Table of Contents I

notes

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

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
RR	u1, e1

notes

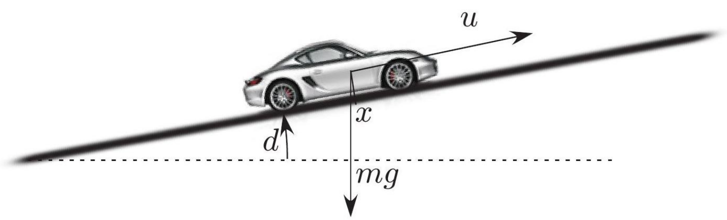
Main ILO of sub-module "what is control"

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



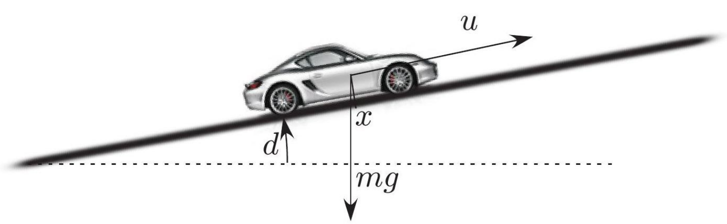
main parameters:

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

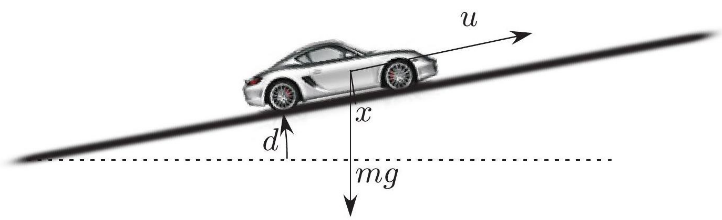
from Euler backwards discretization: $x^{++} = a_1 x^{+} + a_0 x + u + f(d)$

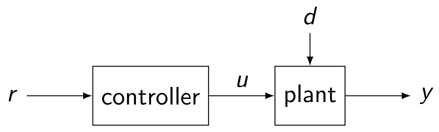
control objective: minimize $|y[k] - r[k]|$ with $r[k] =$ wished speed

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 $d[k]$ will happen, and I compensate for that"





$$u[k] = \text{something that compensates that } \widehat{d}[k]$$

notes

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

Open loop / feedforward control: "I think I know which $d[k]$ will happen, and I compensate for that"

$$u[k] = \text{something that compensates that } \widehat{d}[k]$$

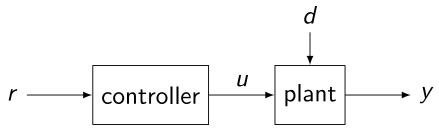
problem: if $\widehat{d} - d$ is big, then we expect $y - r$ to be big too, and we won't be able to note this!

notes

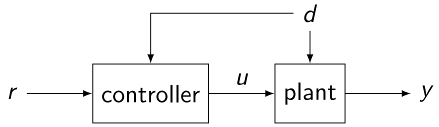
- of course pure feedforward / open loop control may be problematic, especially for the fact that there won't be ways to detect if there are big errors and react to them

Note that open loop $\neq$ 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 1

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

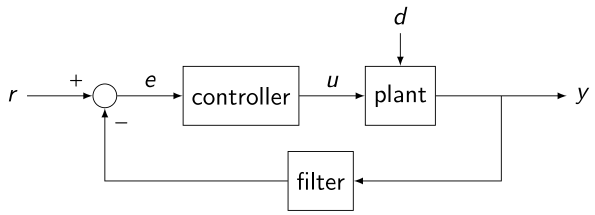
Solution 1:

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

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)

