Ch. 11 Robot Control

14.7.23

Controller converts task specifications to forces and torques at the actuators.

Control strategies (Control objectives)

- Motion control: e.g. moving object from one place to another
- Force control: e.g. applying a polishing wheel to workpiece
- hybrid motion-force control: e.g. writing on a chalk board
  force to press chalk against board, motion to move chalk across board (forces and motions in diffent directions)
- impedance control: e.g. robot acts as a haptic display

Strategy depends on task and environment.
force-control goal makes sense when end-effector is in contact with something (not when moving in free space)

Fundamental constraint imposed by mechanics:
Robot can not independently control motion and force in the same direction.

Feedback control:
- measure actual behaviour of robot: position, velocity and force sensors
- compare with desired behaviour
- send control signal to actuators

Ch. 11.1 Control System Overview

Sensors:
- Potentiometers
- Encoders
- Resolvers (Koordinatenwandler) for joint position and angle sensing
- Tachometers for joint velocity sensing
- Joint force-torque sensors
- Multi-axis force-torque sensors (at "wrist" between arm and end-effector)

Instead of tachometer: numerically-difference position signal (with digital filter) at successive steps.
Add low-pass filter to reduce high-frequency signal parts.

The controller samples the sensors and updates its control signals to actuators at rate:

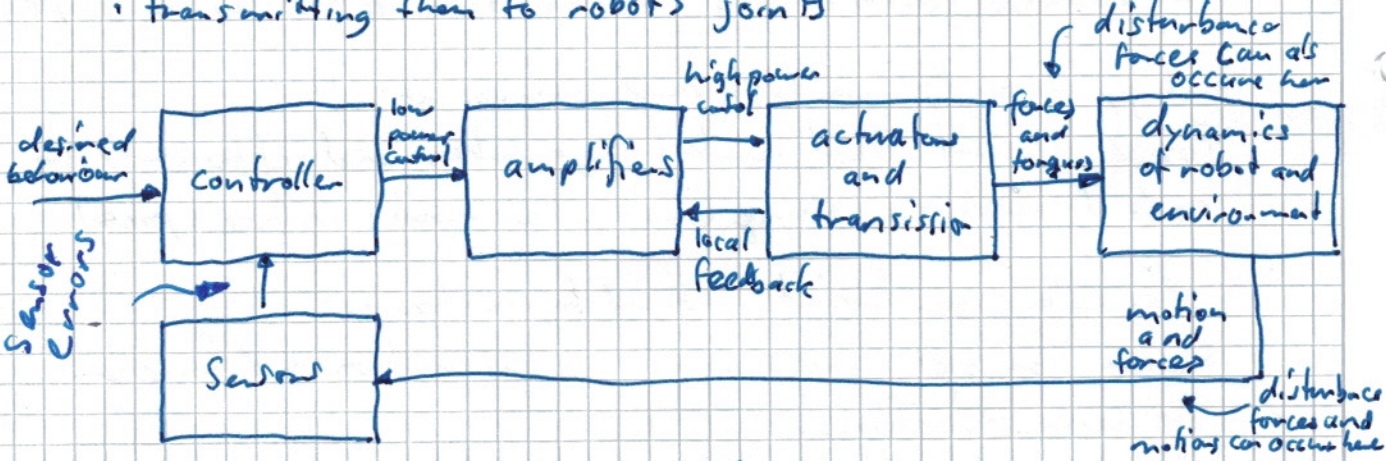
hundreds to few thousands Hz

Higher control update rates are of limited benefit because of time constants of dynamics of robot and environment.

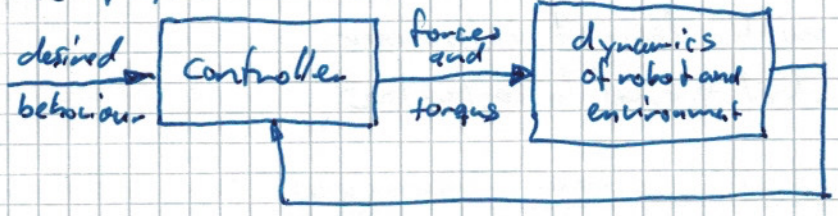
In further analysis: continuous time (sampling time = 0)

There are different technologies for:

- creating mechanical power (typically current is proportional to torque produced by motor)
- transforming speeds and forces
- transmitting them to robot's joints



- Combine joints amplifier, actuator and transmission in a "box"
- Assume perfect sensors



Ignoring mechanical flexibility, vibrations in joint and links, backlash of gears and transmissions, actuator saturation limits, limited sensor resolution...

In real robot designs these effects need to be taken into account.

Ch. 11.2. Error Dynamics

15.7.23

Controlling a single joint (can be generalized to the case of multi-joint robot.

Control objective: Motion Control

Joint error: $\Theta_e(t) = \Theta_d(t) - \Theta(t)$

with: $\Theta_e(t)$: joint error
$\Theta_d(t)$: desired joint position (desired motion) ← reference input
$\Theta(t)$: actual joint position

Differential equation for evolution of joint error $\Theta_e(t)$ is called <u>error dynamics</u>.

Feedback controller: create error dynamics such that $\Theta_e(t) \to 0$ for $t \to \infty$
(or very small)

↳ error $\Theta_e(t)$ should tend to zero (or very small value) as time increases. and as fast as possible

<u>Error Response</u> (unit step response)

Test quality of controller.

Impulse response (unit error response)

$\Theta_e(0) = 1$

$\dot{\Theta}_e(0) = \ddot{\Theta}_e(0) = \dots = 0$

Typical error response $\Theta_e(t)$:

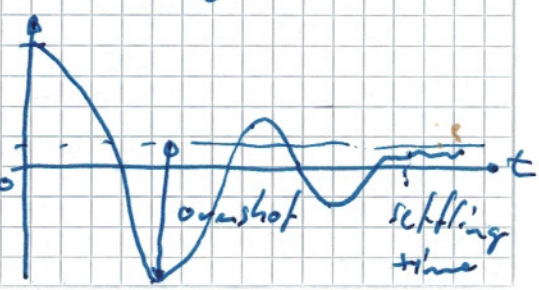
- steady-state error } steady-state response
- overshoot
- settling time } transient response

Steady state error e_{ss}: asymptotic error $\Theta_e(t)$ as $t \to \infty$

Overshoot: if error response overshoots final steady state error:

$$\text{overshoot} = \left| \frac{\Theta_{e,min} - e_{ss}}{\Theta_e(0) - e_{ss}} \right| \times 100\%$$

