Table of Contents I

notes

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

Introduction to Open-Loop Controller Design

notes

Contents map

developed content units	taxonomy levels
closed-loop control	u1, e1
open-loop control	u1, e1

prerequisite content units	taxonomy levels
LTI system	u1, e1

notes

Main ILO of sub-module "Introduction to Open-Loop Controller Design"

Explain the difference between open-loop and closed-loop control architectures, using the car speed control example as context.

Formulate the mathematical model of a physical system (car dynamics) and linearize it around a desired equilibrium point.

Design a basic open-loop controller by inverting the system model or its DC gain, and analyze its limitations in terms of disturbance rejection and parameter sensitivity.

Evaluate the trade-offs between response speed and actuator effort when implementing pole cancellation in open-loop control design.

Assess the practical challenges of implementing open-loop control in real-world systems, using the automatic gate example as a case study.

notes

- by the end of this module you shall be able to understand and apply basic open-loop control design principles

Roadmap

- overview of the structure
- building intuition through an example

notes

- In this part of the course, we're going to take our first steps into the world of controller design. Specifically, well look at open-loop control, and well use a simple but illustrative example controlling the speed of a car to build our intuition.
- The idea is to first understand the control architecture, and then go step-by-step through the design procedure using a model of the system.

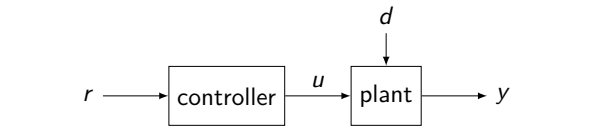
What do we learn now?

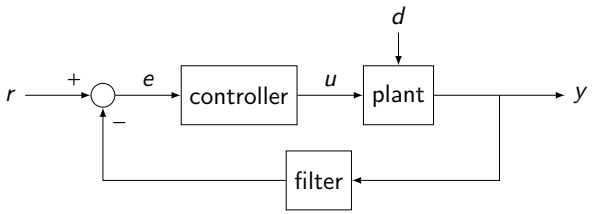
introduction to how to design an open-loop controller

notes

- The goal now is to understand how we can design an open-loop controller. That means we're going to find a way to compute the control input $u(t)$ that will hopefully drive the system output $y(t)$ to follow a desired behavior, without relying on feedback.
- This is conceptually simpler than feedback control, but as well see, it comes with significant limitations.

Basic Control Architectures: open-loop vs. closed-loop (or feedback)

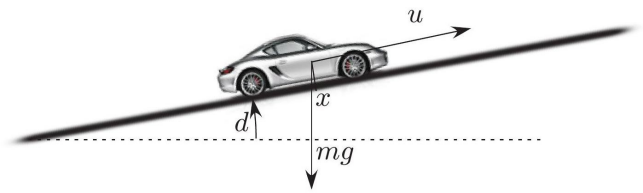




notes

- Let me briefly highlight the difference between open-loop and closed-loop control. In open-loop control, we decide on the control action $u(t)$ purely based on the reference $r(t)$ and a model there is no checking whether the output actually follows.
- In closed-loop control, we use feedback: we measure the output $y(t)$, compare it with the reference $r(t)$, and adjust $u(t)$ accordingly.
- In this course, we will mostly focus on feedback control, but its important to understand open-loop control as a baseline.

Example: car speed control



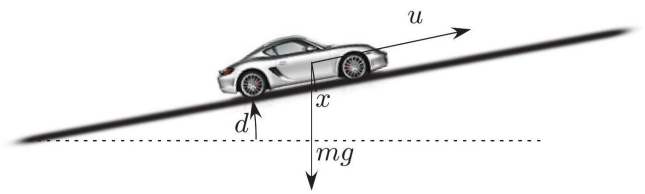
Signals:

- output: $y(t)$ car speed
- input: $u(t)$ force generated by an engine
- disturbance: $d(t)$ road slope

notes

- Let's consider a car moving on an inclined road. We want to control its speed that's our output, $y(t)$.
- The force generated by the engine is our control input, $u(t)$. And then we have the road slope, $d(t)$, acting as a disturbance that affects the motion of the car but is not something we can directly control.