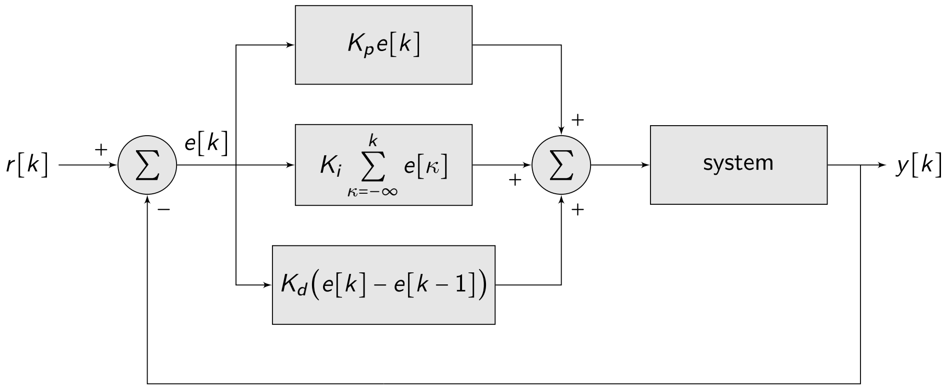
Main dichotomy on how to build a feedback controller

- 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

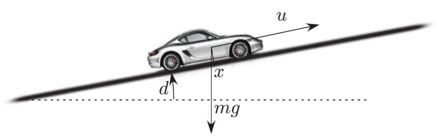


implicit assumption: we can measure $y[k]$*!* (see also `https://www.youtube.com/watch?v=UR0hOmjaHp0`!)

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



$$u[k] = \quad K_p \text{ error right now}$$
$$+K_i \text{ sum of all past errors}$$
$$+K_d \text{ current tendency of the error}$$

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 2

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:

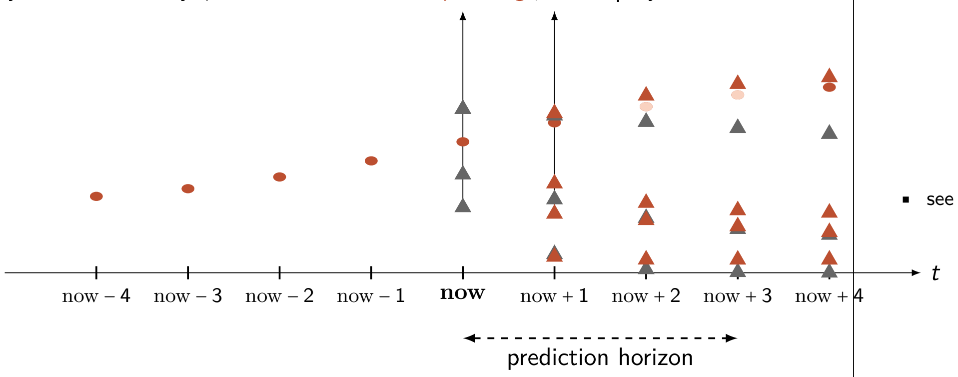
A PID controller is not guaranteed to work well in all cases; its performance depends on proper tuning. Poorly tuned PID controllers can lead to instability, slow response, or excessive oscillations. The three gains (proportional (K_p), integral (K_i), and derivative (K_d)) must be adjusted based on 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.
Tuning methods like Ziegler-Nichols, Cohen-Coon, or optimization-based approaches help in achieving a well-performing PID controller. Therefore, the correct answer is: *no, it depends on how well tuned it is*.

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 = "best" forecasted y execute $u(\text{now})$ measure, and repeat



notes

- see also https://www.youtube.com/watch?v=UR0hOmjaHp0!

Question 3

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.

- what is control 16

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.
- **Computation Time:** MPC solves an optimization problem at every step. If computations take too long, real-time implementation may fail.
- **Disturbances & Uncertainty:** If disturbances or modeling errors are not

notes

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

Question 4

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.

