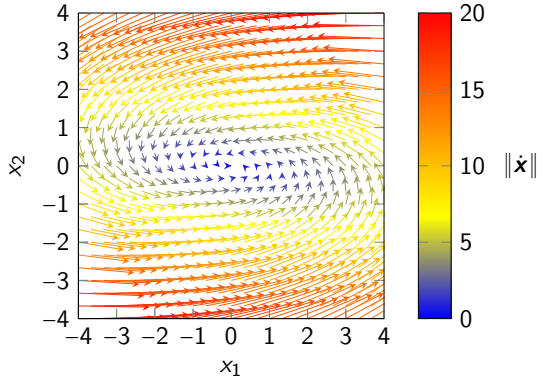



**Solution 1:**

The eigenspaces are associated to that subspaces identified by a series of aligned quivers. The eigenvalues are positive or negative depending on the movement. If there are complex eigenvalues then there are spiral like movements.

## Question 87

Which eigenvalues and eigenspaces would you say characterize the system matrix $A$, looking just at this phase portrait?






**Solution 1:**

notes

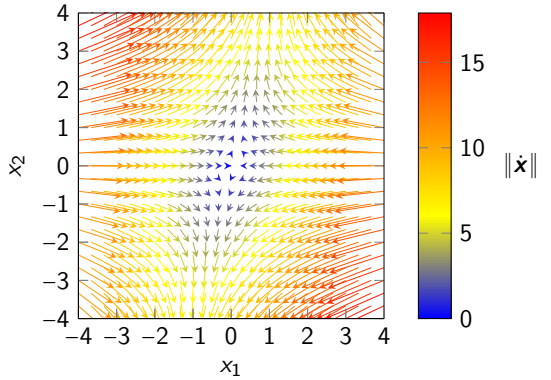
- see the associated solution(s), if compiled with that ones :)

notes

- see the associated solution(s), if compiled with that ones :)

## Question 88

Which eigenvalues and eigenspaces would you say characterize the system matrix $A$, looking just at this phase portrait?



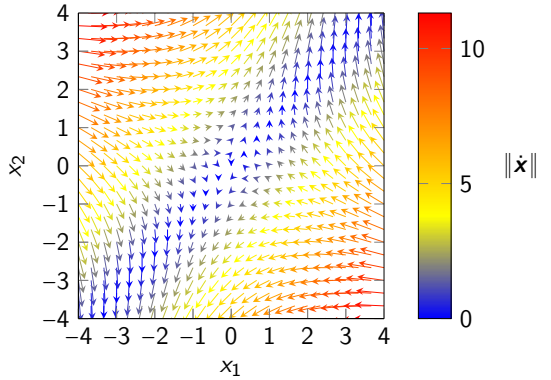



**Solution 1:**

The eigenspaces are associated to that subspaces identified by a series of aligned quivers. The eigenvalues are positive or negative depending on the movement. If there are complex eigenvalues then there are spiral like movements.

## Question 89

Which eigenvalues and eigenspaces would you say characterize the system matrix $A$, looking just at this phase portrait?






**Solution 1:**

notes

- see the associated solution(s), if compiled with that ones :)

## Recap of sub-module

## "Connections between eigendecompositions and free evolution in continuous ti

- the eigenvalues of the system matrix $A$ give the growth / decay rates of the modes $e^{\alpha t}$ of the free evolution of the system
- along eigenspaces, the trajectory of the free evolution is "simple", i.e., aligned with that eigenspace
- the kernel of the system matrix gives us the equilibria corresponding to $u = 0$



notes

- the most important remarks from this sub-module are these ones

# Metacognition Activities

notes

# In-Class Metacognition Activities



notes

## Concept Mapping Challenge

- **Activity:** Draft individual concept maps connecting key course concepts (e.g., ODE classifications, phase portraits). Compare and refine in small groups.
- **Focus:** Reflect on conceptual structures and interrelations.
- **Debrief:** Share one insight on evolving understanding.

notes

- TODO

## Error Analysis Workshop

- **Activity:** Analyze common mistakes related to ODEs or phase portraits. Identify causes and prevention strategies.
- **Focus:** Reflect on common problem-solving errors.
- **Debrief:** Discuss validation strategies.



notes

- TODO

## Think-Aloud Pair Problem Solving

- **Activity:** Solve a problem aloud while a partner asks reflective questions. Swap roles afterward.
- **Focus:** Encourage awareness of reasoning strategies.
- **Debrief:** Discuss insights on different approaches.

notes

- TODO

# At-home Self-paced Metacognition Activities



notes

## Reflection Journal

- **Task:** Write reflections after problem-solving sessions, addressing: intuitive concepts, challenges, strategies used, and misconceptions.
- **Focus:** Promote awareness of learning strategies.

notes

- TODO

## Self-Explanation Videos

- **Task:** Record a short video explaining a concept (e.g., linearization validity). Reflect on uncertain parts.
- **Focus:** Reinforce understanding by articulation.



notes

- TODO