Masood Dehghan

Sep. 2018

CMPUT 399
Intro Robotics & Mechatronics:
Control

with slides from Chad Jenkins (U Michigan) and Oussama Khatib (Stanford)

Outline

- Control
  - 2^{nd} Order ODEs
  - Natural (passive) systems
  - PID Control

Motivation

- Control >>> design the behavior of your mechatronic system.

Boston Dynamics' Atlas robot can backflip now

Natural (passive) systems

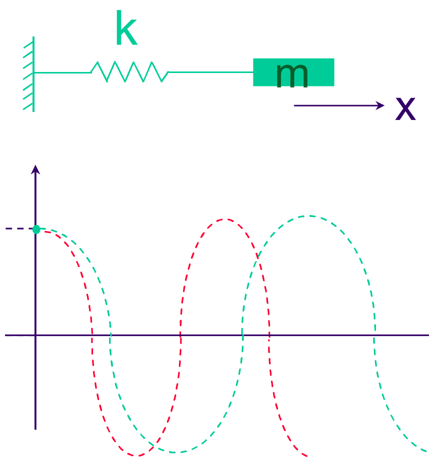
- Mass-spring system 2nd order system:

Newton's Law: $F = m.a$

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

$$m\ddot{x} + kx = 0$$

$$x(t) = x_0 \cos\left(\sqrt{\frac{k}{m}}\, t + \phi\right)$$



Natural Frequency $\omega_n = \sqrt{\frac{k}{m}}$

Frequency increases with stiffness and inverse mass

$$\ddot{x} + \omega_n^2 x = 0$$

Normalized

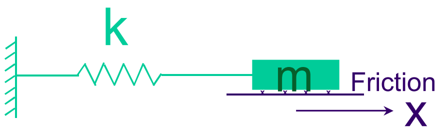
$$x(t) = c\, \cos(\omega_n t + \phi)$$

https://www.myphysicslab.com/springs/single-spring-en.html

Dissipative systems

- Dissipative systems

Dissipative Systems



Viscous friction: $f_{friction} = -b\dot{x}$

$$m\ddot{x} = -kx - b\dot{x}$$

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

Normalized: $$\ddot{x} + \frac{b}{m}\dot{x} + \frac{k}{m}x = 0$$

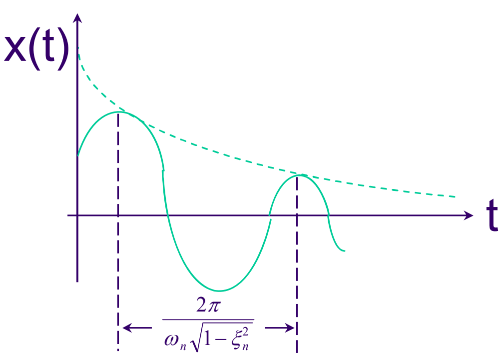
2^{nd} order system

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

$$\ddot{x} + 2\xi_n\omega_n\dot{x} + \omega_n^2 x = 0$$

Natural frequency $\omega_n = \sqrt{\frac{k}{m}}$; $\xi_n = \frac{b}{2\sqrt{km}}$ Natural damping ratio

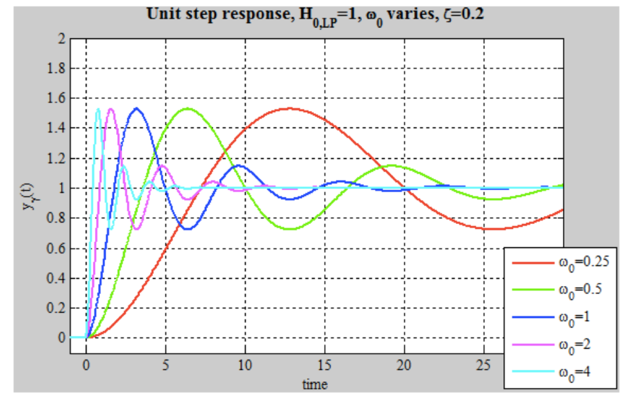
$$x(t) = ce^{-\xi_n\omega_n t}\cos(\underbrace{\omega_n\sqrt{1-\xi_n^2}}_{\omega}\,t + \phi)$$