$\Theta_{e,min}$: least positive value achieved by error



Settling time (usually 2%):
first time T such that $|\theta_e(t) - e_{ss}| \leq 0.02(\theta_e(0) - e_{ss})$

Good error response:
- little (or no) steady-state error
- little or no overshot
- a short (2%) settling time

Linear Error Dynamics

Error dynamics: dynamics of robot and controller in response of reference input

Linear Systems.
Error dynamics described by linear ordinary differential equations:

$$a_p \theta_e^{(p)} + a_{p-1}\theta_e^{(p-1)} + a_{p-2}\theta_e^{(p-2)} + \dots + a_2\ddot{\theta}_e + a_1\dot{\theta}_e + a_0\theta_e = c$$

a pth order differential equation

For homogenous ($c=0$) linear error dynamics:

$$\theta_e^{(p)} = -\frac{1}{a_p}\left(a_{p-1}\theta_e^{(p-1)} + \dots + a_2\ddot{\theta}_e + a_1\dot{\theta}_e + a_0\theta_e\right)$$

$$= -a'_{p-1}\theta_e^{(p-1)} - \dots - a'_2\ddot{\theta}_e - a'_1\dot{\theta}_e - a'_0\theta_e$$

pth ordered differential equation $\Rightarrow$ p coupled first order differential eqs.

with $x = (x_1, \dots x_p)$ where

$$\begin{cases} x_1 = \theta_e \\ x_2 = \dot{x}_1 = \dot{\theta}_e \\ \vdots \\ x_p = \dot{x}_{p-1} = \theta_e^{(p-1)} \end{cases}$$

$$\dot{x}_p = -a'_0 x_1 - a'_1 x_2 - \dots - a'_{p-1} x_p$$

Then $\dot{x}(t) = Ax(t)$ where:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -a'_0 & -a'_1 & -a'_2 & \dots & -a'_{p-2} & -a'_{p-1} \end{bmatrix} \in \mathbb{R}^{p \times p}$$

$A \notin so(3)$

Solution to ODE: $\dot{x}(t) = Ax(t) \rightarrow x(t) = e^{At}x(0)$ $A \notin se(3)$

16.7.25

Analogy:

- scalar first-order differential eq $\dot{x}(t) = a x(t) \Rightarrow x(t) = e^{at} x(0)$
- vector differential eq $\dot{x}(t) = A x(t) \Rightarrow x(t) = \underbrace{e^{At}}_{\text{matrix exponential}} x(0)$

Convergence:

- scalar: if $a < 0 \Rightarrow \dot{x}(t) = a x(t)$ converges to $x = 0$ for any initial condition
- vector: if A is negative definite $\Rightarrow \dot{x}(t) = A x(t)$, converges to $x = 0$
  - ↳ eigenvalues of A (may be complex) have negative real components

Eigenvalues of A are given by the roots of the characteristic polynomial of A: (s: eigenvalues of charact. eqn.)

$$\det(sI - A) = s^p + a'_{p-1} s^{p-1} + \dots + a'_2 s^2 + a'_1 s + a'_0 = 0$$

↳ coefficients $a'_0, \dots, a'_{p-1}$ must be positive then roots of characteristic polynom have negative real components
  - ↳ condition holds for $p = 1$ and $p = 2$
  - for higher order systems other conditions must hold!

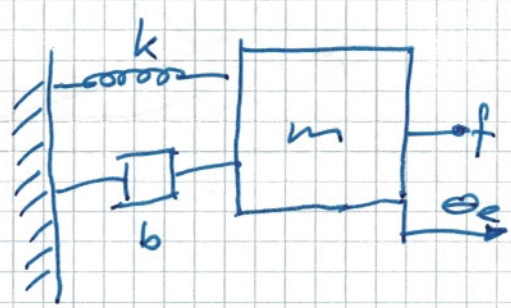
Error dynamics:

- stable: each root of characteristic polynom has neg. real comp
- unstable: otherwise $\Rightarrow \|\Theta_e(t)\|$ can grow without bounds as $t \to \infty$

Mass-Spring-Damper analogy

Mechanical analogy for second-order error dynamics

- Θ_e: position of mass m
- f: external force applied to m
- damper applies force $-b\dot{\Theta}_e$ to the mass (b: damping constant)
- spring applies force $-k\Theta_e$ to m (k: spring constant)



↳ Equation of motion of m:

$$m\ddot{\Theta}_e + b\dot{\Theta}_e + b\Theta_e = f$$

As m approaches zero:

$$b\dot{\theta}_e + k\theta_e = f$$

↳ first-order dynamics
⇒ external force generates velocity instead of acceleration

First-Order Error Dynamics

For homogenous case ($f = 0$):

$$\dot{\theta}_e(t) + \frac{k}{b}\theta_e(t) = 0$$

or

$$\dot{\theta}_e(t) + \frac{1}{\tau}\theta_e(t) = 0, \text{ with } \tau = \underbrace{\frac{b}{k}}_{\text{time constant}}$$

↳ solution:

$$\theta_e(t) = e^{-t/\tau}\theta_e(0)$$

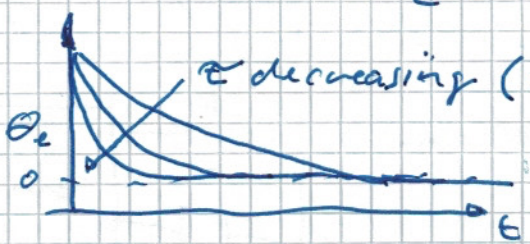
τ: time constant: first order decay to ~37% of initial value

Error response initial condition: $\theta_e(0) = 1$
Steady state error: 0, no overshoot, 2% settling time:

$$\frac{\theta_e(t)}{\theta_e(0)} = 0{,}02 = e^{-t/\tau}$$

↳ Solving: $\ln(0{,}02) = -t/\tau$

$$t = 3.9\tau$$



Second-Order Error Dynamics

21.7.23

$$\ddot{\theta}_e(t) + \frac{b}{m}\dot{\theta}_e(t) + \frac{k}{m}\theta_e(t) = 0$$

Standard second-order form:

$$\ddot{\theta}_e(t) + 2\zeta\omega_n\dot{\theta}_e(t) + \omega_n^2\theta_e(t) = 0$$

