PID CONTROL

Roberto Martin-Martin

Assistant Professor of Computer Science.

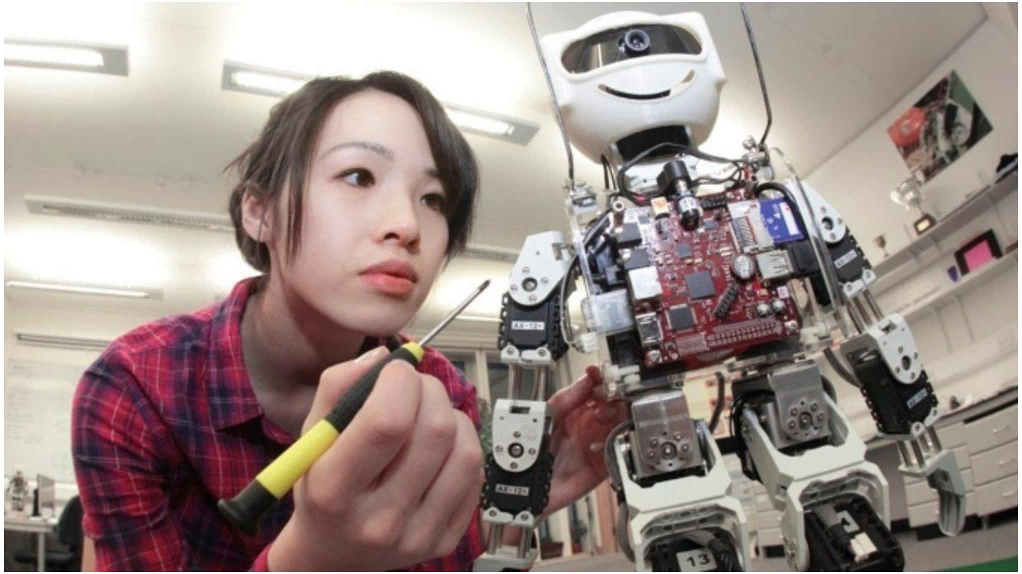
Slides based on Peter Stone’s and Ben Kuipers’

Big Recap – How to Build our Robots?

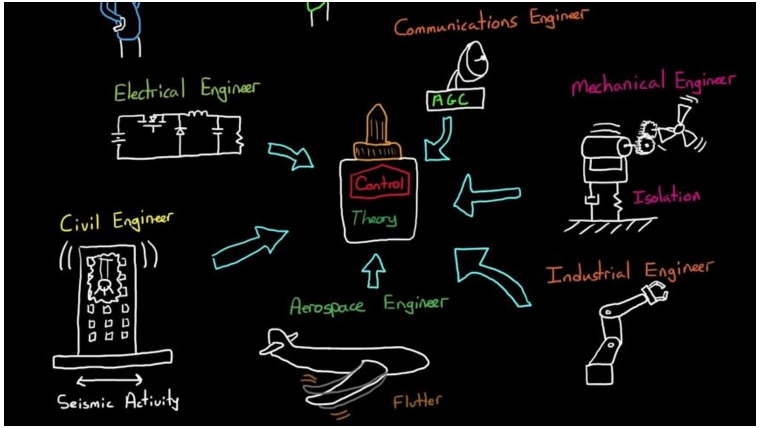
- Statics, Friction, Grasping
  - Newton, FBD, wrenches, stiction, kinetic friction, viscous friction
- Physics of Materials
  - Physics of deformable bodies, strain-stress diagrams
- Articulations
  - Types of joints, Grueber's Formula
- Analog Electronics
  - Current, Voltage, R, C, L, Kirchoff's laws, power
- Digital Electronics
  - Transistors, gates, truth tables, AND, OR, … Boolean algebra
- State Machines
  - DFA, Deterministic/Stochastic DFAs, Petri Nets, Hybrid Automata
- Mechatronics
  - Types of actuators, motors, torque/speed curve, encoders

Everything to build a robot

But how do we move it?



What will you learn today?



- What is control and what is it for?
- Simple controllers
  - Bang-Bang control
  - Proportional control (P)
  - Integral control (I)
  - Derivative control (D)
- PD Controller behavior (over/under/critically damped)
- Type of controller (feedback vs. feedforward)

What is control and what is it for?

$$\dot{smth} = \frac{\delta smth}{\delta t}$$

- Control deals with commanding a dynamical system, a system that changes its state over time
- With control, we influence those changes, ideally towards our desires
- Control is a mechanism to produce **inputs** to the dynamical system to try to guide its **state** towards a desired state
- Of course, we are assuming that the control has an effect in the state: $\frac{\delta F}{\delta u} \neq 0$

$$\dot{x} = F(x, u)$$

$$y = G(x)$$

$$u = H_i(y)$$

$$\dot{x} = F(x, H_i(G(x)))$$

A very special case: Linear Dynamics

- Many systems in robotics can be assumed to have linear dynamics
- Many others can be assumed to be ”linearizable”
- If the system is linear, we can use matrix algebra:

$$\dot{x} = F(x, u) = Ax + Bu$$

- If the relationship between the state and the observation is also linear, we can do the same here:

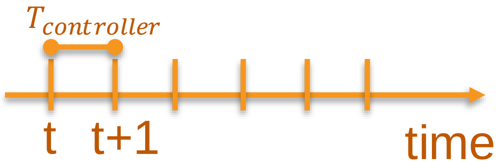
$$y = G(x) = Gx$$

Important: Discrete time system

- Our controller is going to act at discrete time steps
- In most cases in robotics we assume a discrete system because:
  - We have a sensor that provides signals at discrete time steps
    - Camera providing images at N fps
  - We compute the controller response in a computer (discrete time!)
- Controller frequency: $f_{controller} = \frac{1}{T_{controller}}$

$$x_{t+1} = F(x_t, u_t)$$
$$y_t = G(x_t)$$
$$u_t = H_i(y_t)$$

$$x_{t+1} = F(x_t, H_i(G(u_t)))$$

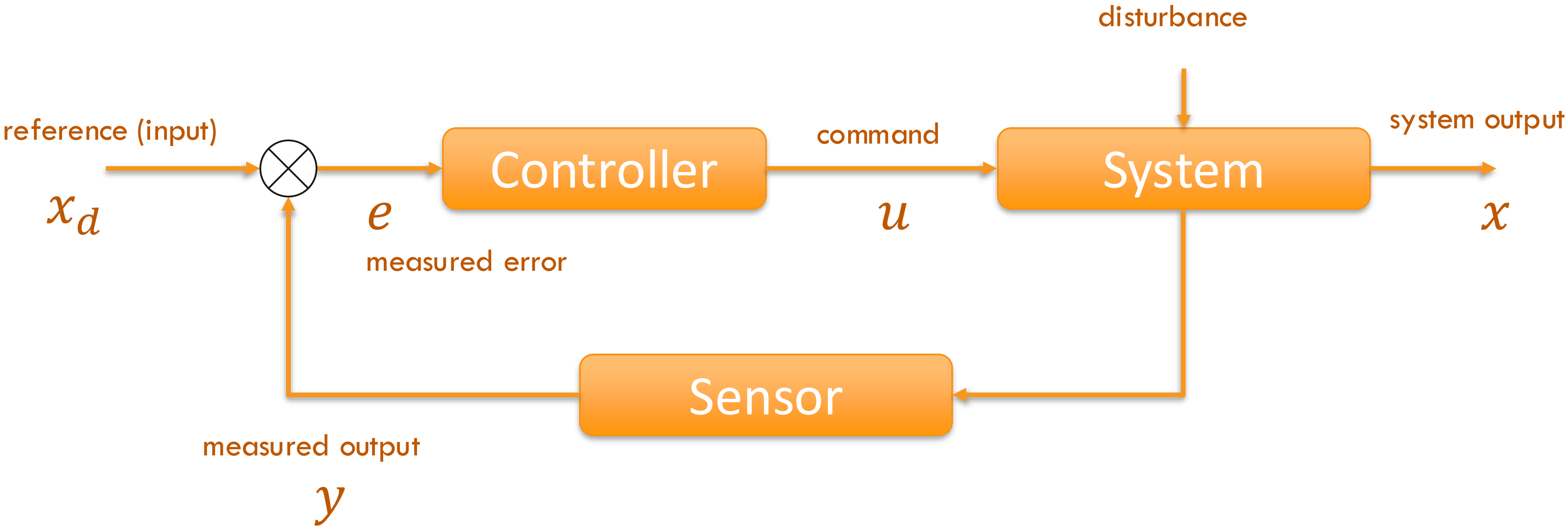


VIP: Linear Dynamics in Discrete-Time Systems

$$x_{t+1} = F(x_t, u_t) = Ax_t + Bu_t$$

$$\mathrm{y}_{\mathrm{t}} = G(x_t) = Gx_t$$

Diagram of the System and Controller



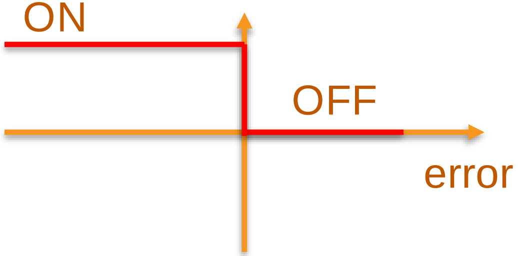
The intuition behind control

- Use action u to "push back" toward error e = 0
  - error e depends on state x (via sensors y)
- What does pushing back do?
  - Depends on the structure of the system: what can we control
  - For example, position vs. velocity vs. acceleration vs. force/torque control
- How much should we "push back"?
  - What does the magnitude of u depend on?
  - This defines different types of control implementation

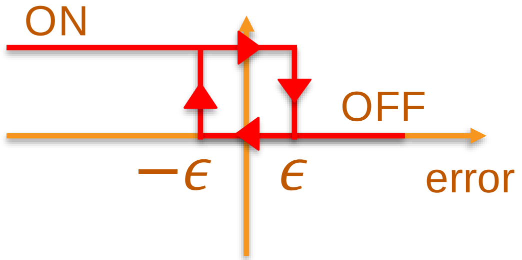
How much should we “push back”?