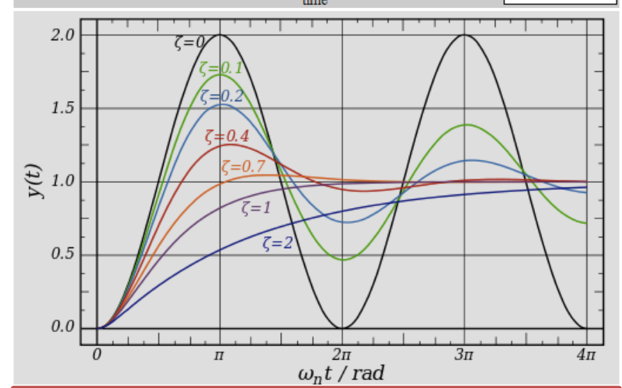


ω — damped Natural frequency

$$\omega = \omega_n\sqrt{1-\xi_n^2}$$

https://www.myphysicslab.com/springs/single-spring-en.html





Higher **ω** : faster response

Higher **ζ**: slower response

ζ = 1 : optimal choice

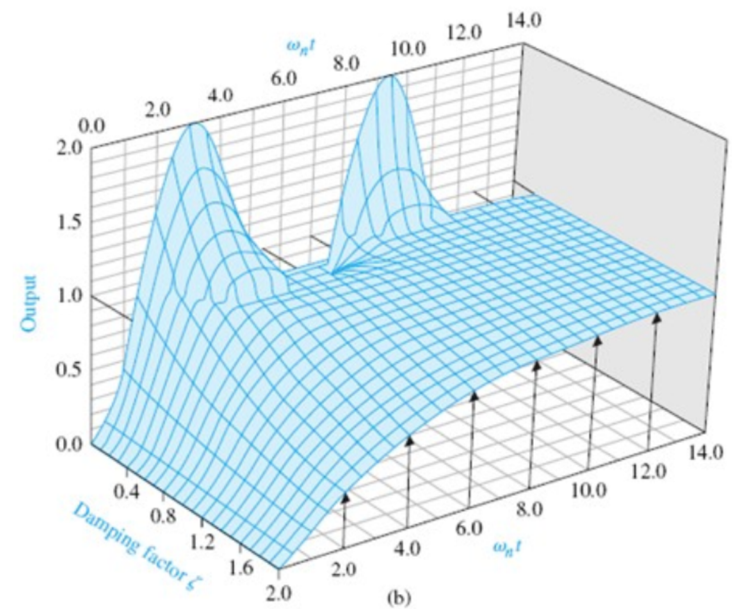
2nd order system

https://www.myphysicslab.com/springs/single-spring-en.html

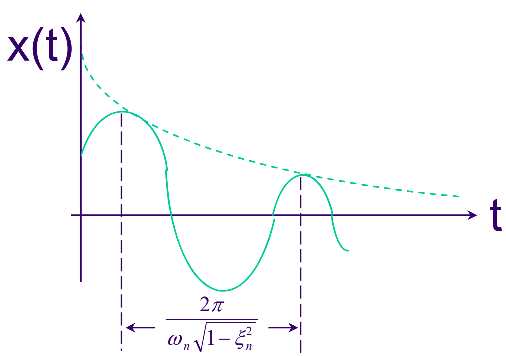
$$\ddot{x} + \frac{b}{m}\dot{x} + \frac{k}{m}x = 0$$

$$\ddot{x} + 2\xi_n\omega_n\dot{x} + \omega_n^2 x = 0$$

Natural frequency $\omega_n = \sqrt{\frac{k}{m}}$; $\xi_n = \frac{b}{2\sqrt{km}}$ Natural damping ratio



$$x(t) = ce^{-\xi_n\omega_n t}\cos(\underbrace{\omega_n\sqrt{1-\xi_n^2}}_{\omega}\,t + \phi)$$