ω_n: natural frequency, $\omega_n = \sqrt{\frac{k}{m}}$

ζ: damping ratio $\zeta = b/2\sqrt{k \cdot m}$

The 2 roots of the characteristic polynomial (Eigenvalues)

$$s^2 + 2\zeta\omega_n \cdot s + \omega_n^2 = 0$$

are

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1} \quad \text{and}$$

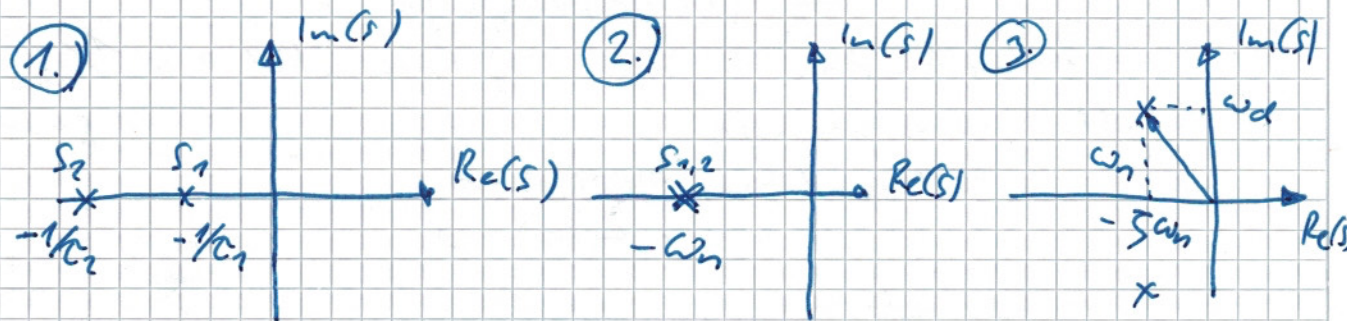
$$s_2 = -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1}$$

Second-order error dynamics is <u>stable</u> if (and only if)

$$\zeta\omega_n > 0 \text{ and}$$
$$\omega_n^2 > 0$$

If stable, 3 types of solution:

1. Overdamped (roots are real and distinct)
2. Critically damped (roots are real and equal)
3. Underdamped (roots are complex conjugate)



overdamped ($\zeta > 1$) | Critically damped ($\zeta = 0$) | underdamped ($\zeta < 1$)

22.7.23

For second-order error dynamics (in standard second-order form) :
↳ if stable (real parts < 0)

$$\ddot{\Theta}_e(t) + 2\zeta\omega_n \dot{\Theta}_e(t) + \omega_n^2 \Theta_e(t) = 0$$

① Overdamped:

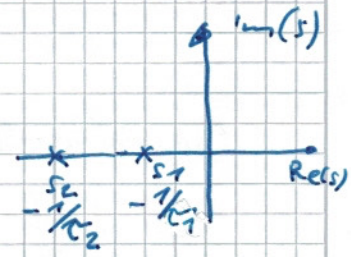
- $\zeta > 0$
- roots $s_{1,2}$ are real and distinct
- solution to second-order error dynamics differential eq.

$$\Theta_e(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

- where c_1 and c_2 can be calculated from initial conditions
- time constants $\tau_1 = -1/s_1$, $\tau_2 = -1/s_2$
- 'slower' time constant given by less negative root s_1

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1}$$

- Unit error response:
  - initial conditions · $\Theta_e(0) = 1$
    · $\dot{\Theta}_e(0) = 0$
  - constants c_1, c_2: · $c_1 = \frac{1}{2} + \frac{\zeta}{2\sqrt{\zeta^2-1}}$
    · $c_2 = \frac{1}{2} - \frac{\zeta}{2\sqrt{\zeta^2-1}}$

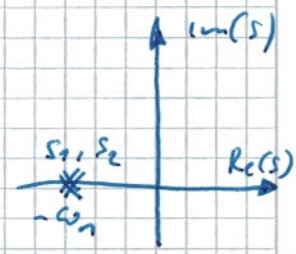


② Critically Damped

- $\zeta = 0$
- roots $s_1 = s_2 = -\omega_n$: real and equal
- solution to second-order error dynamics differential eq.

$$\Theta_e(t) = (c_1 + c_2 t) e^{-\omega_n t}$$

- time constant of decaying exponential $\tau = 1/\omega_n$
- Unit error response:
  - initial conditions · $\Theta(0) = 1$
    · $\dot{\Theta}(0) = 0$
  - constants c_1, c_2: · $c_1 = 1$
    · $c_2 = \omega_n$



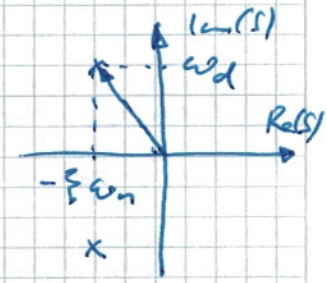
22.7.23

③ Underdamped

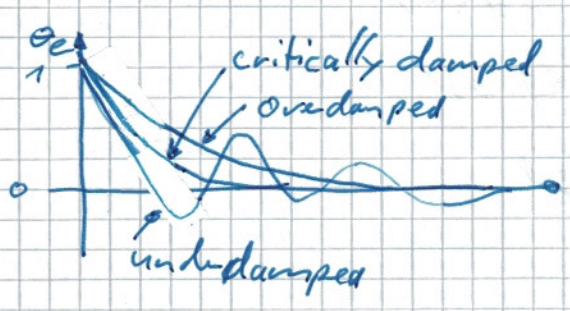
- $\xi < 1$
- roots $s_{1,2}$ are complex conjugats
  - $s_{1,2} = -\xi\omega_n \pm j\omega_d$
  - where $\omega_d = \omega_n\sqrt{1-\xi^2}$: damped natural frequency
- solution to second-order error dynamics differential eq.

$$\theta_e = (c_1\cos(\omega_d t) + c_2\sin(\omega_d t))e^{-\xi\omega_n t}$$

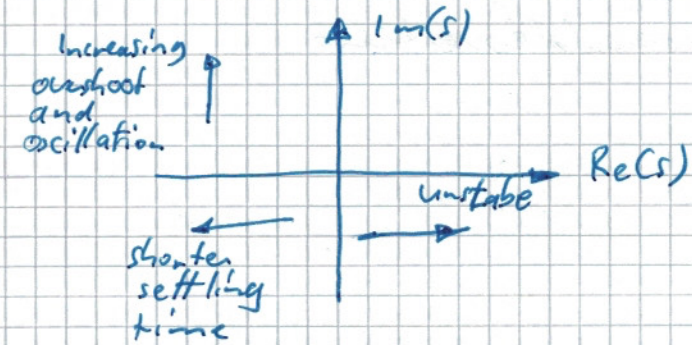
  - decaying exponential (time constant $\tau = 1/(\xi\omega_n)$ multiplied by sinusoid
- Unit error response:
  - initial conditions: $\theta_e(0) = 1$, $\dot{\theta}_e(0) = 0$
  - constants c_1, c_2: $c_1 = 1$, $c_2 = \frac{\xi}{\sqrt{1-\xi^2}}$



Error responses



Root Locations and properties of transient response



- these relationships also for higher-order systems with more than two roots.