- what is control 17

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.

notes

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

Question 5

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:

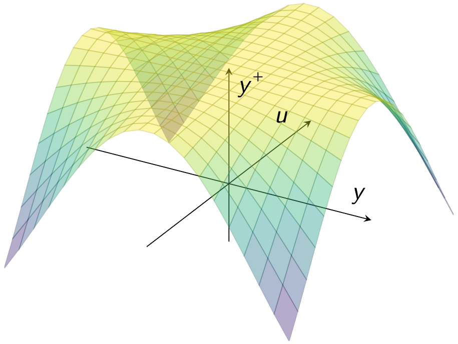
- **System Variability:** Closed-loop control is beneficial when the system is subject to disturbances or uncertainties, as it can adjust in real time. However, for predictable systems with no disturbances, open-loop control may be simpler and more efficient.

notes

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

But eventually, what is control?

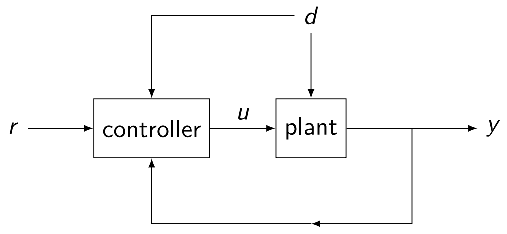
an algorithm to compute $u[k]$ starting from the available information



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 6

Which of the following is a key disadvantage of open-loop control compared to closed-loop control?

Potential answers:

I: **(wrong)** It requires more computational resources.
II: **(correct)** It cannot correct for disturbances or system variations.
III: **(wrong)** It is slower to respond to changes.
IV: **(wrong)** It is more expensive to implement.
V: **(wrong)** I do not know.

Solution 1:

The key disadvantage of open-loop control is that it cannot correct for disturbances or system variations. Unlike closed-loop control, which uses feedback to adjust the control action, open-loop control operates without feedback, making it unable to compensate for unexpected changes or errors in the system.

notes

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

Question 7

What is the primary purpose of the integral term (K_i) in a PID controller?

Potential answers:

I: **(wrong)** To amplify the current error.
II: **(correct)** To eliminate steady-state error by summing past errors.
III: **(wrong)** To predict future errors based on the current trend.
IV: **(wrong)** To reduce the sensitivity to noise.
V: **(wrong)** I do not know.

Solution 1:

The primary purpose of the integral term (K_i) in a PID controller is to eliminate steady-state error by summing past errors over time. This ensures that the system reaches the desired setpoint without any residual error.

notes

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

Question 8

What is a key advantage of Model Predictive Control (MPC) over PID control?

Potential answers:

I: **(wrong)** It requires less computational power.
II: **(wrong)** It is easier to tune.
III: **(correct)** It can handle constraints and optimize future behavior.
IV: **(wrong)** It is always more stable.
V: **(wrong)** I do not know.

Solution 1:

A key advantage of Model Predictive Control (MPC) over PID control is its ability to handle constraints and optimize future behavior. MPC uses a model of the system to predict future states and optimize control actions over a finite horizon, making it suitable for complex, constrained systems.

notes

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

Question 9

In a combined feedforward and feedback control system, what is the role of the feedforward component?

Potential answers:

I: **(wrong)** To correct for errors after they occur.
II: **(correct)** To compensate for known disturbances before they affect the system.
III: **(wrong)** To measure the system output and adjust the control action.
IV: **(wrong)** To reduce the computational load of the feedback controller.
V: **(wrong)** I do not know.

Solution 1:

In a combined feedforward and feedback control system, the role of the feedforward component is to compensate for known disturbances before they affect the system. This proactive approach helps reduce the impact of disturbances, while the feedback component corrects for any remaining errors.

what is control 5

notes

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

Question 10

What is the main goal of automatic control in the context of system dynamics?

Potential answers:

I: **(wrong)** To increase the complexity of the system.
II: **(correct)** To manipulate the system's dynamics to achieve a desired behavior.
III: **(wrong)** To eliminate all disturbances from the system.
IV: **(wrong)** To reduce the need for sensors and actuators.
V: **(wrong)** I do not know.

Solution 1:

The main goal of automatic control in the context of system dynamics is to manipulate the system's dynamics to achieve a desired behavior. This involves designing control algorithms that compute appropriate control inputs ($u[k]$) based on available information, ensuring the system behaves as intended.

what is control 6

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