damped Natural frequency

$$\omega = \omega_n\sqrt{1-\xi_n^2}$$

Higher **ω** : faster response

Higher **ζ**: slower response

ζ = 1 : optimal choice

2nd order systems are Practical

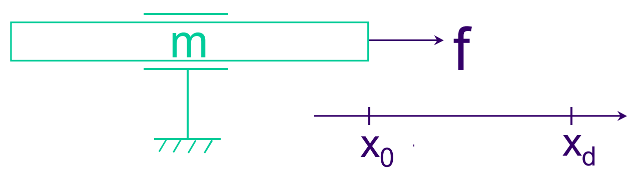
Cool animation with multiple spring-dampers:

https://codepen.io/dissimulate/pen/KrAwx

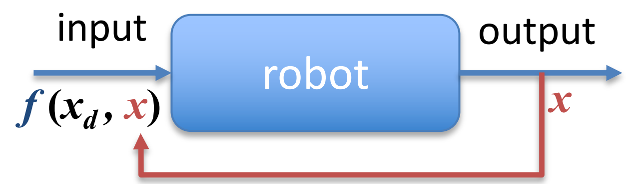
Control >>> Design how your system behave.

Use **Control** to make robots behave like a **dissipative** system

1-dof Robot Control : goal of controller is to move robot to x_d



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



$$m\ddot{x} = f = -k_p(x - x_d) - k_v\dot{x}$$

$$m\ddot{x} + k_v\dot{x} + k_p(x - x_d) = 0$$

Velocity gain (k_v), Position gain (k_p)