Example: car speed control



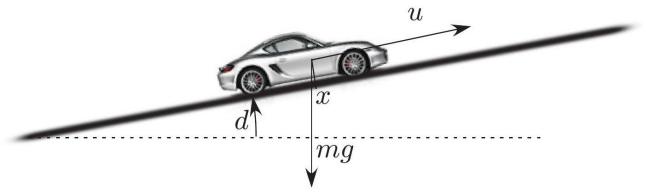
Goal:

find $u(t) = \phi(\star)$ such that $y(t) \simeq \alpha r(t)$ regardless of d

notes

- Our goal is to find a function $\phi(\star)$ that is, the control law which takes in known quantities and returns the control input $u(t)$ that will drive $y(t)$ to approximately follow $\alpha r(t)$, no matter what the disturbance $d(t)$ is doing.
- α is a scaling factor it might be 1, or something else depending on the system and units.

Example: car speed control



Model parameters:

- m: car mass
- b: friction coefficient
- g: gravitational acceleration

notes

- These are the key physical parameters of our model. We'll use them to build a mathematical model of how the car moves.
- m is the mass of the car heavier cars are harder to accelerate. b is a friction or damping coefficient, representing things like air resistance and internal friction. g is the acceleration due to gravity.

how to do it?

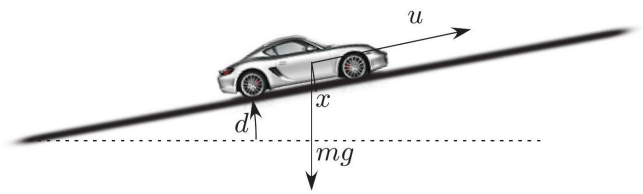
Strategy:

- find the system model
- linearize it around the desired equilibrium
- find the transfer function of the system
- design the controller so that the closed-loop system transfer function is the one we want

notes

- This is our plan. First, we need a model how does the system behave?
- Then we linearize it, because most controller design tools assume linear systems.
- After that, we move to the frequency domain using the Laplace transform, and describe the system with a transfer function.
- Finally, we design a controller so that the resulting closed-loop behavior matches our design goals.

Example: car speed control



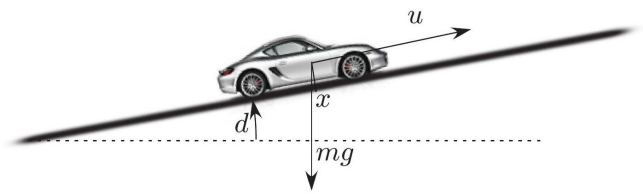
Model:

$$m\dot{y}(t) = -by(t) + u(t) - mg\sin\left(d(\int ydt)\right)$$

notes

- This is the nonlinear model. The dynamics include the slope of the road, which depends on the cars position thats why you see the integral of $y(t)$ in there.
- The term with the sine is what makes this model nonlinear and harder to work with directly.

Example: car speed control



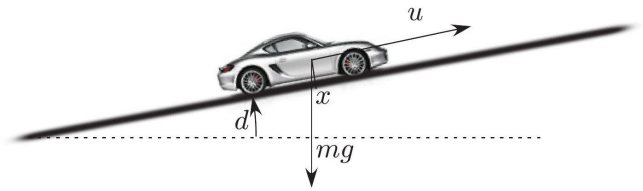
Linearized model:

$$m\dot{y}(t) = -by(t) + u(t) - mgd(t)$$

notes

- After linearization, things simplify. The sine term becomes just $d(t)$, assuming small angles.
- Now we have a linear differential equation, which is much easier to handle and is suitable for our controller design.

Example: car speed control



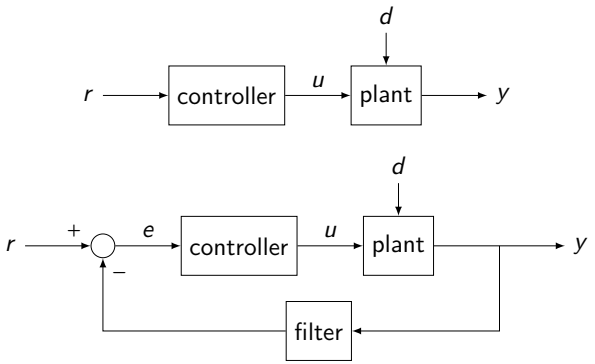
Laplace (model):

$$Y(s) = \frac{1}{ms + b}\big(U(s) - mgD(s)\big)$$

notes

- Taking the Laplace transform of the linear model gives us this transfer function form.
- It clearly shows how $Y(s)$ depends on both $U(s)$ and the disturbance $D(s)$. Our job is to choose $U(s)$ to get $Y(s)$ as close as possible to what we want, ideally cancelling the effect of $D(s)$.

