Control systems: from GW interferometers to quad-copters

Bas Swinkels[1]

Andreas Freise[1,2]

[1]Nikhef GW group, [2]VU

Topical Lectures
14/04/2021

Course schedule

- Now: about 1.5-2 h lecture about control systems
- Rest of the afternoon and tomorrow afternoon: implement your own control system using Jupyter notebook

Goal/Disclaimer

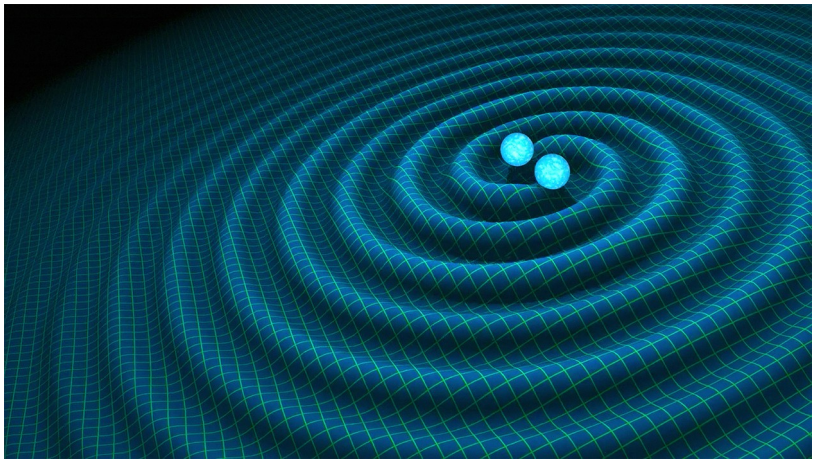
- Goal of this lecture and lab work is to get a quick overview of control systems and why we need them for detecting gravitational waves
- We don’t have time to explain control theory in detail (that is a complete field of study), idea is just to get some practical experience
- Control of a GW interferometers involves many interesting aspects, but they are not good toy models. We will try to have some fun with simulating a quad-copter instead

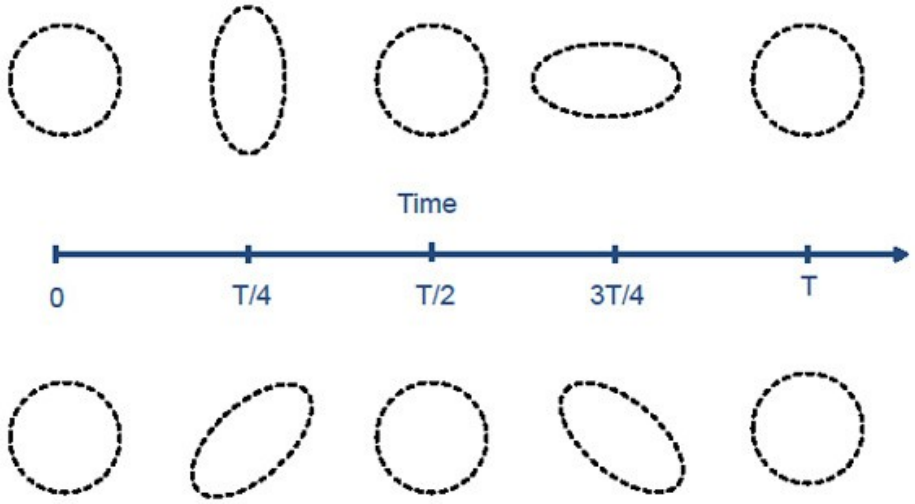
Lecture overview

- Introduction to detecting gravitational waves, and why this requires control systems
- Some background theory: Laplace transformation, transfer functions
- Basics of control systems
- Practical example: controlling a quad-copter

Gravitational wave detection, why do we need control?

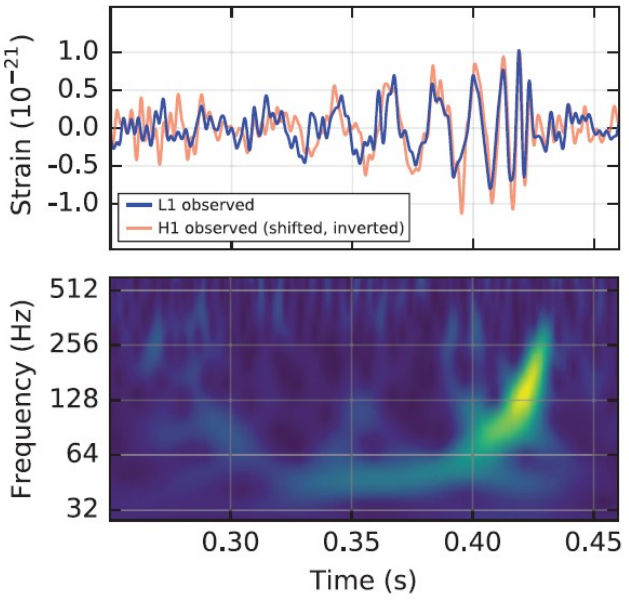
What are Gravitational Waves?

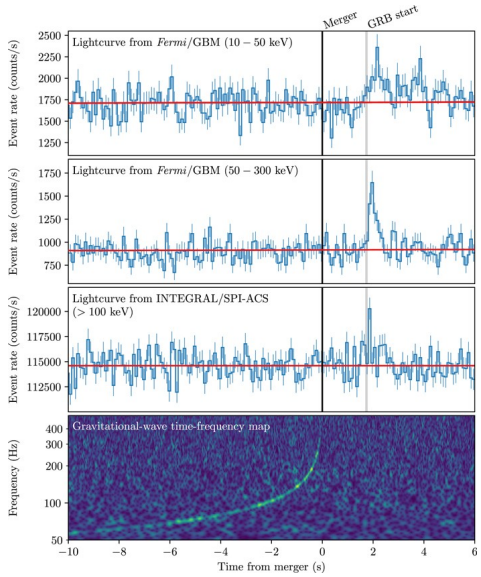


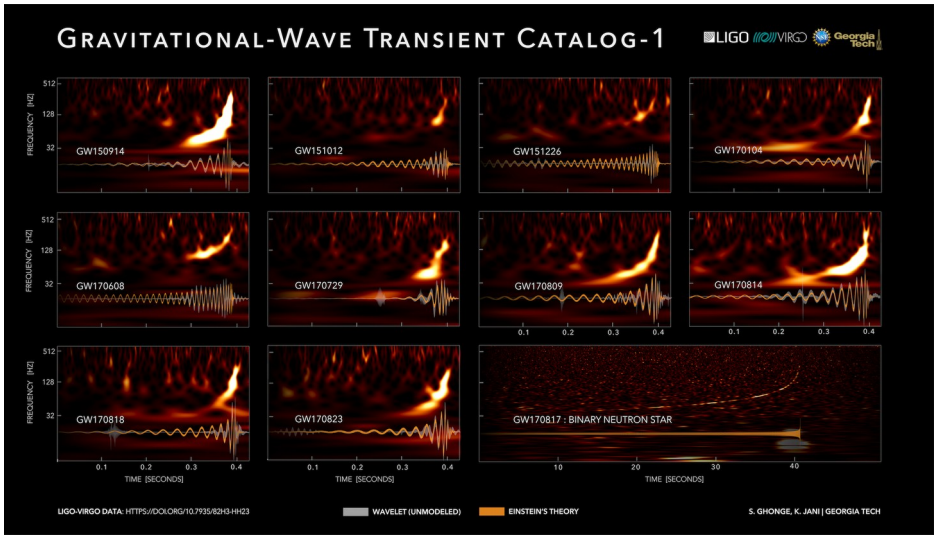


- 'Ripples in the fabric of space-time' that propagate with the speed of light
- Natural wave solution of General Relativity
- A GW stretches and squeezes space-time in transverse direction, 2 possible polarizations
- Gravitational wave strain: $h = \frac{\delta L}{L}$
- Generated when masses are accelerated non-symmetrically (change of quadrupole moment)
- Extremely weak, h = 10^{-21} for typical astronomical sources

Many detections

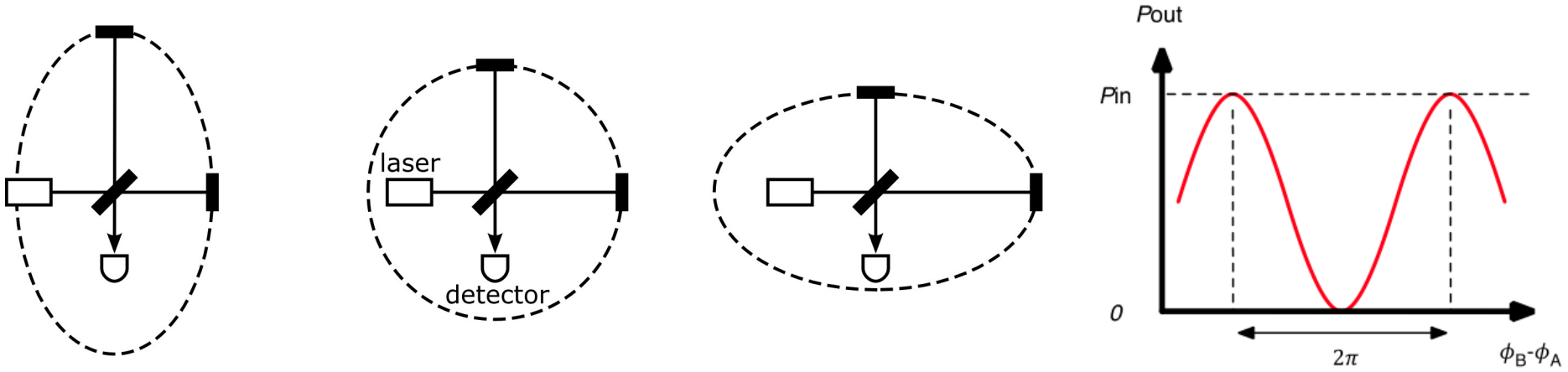






- First indirect evidence of GW by radio telescopes in 1974: slowly inspiraling Hulse-Taylor binary 
- First detection of collision by 2 black holes (GW150914) by two LIGO interferometers in the US 
- Since then triple detections of 2xLIGO and Virgo interferometer in Italy (GW170814)
- First detection of binary neutron star collision (GW170817), with detection of GRB and optical afterglow by telescopes: start of multi-messenger astronomy
- About 50 detections in the years after that!
- Very exciting physics and astronomy, but won't discuss that today