PD – control

Proportional + Derivative

$$\ddot{x} + \frac{k_v}{m}\dot{x} + \frac{k_p}{m}(x - x_d) = 0$$

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

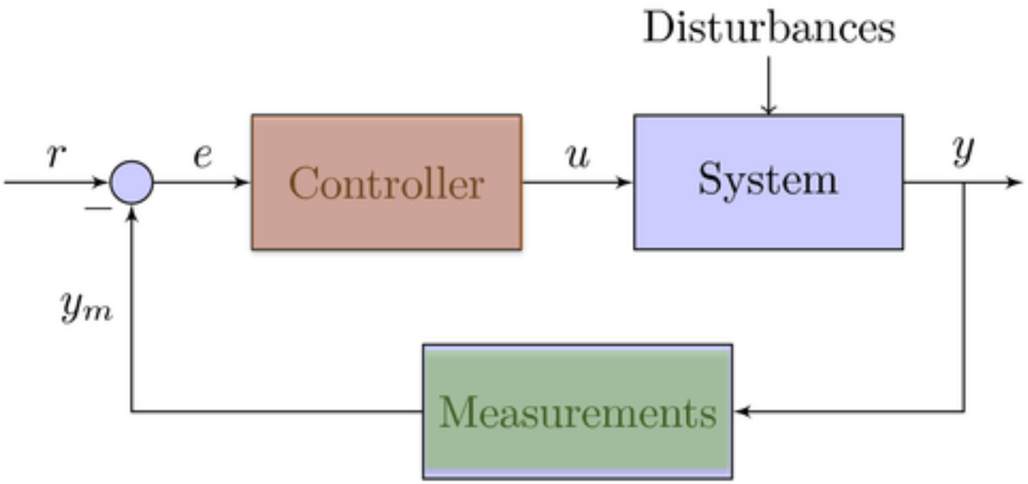
Change behavior with k_p, k_d

$\xi = \dfrac{k_v}{2\sqrt{k_p m}}$ closed loop damping ratio

$\omega = \sqrt{\dfrac{k_p}{m}}$ closed loop frequency

Control Systems

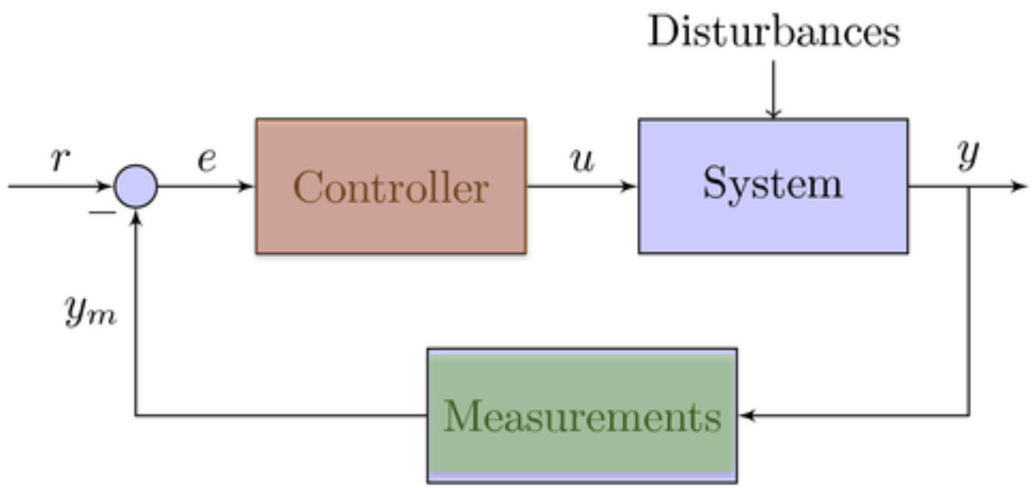
- **Overall block diagram**
  - **System (+actuator)**
  - **Feedback (sensors)**
  - **Controller (design)**



- **Control Methods:**
  - **Model-based**
    - **Requires an approximate (linear / nonlinear) model of system.**
    - **Uncertainties (un-modeled dynamics) are modeled as disturbance**
  - **Model-free**
    - **Learning-based approaches, e.g. RL , adaptive, etc.**

PID control

Proportional + **I**ntegral + **D**erivative

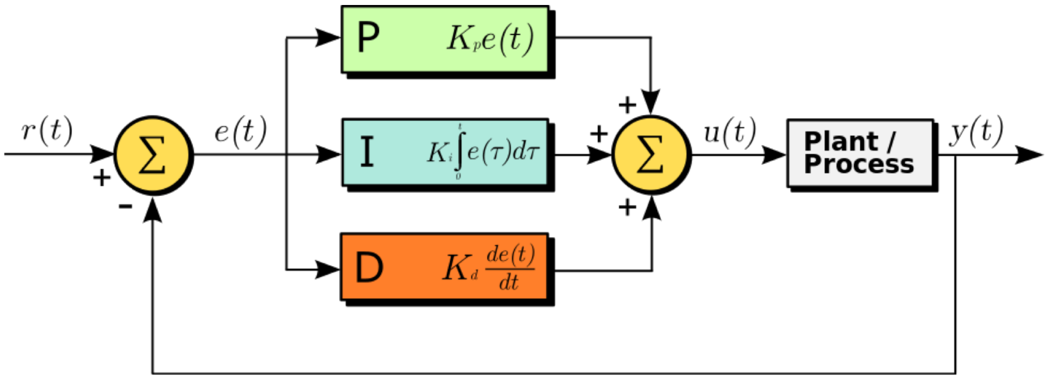


- **PID:**
  - **simple, easy to implement, practical, model-free**
  - **need heuristics for tuning the gains**
- **P**roportional + **I**ntegral + **D**erivative Control

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \dot{e}(t)$$

PID control

Proportional + **I**ntegral + **D**erivative



- **P**roportional + **I**ntegral + **D**erivative Control

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d\, \dot{e}(t)$$

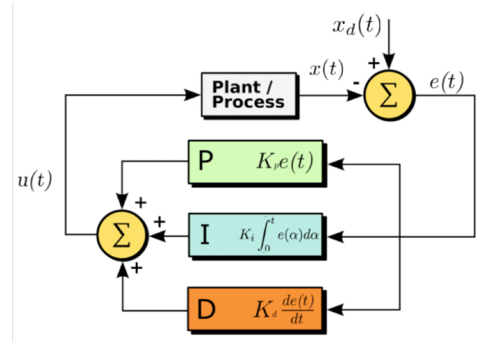
  - **P** term is proportional to the current value
  - **I** term accounts for past values
  - **D** term is a best estimate of the future trend