Fundamental architectures for car speed control: open-loop vs. closed-loop (or feedback)

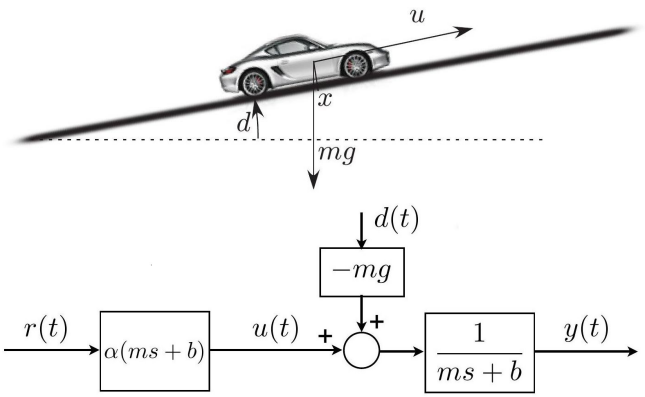


notes

- This figure shows the key difference between open-loop and feedback control architectures.
- In open-loop, we compute $u(t)$ based on the reference and model no correction is done if the system doesn't behave as expected.
- In feedback control, we constantly compare the actual output with the reference, and adjust $u(t)$ accordingly. This is much more robust to disturbances and model inaccuracies.

Car speed control in open loop

= inversion of the system model

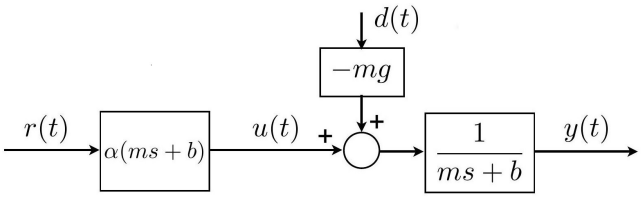


goal: $y(t) \simeq \alpha r(t)$ i.e. $W_{ry}(s) = \alpha$

notes

- Here, we are inverting the model that is, we try to solve for $u(t)$ that makes $y(t)$ behave like $\alpha r(t)$.
- This approach works perfectly on paper if the model is exact and there are no disturbances. But reality is messier.
- Also, as we'll see, this kind of controller may not be implementable in practice.

What are the problems?



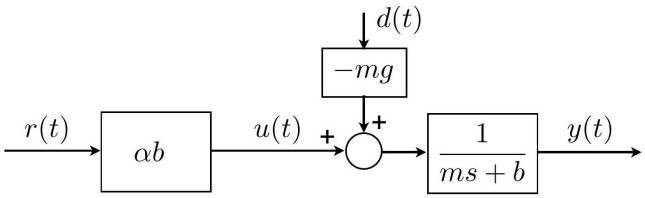
How to solve this? We settle for the output to follow the reference only at steady state
$\Longrightarrow$ we only invert the DC gain

notes

- The first issue is that the controller is non-causal it requires knowledge of future values of the reference. Thats not physically realizable.
- The second issue is sensitivity to modeling errors and disturbances if anything deviates from our model, we lose performance.
- One way to mitigate this is to design the system to only track the reference in steady-state that is, we try to match the final value, not the full trajectory.

Car speed control in open loop

= inversion of the system model gain



$$\Longrightarrow W_{ry}(s) = \frac{\alpha b}{ms+b} \Longrightarrow W_{ry}(0) = \alpha \Longrightarrow y(\infty) = \alpha r(\infty)$$

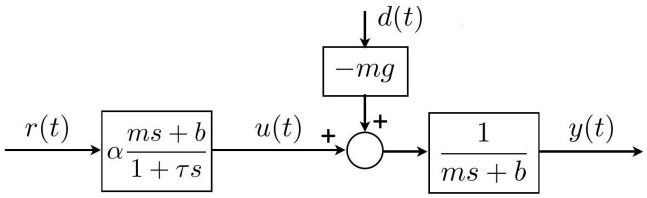
but the rise time depends on m and b!

notes

- So here's the problem with this simple approach: while we can set the final speed correctly through , we have no control over how quickly the car reaches that speed.
- The response speed is entirely determined by the car's mass m and friction coefficient b - parameters we often can't change in practice.
- Imagine you're driving and want to accelerate to 100 km/h. With this method, you'd reach the right speed eventually, but you couldn't choose whether it takes 5 seconds or 30 seconds!