If second-error dynamics are stable:
steady state error is zero: $e_{ss} = 0$

11.3. Motion Control with Velocity Inputs 25.7.23

Typically: control of forces and torques directly at robot's joints

Robot's dynamics transform those controls to joint accelerations

Sometimes: direct control of joint velocities e.g.:
- stepper motors: velocity of joint depends on pulse train sent to stepper
- amplifier of electric motor in velocity control mode
- high level control of a wheeled mobile robot

Motion control task can be expressed in joint space or task space

Motion Control of a Single Joint

Feed Forward Control

Given desired joint trajectory $\Theta_d(t)$

→ commanded velocity $\dot{\Theta}(t)$ as

$$\dot{\Theta}(t) = \dot{\Theta}_d(t)$$

- Simplest type of control.
- no feedback (sensors)
- feed-forward or open-loop control
- position errors will accumulate over time

Feedback Control (single joint) 25.7.23

Measure actual position of joint and implement feedback controller. Otherwise position error will accumulate over time.

P Control and First-Order Error Dynamics

Simplest feedback controller:

$$\dot{\theta}(t) = K_p(\theta_d(t) - \theta(t)) = K_p\,\theta_e(t) \qquad K_p > 0: \text{control gain}$$

Control gain K_p acts like a spring: tries to pull actual joint position to desired joint position.

Setpoint Control:

$\theta_d(t) = c$: constant

$\rightarrow \dot{\theta}(t) = 0$

$$\dot{\theta}_e(t) = \dot{\theta}_d(t) - \dot{\theta}(t) \Rightarrow \dot{\theta}(t) = -\dot{\theta}_e(t)$$

P controller: $\dot{\theta}(t) = K_p \cdot \theta_e(t)$

$$\rightarrow -\dot{\theta}_e(t) = K_p \cdot \theta_e(t)$$

$$\dot{\theta}_e(t) + K_p \cdot \theta_e(t) = 0$$

First-order error dynamics equation with time constant $\tau = 1/K_p$

$\rightarrow$ steady state error is zero

- no overshoot
- 2% settling time: $4/K_p$
- Larger $K_p \rightarrow$ faster response

Not Setpoint Control:

$\theta_d(t)$ not constant

but $\dot{\theta}_d(t) = c$: constant

Error Dynamics (P controller): $\dot{\theta}_e(t) = \dot{\theta}_d(t) - \dot{\theta}(t)$

$$= c - K_p\,\theta_e(t)$$

$$\Rightarrow c = K_p\,\theta_e(t) + \dot{\theta}_e$$

$$\dot{\theta}_e(t) + K_p\,\theta_e(t) = c$$

↳ First-order nonhomogenous linear diff. equation with solution:

$$\theta_e(t) = \frac{c}{K_p} + \left(\theta_e(0) - \frac{c}{K_p}\right) e^{-K_p t}$$

this converges to nonzero c/K_p as time goes to ∞ ($t \to \infty$)

↳ steady state error e_{ss} is <u>nonzero</u>

- the joint position always lags behind moving reference

Steady state error can be made small when choosing a big K_p. But there are limits:

- real joints have velocity limits (large commanded velocities are associated with large K_p)
- large K_p can cause instability when implemented by discrete-time digital controller

PI Control and Second-Order Error Dynamics

19.8.25

Alternative to large K_p in P-Controller:
introduce integral part (term)
↳ PI-Controller

$$\dot{\Theta}(t) = K_p \Theta_e(t) + K_i \int_0^t \Theta_e(t)\, dt$$

variable of integration: t = current time

Error dynamics for <u>constant</u> $\dot{\Theta}_d(t)$:

$$\dot{\Theta}_e(t) + K_p \Theta_e(t) + K_i \int_0^t \Theta_e(t)\, dt = c$$

⇒ time derivative:

$$\ddot{\Theta}_e(t) + K_p \dot{\Theta}_e(t) + K_i \Theta_e(t) = 0$$

Standard second-order form:

$$\ddot{\Theta}_e(t) + 2\zeta\omega_n \dot{\Theta}_e(t) + \omega_n^2 \Theta_e(t) = 0$$

with:
- $\omega_n = \sqrt{K_i}$: natural frequency
- $\zeta = K_p/(2\sqrt{K_i})$: damping ratio

Mass-spring-damper analogy:
- K_p plays role of b/m : larger K_p ⇒ larger damping constant b
- K_i plays role of k/m : larger K_i ⇒ larger spring constant c

Stability:
PI-controlled error dynamics equation is stable if
- $K_i > 0$ and
- $K_p > 0$

Roots of characteristic equation:

$$s_{1,2} = -\frac{K_p}{2} \pm \sqrt{\frac{K_p^2}{4} - K_i}$$

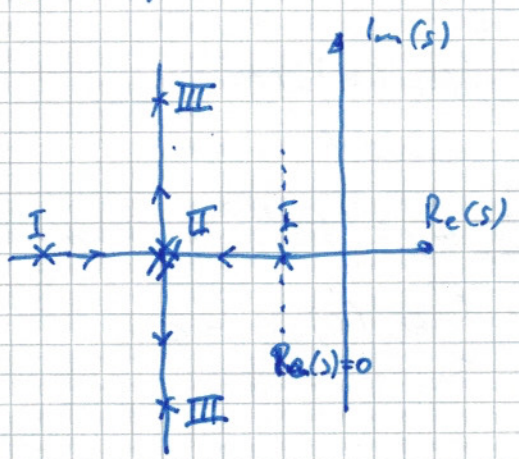
Characteristic equation:

$$s^2 + K_p s + K_i = 0$$

Root Locus

Plot of the roots as only one parameter is varied.

e.g: $K_p = 20$ K_i grows from 0



- $K_p = 20$, $K_i = 0$: (I)
  - $s_1 = 0$, $s_2 = -20$
  - over damped (roots are real)
    - $\xi = K_p/(2\sqrt{K_i}) > 1$
    - $t_1 = -1/s_1$ ("slow" root)
    - sluggish error response
- $K_p = 20$, $K_i = 100$: (II)
  - $s_1 = s_2 = -10 = -\omega_n = K_p/2$
  - critically damped
    - $\xi = 1$
    - 2% settling time: $4t = 4/(\xi\omega_n) = 0.4$
    - no overshoot or oscillation
- $K_p = 20$, $K_i > 100$: (III)
  - $s_{1,2} = -10 \pm j\sqrt{K_i - 100}$: complex conjugates
  - underdamped
    - $\xi < 1$
    - overshoot and oscillation
    - saner settling time: $t = 1/(\xi\omega_n)$

Choose K_p and K_i according to practical limitations but they should yield critical damping.