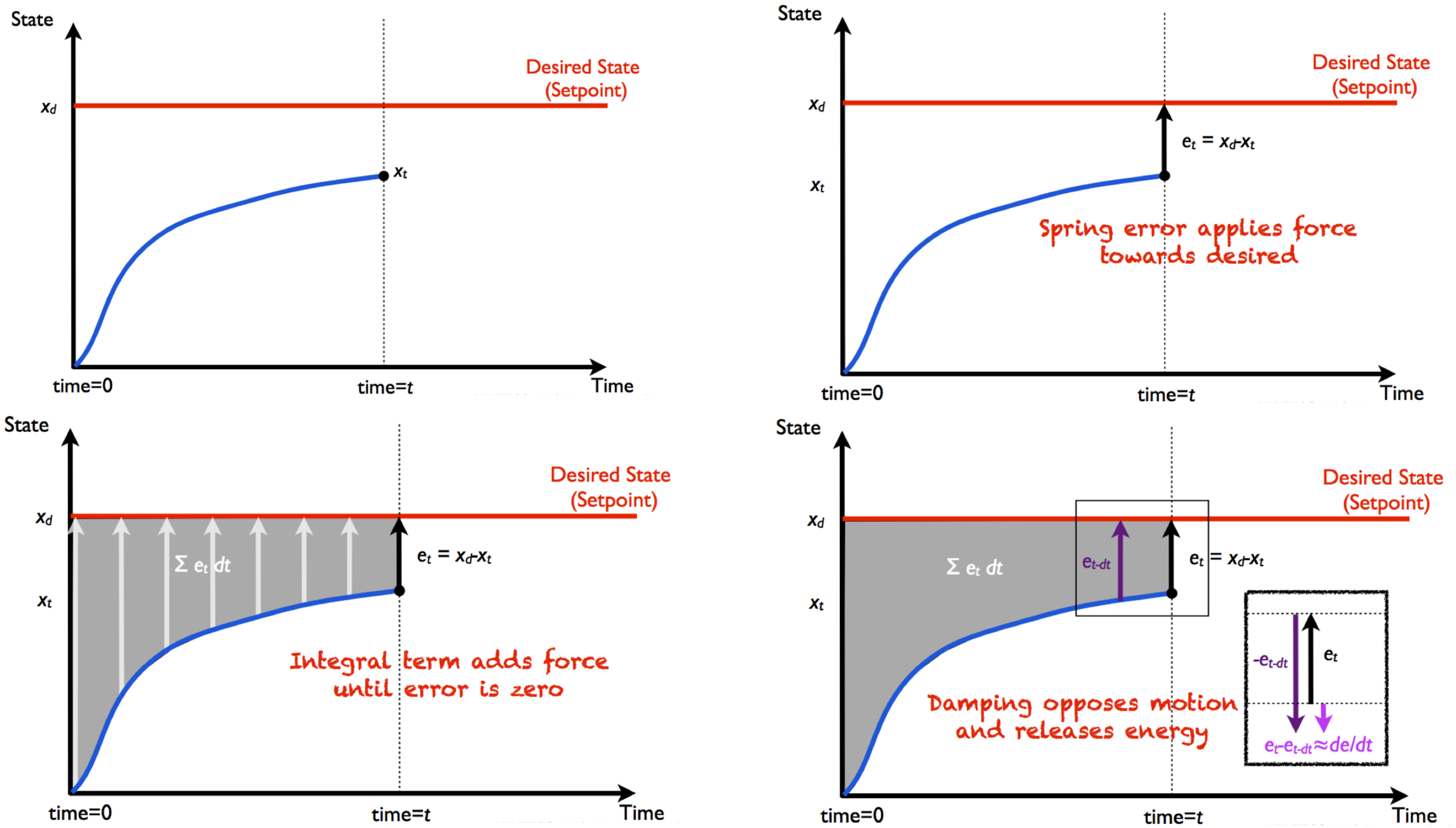
  - Based on mass-spring-damper system

PID control

$$e(t) = x_d(t) - x(t)$$

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d\, \dot{e}(t)$$





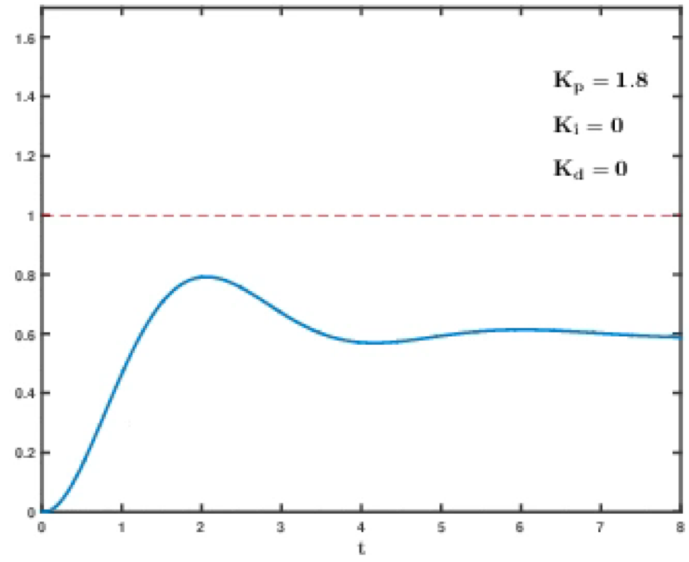
PID control

- **P**roportional + **I**ntegral + **D**erivative Control

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \dot{e}(t)$$

Effects of *increasing* a parameter independently[20][21]

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small



[wikipedia: PID control]

PID: digital implementation

- Discrete integral $$\int_0^t e(\tau)d\tau = \sum_{i=1}^{k} e(t_i)\Delta T$$
- Derivative (backward finite difference) $$\dot{e}(t) = \frac{e_{t_k} - e_{t_{k-1}}}{\Delta T}$$

Pseudocode

```
previous_error = 0
integral = 0
loop:
  error = setpoint - measured_value
  integral = integral + error*dt
  derivative = (error - previous_error)/dt
  output = Kp*error + Ki*integral + Kd*derivative
  previous_error = error
  wait(dt)