Car speed control in open loop

= inversion of the system model gain and cancellation of the stable pole



$$\Longrightarrow W_{ry}(s) = \frac{\alpha}{\tau s + 1} \Longrightarrow W_{ry}(0) = \alpha \Longrightarrow y(\infty) = \alpha r(\infty)$$

and the rise time depends on the controller parameters

notes

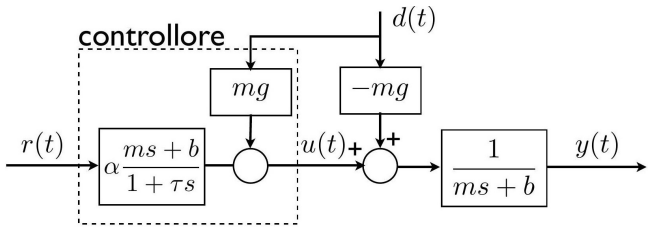
- Now we've improved our controller to also shape the response time through . But there's a catch!
- The faster we want the system to respond (smaller), the harder we're pushing our actuators - in this case, the car's engine.
- Think about flooring the accelerator pedal - you get quick acceleration, but at what cost? Fuel efficiency suffers, engine wear increases, and passengers get uncomfortable.
- There's always this trade-off between performance and practical limitations in real systems.

What if we could measure or predict d?

notes

- Let me ask you this: In real driving, what's the biggest factor affecting your speed besides the accelerator? That's right - hills!
- If we could measure the slope ahead (maybe with sensors or GPS data), could we use that information to improve our control?
- This is like when you see a hill coming and press the accelerator a bit more before you start slowing down.
- But here's the question: How practical is it to measure all possible disturbances perfectly? And what happens if our measurements are slightly off?

Open-loop control of a car's speed



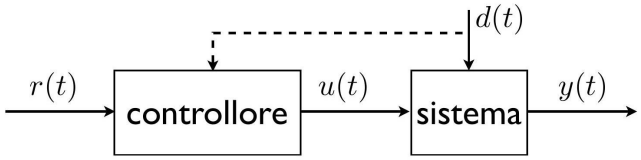
$$Y(s) = W_{ry}(s)R(s) + W_{dy}(s)D(s)$$

$$W_{dy}(s) = (mg - mg)\frac{1}{ms+b} = 0$$

notes

- Here we see mathematically how measuring and compensating for disturbances could theoretically eliminate their effect completely.
- In practice, this would be like having a perfect hill detection system that automatically adjusts your accelerator exactly right.
- But remember, this requires perfect knowledge - we'd need to know the exact mass of the car, the exact slope angle, and react instantly.
- What happens if we overestimate the slope? We might accelerate too much. Underestimate? We still slow down.
- This shows both the potential and limitations of disturbance compensation.

Generalization of the example



- Objectives of open-loop control: $W_{ry}(s) \simeq \alpha \quad W_{dy}(s) \simeq 0$
- Problems of open-loop control:
  - need to know exactly the model and its parameters
  - need to measure disturbances
  - impossibility to control unstable systems

notes

- Let me summarize the key takeaways from our car example to general systems.
- Open-loop control is like trying to shoot a target blindfolded - if you know exactly how to aim and nothing changes, you might hit it.
- But in reality, wind might blow (disturbances), your arm might shake (parameter variations), or the target might move (model uncertainties).
- The fundamental issue is there's no feedback - no way to see if you're actually hitting the target and adjust accordingly.
- This is why in most real-world applications, we need something more robust - which brings us to closed-loop control.

Practical example: DIY automatic gate in open loop

Recipe:

- buy and install the gate
- do some experiments to understand the linear regime
- create a model from the data
- invert it
- build an RCL circuit that implements that controller

notes

- Imagine you're actually building this automatic gate controller. Sounds straightforward, right?
- But what happens when temperatures change and the gate's friction increases in winter?
- Or when leaves get stuck in the mechanism adding unexpected resistance?
- Your carefully designed open-loop controller wouldn't know to push harder - the gate would just stop short of closing completely.
- This is why you see real automatic gates using feedback - sensors that check if the gate actually reached the end position.
- Open-loop works fine in ideal, controlled conditions, but reality is rarely so cooperative!

Summarizing

Explain the difference between open-loop and closed-loop control architectures, using the car speed control example as context.

Formulate the mathematical model of a physical system (car dynamics) and linearize it around a desired equilibrium point.

Design a basic open-loop controller by inverting the system model or its DC gain, and analyze its limitations in terms of disturbance rejection and parameter sensitivity.

Evaluate the trade-offs between response speed and actuator effort when implementing pole cancellation in open-loop control design.

Assess the practical challenges of implementing open-loop control in real-world systems, using the automatic gate example as a case study.

notes

- you should now be able to understand and apply basic open-loop control design principles

Most important python code for this sub-module

notes

The python.control library

. . . as virtually in all the modules of this part of the course

notes

- the control library is basically essential

Self-assessment material

notes

Question 1

What is the fundamental limitation of open-loop control compared to closed-loop control?