If desired $\dot{\Theta}_d(t)$ is anything other than constant the PI controller cannot eliminate steady state error completely. Well designed PI-controller can provide better tracking performance than a P-controller.

Feedforward Plus Feedback Control

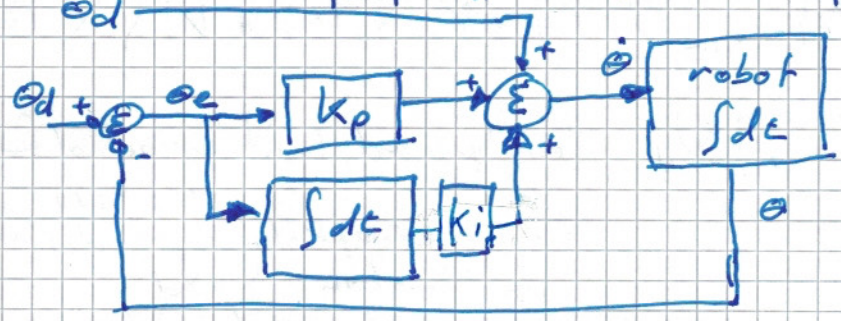
25.8.23

Feedback control: error is required for joint to move
→ use knowledge of desired trajectory/motion $\Theta_d(t)$ to move before any error accumulates

Combine advantages of feedforward and feedback control:

$$\dot{\Theta}(t) = \dot{\Theta}_d(t) + K_p \Theta_e(t) + K_i \int_0^t \Theta_e(t)\,dt$$

feedforward — proportional — integral



> This is a preferred control law for producing commanded velocities

11.3.2 Motion Control of a Multi-joint Robot

PI feedback plus feedforward controller generalizes to robots with n joints:

- reference position: $\Theta_d(t) \in \mathbb{R}^n$
- actual position: $\Theta(t) \in \mathbb{R}^n$
- gains: $K_i = I k_i \in \mathbb{R}^{n \times n}$, $K_p = I k_p \in \mathbb{R}^{n \times n}$

I: identity matrix; $k_i \in \mathbb{R}^+$, $k_p \in \mathbb{R}^+$ } positive scalars

Each joint is subject to stability and performance analysis as a single joint.

11.3.3 Task-Space Motion Control

$X(t) \in SE(3)$: configuration of robots end-effector
$V_b(t)$: end-effector twist in $\{b\}$ frame
↳ $[V_b] = X^{-1}\dot{X}$

Desired motion: · $X_d(t)$
· $[V_d] = X_d^{-1}\dot{X}_d$

Task-space control law:

$$V_b(t) = [Ad_{X^{-1}X_d}]V_d(t) + K_p X_e(t) + K_i \int_0^t X_e(t)\,dt$$

with: · $[Ad_{X^{-1}X_d}]V_d$: feedforward twist in actual frame X (X_{sd}) when end-effector is at desired configuration: $X = X_d$ this reduces to V_d

· $X_e(t)$: Error is <u>not</u> $X_d(t) - X(t)$ since we cannot substract two elements of $SE(3)$ because twist that takes X to X_d in unit time $\Rightarrow [X_e] = \log(X^{-1}X_d) \in se$

· Diagonal gain matrices K_p and K_i:
  · top 3 entries (for angular velocities)
  · bottom 3 entries (for linear velocities)
  ↳ can have different gain values for angular and for linear entries

Commanded joint velocities $\dot{\theta}$ realizing V_b from control law:

$$\dot{\theta} = J_b^{\dagger}(\theta)V_b$$

where $J_b^{\dagger}(\theta)$: pseudo-inverse of body Jacobian

Ch. 11.4 Motion Control with Torque or Force Inputs 10.9.25

Often: motor amplifier input is desired torque or force

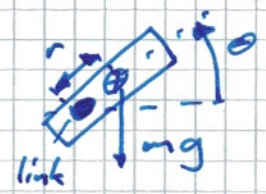
This allows to use dynamic model of robot in design of control law.

Also:

- Stepper-motor-controlled robots: generally low/predictable force/torque requirements
- velocity-control modes of off-the-shelf amplifiers of motors do not make use of dynamic model of robot

↳ Controller that generates joint torques and forces to track desired trajectory (joint space or task space)

Motion Control of a single Joint



- single motor attached to single link
- τ: torque of motor
  θ: angle of link

$g > 0$: gravitational acceleration
M: scalar inertia about axis of rotation
m: Mass of link
r: distance of axis to center of mass

Dynamics:

$$\tau = M\ddot{\theta} + mgr\cos(\theta)$$

↑ no dissipation

Simple model for friction forces:

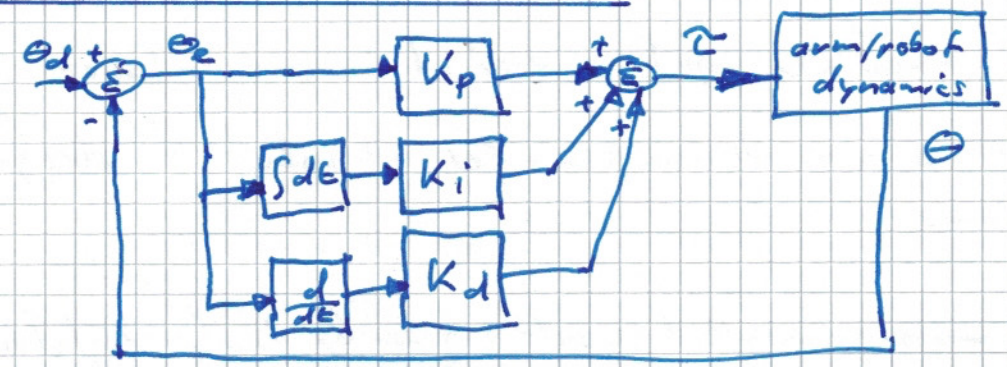
$$\tau_{fric} = b\dot{\theta} \qquad b \geq 0$$