Interferometric GW detection

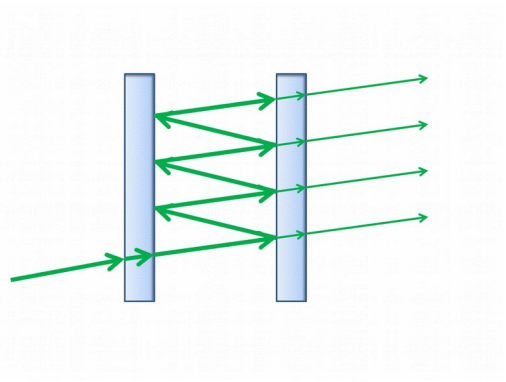


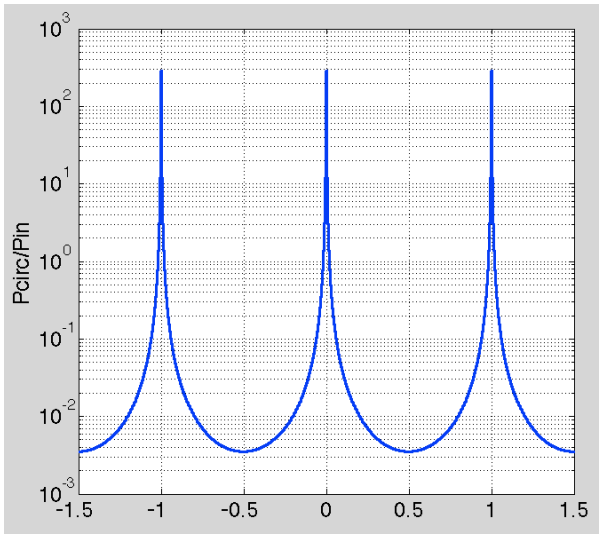
- Michelson interferometer is a natural fit for measuring gravitational waves: GW cause a differential change of arm length
- Output of interferometer shows periodic ‘fringes’ when difference in arm lengths changes by half a wavelength (Nd;YAG laser at 1.064 um)
- A simple Michelson is not sensitive enough to detect GW, need several extra tricks ...

Increase arm length

- Classical Michelson interferometer: beam-splitter and two end mirrors
- $dL = L * h$, so increasing arm length for a given strain will increase the displacement. Virgo interferometer has 3 km arms, so 6 km of ultra-high vacuum pipe
- Photodiode signal limited by shot noise. For SNR reasons, you want to be close to 'dark fringe', need to stabilize difference of arm length to fraction of wavelength

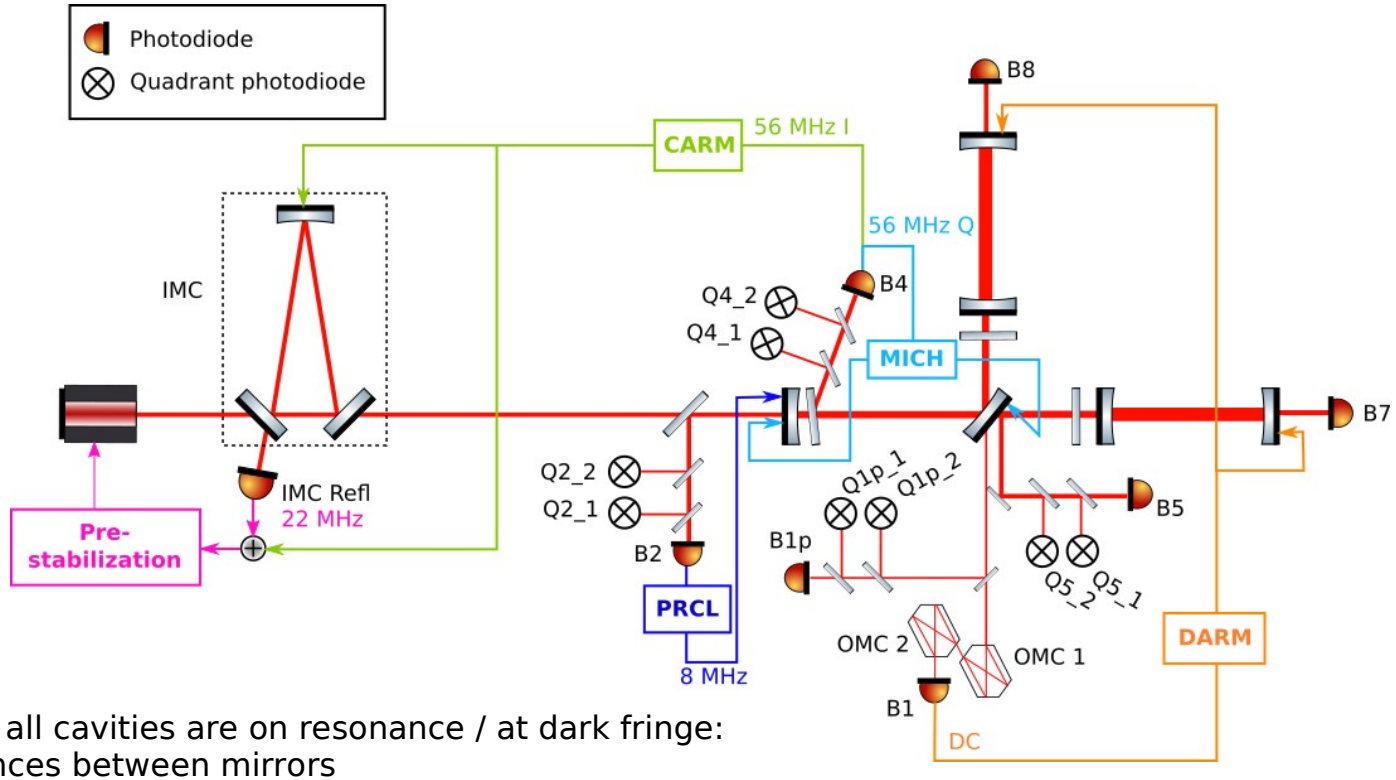
Add Fabry-Perot cavities





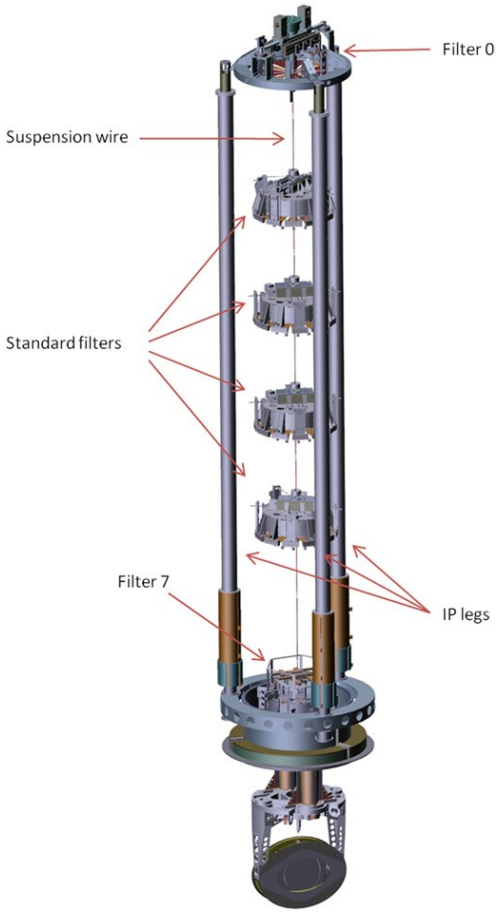
- Ideally, you want arm lengths of a few hundred km, but not enough money/space
- Use optical resonators (Fabry Perot cavities) in the arms to make the light bounce around about 450 times, so effective arm-length ~400 km
- Resonators only work when you are on resonance, when cavity length is integer number of half-wavelengths. Need to control distance between mirrors with picometer accuracy