Potential answers:

I: **(wrong)** It requires more computational power
II: **(correct)** It cannot compensate for unmeasured disturbances or model inaccuracies
III: **(wrong)** It only works for nonlinear systems
IV: **(wrong)** It requires more sensors than closed-loop control
V: **(wrong)** I do not know

Solution 1:

The key limitation of open-loop control is its inability to compensate for disturbances or model errors since it doesn't use feedback. While closed-loop control can adjust based on the actual output, open-loop control blindly applies its predetermined control action regardless of what actually happens to the system.

notes

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

Question 2

Why do we typically linearize nonlinear system models before designing controllers?

Potential answers:

I: **(wrong)** Because all physical systems are fundamentally linear
II: **(wrong)** Because nonlinear controllers cannot be implemented in practice
III: **(correct)** Because most controller design tools and analysis methods are developed for linear systems
IV: **(wrong)** Because linearization increases system stability
V: **(wrong)** I do not know

Solution 1:

We linearize nonlinear models because the vast majority of controller design techniques (including transfer function analysis, frequency response methods, and pole placement) are developed for linear systems. Linearization allows us to apply these powerful tools while maintaining reasonable accuracy near the operating point.

notes

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

Question 3

In the car speed control example, why can't perfect disturbance rejection be achieved in practice through open-loop control?

Potential answers:

I: **(wrong)** Because disturbances cannot be measured under any circumstances
II: **(correct)** Because it requires perfect knowledge of both the system model and disturbance characteristics
III: **(wrong)** Because open-loop controllers are inherently unstable
IV: **(wrong)** Because the car's mass changes during operation
V: **(wrong)** I do not know

Solution 1:

Perfect disturbance rejection would require exact knowledge of both the system parameters (mass, friction coefficient) and the precise characteristics of the disturbance (road slope). In reality, both system parameters and disturbances have

notes

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

Question 4

What is the main practical issue with designing an open-loop controller by perfectly inverting the system model?

Potential answers:

I: **(wrong)** It makes the system too fast
II: **(correct)** It often results in a non-causal controller that requires knowledge of future inputs
III: **(wrong)** It requires solving differential equations in real-time
IV: **(wrong)** It makes the control signal too smooth
V: **(wrong)** I do not know

Solution 1:

Perfect model inversion typically leads to non-causal controllers that would need to anticipate future reference signals, which is physically impossible to implement in real-time systems. This is why practical open-loop designs often settle for steady-state accuracy rather than perfect tracking.

notes

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

Question 5

In the car speed control example, why might choosing a very small time constant in the controller be problematic?

Potential answers:

I: **(wrong)** It would make the controller too simple
II: **(correct)** It would require unrealistically large control forces from the engine
III: **(wrong)** It would make the car accelerate too slowly
IV: **(wrong)** It would prevent the car from reaching the desired speed
V: **(wrong)** I do not know

Solution 1:

A very small (fast response) would demand extremely large control forces because the controller would try to achieve the desired speed almost instantaneously. Real engines have limited power, and such demands could lead to actuator saturation, excessive fuel consumption, or mechanical stress.

notes

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

Recap of module "Introduction to Open-Loop Controller Design"

- open-loop control is structurally simple but not very robust

notes

- If you remember just one thing from this module, it's this: open-loop control is like riding a bike with your eyes closed.
- It might work if the path is perfectly straight and predictable, but the moment something unexpected happens, you're in trouble.
- The simplicity is appealing - no sensors needed, just pure calculation. But this simplicity comes at the cost of fragility.
- As engineers, we need to understand both the theoretical appeal and practical limitations of each approach.

Introduction to closed-loop controller design

notes

Contents map

developed content units	taxonomy levels
closed-loop control	u1, e1

prerequisite content units	taxonomy levels
LTI system	u1, e1

notes

Main ILO of sub-module "Introduction to closed-loop controller design"

Explain the fundamental differences between open-loop and closed-loop control systems in terms of error correction and disturbance rejection

notes

- you should now be able to distinguish control architectures

Roadmap

- what it is
- examples

notes

- Today we're starting a crucial new topic that addresses all the limitations we saw in open-loop control.
- First, we'll understand what exactly closed-loop control is at a conceptual level.
- Then we'll look at concrete examples to build our intuition before diving into the mathematics.
- By the end of this lecture, you should understand why feedback is so powerful despite its added complexity.