↳ $\tau = M\ddot{\theta} + mgr\cos(\theta) + b\dot{\theta}$

↳ $\tau = M\ddot{\theta} + h(\theta, \dot{\theta})$

h: all terms depending only on state not on acceleration

Feedback Control PID Control



$$\tau = K_p \theta_e + K_i \int \theta_e(\epsilon)\, d\epsilon + K_d \dot{\theta}_e$$

- $K_p \in \mathbb{R}^+$: acts as virtual spring → reduces position error
- $K_i \in \mathbb{R}^+$: " virtual damper → reduces velocity error
- $K_d \in \mathbb{R}^+$: reduces or eliminates steady-state error

- PD Control and Second-Order Error Dynamics
  - like PID with $K_i = 0$
  - robot moves in horizontal plane : $g = 0$ (or $q = 0$)
    - Dynamics : $\tau = M\ddot{\theta} + mgr\cancel{\cos(\theta)} + b\cdot\dot{\theta}$
    - Control Law PD: $\tau = K_p \underbrace{\theta_e}_{\theta_d - \theta} + K_d \underbrace{\dot{\theta}_e}_{\dot{\theta}_d - \dot{\theta}}$

$$\hookrightarrow M\ddot{\theta} + b\dot{\theta} = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta})$$

- Set-point control : $\theta_d = c$ (constant)

$$\dot{\theta}_d = \ddot{\theta}_d = 0$$

$$\hookrightarrow \theta_e = \theta_d - \theta$$
$$\dot{\theta}_e = -\dot{\theta}$$
$$\ddot{\theta}_e = -\ddot{\theta}$$

$$\hookrightarrow M\ddot{\theta}_e + (b + K_d)\dot{\theta}_e + K_p \theta_e = 0$$

Standard second-order-form:

$$\ddot{\theta}_e + \frac{b + K_d}{M}\dot{\theta}_e + \frac{K_p}{M}\theta_e = 0$$

$$\hookrightarrow \ddot{\theta}_e + 2\zeta\omega_n \dot{\theta}_e + \omega_n^2 \theta_e = 0$$

damping ratio: $\zeta = \frac{b + K_d}{2\sqrt{K_p M}}$ natural frequency $\omega_n = \sqrt{\frac{K_p}{M}}$

Stability : $b + K_d > 0$, $K_p > 0$

↳ the steady state error = 0

- choose K_d and K_p such that $\zeta = 1$ (critical damping)
- K_p high, subject to practical issues: fast response

PID Control and Third-Order Error Dynamics 10.9.23

- Set-point control
- link moves in vertical plane ($g > 0$)
  - ↳ steady state error $\neq 0$
    - Integral term allows to eliminate steady-state error

Setpoint error dynamics:

$$M\ddot{\Theta}_e + (b + K_d)\dot{\Theta}_e + K_p\Theta_e + K_i \int \Theta_e(t)dt = \tau_{dist}$$

with: $\tau_{dist} = m \cdot g \cdot r \cos(\Theta)$: disturbance torque

Derivative on both sides:

$$M\Theta_e^{(3)} + (b K_d)\ddot{\Theta}_e + K_p\dot{\Theta}_e + K_i\Theta_e = \dot{\tau}_{dist}$$

If $\tau_{dist} = c \Rightarrow \dot{\tau}_{dist} = 0$

Characteristic equation:

$$s^3 + \frac{b + K_d}{M}s^2 + \frac{K_p}{M}s + \frac{K_i}{M} = 0$$

stable if all roots have negative real parts $\Rightarrow \Theta_e$ converges to zero

Conditions for all roots to have negative real components:

$$K_d > -b$$

$$K_p > 0$$

$$\frac{(b + K_d)K_p}{M} > K_i > 0 \quad \text{: upper and lower bounds!}$$

Design strategy:

- choose K_p and K_d for good transient response
- choose K_i large enough to eliminate steady state error but small enough to stay stable
  - ↳ integrator anti-windup places limit on how large error integral is allowed to grow (to stay stable)

PID: Not only for set-point control. Works well where $\dot{\Theta}_d(t) \neq 0$
But integral control will not eliminate tracking error along arbitrary trajectories.

Pseudocode PID Control:

```
time = 0, eint = 0          // dt: servo cycle time
                            // eint: error integral
qprev = senseAngle
loop
  [qd, qdotd] = trajectory(time) // from trajectory generator
  q = senseAngle
  qdot = (q - qprev)/dt     // simple velocity calculation
  qprev = q
  e = qd - q
  edot = qdotd - qdot
  eint = eint + e*dt
  tau = Kp*e + Kd*edot + Ki*eint
  commandTorque(tau)
  time = time + dt
endloop
```

Feedforward Control

Use model of robot dynamics to generate torques.

Model:

$$\tau = \tilde{M}(\theta)\ddot{\theta} + \tilde{h}(\theta, \dot{\theta})$$

perfect model if $\tilde{M}(\theta) = M(\theta)$ and $\tilde{h}(\theta, \dot{\theta}) = h(\theta, \dot{\theta})$

Given θ_d, $\dot{\theta}_d$ and $\ddot{\theta}_d$ from trajectory generator:

Feedforward torque $\tau(t) = \tilde{M}(\theta_d(t))\ddot{\theta}(t) + \tilde{h}(\theta(t), \dot{\theta}(t))$

Pseudo Code:

```
time = 0          // dt: servo cycle time
loop
  [qd, qdotd, qdotdotd] = trajectory(time)  // trajectory generator
  tau = Mtilde(qd)*qdotdotd + htilde(qdotd, qd)  // calc. dynamics
  commandTorque(tau)
  time = time + dt
endloop
```

There are always modelling errors ⇒ combine feedforward with feedback control.

11.4.1.3 Feedforward Plus Feedback Linearization 10.9.23

- Feedback: no model of robot and environment will be perfect
- Feedforward: good model can improve performance and analysis

PID error dynamics:

$$\ddot{\theta}_e + K_d \dot{\theta}_e + K_p \theta_e + K_i \int \theta_e(t)\,dt = 0$$