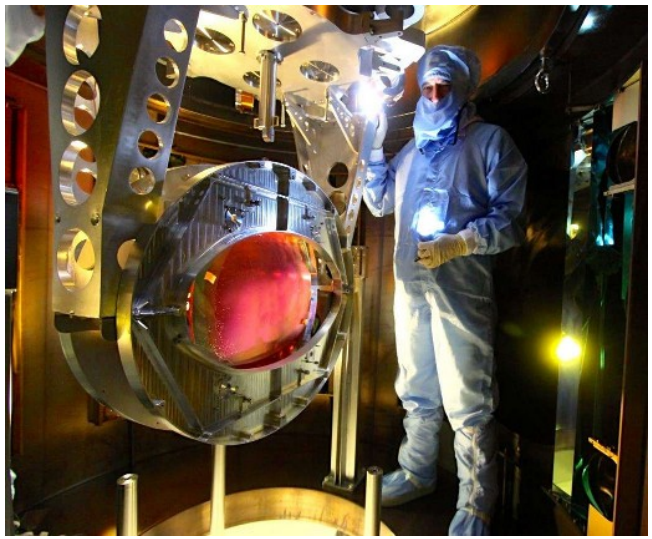
Longitudinal control

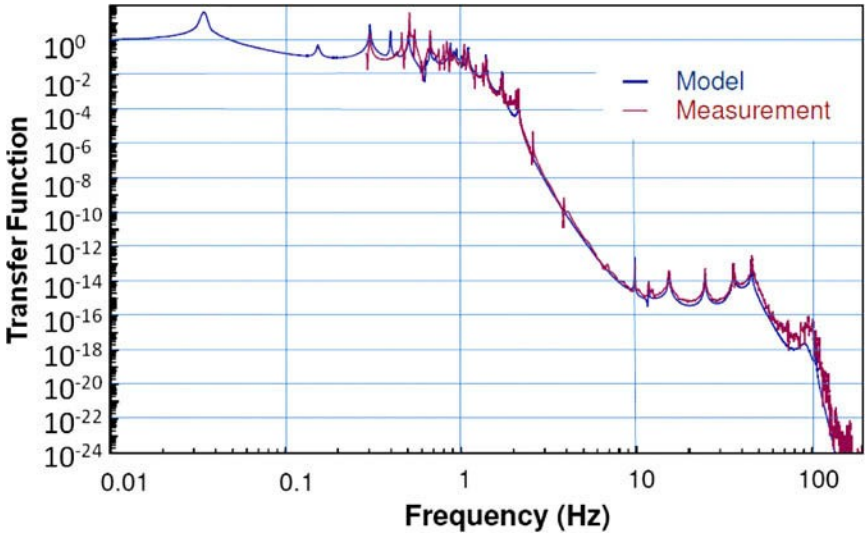


- Interferometer is only sensitive when all cavities are on resonance / at dark fringe: use real-time system to control distances between mirrors
- We use several optical signals that indicate how far we are away from resonance of cavities (modulation/demodulation technique: Pound-Drever-Hall)
- Push mirrors back to resonance using voice-coil actuators. Effect of GW signal is suppressed by control loop, need to correct for this in calibration.
- Since we use light as our ruler, we also need to stabilize the laser frequency to about 1 mHz
- Also need to keep mirrors aligned with microrad accuracy
- Control system is key part of measurement technique!

Seismic isolation



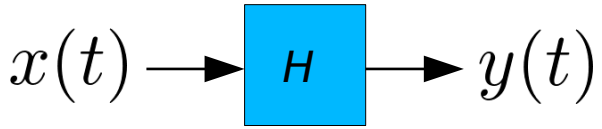




- To measure a strain of 1e-21, we need our mirrors to be stable by 1e-18 meters over a few km.
- Ground is vibrating by about fraction of a micrometer, need 10 orders of magnitude attenuation above 10 Hz!
- Use combination of active pre-isolation stage (inertial free platform balancing on inverted pendulum, using accelerometers and position sensors) and passive multi-stage pendulums and blade springs

Background: Laplace transform transfer functions

Linear, time-invariant systems



- To control a system, we need to understand how its outputs depend on its inputs
- Control theory is mostly limited to studying linear, time-invariant (LTI) systems
- Linear system: if $x_1(t) \rightarrow y_1(t)$ and $x_2(t) \rightarrow y_2(t)$
  then $\alpha x_1(t) + \beta x_2(t) \rightarrow \alpha y_1(t) + \beta y_2(t)$
- Real systems are non-linear, but can always do linear approximation around working point
- Time invariant: if $x(t) \rightarrow y(t)$ then $x(t-\tau) \rightarrow y(t-\tau)$
- Causality: require that input at $t=t_0$ does not have an effect on output for $t < t_0$
- Important observation: if input of LTI system is sine at frequency *f*, output is also a sine at *f*, only changed in phase and amplitude. Useful to think about systems in frequency domain

Laplace transform

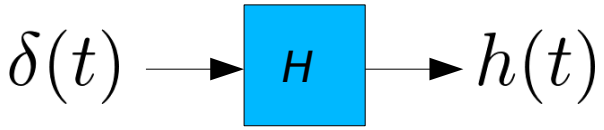
- Definition: $X(s) = \mathcal{L}\{x(t)\} = \int_0^\infty \exp(-st)x(t)\,dt$

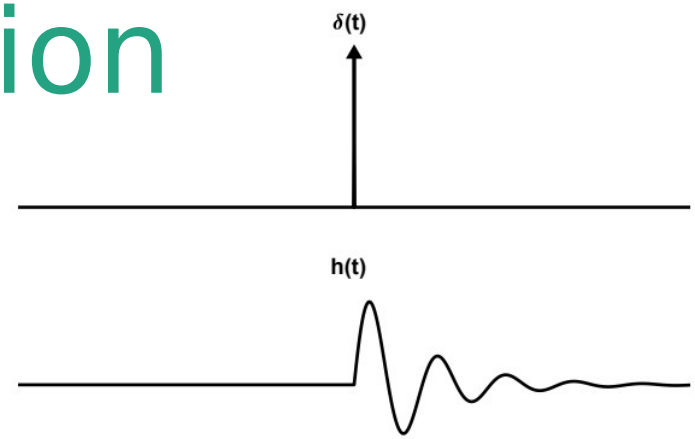
  with $s = \sigma + i\omega$ a complex variable. Somewhat related to Fourier transform $X(\omega)$
- Rather than calculate integral by hand, you normally use a table of known transformations of some basic functions to go from '*time domain*' to '*s domain*' and back
- Most important one is transform of derivative: $\mathcal{L}\left\{\frac{d}{dt}x(t)\right\} = sX(s) - x(0)$

  intuitively similar to $\frac{d}{dt}e^{i\omega t} = i\omega e^{i\omega t}$
- More in general: $\mathcal{L}\left\{\frac{d^n}{dt^n}x(t)\right\} = s^n X(s) + \cdots$

  where extra terms depend on initial value of x and its derivatives. We are mostly interested in steady-state behavior and not in initial transients, so extra terms are usually ignored

Impulse response & transfer function





- Response of a delta-function to input of a system gives the **impulse response** *h*. Note that for a causal system, $h(t) = 0$ for $t < 0$
- For a LTI systems, output *y(t)* of an arbitrary input signal *x(t)* can be calculated by convolving the input signal with the impulse response:

$$y(t) = (h * x)(t) = \int_{\infty}^{\infty} h(\tau)x(t-\tau)d\tau$$

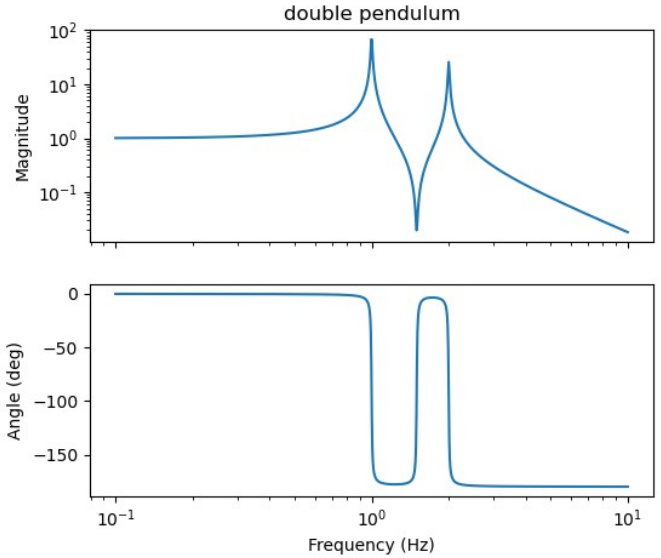
- Laplace transform of a convolution in the *time domain* is just a product in the *s domain*:

$$Y(s) = H(s) \cdot X(s)$$

- Laplace transform of the impulse response is called the **transfer function** of the system:

$$H(s) = \frac{Y(s)}{X(s)}$$

Bode plot



- Transfer function *H(s)* can be seen as frequency response of system by substituting $s = i\omega$
- Several ways of plotting this, most common one is the **Bode plot**:
  - subplot with log of |*H*| vs log of frequency
  - subplot with phase(H) vs log of frequency
- Asymptotes when a system is proportional to $\boldsymbol{s^n}$ in a certain frequency range:
  - a loglog plot of the magnitude will show a straight line of slope *n*,
    e.g. a second order low-pass filter has ‘slope -2’ at high frequencies (goes down with $1/f^2$)
  - (for simple systems) the angle shows steps at multiples of *n* * -90°

Transfer function from equations of motion

- Response of physical systems can usually be described by differential equations. For an LTI system, this has the form

$$\left(a_n \frac{d^n}{dt^n} + \cdots + a_2 \frac{d^2}{dt^2} + a_1 \frac{d}{dt} + a_0\right) y(t) = \left(b_n \frac{d^n}{dt^n} + \cdots + b_2 \frac{d^2}{dt^2} + b_1 \frac{d}{dt} + b_0\right) x(t)$$

- Doing the Laplace transform:

$$\left(a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0\right) Y(s) = \left(b_n s^n + \cdots + b_2 s^2 + b_1 s + b_0\right) X(s)$$

- Transfer function is calculated as output over input, always yields a rational function:

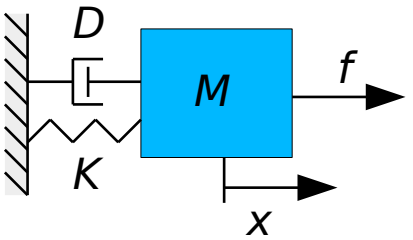
$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_n s^n + \cdots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0}$$

- See **`tf`** function in Matlab, or **`TransferFunction`** object in scipy.signal