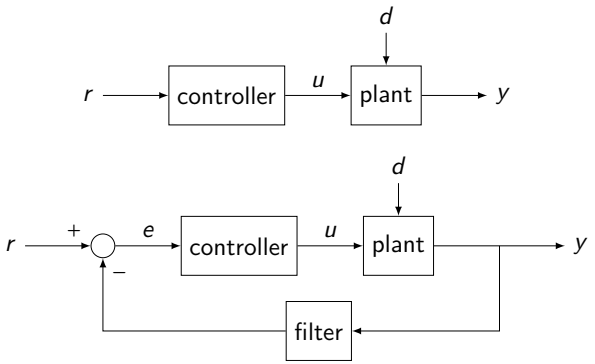
What do we learn now?

introduction to how to design a closed-loop controller

notes

- This is where control theory gets really exciting! We're moving from fragile open-loop systems to robust closed-loop designs.
- You'll learn the fundamental principles that enable everything from cruise control to spacecraft navigation.
- The key idea is simple: use feedback to automatically correct errors, but the implications are profound.
- Over the next few lectures, you'll gain tools to design systems that can maintain performance despite uncertainties and disturbances.

Fundamental control architectures: open-loop vs. closed-loop (or feedback)

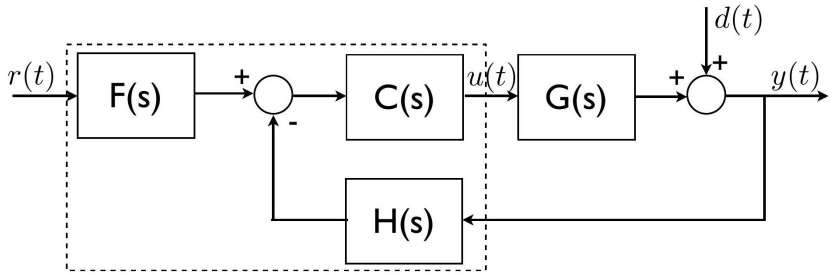


notes

- Let me explain why feedback is such a game-changer. In closed-loop control, we're constantly checking our actual output against our desired output - it's like riding that bike with your eyes open!
- The magic happens because the controller automatically adjusts its actions based on the tracking error. No need for perfect knowledge of the system or disturbances.
- Think about how you shower: you don't calculate exactly how much to turn the tap - you feel the water temperature and adjust continuously. That's feedback control!
- The closed path of signals creates a self-correcting system. Any deviation creates forces that try to eliminate that deviation.
- This is why feedback systems can handle uncertainties and disturbances that would completely derail open-loop systems.

Notation

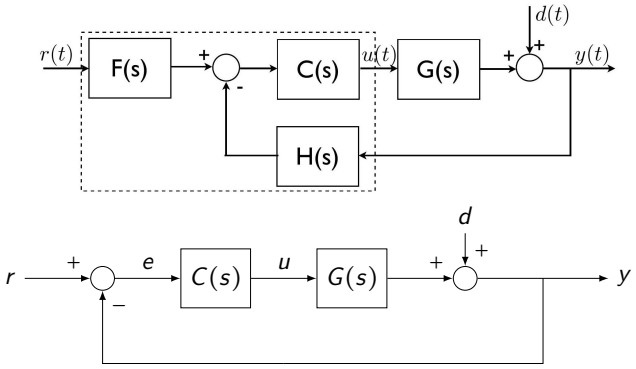
- $G(s)$: Transfer function of the system to be controlled
- $H(s)$: Transfer function of the sensor
- $C(s)$: Transfer function of the controller
- $F(s)$: Transfer function of the shaping filter



notes

- Before we dive deeper, let's get comfortable with the standard notation we'll be using throughout this course.
- Notice that disturbances can enter at multiple points - not just at the input. A wind gust affects a car differently than a change in road slope!
- The sensor block H(s) is crucial - real feedback depends on measuring the output, and no sensor is perfect.
- The shaping filter F(s) lets us modify how the reference signal is presented to the system - like smoothing out abrupt commands.
- Understanding this general structure will help you analyze any feedback system you encounter.

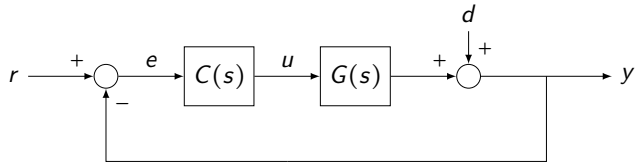
By moving blocks we can switch to standard notation



notes

- So heres whats going on: by cleverly rearranging the blocks in our diagram, we can bring it into a more standard formone that matches the canonical feedback loop format we typically analyze. This doesn't change the system's behavior, but it makes the math and reasoning that follow much more convenient. Its like tidying your workspace before diving into a complex tasksame tools, but a much smoother process.