- How do we set the value of u?
- Some options:
  - Bang-Bang control
  - Proportional control
  - Integral control
  - Derivative control

Bang-Bang Control



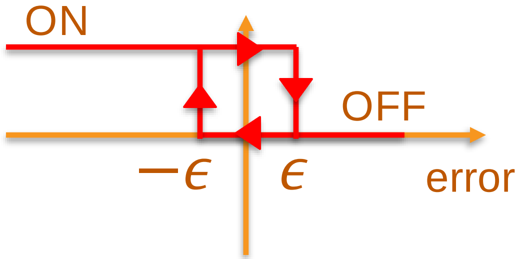
- Push back, against the direction of the error
  - with constant action u
- Error is $e = x - x_d$
  - if $e < 0$ then $u = ON$ → $\dot{x} = F(x,u) > 0$
  - if $e > 0$ then $u = OFF$ → $\dot{x} = F(x,u) < 0$
- Problem: **chatter** around $e = 0$ → constant switch ON/OFF → Bad for the system!
  - We add some hysteresis
    - if $e < -\epsilon$ then $u = ON$ → $\dot{x} = F(x,u) > 0$
    - if $e > +\epsilon$ then $u = OFF$ → $\dot{x} = F(x,u) < 0$
    - If $-\epsilon < e < \epsilon$ then u depends on “history” (where do I come from?)



https://pollev.com/robertomartinmartin739

Exercise: Bang-Bang Control

- HVAC system
  - x = temperature
  - $f_{controller} = 2Hz$
  - u = HVAC ON/OFF
    - ON → $x_{t+1} = x_t - 2$
    - OFF → $x_{t+1} = x_t + 2$
  - We observe directly the temperature (y=x)
  - With/without hysteresis
- Starting state: $x_0 = 90F$
- Desired state: $x_d = 75F$



Bang-Bang Control

- Pro: Super simple!
  - Easy computation
  - Easy implementation
- Cons:
  - Does not "converge"
    - It keeps oscillating around the desired value
    - That can be bad for the system
  - Not very efficient:
    - <u>No matter the "magnitude" of e, our response is the same</u>

Proportional Control

- Why not making u proportional to the “amount” of error?
- Proportional control:
  - Push back, proportional to the error

$$u = -k_p e + u_b = -k_p(x - x_d) + u_b$$

  - Set u_b so that $\dot{x} = F(x_d, u_b) = 0$
  - Example: driving. You (the controller) “converge” to pushing the pedal some amount (u_b) so that your velocity is constant at your desired value

https://pollev.com/robertomartinmartin739

Exercise

- We want to control a robot arm holding an object
- We control the velocity of the robot's joint
- The state of the robot is given by its joint position and velocity
- Because of the weight of the object, the arm drops at 1 rad/s
- q is the vector of joint values
- Frequency of the system is 1Hz

Proportional Control

- We make our control proportional to the error:

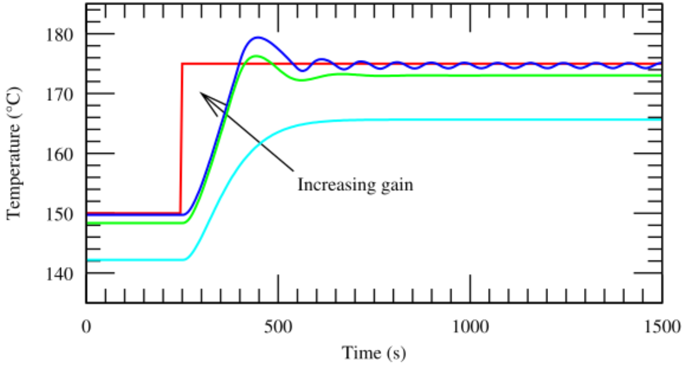
$$u = -k_p e + u_b$$

- For a(ny) linear system, we get exponential convergence

which with our controller leads to

$$x(t) = Ce^{-\alpha t} + x_d$$

$\alpha = k_p b - a$ (if SISO system all real numbers)



Derivation (we will call it k_p instead of k_1 and we call it x_d instead of x_{set}):

$$e = x - x_{set}$$

$$u = -k_1 e + u_b$$

Control is proportional to the error!

$$\begin{aligned}\dot{x} = ax + bu &= ax + b(-k_1 e + u_b)\\ &= ax + b(-k_1(x - x_{set}) + u_b)\\ &= -(k_1 b - a)x + (k_1 x_{set} + u_b)b\\ &= -\alpha x + \beta\end{aligned}$$

$$x = Ce^{-\alpha t} + \frac{\beta}{\alpha}$$

To ensure that $x(\infty) = x_{set}$, we must set $u_b = -\frac{a}{b} x_{set}$.