Example: damped mass on a spring

- Equation of motion for mass *M*, viscous damping *D*, spring constant *K*:

$$M\frac{d^2}{dt^2}x(t) + D\frac{d}{dt}x(t) + Kx(t) = f(t)$$



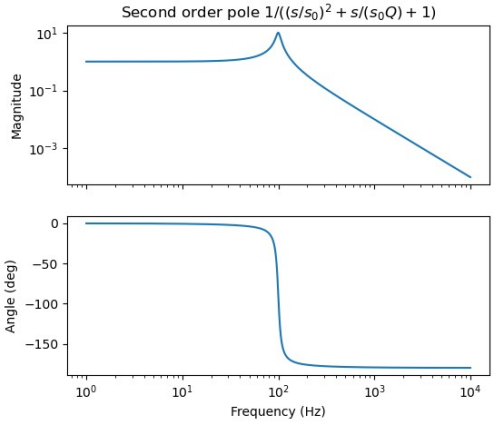
- Apply Laplace transform (ignoring initial conditions):

$$Ms^2X(s) + DsX(s) + KX(s) = F(s)$$

- Calculate transfer function from force to position:

$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{Ms^2 + Ds + K}$$

- Calculate frequency response by substituting $s = i\omega$



Poles and zeros

- Definition of transfer function:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_n s^n + \cdots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0}$$

- Both polynomials can be factored into 1st order and 2nd terms corresponding to real or complex zeros (*s* for which numerator = 0) or poles (s for which denominator=0)

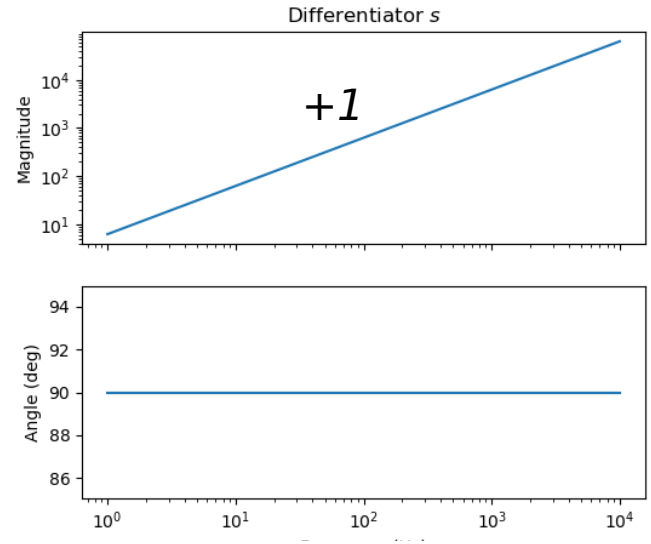
$$H(s) = K \frac{(s - z_1)(s - z_2)(s^2 + s s_{z3}/Q_{z3} + s_{z3}^2)\cdots}{(s - p_1)(s - p_2)(s - (s^2 + s s_{p3}/Q_{p3} + s_{p3}^2)\cdots}$$

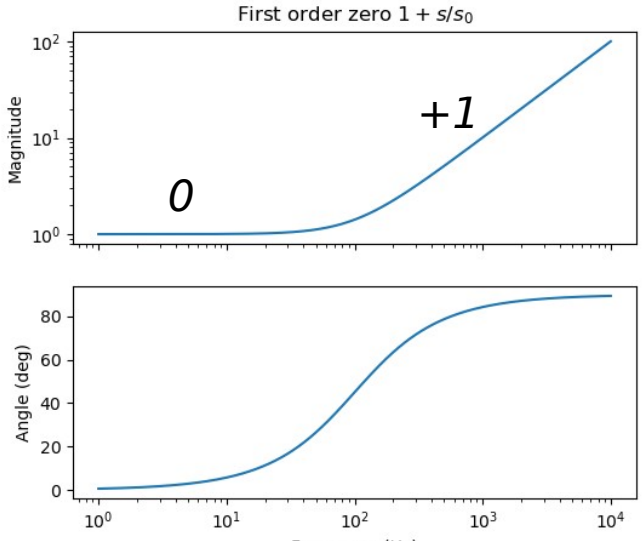
- The 2nd order terms can be further split in conjugate complex pair of poles/zeros

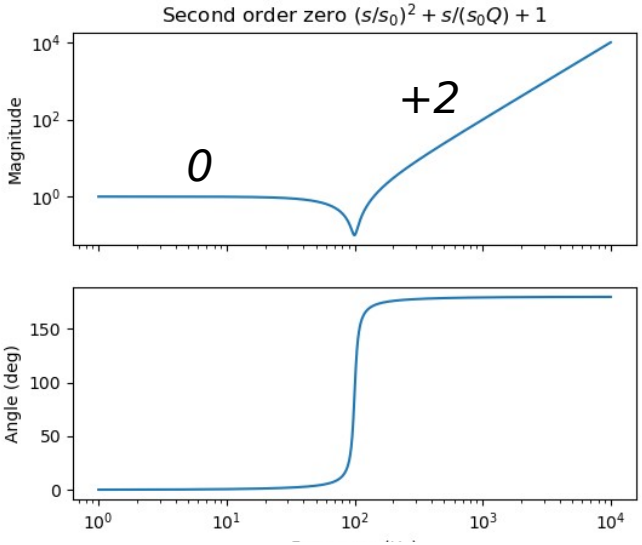
$$H(s) = K \frac{(s - z_1)(s - z_2)(s - (\sigma_{z3} + i\omega_{z3}))(s - (\sigma_{z3} - i\omega_{z3}))\cdots}{(s - p_1)(s - p_2)(s - (\sigma_{p3} + i\omega_{p3}))(s - (\sigma_{p3} - i\omega_{p3})\cdots}$$

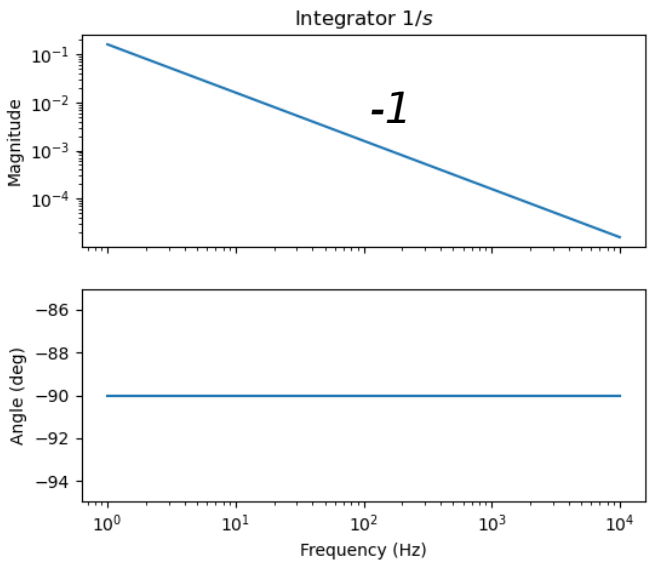
- Location of poles and zero in complex plane uniquely define system response. All poles must be in the left-hand plane for the system to be stable
- See **zpk** function in Matlab or **ZerosPolesGain** object in scipy.signal