But what would we ideally want?



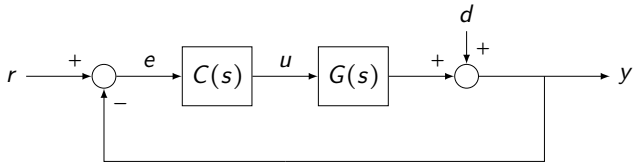
$$W_{ry}(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} \approx 1 \quad \Longrightarrow \quad y(t) \approx r(t)$$

$$W_{dy}(s) = \frac{1}{1 + C(s)G(s)} \approx 0 \quad \Longrightarrow \quad y(t) \approx \text{ not affected by } d(t)$$

notes

- Now, lets dream a bit. Ideally, we want our system output $y(t)$ to follow the reference $r(t)$ as closely as possible. Thats what the first equation tells uswhen the loop gain $C(s)G(s)$ is large, the transfer function from reference to output approaches 1. At the same time, wed love it if disturbances $d(t)$ had no effect on the output. And look at the second equation: when $C(s)G(s)$ is large, the disturbance gets suppressed. So we get great tracking and disturbance rejectionthis is the holy grail of feedback control!

And how is it done 90% of the time?



With

- $C(s) = K_p$, proportional controller
- $C(s) = K_p + \frac{K_i}{s}$, proportional-integral controller
- $C(s) = K_p + K_d s$, proportional-derivative controller
- $C(s) = K_p + \frac{K_i}{s} + K_d s$, proportional-integral-derivative controller

notes

- So how do we achieve that ideal behavior we just talked about? In the vast majority of practical applications, we do it with a PID controller or one of its simpler forms. The P, PI, PD, and full PID controllers are the workhorses of the industry. They're simple, powerful, and surprisingly effective in many situations. If you're wondering why they work so well, I really recommend this video: `https://www.youtube.com/watch?v=UR0hOmjaHp0&pp=ygU0cGlkIGNvbnRyb2xsZXI%3D`. It explains not just the math but also the intuition behind each term.

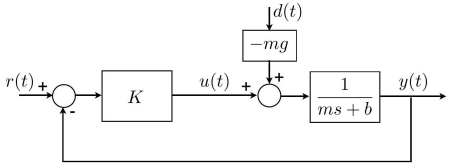
And how to build a PID?

- analog: `https://www.youtube.com/watch?v=Sw3NEA3GEnI`
- digital: in a few modules

notes

- You might be wonderingOK, I get what a PID is, but how do we actually build one? Well, you can implement it in hardware, using op-amps in an analog circuitlike in this excellent video: `https://www.youtube.com/watch?v=Sw3NEA3GEnI`. Or, as is more common today, you can implement it digitallyusing code, blocks in a control system library, or on a microcontroller. Digital implementations give you a lot of flexibility and are the standard in most modern applications.

Example: car speed control with a P controller



with $C(s) = K$ and $G(s) = \dfrac{1}{ms+b}$, implying

$$W_{ry}(s) = \frac{C(s)G(s)}{1+C(s)G(s)} = \frac{K}{ms+b+K} \approx 1 \quad \Longrightarrow \quad y(t) \approx r(t)$$

$$W_{dy}(s) = \frac{-mgG(s)}{1+C(s)G(s)} = \frac{-mg}{ms+b+K} \approx 0 \quad \Longrightarrow \quad y(t) \approx \text{ not affected by } d(t)$$

notes

- Lets go through the math here so you see where these expressions come from. First, for the reference-to-output path:

$$W_{ry}(s) = \frac{C(s)G(s)}{1+C(s)G(s)} = \frac{\frac{K}{ms+b}}{1+\frac{K}{ms+b}} = \frac{\frac{K}{ms+b}}{\frac{K+ms+b}{ms+b}}$$

which simplifies to $\dfrac{K}{ms+b+K}$. Similarly, for the disturbance path:

$$W_{dy}(s) = \frac{-mgG(s)}{1+C(s)G(s)} = \frac{\frac{-mg}{ms+b}}{1+\frac{K}{ms+b}} = \frac{\frac{-mg}{ms+b}}{\frac{ms+b+K}{ms+b}}$$

so you get $\dfrac{-mg}{ms+b+K}$. The takeaway is simple: by increasing K, we make $W_{ry}(s)$ closer to 1 and $W_{dy}(s)$ closer to 0better tracking, better disturbance rejection.