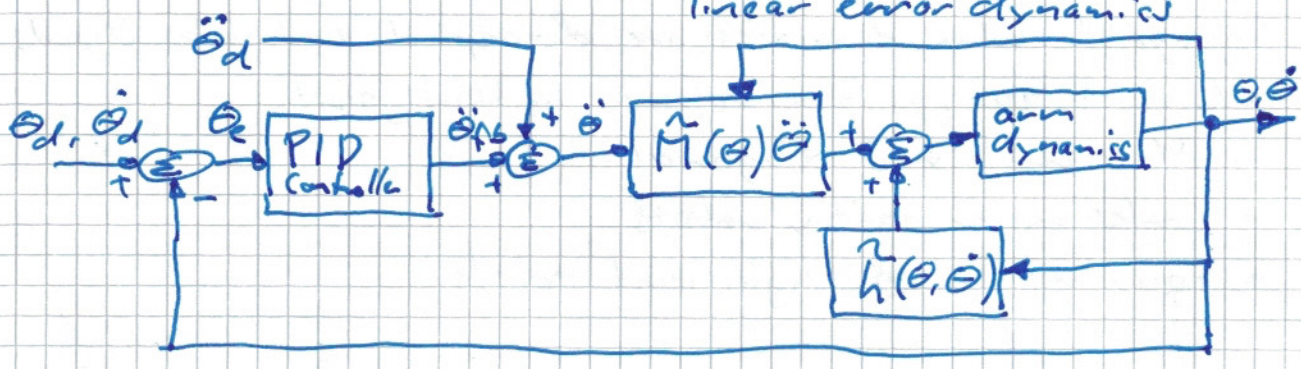
Since $\ddot{\theta}_e = \ddot{\theta}_d - \ddot{\theta}$ $\Rightarrow$ $\ddot{\theta} = \ddot{\theta}_d - \ddot{\theta}_e$: commanded acceleration

$$\hookrightarrow \ddot{\theta} = \ddot{\theta}_d + K_d \dot{\theta}_e + K_p \theta_e + K_i \int \theta_e(t)\,dt$$

Model: $\{\tilde{M}, \tilde{h}\}$

$$\hookrightarrow \tau = \tilde{M}(\theta)\left(\ddot{\theta}_d + K_p \theta_e + K_i \int \theta_e(t)\,dt + K_d \dot{\theta}_e\right) + \tilde{h}(\theta, \dot{\theta})$$

Feedback linearization: feedback of θ and $\dot{\theta}$ generate linear error dynamics



in practice K_i is often chosen to be zero (PD control)

Pseudo code

```
time = 0          // dt: cycle time
eint =            // error integral
qprev = senseAngle
loop
  [qd, qdotd, qdotdotd] = trajectory(time) // from trajectory
                                           // generator
  q = senseAngle
  qdot = (q - qprev)/.dt        // simple velocity calculation
  qprev = q
  e = qd - q
  edot = qdotd - qdot
  eint = eint + e*dt
  tau = Mtilde(q)*(qdotdotd + Kp*e + Kd*edot + Ki*eint)
                + htilde(q, qdot)
  CommandTorque(tau)
  time = time + dt
endloop
```

Motion control of a Multi-Joint Robot

n-joint robot control works analogous to single-joint robot but with vectors (and matrices!)

$$\tau = M(\theta)\,\ddot{\theta} + h(\theta, \dot{\theta})$$

$M(\theta)$: $n \times n$ positive definite mass matrix (function of configuration θ)

The components of the dynamics are coupled:

acceleration of a joint is a function of positions, velocities and torques of other joints

Decentralized Multi-Joint Control

Simplest method for multi-joint robot. Apply independent controller at each joint.

Appropriate when dynamics are decoupled: each joint depends only on torque, position-, velocity of that joint.

↳ Mass matrix is diagonal

- → Cartesian or gantry robots
- → equivalent to 3 single-joint robots

Centralized Multi-Joint Control

When · gravity forces and torques are significant and coupled
· or $M(\theta)$ not (nearly) diagonal matrix

Generalize Computed Torque Controller to multi-joint robot.

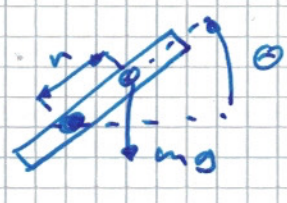
Configurations θ, θ_d and error θ_e are n-vectors and K_p, K_i, K_d are positive definite matrices

$$\tau = \tilde{M}(\theta)\left(\ddot{\theta}_d + K_p\theta_e + K_i\int\theta_e\,dt + K_d\dot{\theta}_e\right) + \tilde{h}(\theta, \dot{\theta})$$

Typically gain matrices as k_pI, k_iI, k_dI with k_p, k_i, k_d positive scalars

↳ can be optimized (approximated in some cases see book (and Krasovskii-LaSalle principle)

PID Feedforward Plus Feedback Linearization (Summary Course Video)



$$\tau = M\ddot{\theta} + h(\theta, \dot{\theta})$$

PID Feedback control:

$$\tau = K_p\theta_e + K_i\int\theta_e(t)dt + K_d\dot{\theta}_e$$

- can give good performance
- but needs error before it will command a torque

↳ Add model (dynamics) of robot: no need to wait for error to command torque

Dynamic Model: $\{\tilde{M}, \tilde{h}\}$

- based on estimates of inertia of robot
- gravitational terms and friction

perfect model if $\tilde{M} = M$ and $\tilde{h} = h$ at all times

Feedforward control: $\tau = \tilde{M}\ddot{\theta}_d + \tilde{h}(\theta_d, \dot{\theta}_d)$

Desired position, velocity and acceleration comes from known desired trajectory: there is no feedback from joint

Combine good dynamic model and stabilization of PID controller:

Computed Torque Control

$$\tau = \tilde{M}\underbrace{\left(\ddot{\theta}_d + K_p\theta_e + K_i\int\theta_e\,dt + K_d\dot{\theta}_e\right)}_{\text{acceleration: sum of feedforward acceleration at this time plus acceleration generated by PID}} + \underbrace{\tilde{h}(\theta, \dot{\theta})}_{\text{torque needed to balance friction and gravity at current state}}$$

↳ $\tilde{M}$ model turns feedforward + feedback acceleration into joint torque

↳ if error $\theta_e = 0 \Rightarrow$ feedforward

Integral term can be eliminated ($K_i = 0$) for stability reasons

Force Control

17.9.23

- Apply forces and torques to environment
- don't create motions

Pure force control is an abstraction: robots are usually able to move in at least some directions

Ideal force control: force applied by end-effector is unaffected by disturbance forces applied to the end-effector.

Manipulator Dynamics:

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) + b(\theta) + J^T(\theta) F_{tip} = \tau$$

F_{tip}: wrench applied to environment by end-effector.