  goto loop
```

PID gain tuning

- Start will all gains at zero: $K_p=K_d=K_i=0$
- Increase K_p until system roughly meets desirable state
  - Overshoot & oscillation are acceptable
- Increase K_d until system is stable
- Increase K_i until system consistently reaches x_d
- Refine gains to improve performance

- Will test this in lab 2, for Lego actuators

Tracking Control

What if we want to track a trajectory, i.e. x_d is changing with time: $x_d(t)$

$x_d(t)$; $\dot{x}_d(t)$; and $\ddot{x}_d(t)$

$$m\ddot{x} = f \;\Rightarrow \ddot{x} = \frac{f}{m} = f'$$

$$f' = \ddot{x}_d - k_v(\dot{x} - \dot{x}_d) - k_p(x - x_d)$$

Control: $f' = \ddot{x}_d - k_v'(\dot{x} - \dot{x}_d) - k_p'(x - x_d)$

Closed-loop System:

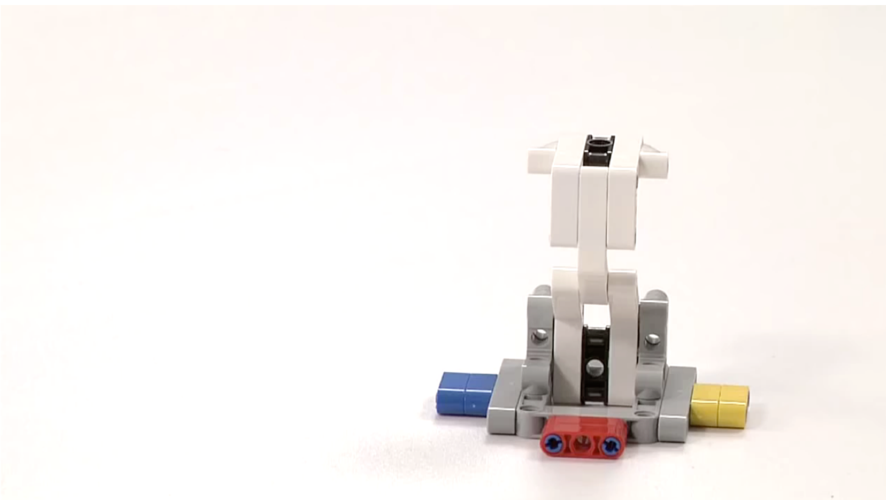
$$(\ddot{x} - \ddot{x}_d) + k_v'(\dot{x} - \dot{x}_d) + k_p'(x - x_d) = 0$$

with $e \equiv x - x_d$

$$\ddot{e} + k_v'\dot{e} + k_p'e = 0$$

Motivation

- Control >> Design how your mechatronic system behave.



