Robot Control: Introduction

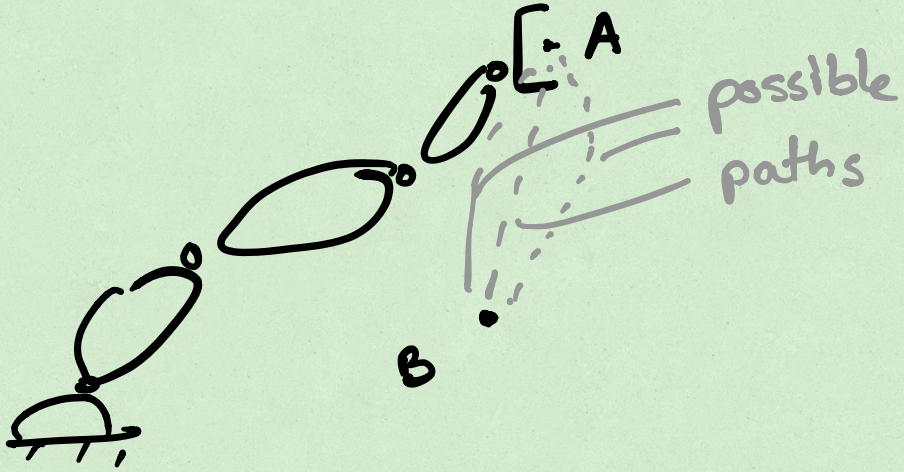
Control problem: determine joint inputs required for the end-effector to execute a desired action

examples of Joint inputs:
- motor torques
- voltage to the motors
- joint velocities (easy if you use servomotors)

Problems encountered may vary with the mechanical design of the robot (a Cartesian robot vs an elbow robot or use of gears)

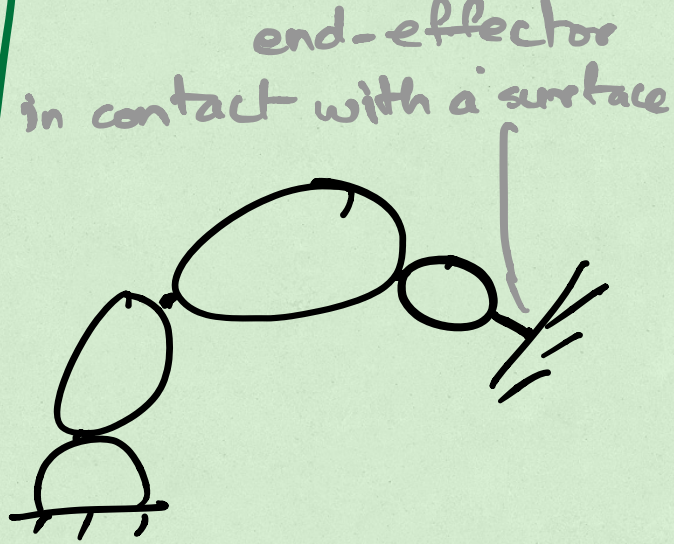
Motion vs force control

Position / Motion control



- Point to point (PTP) control:
  move robot from position A to B. (position here refers to position and orientation)
- Trajectory tracking problem:
  Move robot along a specified path $x(t)$

Force control



- cannot move the end-effector 'inside' the surface!
- regulate contact force normal to the surface

Control design and challenges

Control schemes are based on

feedforward : commands generated at planning stage

feedback : commands generated using robot state measures

Control law can be defined either in Joint or Cartesian (Task) space.

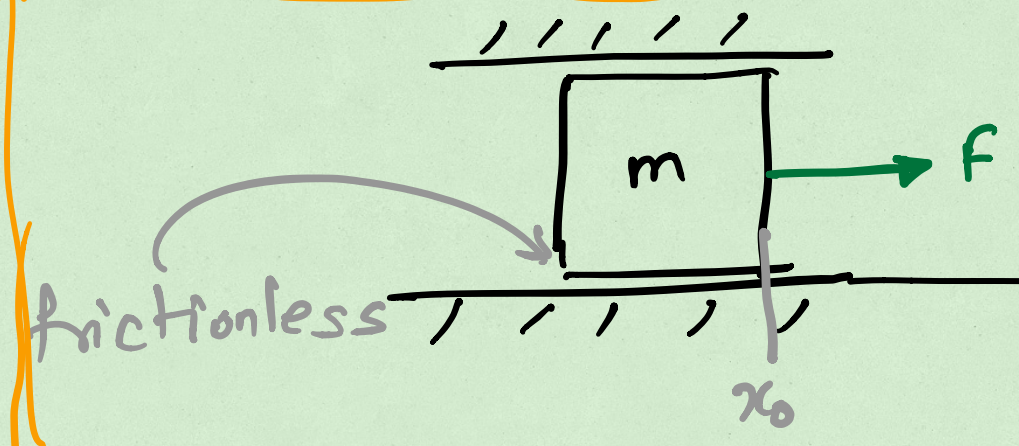
Control action always occurs at joint level (actuators driving the robot)

Challenges posed by:
- unmodeled effects (friction, flexibility of transmission)
- external disturbances
- errors (initial or due to dist.)
- uncertain robot parameters etc.