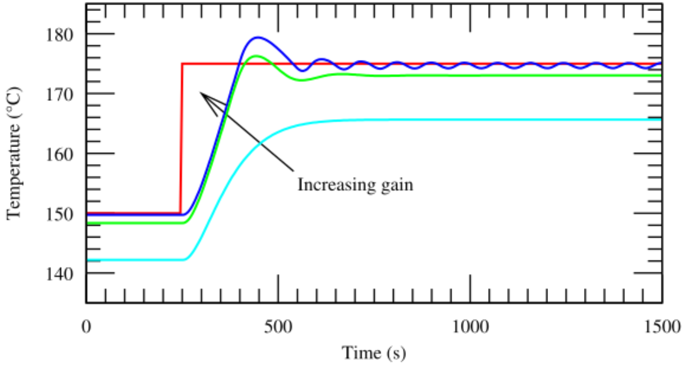
$$x(t) = Ce^{-(k_1 b - a)t} + \frac{(k_1 x_{set} + u_b)b}{k_1 b - a}.$$

Proportional Control

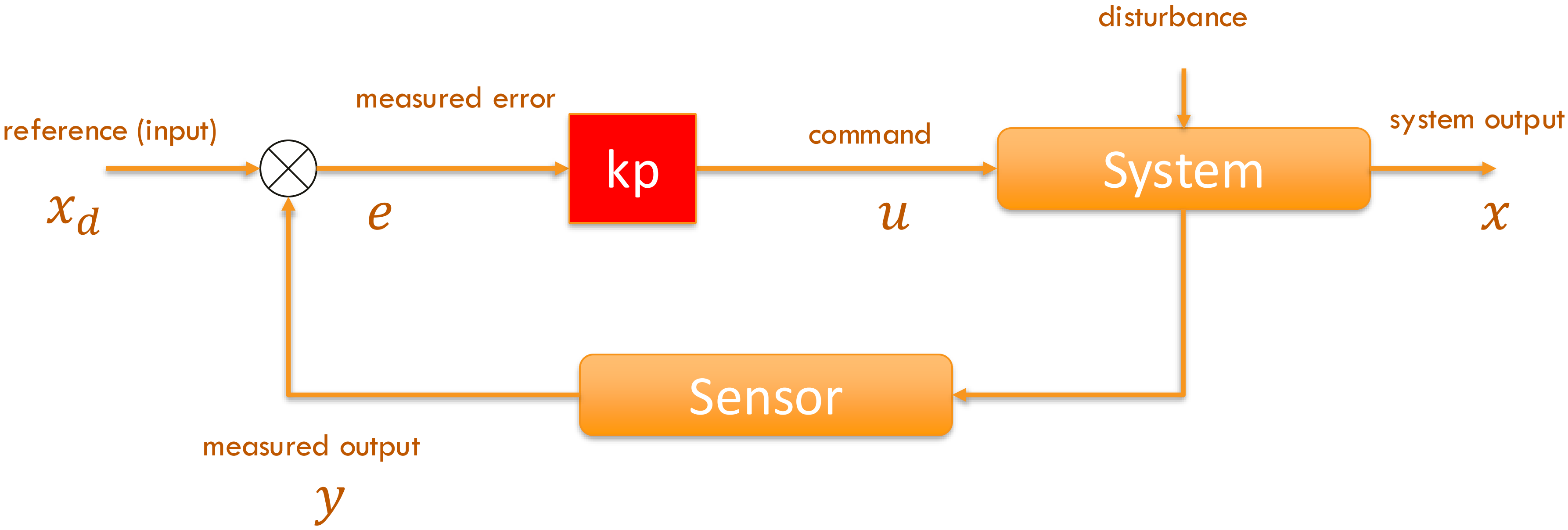
- For a linear system, we get exponential convergence

$$x(t) = Ce^{-\alpha t} + x_d$$

- The controller gain k_p determines how quickly the system responds to error



Proportional Control: Diagram



Proportional Control: Issues

- Relatively simple control but
  - May have a steady-state error if the system has disturbances
  - Depending on k_p may not reach the desired configuration, need a very long time, or overshoot and oscillate (VIP: different behaviors of a controller)

Steady-State Error

- Cause: Disturbances
  - What are disturbances? Unmodeled dynamics

$$\dot{x} = F(x, u) + d$$

- Our controller was designed to lead to $\dot{x} = 0$ at the desired point ($\mathrm{e} = 0$), but due the disturbances, at the desired point:

$$\dot{x} = F(x_d, u_b) + d = d \neq 0$$

- We need to adapt u_b to different disturbances d

Solution: Adaptive control

- We need controllers at different time scales

$$u = -k_p e + u_{bv}$$

$$\dot{u}_{bv} = f(e) = -k_I e$$

$$with\ k_I \ll k_p$$

- This eliminates the steady-state error because the slower controller adapts u_{bv}