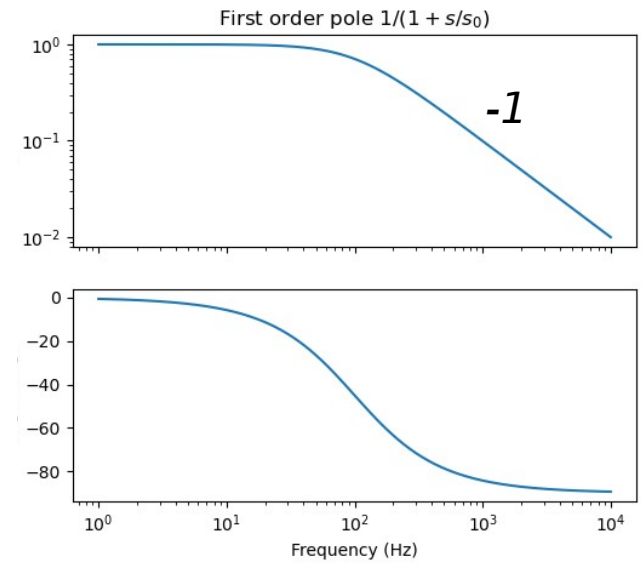
Bode plots of basic blocks

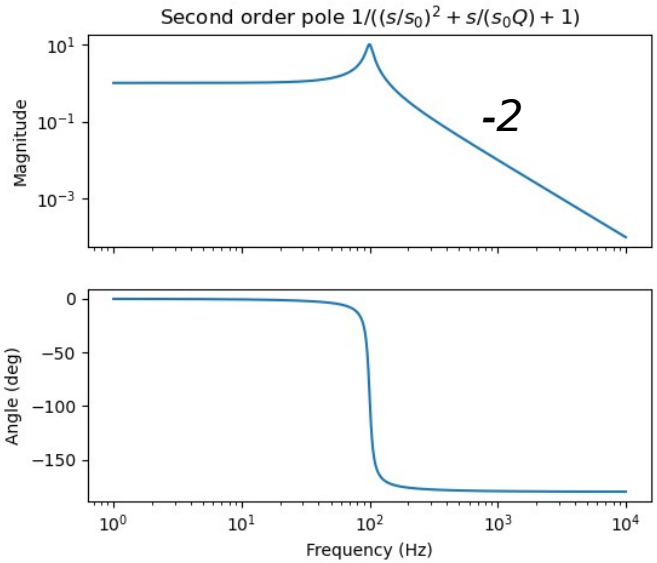










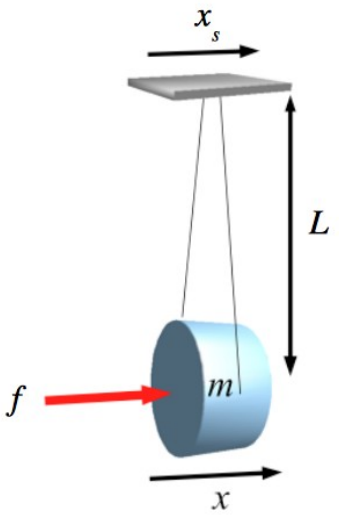


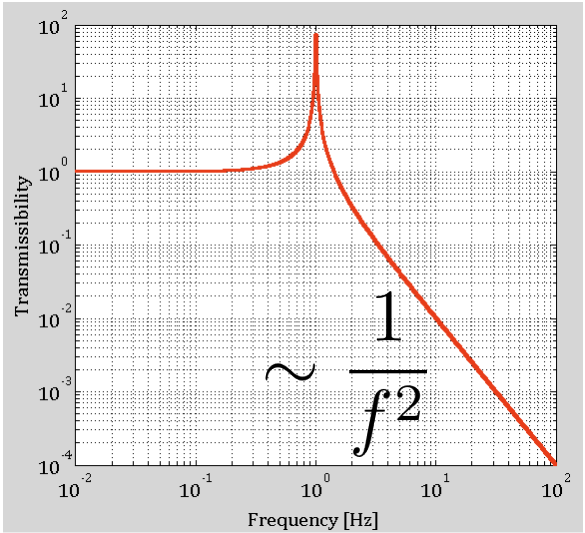
- Transfer functions with arbitrary shape can be obtained by multiplying basic blocks

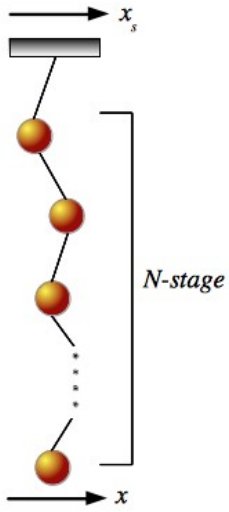
GW example: pendula

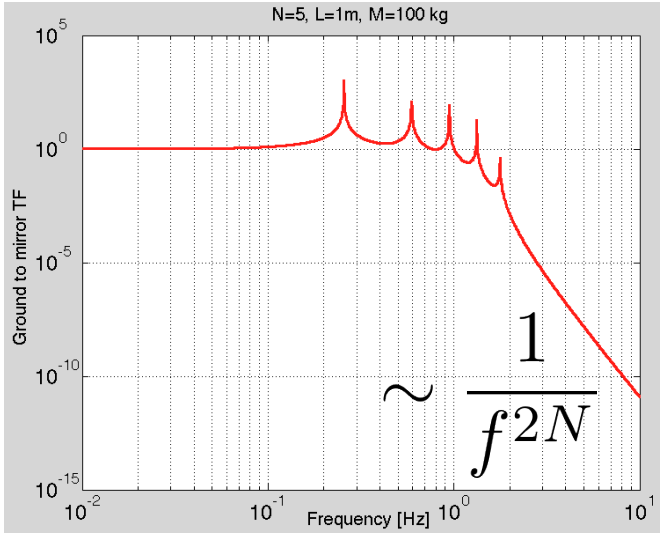
- To isolate our mirrors from seismic noise, we suspend them as a pendulum. Has a TF like a 2nd order pole with very high Q. This attenuates noise above resonance with $1/f^2$
- This is not enough, we need attenuation of 10 orders of magnitude above 10 Hz!
- Solution: more pendula in series. TF shows resonances for many modes

$$f = \frac{1}{2\pi}\sqrt{\frac{g}{L}}$$





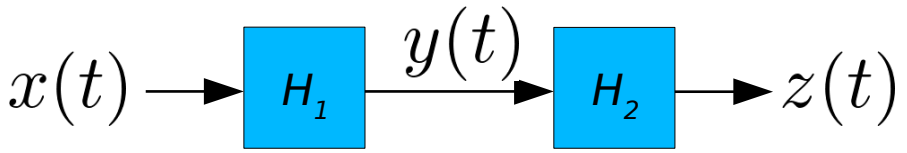




Combining systems

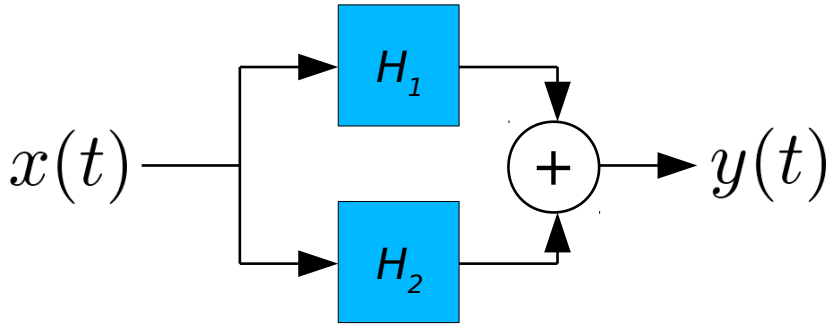
- In series:

$$z(s) = H_2(s)y(s)$$
$$= H_2(s)H_1(s)x(s)$$



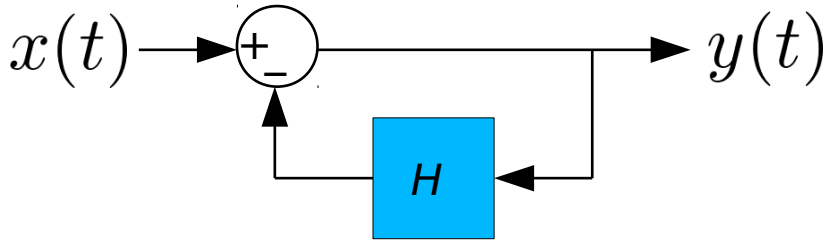
- Parallel:

$$y(s) = H_1(s)x(s) + H_2(s)x(s)$$
$$= (H_1(s) + H_1(s))x(s)$$



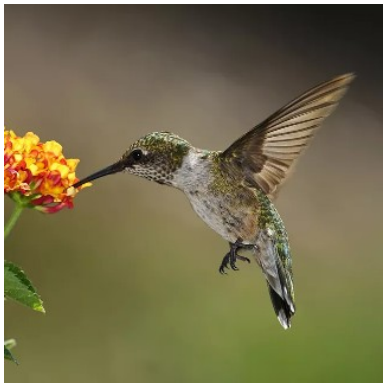
- Feedback:

$$y(s) = x(s) - H(s)y(s)$$
$$y(s) = \frac{1}{1 + H(s)}x(s)$$



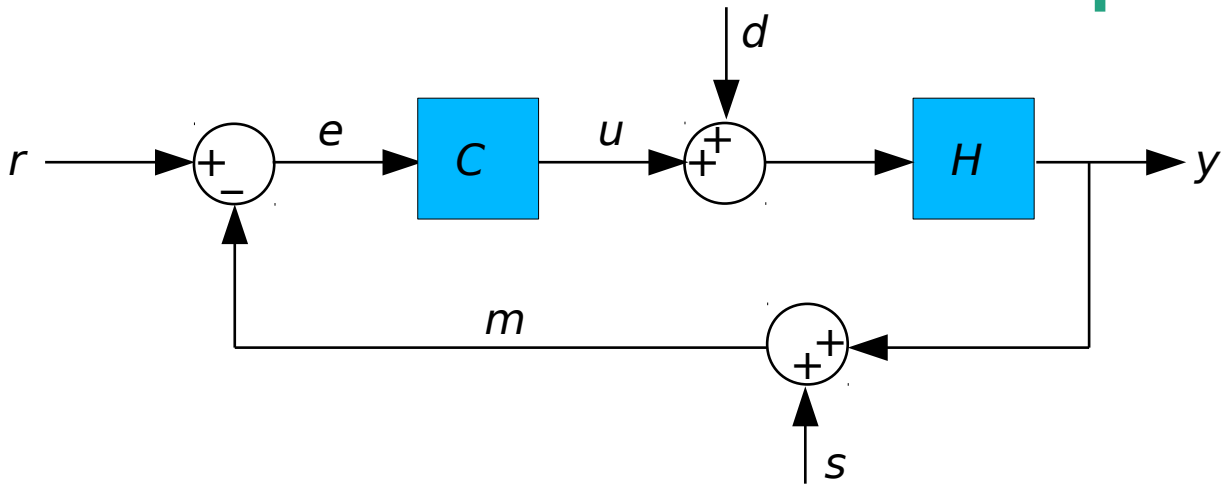
Control systems

Feedback/control is everywhere

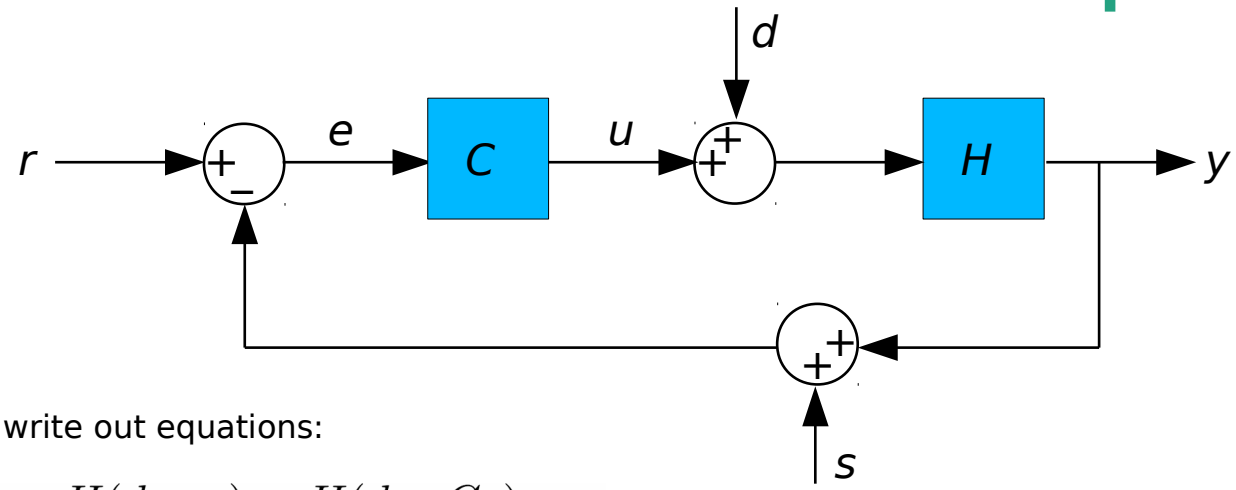

- If you have any system of which you can measure the state (e.g. with a sensor), and can influence its state (e.g. with an actuator), you can implement a controller to make the output do exactly what you want (within some limits)
- Examples of control are everywhere:
  - Electronics: OpAmp with feedback, phased lock loop, tracking grooves of a CD
  - Biological systems: control sugar level in blood with insulin, walking, flying, balancing
  - Transport: cruise control in car, autopilot, launching/landing spacecraft
  - At home: thermostats in your house, oven
  - In industry: process control, wafer stage movement at ASML, robotics
  - In science: e.g. null measurements
  - In finance/business
- Many systems are easy to understand, but getting the best performance out of a system with fast dynamics, proving stability, robustness in various conditions .... is a whole discipline (control engineering) with some advanced mathematics

Basic control loop



- Main signals and systems:
  - system you want to control ***H***, the 'plant' with external disturbance ***d***
  - output or 'process variable' ***y***, the thing we want to control
  - set-point or reference value ***r***: what we want **y** to be
  - measurement ***m*** of output, corrupted by sensing noise ***s***
  - error ***e,*** difference between set-point and measurement
  - controller ***C***, system that calculates actuator command ***u***
    based on error **e**

Basic control loop



- Straightforward to write out equations:

$$
\begin{aligned}
y &= H(d+u) = H(d+Ce) \\
&= H(d+C(r-(y+s)) \\
(1+HC)y &= Hd + HCr - HCs \\
y &= \frac{H}{1+HC}d + \frac{HC}{1+HC}(r-s)
\end{aligned}
$$

- Some definitions
  - Open loop transfer function: product of all systems going around loop HC
  - Closed loop transfer function: $\frac{1}{1+HC}$
  - Complementary closed-loop transfer function: $\frac{HC}{1+HC}$

Open vs closed loop

- For a control system to be stable, the open-loop transfer function should roughly have the shape of integrator: infinite gain at low frequencies, zero gain at high frequencies
- Unity gain frequency (UGF): frequency for which |H(s)C(s)| = 1. Control is effective below this frequency, does almost nothing (apart from making noise) above.
- Uncontrolled system: output only moves due to external disturbance ***d***:
  $y = Hd$
- Closed loop system (ignoring ***r***, ***s***) will show output
  y = H/ (1 + HC)d
  so output is suppressed by the closed loop transfer function 1/(1+HC): disturbance perfectly suppressed at low frequencies, almost no effect above UGF
- When moving the set-point ***r*** (ignoring s, d), the output is
  y = HC / (1+HC) r
  so this goes with the complementary closed-loop transfer function. Output perfectly follows reference at low frequencies, while it does nothing at high frequencies
- Influence of sensing noise ***s*** is similar to changes in set-point: at low frequencies the output follows the sensing noise. You cannot control better than your sensor!

Loop stability

- A closed loop system can become unstable if the controller does not have the right transfer function (or when it has the wrong sign!)
- Not enough time to explain all details, but a system is only stable if slope of open-loop transfer function has a ‘slope -1’ around the UGF, similar to an integrator. Slope can be steeper at lower frequencies to get more loop gain (and thus more suppression), and steeper at higher frequencies to avoid introducing too much sensor noise above UGF.
- You want to the system to remain stable, even if your system is not perfectly calibrated, or if if its response slowly changes over time: **robustness**. This can be achieved by taking enough margin when designing your loop: gain stability, phase stability

System identification

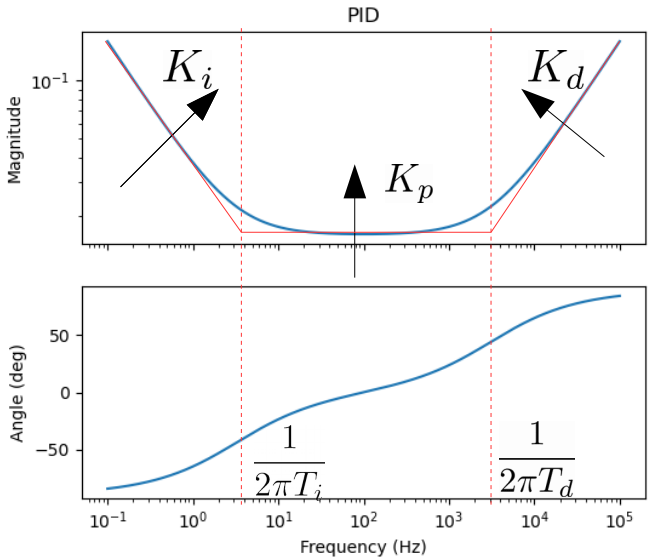
- To design a good controller C, you need to know the response of your plant H. Can be based on physics of the system, but that is always a simplification. Ideally do measurements on the actual system that includes all gory details (e.g. internal mechanical resonances, limits of electronics, spurious couplings ...). Known as **system identification**
- Easiest is open loop measurements: add known signal (impulse/step response, white noise, ...) to input, then measure output
- Not always possible if the system doesn't work without control (non-linear, unstable, ...)/ Leads to chicken-egg problem: need closed loop to measure H, which needs controller C, which needs measurement of H in closed loop ...
- In practice: use modeling, test stands with a subset/simplified version of hardware or just trial and error to close loop with basic controller, then iterate with doing measurement and improving controller until you are get good performance

PID controller

- General purpose controller, used almost everywhere, e.g. for temperature stabilization of industrial process
- 3 main parameters: **P**roportional, **I**ntegral, **D**ifferential:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{d}{dt} e(t)$$

- Not always all 3 terms needed, e.g. setting K_d=0 gives a PI controller
- Alternative definition: $u(t) = K_p \left(e(t) + \frac{1}{T_d} \int_0^t e(t')dt' + T_i \frac{d}{dt} e(t) \right)$



PID controller

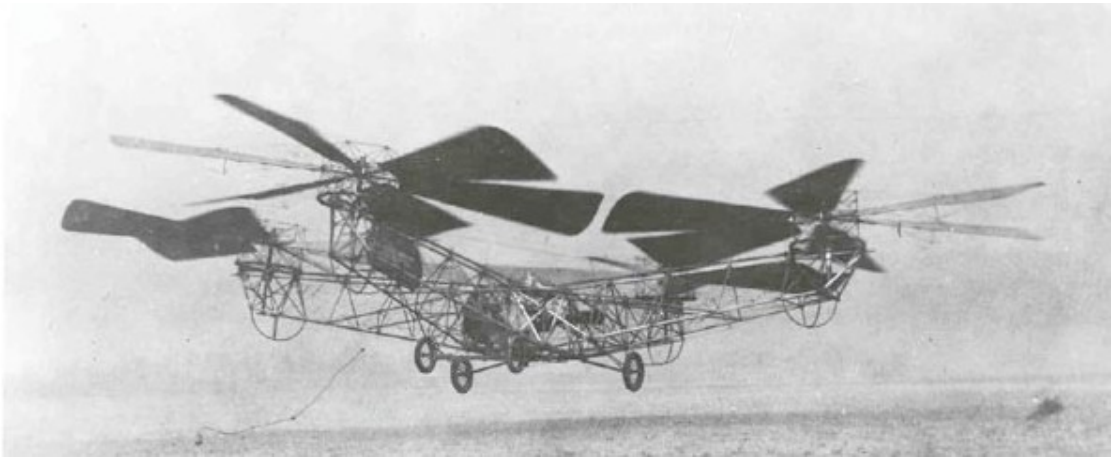
- In practice, you need to think about some details:
  - avoid that integrator accumulates large value when actuator saturates, known as anti wind-up protection
  - proper way to handle live changing of PID parameters
  - ways of reacting to changes in set-point vs changes in disturbance
  - differentiating term can cause issues with high frequency sensor noise: add extra low-pass filter above UGF
  - see e.g. http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
- Many ways to tune PID parameters, bit of an art form:
  - typically increase K_p until the system oscillates, then reduce a bit or compensate with others, see e.g. Ziegler–Nichols method
  - look at response of step on set-point. If you have a non-zero steady state error, you need more K_i. K_d can help reducing overshoot.
  - just use trial and error, likely what you will do tomorrow

Digital control

- In practice, control systems are usually implemented using a digital system, allows for cheap and flexible systems
- PID controller: use discrete integrator (sum) and differentiator (finite difference)
- For more complex controllers: use Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) implementations. In practice use series of 2^{nd} order IIR filters (second order stages, or bi-quads).
- Inputs and outputs don't measure continuous values, but sampled/reconstructed using Digital-to-Analog (ADC) and Analog-to-Digital Converters (DAC) with finite number of bits -> extra source of noise
- Outputs are calculated once per time step, which leads to delays. This will influence the stability and puts a limit on the achievable UGF
- Mathematics for discrete systems is different: from S-domain (Laplace) to Z-domain.
- In practice, many of these details can be neglected, as long as you have enough bits, and sample frequency is fast enough