F_{tip} and $J(\theta)$ in same frame (space frame or end-effector frame)

Force Control task: robot moves slowly (or not at all)

↳ ignore velocity and acceleration

↳ $g(\theta) + J^T(\theta) F_{tip} = \tau$

Absence of direct measurement of the force-torque at end-effector

↳ joint-angle feedback alone can be used to implement force control law:

$$\tau = \tilde{g}(\theta) + J^T(\theta) F_d$$

where: $\tilde{g}(\theta)$: model of gravitational torques

F_d: desired wrench

Control law requires:

- good model for gravity compensation
- precise control of torques produced at robot joints
  - ↳ in case of DC motor without gearing: torque control by current control of motor

PI Force Control

19.9.23

Derivative control typically not relevant

- force measurements are noisy: time derivating meaningless
- direct control of joint forces and torques and simple rigid body model
  - ↳ direct transmission of end-effector forces
  - ↳ no dynamics that integrate control commands to produce desired behavior.

PI controller law with feedforward and gravity compensation:

$$\tau = \tilde{g}(\theta) + J^T(\theta)\left(F_d + K_{fp} F_e + K_{fi} \int F_e(t)\, dt\right)$$

where: $F_e = F_d - F_{tip}$

- K_{fp}: proportional gain matrix
- K_{fi}: integral " "

↳ simple control law. But potentially dangerous!

- if there is nothing to push against: robot will accelerate in failing attempt to create end-effector forces

⇒ add velocity damping: typical force-control task requires little motion ⇒ limit acceleration

$$\tau = \tilde{g}(\theta) + J^T(\theta)\left(F_d + K_{fp} F_e + K_{fi} \int F_e(t) dt - K_{damp} V\right)$$

where: K_{damp}: velocity damping (positive definite).

Ch. 11.6 Hybrid Motion-Force Control

22.9.23

Control motions and forces.

n-dimensional task-space:

- $2n$ forces and motions
- free to specify n forces and motions at any time
  - often n are specified by environment

> Don't specify forces and motions in the 'same direction' as they are not independent!

Example: - 2-d environment
- Damper

$$f = B_{env} \cdot v$$

where $B_{env} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$

- $f = (f_1, f_2)$
- $v = (v_1, v_2)$

$\rightarrow$ $f_1 = 2v_1 + v_2$

$f_2 = v_1 + v_2$

- $2n = 4$ velocities and forces
- $n = 2$ freedoms to choose

Examples:

1. Specify f_1 and v_1 independently as B_{env} is not diagonal
   - then v_2 and f_2 are determined by environment
2. We cannot independently control f_1 and $2v_1 + v_2$ as they are in the same direction!

Natural and Artificial Constraints

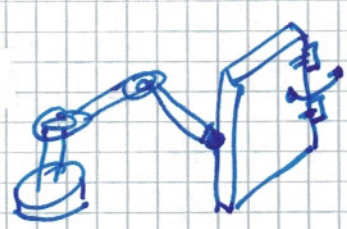
- Particulary interesting case:
  - environment is infinitely stiff (rigid constraints) in k directions
  - unconstrained in $n-k$ directions

↳ we cannot chose which of the $2n$ motions and forces to specify.

The contact with the environment choses
- k directions in which the robot can freely apply forces
- $n-k$ direction of free motion

Example 1:

- Task space: $n=6$, $SE(3)$
- Robot grasping cabinet door



↳ $6-k=1$ motion freedoms (rotating about hinges)

$k=5$ force freedoms: robot can apply any wrench that has zero moment about hinges (not moving door)

Example 2:

- Robot writing on chalkboard
- $k=1$: controlling force into board, cannot penetrate board
- $6-k=5$: freedom to move
  - 2 for motion of tip
  - 3 orientation of chalk
- two caveats:
  1. friction of chalk on board
  2. robot could move away from board leading to 6 d.o.f.

↳ simplify: all constraints are equality constraints

i.e. inequality constraint of board: chalk cannot penetrate board is treated as equality constraint: robot cannot move away from board

22.9.23

Example 3:

- robot erasing a frictionless chalkboard
- $X(t) \in SE(3)$
- configuration of of eraser (block) frame $\{b\}$ relative to space frame $\{s\}$
- Body frame twist $V_b = (\omega_x, \omega_y, \omega_z, v_x, v_y, v_z)$
  - " wrench $\mathcal{F}_b = (m_x, m_y, m_z, f_x, f_y, f_z)$
- maintaining contact with board ($k = 3$ constraints on twist):
  - $\omega_x = 0$
  - $\omega_y = 0$
  - $v_z = 0$

↳ these velocity constraints: holonomic
- can be integrated to give configuration

↳ natural constraints, specified by environment
- there are $6-k=3$ natural constraints on wrench, too
  - $m_z = 0$
  - $f_x = 0$
  - $f_y = 0$

↳ we can freely specify:
- any twist satisfying the $k=3$ velocity constraints
- any wrench satisfying the $6-k=3$ wrench constraints

↳ these motion and force specifications are called artificial constraints

Example of artificial constraints with corresponding natural constraints

natural constraint	artificial constraint
$\omega_x = 0$	$m_x = 0$
$\omega_y = 0$	$m_y = 0$
$m_z = 0$	$\omega_z = 0$
$f_x = 0$	$v_x = k_1$
$f_y = 0$	$v_y = 0$
$v_z = 0$	$f_z = k_2 < 0$

The artificial constraints cause eraser to move with $v_x = k_1$ while applying a constant force k_2 against board.

A Hybrid Motion-Force Controller

25.9.23

If environment is rigid:

→ natural constraints on velocity in task space as Pfaffian constraints

$$A(\Theta)\mathcal{V} = 0$$

where $A(\Theta) \in \mathbb{R}^{k \times 6}$ for twists $\mathcal{V} \in \mathbb{R}^6$

→ holonomic and nonholonomic constraints

Task space of robot (see book 8.6), with absence of constraints:

$$\mathcal{F} = \Lambda(\Theta)\dot{\mathcal{V}} + \eta(\Theta, \mathcal{V})$$

where: $\mathcal{T} = J^T(\Theta)\mathcal{F}$: torques/forces created by actuators

→ Constrained dynamics (s.a. book 8.7.):

$$\mathcal{F} = \Lambda(\Theta)\dot{\mathcal{V}} + \eta(\Theta, \mathcal{V}) + \underbrace{A^T(\Theta)\lambda}_{\mathcal{F}_{tip}}$$

where: • $\lambda \in \mathbb{R}^k$: lagrange multipliers

• $\mathcal{F}_{tip}$: wrench of robot against constraint

→ $\lambda = (A\Lambda^{-1}A^T)^{-1}(A\Lambda^{-1}(\mathcal{F} - \eta) - \dot{A}\mathcal{V})$

→ to calculate wrench $\mathcal{F}_{tip} = A^T(\Theta)\lambda$ that robot applies against constraint

→ see book for full control law algorithm