Proportional-Integral Control

- The adaptive controller $\dot{u}_b = -k_I e$ means that:

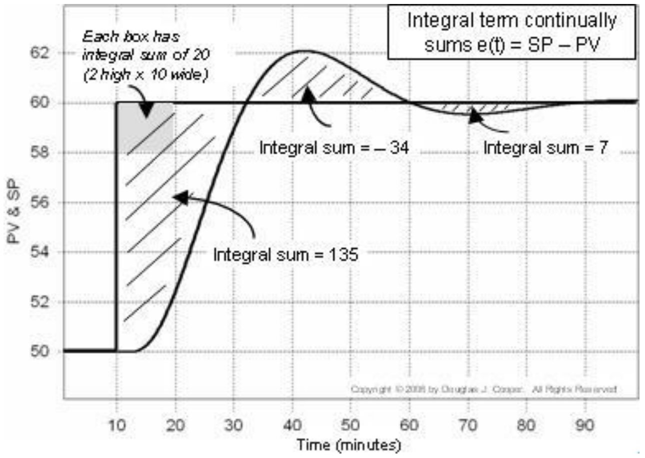
$$u_{bv} = -k_I \int_0^t e \, dt + u_b$$

- And then the control law is:

$$u = -k_p e - k_I \int_0^t e \, dt + u_b$$

- This is called PI control

Examples



When do we use this in robotics

To overcome stiction!

- robot controlled by torques in motors

$u = k_p e = \tau$

- e is small → $u = k_p e = \tau$ small → not enough to overcome stiction ($\tau < \tau_{stiction}$)
- steady-state error

Proportional Control

- Why not making u proportional to the “amount” of error?
- Proportional control:
  - Push back, proportional to the error

$$u = -k_p e + u_b = -k_p(x - x_d) + u_b$$

  - Set u_b so that $\dot{x} = F(x_d, u_b) = 0$
  - Example: driving. You (the controller) “converge” to pushing the pedal some amount (u_b) so that your velocity is constant at your desired value

Steady-State Error

- Cause: Disturbances
  - What are disturbances? Unmodeled dynamics

$$\dot{x} = F(x, u) + d$$

- Our controller was designed to lead to $\dot{x} = 0$ at the desired point ($\mathrm{e} = 0$), but due the disturbances, at the desired point:

$$\dot{x} = F(x_d, u_b) + d = d \neq 0$$

- We need to adapt u_b to different disturbances d

Proportional-Integral Control

- The adaptive controller $\dot{u}_b = -k_I e$ means that:

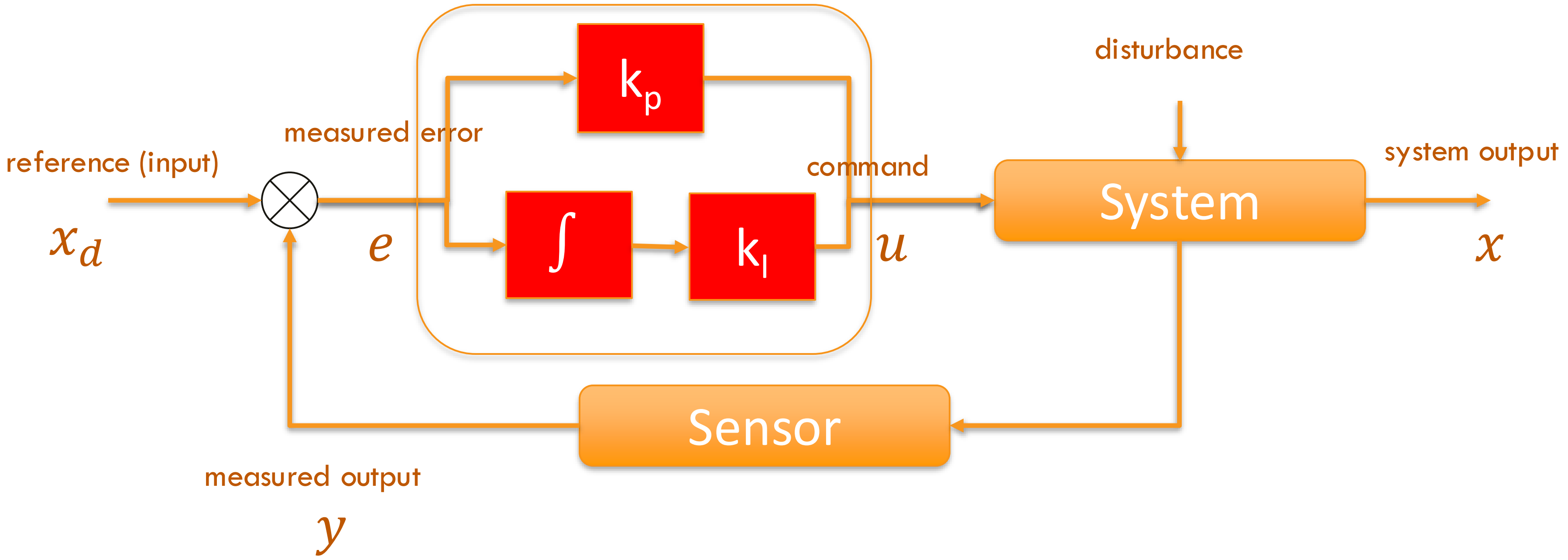
$$u_{bv} = -k_I \int_0^t e\, dt + u_b$$

- And then the control law is:

$$u = -k_p e - k_I \int_0^t e\, dt + u_b$$

- This is called PI control

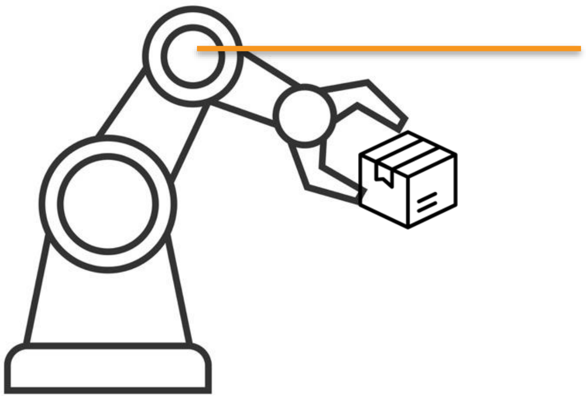
Proportional-Integral Control: Diagram



PollEv.com/robertomartinmartin739

Exercise

- Robot joint trying to reach a desired configuration $q_d = 0$
- Holding something in the hand → disturbance!
- Mass of arm: $m_{arm} = 20kg$
- COM of arm: 50cm from joint
- Mass of box: $m_{box} = 5kg$
- COM of box: 70cm from joint
- Controlled by torque
- Range $[-\pi, \pi]$



Derivative Control

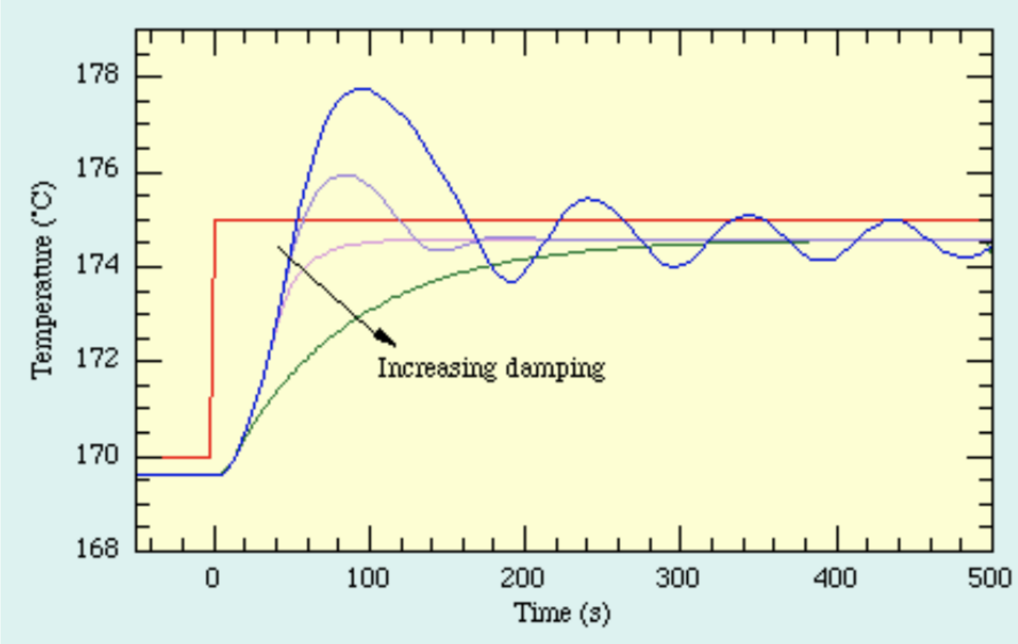
- Remember viscous (damping) friction?
  - Force opposing motion, proportional to velocity
  - $F_{viscous} = -\mu_{viscous} v$
- We are going to include a term in our controller that acts as viscous friction (inverse and proportional to the velocity of the error)