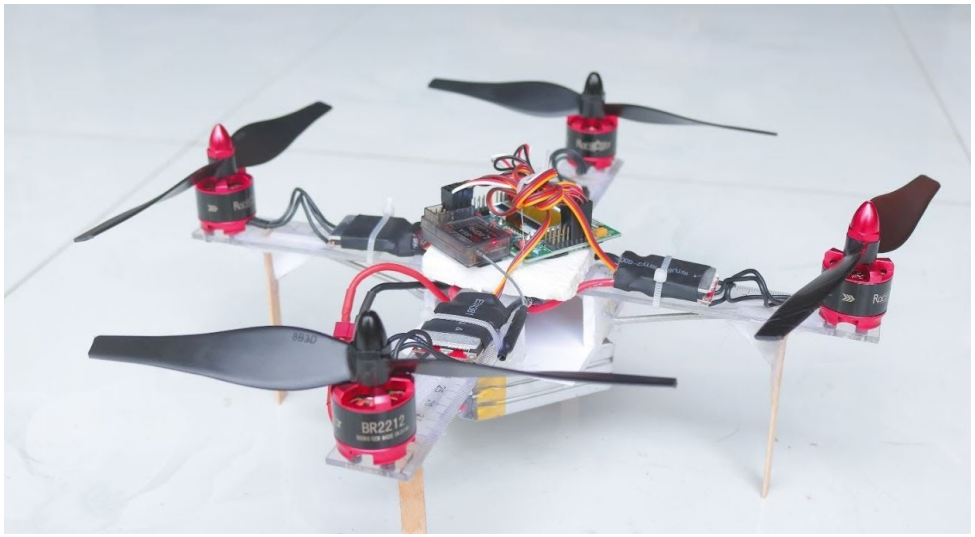
Quadcopter control

Quad-copter



- Why study quad-copters? They are cool, have relatively easy physics but needs some interesting control concepts to make them work
- Old idea, actually predates the conventional helicopter
- Early versions were not a success: "*it suffered from complexity, control difficulties, and high pilot workload, and was reportedly only capable of forward flight in a favorable wind*" (see 'Bothezat helicopter' on wikipedia)
- Modern versions got popular about 15 years ago, due to availability of fast micro-controllers, cheap sensors (GPS, IMU) and control loops that help the pilot
- Since then topic for a lot of control theory studies, see e.g. work by R. D'Andrea at ETHZ:
  https://www.youtube.com/watch?v=w2itwFJCgFQ
  https://www.youtube.com/watch?v=XxFZ-VStApo

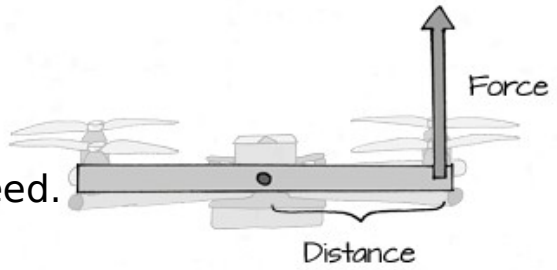
Main components (DIY)

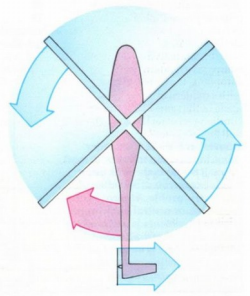


- X-shaped frame
- LiPo battery for power
- a bunch of sensors: GPS, accelerometers, gyroscopes, magnetometers
- 4 actuators: brush-less motors + digital speed controllers + propellers (2 CW, 2 CCW)
- real-time microcontroller (faster than an Arduino, but not a lot)
- optional: remote control, camera
- Can be built for around 100-200 €, maybe next year topical lessons?

Quad-copter physics

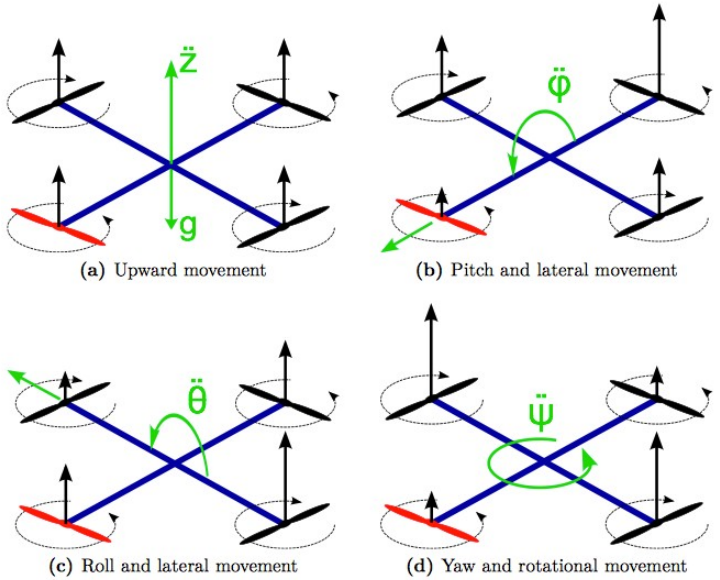
- Basic physics:
  - fixed body mass, moment of inertia (roughly similar for pitch/roll, different for yaw)
  - gravity acts on center-of-mass (COM)
  - each propellers generates a force proportional to square of its speed. Force is applied at offset from the COM, so this also induces a torque in pitch/roll via lever arm
  - each propeller also gives a torque around its axis gives counter to rotation directory, this is why a helicopter has a tail rotor. For quad-copters this is solved by having 2 CW and 2 CCW propellers
  - Atmospheric drag counteracts all movement/rotation
  - Disturbances from wind/turbulence
- Straightforward to get the equations of motion
- Inherently unstable, hard to control by hand by commanding individual motors. Use a control system to stabilize and let it react predictably to steering commands





Quad-copter 3D control

- By using combinations of motors, you can control the quad-copter in 4 degrees-of-freedom (DOF)
  - (a) giving same command to all 4 motors, all torques are canceled, creates purely vertical force
  - (b,c) increasing/decreasing power on opposite corners, the unbalanced forces create torque in pitch/roll
  - increasing power on one opposing pair, lowering on other pair gives no net force, but axial torque no longer cancels: yaw
- requires 4x4 **driving matrix** from DOF to motor



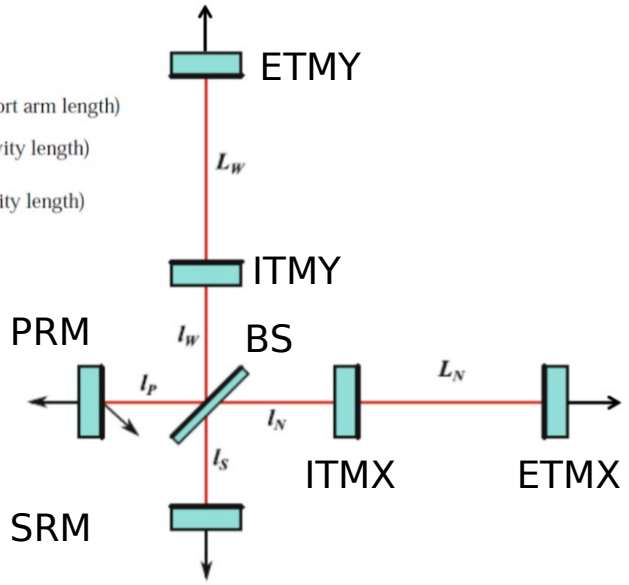
(a) Upward movement (b) Pitch and lateral movement (c) Roll and lateral movement (d) Yaw and rotational movement

http://uav-society.blogspot.com/2014/06/quadcopter-mechanics.html

vertical	pitch	roll	yaw	
1	1	0	-1	motor1
1	0	1	1	motor2
1	-1	0	-1	motor3
1	0	-1	1	motor3

GW: driving matrix of interferometer

$$\text{DARM} = L_N - L_W \text{ (differential arm length)}$$
$$\text{CARM} = \frac{L_N + L_W}{2} \text{ (average arm length)}$$
$$\text{MICH} = l_N - l_W \text{ (differential Michelson short arm length)}$$
$$\text{PRCL} = l_P + \frac{l_N + l_W}{2} \text{ (power recycling cavity length)}$$
$$\text{SRCL} = l_S + \frac{l_N + l_W}{2} \text{ (signal recycling cavity length)}$$



PRCL	SRCL	MICH	DARM	CARM	
1		-1			PRM
	1	1/sqrt(x)			BS
		-1			SRM
					ITMX
					ITMY
			1	1	ETMX
			-1	1	ETMY

- Main optics of a GW interferometer consists of 7 mirrors, which can be pushed individually
- With interferometry we only measure optical path lenghts between mirrors, never absolute positions.
- We have optical photodiodes that contain signals for 5 main DOF (one contains GW!), corresponding to combinations of distances
- 2 DOF uncontrolled, correspond to global translation
- For perfect feedback to the each DOF, we need to send different combinations of forces to the mirrors via a **driving matrix.** Not a unique choice