Motion Control: Preliminaries

Basic concepts we will use for manipulators

Consider a single degree of freedom (DOF) robot with a prismatic joint.



the robot is just a mass (inertia)! governed by

$$F = m\ddot{x}$$

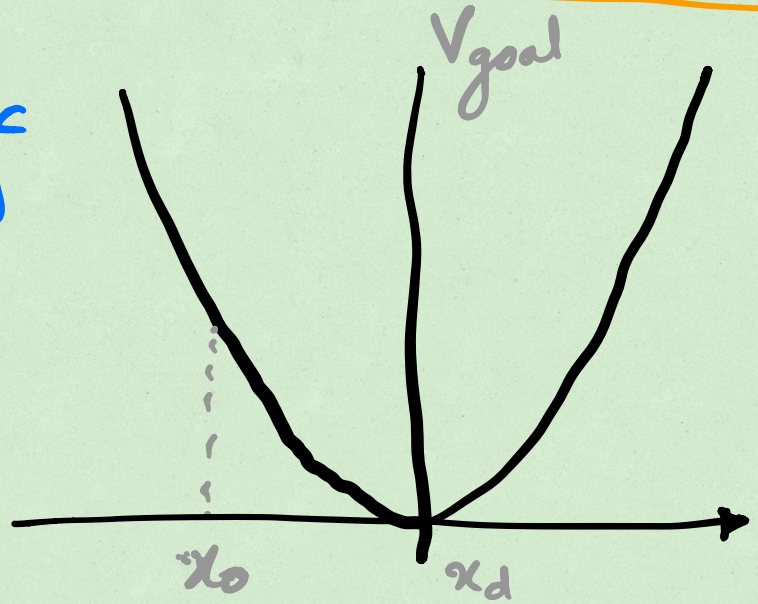
Goal: Find $F(t)$ to move the robot from the initial position x_0 to a desired (goal) position x_d.

Potential Field

Recall: artificial potential fields were useful for path planning

Can we do something similar here?

Quadratic artificial potential field



$$V_{goal} = \frac{1}{2} K_p (x_g - x)^2$$

$K_p > 0$, $V_{goal} \geq 0$

$V_{goal} = 0$ iff $x = x_d$

Corresponding force, $F_g = -\frac{\partial V_{goal}}{\partial x} = K_p (x_d - x)$

Initial control scheme (proportional (P) control)

System: $m\ddot{x} = F$

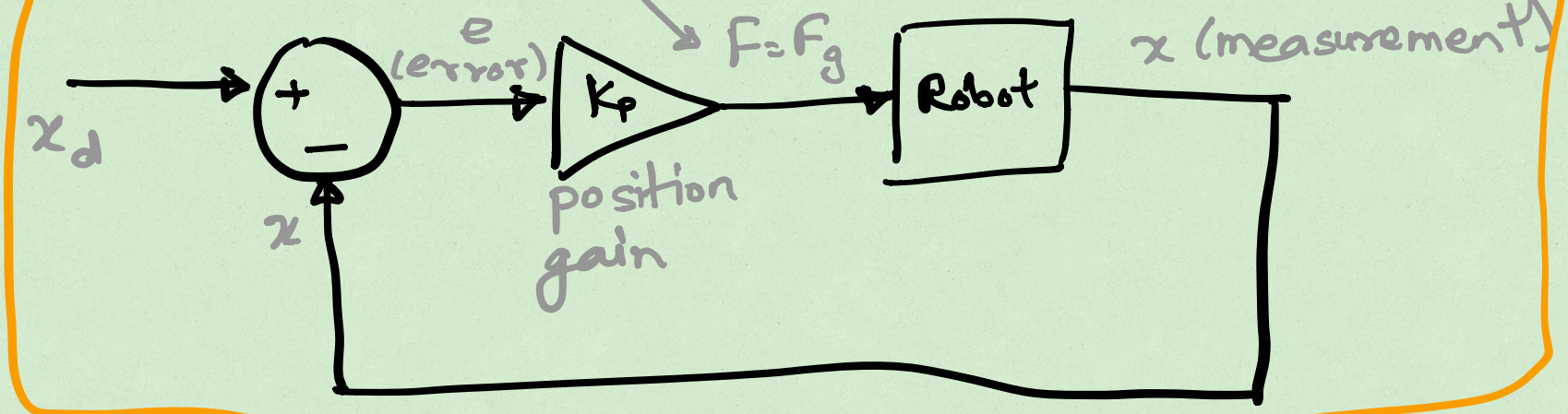
Based on the potential field,

this is like having a virtual spring between current and goal position!