Summarizing

Explain the fundamental differences between open-loop and closed-loop control systems in terms of error correction and disturbance rejection

notes

- you should now be able to distinguish control architectures

Most important python code for this sub-module

notes

The python.control library

. . . as virtually in all the modules of this part of the course

notes

- the control library is basically essential

Self-assessment material

notes

Question 6

What is the primary conceptual advantage of closed-loop control over open-loop control?

Potential answers:

I: **(correct)** Ability to automatically correct errors using feedback
II: **(wrong)** Higher computational efficiency in implementation
III: **(wrong)** Elimination of all system disturbances
IV: **(wrong)** Reduced need for sensors in the system
V: **(wrong)** I do not know

Solution 1:

The primary advantage of closed-loop control is its ability to automatically correct errors using feedback. Unlike open-loop systems, closed-loop systems continuously compare the actual output with the desired reference and adjust accordingly, making them robust to uncertainties and disturbances. The other options are incorrect: closed-loop control often requires more computation (not less),

notes

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

Question 7

Why are PID controllers so widely used in practice despite their simplicity?

Potential answers:

I: **(wrong)** They can perfectly eliminate all system nonlinearities
II: **(correct)** They provide effective performance across many applications with relatively simple implementation
III: **(wrong)** They require no tuning parameters for optimal performance
IV: **(wrong)** They eliminate the need for system modeling entirely
V: **(wrong)** I do not know

Solution 1:

PID controllers are widely used because they offer effective performance across many applications with relatively simple implementation. The proportional term responds to present error, the integral term addresses accumulated past errors, and the derivative term anticipates future errors. While they don't eliminate nonlinearities (first option), do require tuning (third option), and still benefit from system modeling (fourth option), their balanced performance makes them practical for many real-world systems.

notes

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

Question 8

In the standard closed-loop control notation, what does the transfer function $W_{dy}(s) \approx 0$ imply about the system?

Potential answers:

I: **(wrong)** The system cannot track reference signals
II: **(wrong)** The controller has become unstable
III: **(correct)** The system effectively rejects disturbances
IV: **(wrong)** The sensor measurements are inaccurate
V: **(wrong)** I do not know

Solution 1:

When $W_{dy}(s) \approx 0$, it means the transfer function from disturbances to output is nearly zero, indicating effective disturbance rejection. This is a desirable property of well-designed feedback systems. The first option is incorrect because reference tracking is governed by $W_{ry}(s)$. The second option is unrelated to disturbance rejection, and the fourth option concerns sensor performance rather than the

notes

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

Question 9

What fundamental limitation prevents a real control system from achieving perfect tracking ($W_{ry}(s) = 1$) and perfect disturbance rejection ($W_{dy}(s) = 0$) simultaneously?

Potential answers:

I: **(wrong)** The need for digital implementation
II: **(wrong)** Sensor accuracy limitations
III: **(correct)** Fundamental trade-offs between performance, robustness, and stability
IV: **(wrong)** The cost of high-quality actuators
V: **(wrong)** I do not know

Solution 1:

The fundamental limitation arises from trade-offs between performance, robustness, and stability. While increasing controller gain can improve both tracking and disturbance rejection, practical systems face limits due to stability constraints, actuator saturation, and robustness requirements. The other options, while potentially important in specific cases, are not fundamental limitations in the same way as these inherent trade-offs.

notes

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

Question 10

In the car speed control example with a P controller, what happens to both $W_{ry}(s)$ and $W_{dy}(s)$ as the proportional gain K is increased?

Potential answers:

I: **(wrong)** Both transfer functions approach infinity
II: **(wrong)** $W_{ry}(s)$ approaches 0 while $W_{dy}(s)$ approaches 1
III: **(correct)** $W_{ry}(s)$ approaches 1 while $W_{dy}(s)$ approaches 0
IV: **(wrong)** Both transfer functions become oscillatory
V: **(wrong)** I do not know

Solution 1:

As K increases, $W_{ry}(s) = \dfrac{K}{ms + b + K}$ approaches 1 (better tracking), while $W_{dy}(s) = \dfrac{-mg}{ms + b + K}$ approaches 0 (better disturbance rejection). This demonstrates the dual benefits of high-gain feedback. The first option is incorrect as neither transfer function grows without bound. The second option reverses the

notes

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

Recap of module "Introduction to closed-loop controller design"

- feedback control is more promising, but requires designing more things compared to open loop

notes

- Before we wrap up, lets recap the key message: feedback control can dramatically improve system performancebetter tracking, disturbance rejection, and robustness. But it doesnt come for free. You need to carefully design your controller, and that takes insight, math, and often some trial and error. Still, its usually worth it.