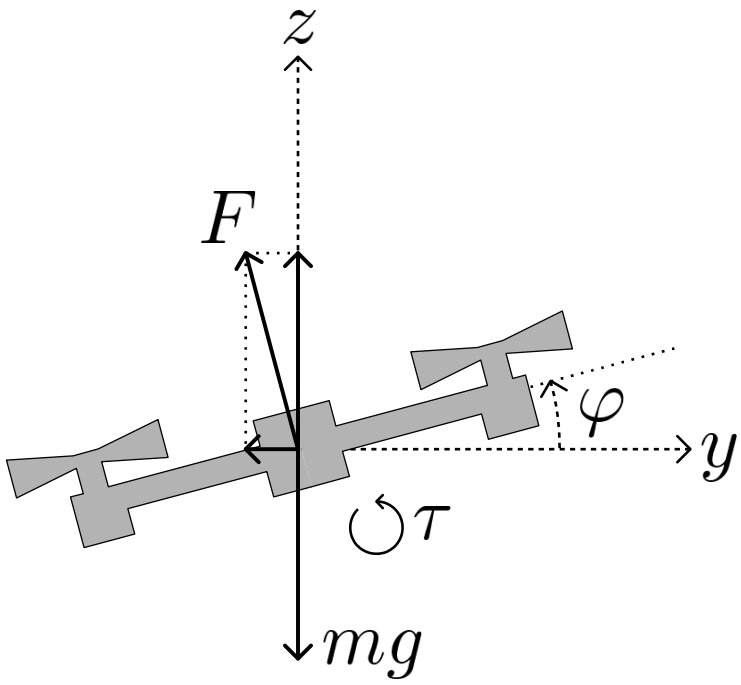
2D Quad-copter equations of motion

- Unfortunately we have to stick to simulations this year
- Keep it simple: simulate 2D case with 2 propellers. System with 2 inputs, 3 outputs
- Equations of motion (ignoring drag):

$$m\frac{d^2}{dt^2}y = -\sin(\varphi)F$$

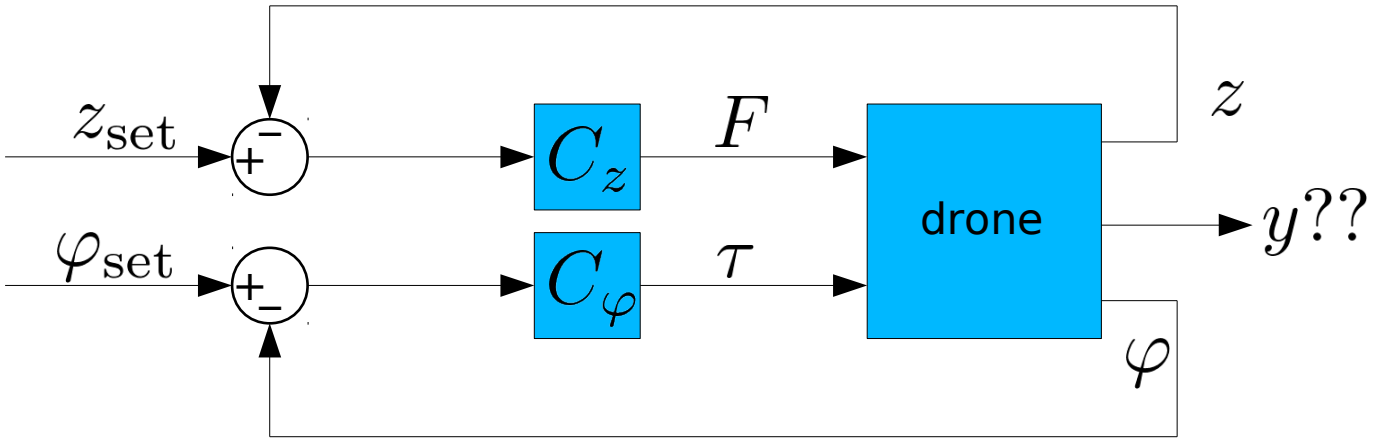
$$m\frac{d^2}{dt^2}z = \cos(\varphi)F - mg$$

$$I_{xx}\frac{d^2}{dt^2}\varphi = \tau$$



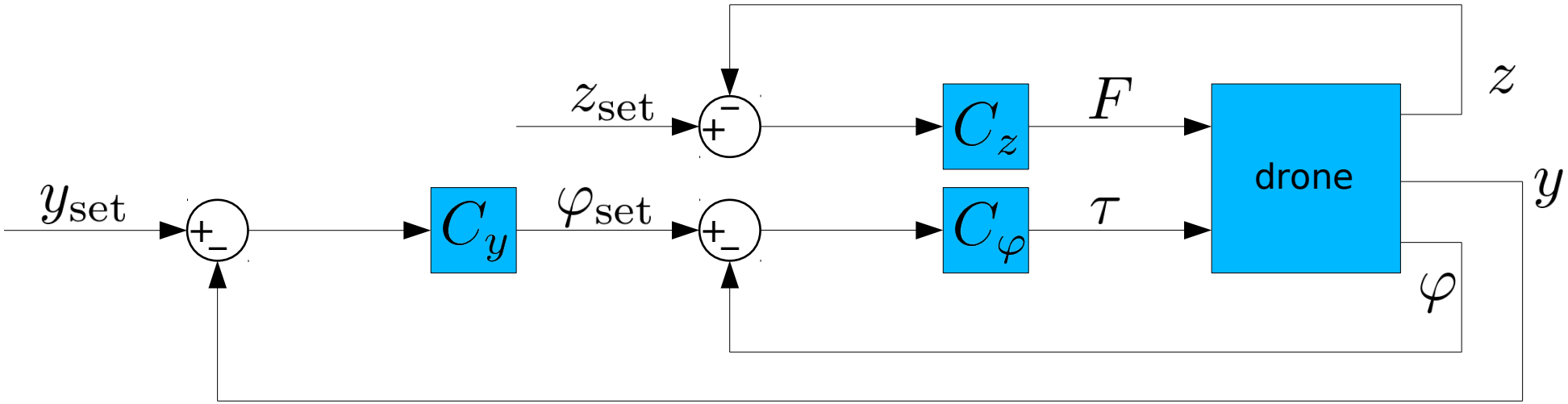
- Needs a trivial driving matrix to go from desired (τ, F) to commands for left/right motor.
- 3D case is not much more difficult, but adds some complications due to coordinate transformations between fixed world and rotating drone frame

Basic control



- Issue: for 2D case, we only have 2 actuators, so we can only control 2 of 3 DOF (for 3D case 4 actuators for 6 DOF)
- In steady-state, the force of the rotors need to balance gravity. This means φ should be most of the time close to zero. Obvious control strategy is to use the force *F* to actuate on *z*-axis, and use torque τ to actuate on the roll axis φ. These DOF can be controlled independently.
- Note that *y*-axis is uncontrolled! Not a very useful strategy if you want to go somewhere ...
- Observation: when $\varphi \neq 0$, you apply a small force along *y*: $F_y = -F\sin(\varphi) \approx -F\sin\varphi$

Cascaded control



- Solution: give up freedom to control roll and use set-point of roll loop as actuator for sideways loops
- Outer loop that works on set-point of inner loop: known as **cascaded control**
- Not obvious how such a double loop can be made stable
- Remember that well below the UGF of the inner loop, its TF from set-point to output is flat. So the double loop can be made stable if UGF of inner loop is much faster (> factor 5) than UGF of outer loop.

End of lecture

- Now you can try for yourself:

1 Topical Lectures April 2021

1.1 Drone control, part 1: keyboard flying is hard

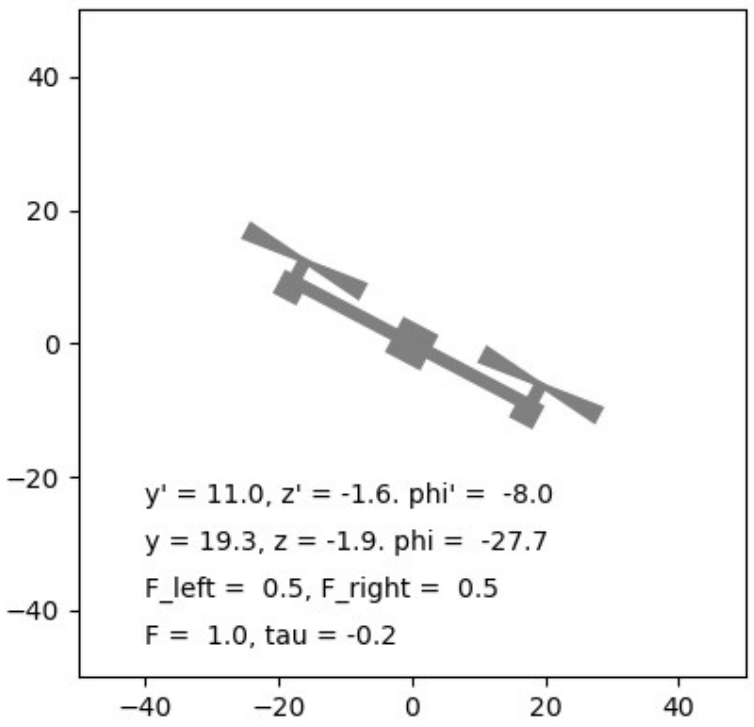
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import time
import module
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# For the interactive plots we need the QT backend
%matplotlib qt
```

In [2]:
```
class Control():
    def __init__(self, _drone, _plotter):
        self.drone = _drone
        self.plotter = _plotter

        # values for keyboard control
        self.key_dF = 0.03
        self.key_dF0 = 0.005

        self.stop = False
        self.reset()
```



Backup slides

Sensing

- For now, we assume we have perfect sensors to measure the complete state (positions/angles/speeds)
- In reality, you only have only access to a limit set of noisy measurements
- Under certain conditions, it is possible to reconstruct the full state of the system using only a limited set of sensors (observability)
- One optimal way of doing this is a Kalmann filter. Basically you run a simulation of your physical system, then adjust state until simulated measurements correspond to the observed measurements, with optimal weighting of all noises

Delay

- Every system has some delay
- Can be modeled as system with gain 1, frequency dependent phase: $TF = e^{-j\omega\tau}$
- Leads to lagging of phase, which will ultimately limit bandwidth of achievable loop
- For a digital system, you have tau = > t_sample, which will ultimately limit the bandwidth of achievable control. Rule of thumb: f_sample > 100 * UGF
- Can use Pade approximation for simulations