Choose, $F = F_g = k_p(x_d - x)$

Actuator force

error in position, e



Actuator force is proportional to the error in robot position !!

Stability

Kinetic energy of the robot, $T = \frac{1}{2} m \dot{x}^2$

Using the Lagrangian,

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x}}\right) - \frac{\partial T}{\partial x} = F = -\frac{\partial V_{goal}}{\partial x}$$

or, $$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x}}\right) - \frac{\partial (T - V_{goal})}{\partial x} = 0$$

L

no non-conservative forces

V_{goal} is like potential energy associated with our closed loop system!!

Energy is conserved $\Rightarrow$ system is stable

What about performance?

$$m\ddot{x} = F = F_g = k_p(x_d - x)$$

if initial position, $x_0 = x_d \rightarrow F = 0 \Rightarrow \ddot{x} = 0$

if $x_0 \neq x_d$, intial potential energy,

$$V = \frac{1}{2} k_p (x_d - x_0)^2$$

But, energy is conserved! (potential energy is converted to Kinetic and viceversa)

robot doesn't stop at x_d ($V = 0$)!!!

Robot motion is oscillatory about $x = x_d$, similar to that of a simple pendulum or mass-spring system.

Velocity feedback (proportional-derivative (PD) control)

Consider a modified control law,

$$F = F_g + F_v = K_p(x_d - x) + F_v$$

where, F_v should be a force that helps bring the robot to rest at $x = x_d$

then, $$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{x}}\right) - \frac{\partial (T - V_{grav})}{\partial x} = F_v$$

Recall: rate of change of energy, $\dot{H} = F_v^T \dot{x}$

to bring the robot to rest at x_d, $\dot{H} < 0$, if $\dot{x} \neq 0$

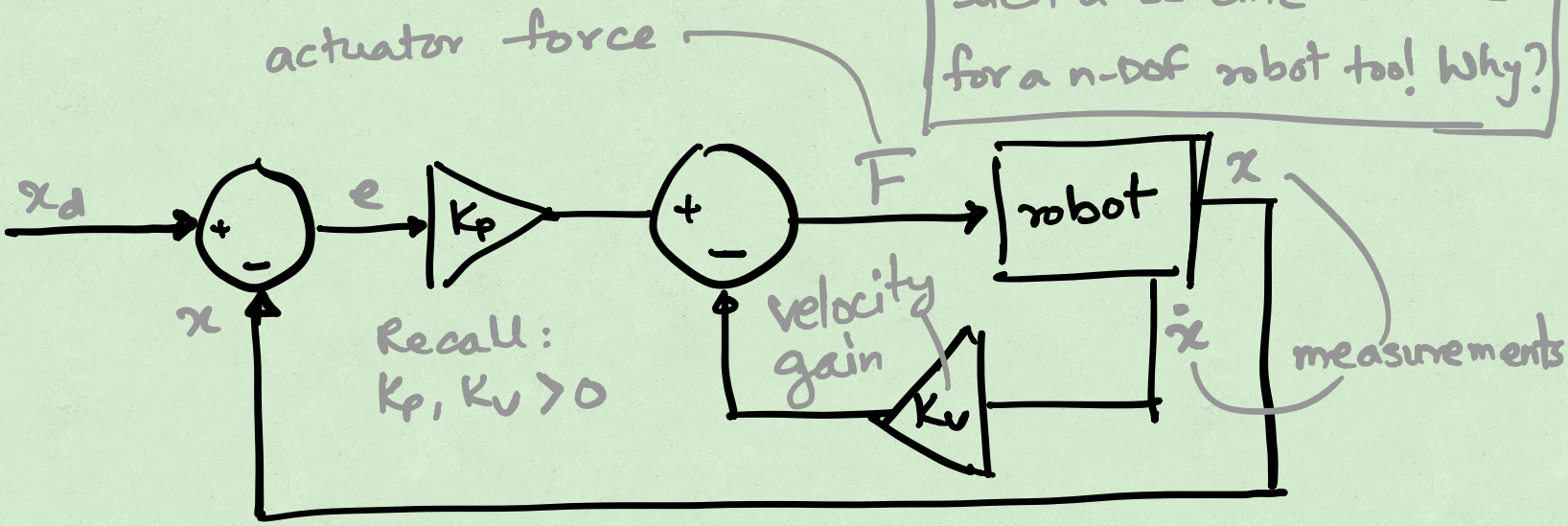
a simple choice is $F_v = -K_v \dot{x}$, $K_v > 0$

overall, $F = K_p(x_d - x) - K_v \dot{x}$

$-K_v \dot{x}$: like a damper

PD control (continued)

We will soon see that such a scheme works for a n-DOF robot too! Why?



PD controller ensures that the robot reaches, and comes to rest at $x = x_d$, eventually.

How does choice of K_p, K_v affect performance (in terms of oscillations, time to settle at goal position, effect of disturbances etc.) ?