$$u = -k_p e - k_d \dot{e}$$

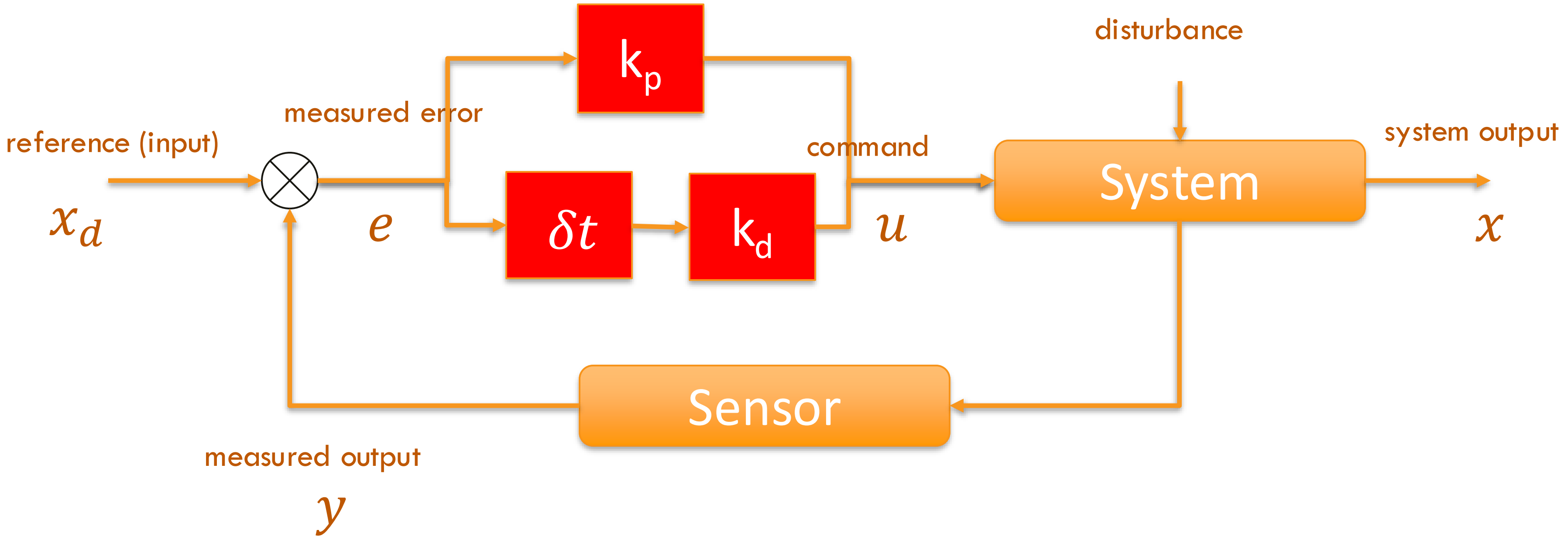
- [Practical issue: computing derivative from measurements can be fragile and amplify noise!]

Derivative Control in Action

Damping fights oscillation and overshoot



Proportional-Derivative Control: Diagram



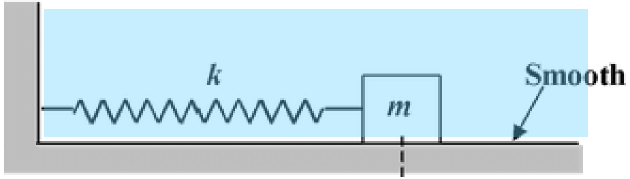
Study: PD-Control and the Mass-Spring-Damper Model

- A controller creates a behavior that resembles the behavior of a mass attached to a spring, immersed in some fluid
- Mass: m
- Fluid viscous friction: μ_{vf}

$$F_{vf} = -F_N \mu_{vf} v = -k_{vf}\dot{x}$$

- Spring constant: k_{spring}
  - Hooke's law: $F_{spring} = -k_{spring}(x - x_{eq}) = -k_{spring}x$
- 2nd Newton's law:

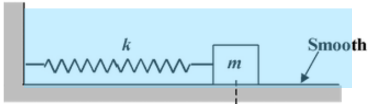
$$\Sigma F = m \cdot a = \mathrm{m}\ddot{x} = -k_{spring}x - k_{vf}\dot{x}$$



Study: PD-Control and the Mass-Spring-Damper Model

- Rearranging and renaming:

$$\ddot{x} + b\dot{x} + cx = 0$$
$$b = \frac{k_{vf}}{m}$$
$$c = \frac{k_{spring}}{m}$$



- Simple 2nd order differential equation. Solution of the form:

$$x(t) = Ae^{r_1 t} + Be^{r_2 t}$$

with r_1 and r_2 the roots of the equation $\ddot{x} + b\dot{x} + cx = 0$

$$r_1, r_2 = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

Study: PD-Control and the Mass-Spring-Damper Model

- The behavior of the system+controller depends on the nature of the roots r_1 and r_2, which depend on the parameters b and c (k and k_{vf})

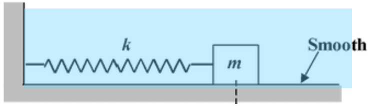
- The system converges to x=0 if its velocity at that point is also 0: $(x, \dot{x}) = (0,0)$