LEGO Mindstorms
EV3Bike
Powered by leJOS & Java

Control of Nonlinear Systems

Non Linearities

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

Control Partitioning

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

with

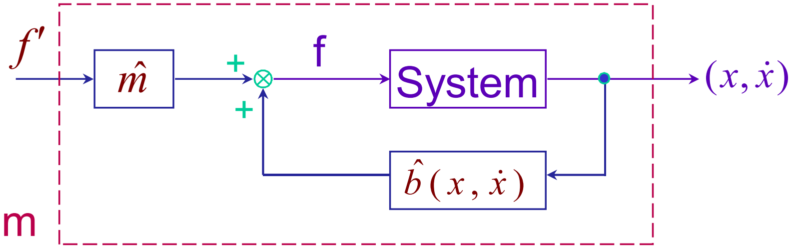
$$\alpha = \hat{m}$$

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

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

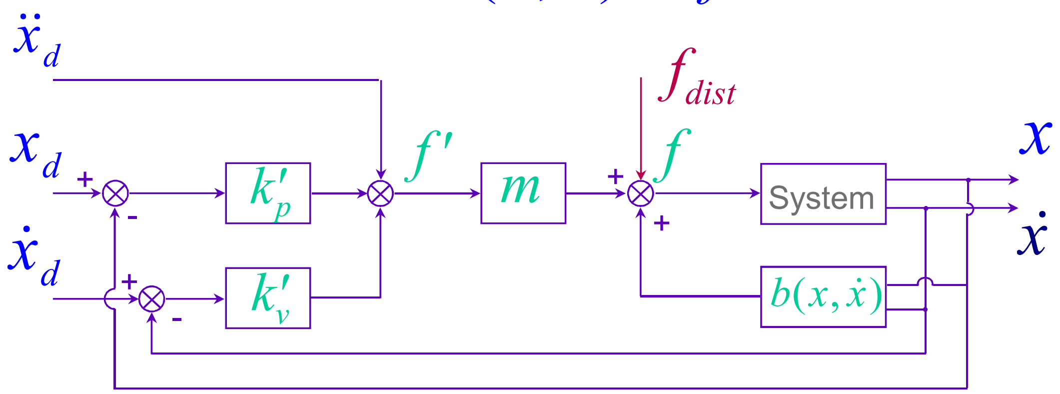
$$\longrightarrow \quad 1.\ddot{x} = f'$$

Unit mass system



Disturbance Rejection

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



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

Control $f = mf' + b(x, \dot{x})$

Closed loop $$\ddot{e} + k'_v\dot{e} + k'_p e = \frac{f_{dist}}{m}$$

Steady-State Error

$$\ddot{e} + k_v'\dot{e} + k_p'e = \frac{f_{dist}}{m}$$

The steady-state $\left(\dot{e} = \ddot{e} = 0\right)$:

$$k_p'e = \frac{f_{dist}}{m}$$

$$e = \frac{f_{dist}}{mk_p'} = \frac{f_{dist}}{k_p}$$

Closed loop position gain (stiffness)

Practical issues for choosing Controller Gains

Performance

High Gains → better disturbance rejection

Gains are limited by
- structural flexibilities
- time delays (actuator-sensing)
- sampling rate

$$\omega_n \le \frac{\omega_{res}}{2}$$ ← lowest structural flexibility

$$\omega_n \le \frac{\omega_{delay}}{3}$$ ← largest delay $\left(\frac{2\pi}{\tau_{delay}}\right)$

$$\omega_n \le \frac{\omega_{sampling-rate}}{5}$$

Tacoma Narrows Bridge (1940)

Rule of thumb:
ω = (3 to 10 Hz)*2π
ζ = 1