PD control (performance)

The PD controller imitates a mass-spring-damper system

damping, stiffness

$$m\ddot{y} + b\dot{y} + Ky = 0$$

This is same as the PD controlled system with the end of the spring fixed at 0.

You should be able to verify that this can be obtained from the earlier one with a simple change of variables, $y = x - x_d$

Recall that when $b = 0$ (no damping), motion is oscillatory with a constant amplitude

Second-order system

$$m\ddot{x} + b\dot{x} + Kx = 0$$

$$\text{or,} \quad \ddot{x} + \frac{b}{m}\dot{x} + \frac{K}{m}x = 0$$

Define, $\omega_n^2 = \frac{K}{m}$; $\xi = \frac{b}{2\omega_n m} = \frac{b}{2\sqrt{Km}}$

ω_n : natural frequency

ξ : damping ratio

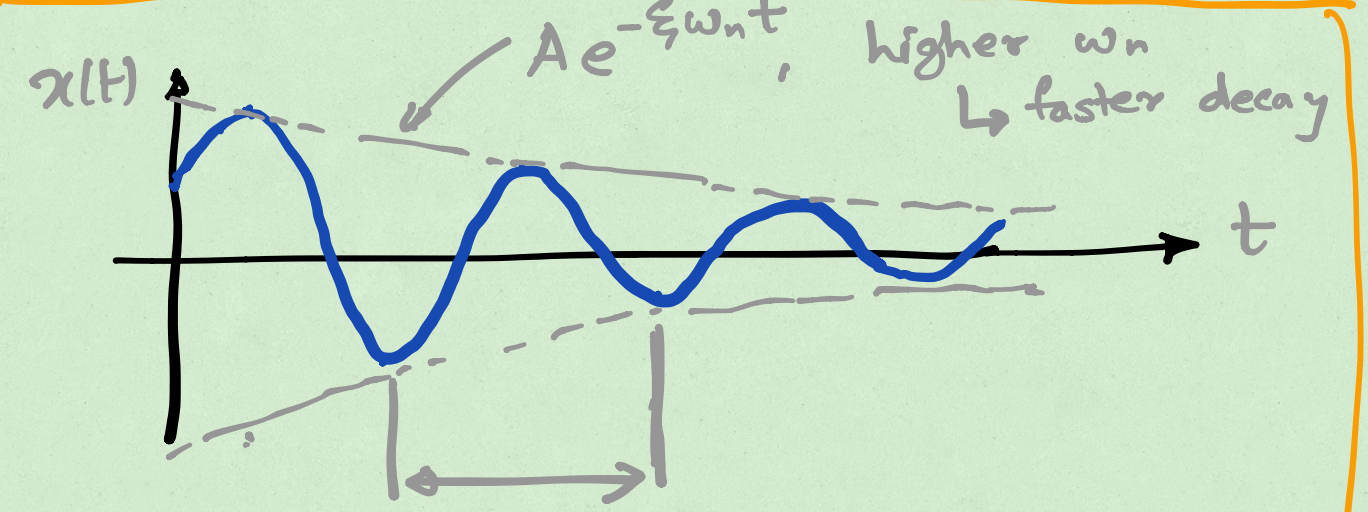
The solution to this system is given by,

$$x(t) = \underbrace{Ae^{-\xi\omega_n t}}_{\text{amplitude decays exponentially}} \cos\left(t\underbrace{\omega_n\sqrt{1-\xi^2}}_{\omega_d} + \phi\right), \quad \xi \leq 1$$

ω_d : damped natural frequency

Second-order system (time-response)

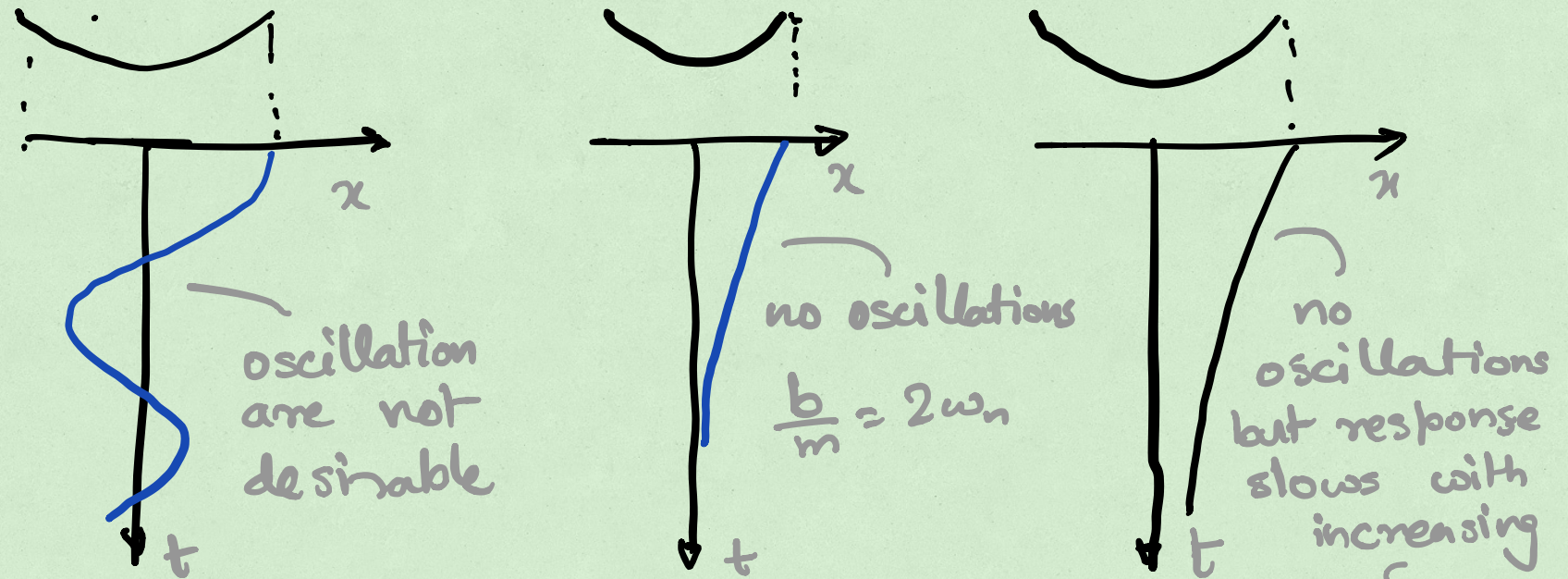
$$x(t) = Ae^{-\xi\omega_n t}\cos\left(t\omega_n\sqrt{1-\xi^2} + \phi\right), \quad \xi \leq 1$$



For $\xi > 1$, the response is exponential with no oscillations.

Second order system (time response)

Increasing damping →



$\zeta < 1$ underdamped

$\zeta = 1$ critically damped

$\zeta > 1$ overdamped

We want to choose K_p & K_v to have a critically damped system ✓

PD control (choice of K_p, K_v)

$$m\ddot{x} = F = K_p(x_d - x) - K_v\dot{x}$$

or, $$m\ddot{x} + K_v\dot{x} + K_p(x - x_d) = 0$$

K_v: velocity gain

K_p: position gain

or, $$\ddot{x} + \frac{K_v}{m}\dot{x} + \frac{K_p}{m}(x - x_d) = 0$$

→ second order mass spring damper!

comparing $\ddot{x} + 2\xi\omega\dot{x} + \omega^2(x - x_d) = 0$

closed loop damping ratio, $\xi = \frac{K_v}{2\sqrt{K_p m}}$

closed-loop natural frequency, $\omega = \sqrt{\frac{K_p}{m}}$

Selection of K_p and K_v

- Typically, we want the robot behavior to be critically damped (reaches goal as fast as it can with no oscillations)

$$\xi = \frac{K_v}{2\sqrt{K_p m}} = 1$$

- Choice of ω determines K_p!
  - we want ω as high as possible (faster response)
  - High K_p is also good to limit deviations due to disturbances!

Selection of K_p & K_v (continued...)

Hence, $K_p = m\omega^2$

$K_v = m(2\xi\omega)$

ω : typically limited by unmodeled dynamics — from flexibility in the robot structure (due to couplings, transmissions etc.), communication delays in the control loop etc.

In practice, start with low ω and increase till the response is free of vibrations.

Control partitioning

Both K_p and K_v are scaled by mass m!

Unit mass system	m-mass system
$K_p' = \omega^2$	$K_p = m K_p'$
$K_v' = 2\xi\omega$	$K_v = m K_v'$

1. Design using the unit-mass system.
2. Scale by mass m to keep the same ω

This works even if m changes with q, as in case of our robot manipulators!

Control Partitioning

$$m\ddot{x} = F \Rightarrow m(1 \cdot \ddot{x}) = mf'$$

$$f' = -K_v'\dot{x} + K_p'(x_d - x)$$

$$F = m(-K_v'\dot{x} + K_p'(x_d - x)) \quad (K_p = mK_p', \; K_v = mK_v')$$

$$= mf'$$

$\downarrow$

$$m\ddot{x} = F = mf'$$

or, $$\ddot{x} = f'$$

or, $$\ddot{x} + K_v'\dot{x} + K_p'(x - x_d) = 0$$

same dynamic behavior as the unit mass system.

Other characteristics may be different.

Handling Non-Linearities

try to model and compensate

$$m\ddot{x} + b(x, \dot{x}) = F$$

$b(x, \dot{x})$: non-linear terms (could be due to friction, gravity etc.)

Control partitioning:

$$F = \alpha f' + \beta$$

where, $\alpha = \hat{m}$, $\beta = \hat{b}(x, \dot{x})$

$\hat{m}$: our estimate of m

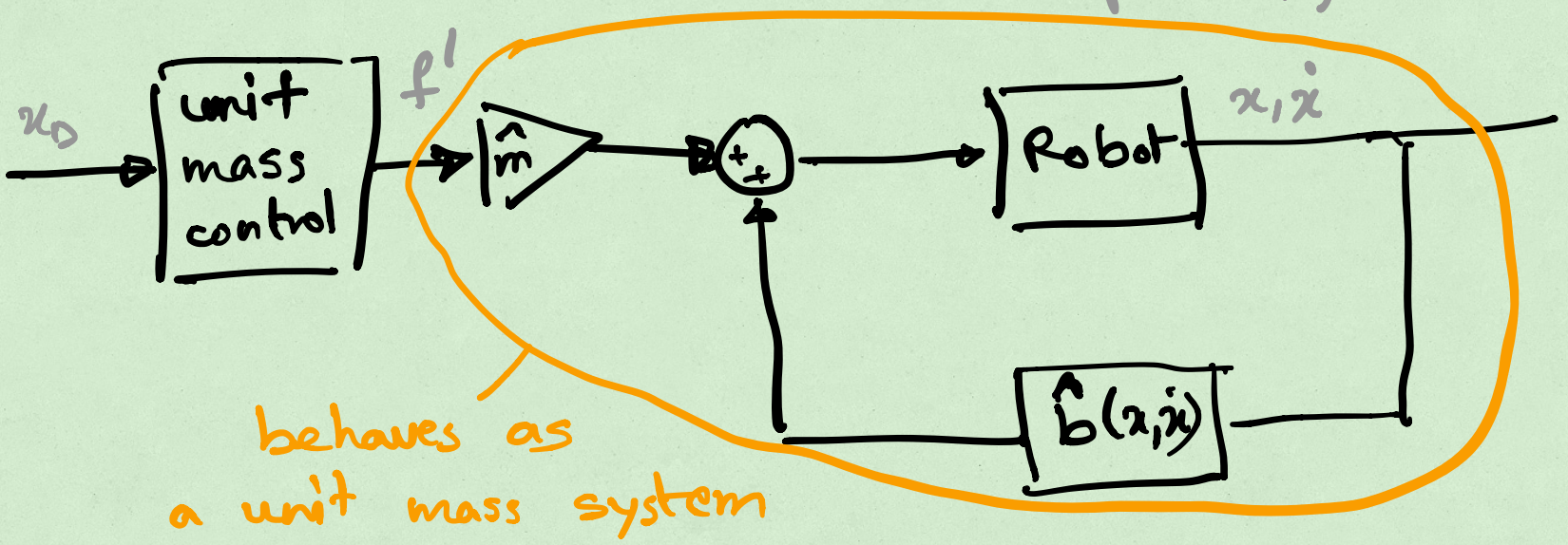
$\hat{b}$: our estimate of b

Handling non-Linearities (continued...)

$$m\ddot{x} + b(x,\dot{x}) = \hat{m}f' + \hat{b}(x,\dot{x})$$

With perfect estimates and measurements, this behaves as a unit mass system!

$$\ddot{x} = f'$$ (in practice, this will be close but not perfect)



Motion control : Trajectory tracking

How can we modify the controller to follow a path ?

i.e. we want to design f' (we have already converted the system to behave as a unit mass system) to follow a path defined by $\ddot{x}_d(t)$, $\dot{x}_d(t)$ and $x_d(t)$

move the free end of the spring and damper (of the earlier PD controller) along the path !

$$f' = \ddot{x}_d - K_v(\dot{x} - \dot{x}_d) - K_p(x - x_d)$$

feed forward — to track the acceleration

Motion control (continued)

Closed loop system:

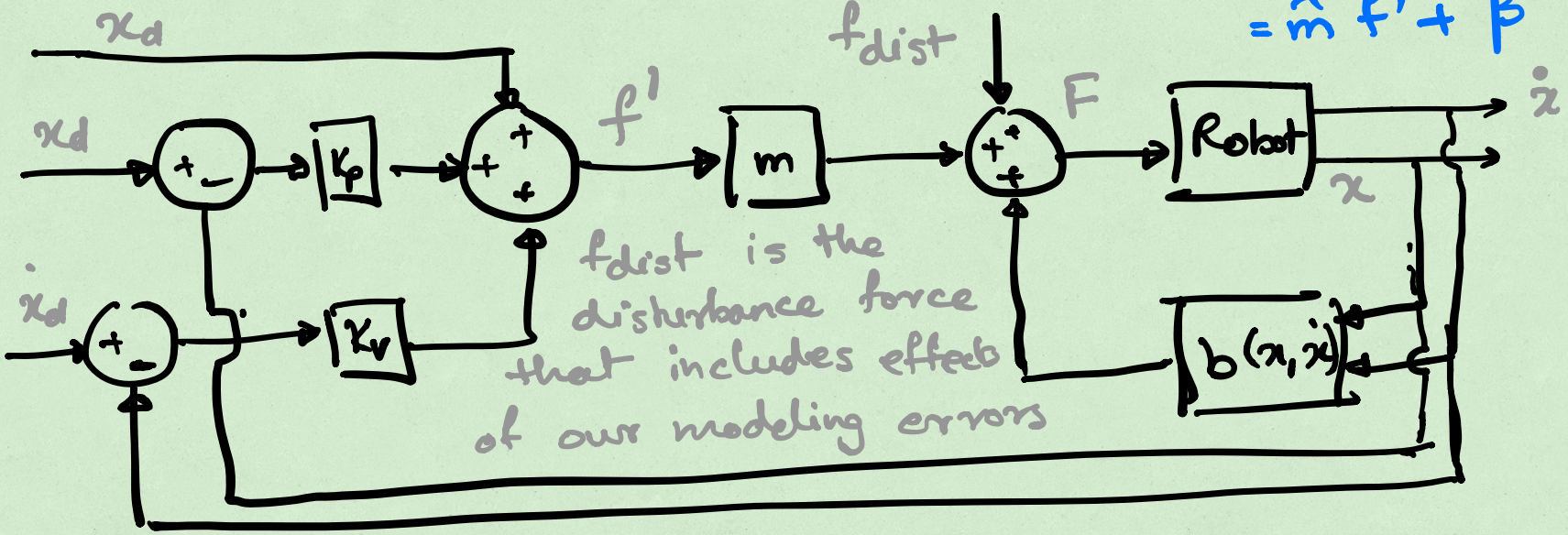
$$(\ddot{x} - \ddot{x}_d) + K_v(\dot{x} - \dot{x}_d) + K_p(x - x_d) = 0$$

linear second order system

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \qquad e = x - x_d \text{ is the error}$$

For the full system, $m\ddot{x} + b(x, \dot{x}) = F$

$= \hat{m} f' + \beta$



Steady-state error

$$m\ddot{x} + b(x,\dot{x}) = F + f_{dist}$$

control: $F = mf' + b(x,\dot{x})$ — any modeling errors are in f_{dist}

Assume: f_{dist} is bounded ($|f_{dist}| < a$ at all times)

Closed-loop: $\ddot{e} + K_v'\dot{e} + K_p e = \frac{f_{dist}}{m}$

Steady state error: error when $\ddot{e} = \dot{e} = 0$

(how far we are from rest position)

if $\ddot{e} = \dot{e} = 0$, $e = \frac{f_{dist}}{m K_p'}$ — recall K_p' determines

if m varies with position so does steady state error!

$\Rightarrow$ Large $K_p \Rightarrow$ small errors

PID control

Add an integral action to address steady-state error.

System: $m\ddot{x} + b(x, \dot{x}) = F + f_{dist}$

Control: $f' = \ddot{x}_d - K_v(\dot{x} - \dot{x}_d) - K_p(x - x_d) - K_i \int (x - x_d)\, dt$

builds up over time and compensates

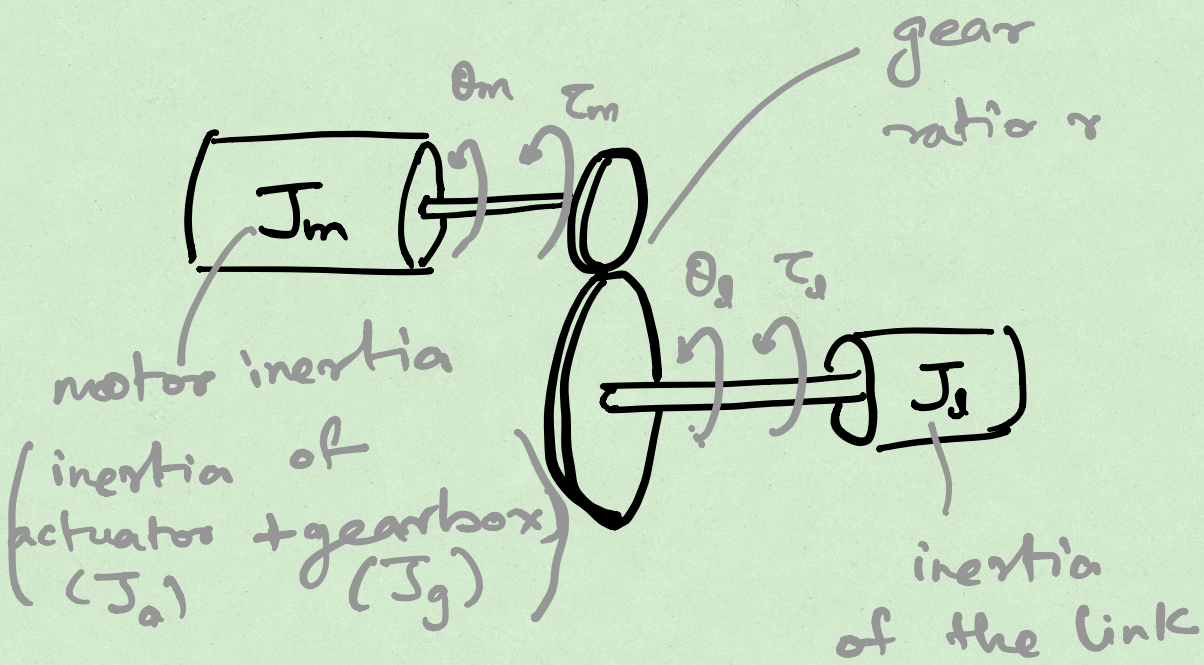
Closed-loop system:

$$\ddot{e} + K_v' \dot{e} + K_p' e + K_i \int e\, dt = \frac{f_{dist}}{m} \quad \text{(constant)}$$

or $$\dddot{e} + K_v' \ddot{e} + K_p' \dot{e} + K_i e = 0 \rightarrow e = 0$$

In practice, integral action is usually used only when close to goal.

Gear reduction



$$\theta_m = r\,\theta_l$$

$$\dot{\theta}_m = r\,\dot{\theta}_l$$

$$\tau_l = r\,\tau_m$$

$r \sim 25\text{-}300$ for most industrial robots

Why use gears?

- small r → need large motors to get required torques → larger motors, increases weight!
- Most motors work efficiently at higher speeds than needed for robot

Gear reduction (continued...)

How do gears change the dynamics?

$$J_m \ddot{\theta} + b_m \dot{\theta} = \tau_m - \frac{\tau_l}{\gamma}$$

$$= \tau_m - \frac{1}{\gamma}\left(J_l \ddot{\theta}_l + b_l \dot{\theta}_l\right)$$

or,
$$\left(J_m + \frac{J_l}{\gamma^2}\right)\ddot{\theta}_m + \left(b_m + \frac{b_l}{\gamma^2}\right)\dot{\theta}_m = \tau_m$$

effective inertia; effective damping

or, on link side
$$\left(J_l + \gamma^2 J_m\right)\ddot{\theta}_l + \left(b_l + \gamma^2 b_m\right)\dot{\theta}_l = \tau_l$$

these can be very large even for small J_m, b_m!

Problems with gears: increased inertia, friction and backlash

Actuator dynamics

- To control using voltage commands to the actuator
- For a permanent magnet DC (PMDC) motor, (other actuators will have different dynamic behavior)

relationship between current, voltage and motor torque is governed by

$$L\frac{di}{dt} + Ri = V - V_b$$

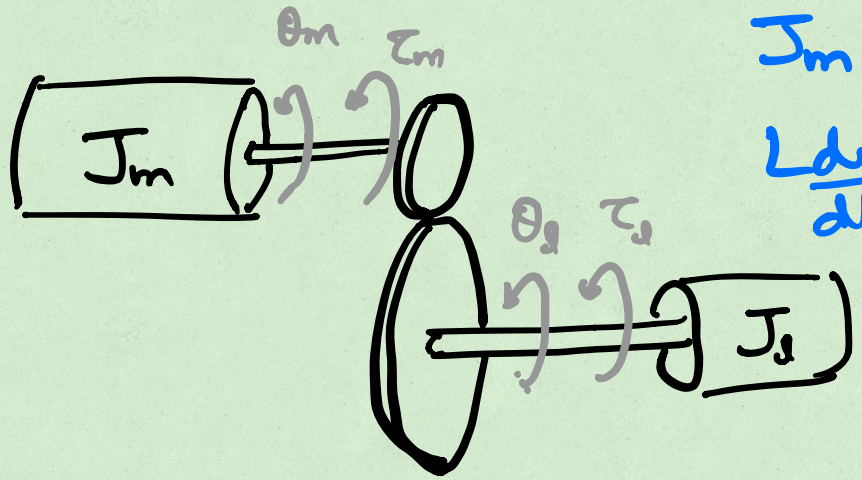
- L: inductance of armature
- R: resistance
- V: applied voltage

$$V_b = K_b \dot{\theta}_m$$

$$\tau_m = K_m i$$

K_b, K_m: motor constants

Actuator dynamics (combined model)



$$J_m \ddot{\theta}_m + B_m \dot{\theta}_m = K_m i - \frac{\tau_g}{r}$$

$$L\frac{di}{dt} + Ri = V - K_b \dot{\theta}_m$$

governed by these two coupled equations

However, motors respond much faster than the robot. This can be used to simplify and model the dynamics using a singe equation (ignore di/dt)

$$J_m \ddot{\theta}_m + \left(B_m + \frac{K_m K_b}{R}\right)\dot{\theta}_m = \frac{K_m}{R} V - \frac{\tau_g}{R}$$