- For the system to converge, both roots must have negative real parts. This requires both c>0 and b>0
  - c>0 → k_{spring}>0, Hooke's law "in the normal way".
    - If c<0 (k_{spring}<0) it would work as an anti-spring: increasing force in the same direction than the displacement, pushing the mass to the infinite
  - b>0 → k_{vf}>0, Damping "in the normal way": opposing velocity.
    - If b<0 (k_{vf}<0) the damping would increase velocity in the same direction of motion, making the system to diverge

Study: PD-Control and the Mass-Spring-Damper Model

$$r_1, r_2 = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

- c>0, b>0. What is the behavior of the system?
  - Depends on the discriminant $D = b^2 - 4c$
  - b small (k_{vf} small) → $D < 0$ → System is underdamped
    - System oscillates with decreasing amplitude
    - It may converge, depending if r_1 and r_2 imaginary parts
  - b large (k_{vf} large) → $D > 0$ → System is overdamped
    - System moves slowly towards convergence, eventually reaching it
  - $D = b^2 - 4c = 0$ → $b = 2\sqrt{c}$. System is critically damped:
    - Moves as quickly as possible to the resting (desired) position without overshooting



- To tune a PD controller to the optimal behavior, we model it as a spring-mass damped system and find the parameters so that the system is critically damped

Study: PD-Control and the Mass-Spring-Damper Model

- Some extra definitions:

$$\ddot{x} + b\dot{x} + cx = 0$$

$$b = \frac{k_{vf}}{m}$$

$$c = \frac{k_{spring}}{m}$$

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0$$

$$\omega_0 = \sqrt{\frac{k_{spring}}{m}} = \sqrt{c}$$

Natural (undamped) frequency

$$\zeta = \frac{k_{vf}}{2\sqrt{mk_{spring}}} = \frac{b}{2\sqrt{c}}$$

Damping ratio (unitles

Study: PD-Control and the Mass-Spring-Damper Model

- Some extra definitions:

$$\omega_0 = \sqrt{\frac{k_{spring}}{m}}$$

$$\zeta = \frac{k_{vf}}{2\sqrt{mk_{spring}}}$$

$$b = 2\zeta\omega_0$$

$$c = \omega_0^2$$

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0$$

$$D = b^2 - 4c = \frac{\mathrm{k}_{\mathrm{vf}}^2}{\mathrm{m}^2} - \frac{4\mathrm{k}_{\mathrm{spring}}}{m}$$

$$D = 0 \rightarrow \frac{\mathrm{k}_{\mathrm{vf}}^2}{\mathrm{m}^2} = \frac{4\mathrm{k}_{\mathrm{spring}}}{m} \rightarrow \frac{\mathrm{k}_{\mathrm{vf}}^2}{4mk_{spring}} = 1 \rightarrow \frac{k_{vf}}{2\sqrt{mk_{spring}}} = 1$$

- Critically damped: ζ=1
- Overdamped: ζ >1
- Underdamped: ζ <1

How does this look like?

- Critically Damped (D=0, ζ=1)

$$x(t) = (A + Bt)e^{-\omega_0 t}$$

- Overdamped (D>0, ζ>1)

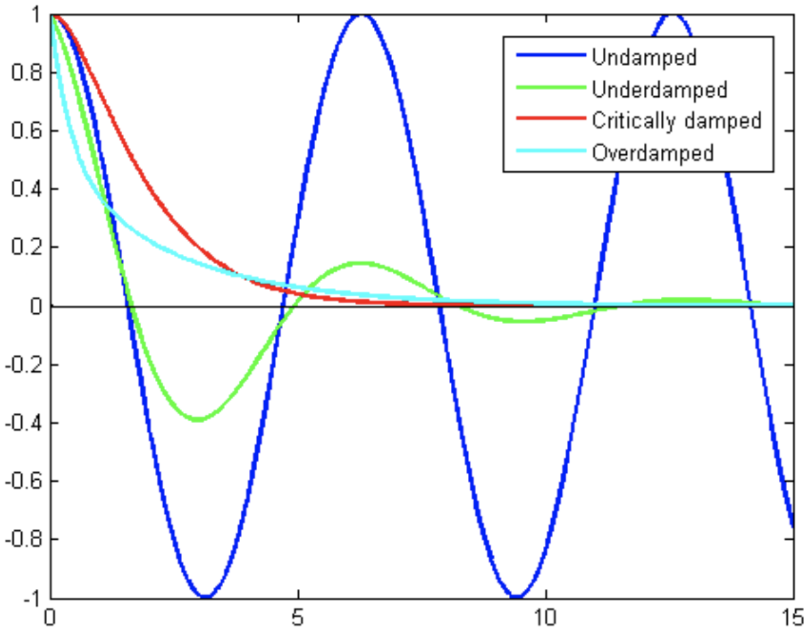
$$x(t) = Ae^{\gamma_+ t} + Be^{\gamma_- t}$$

$$\gamma_\pm = \omega_0(-\zeta \pm \sqrt{\zeta^2 - 1})$$

- Underdamped (D<0, ζ<1)

$$x(t) = e^{-\zeta\omega_0 t}(A\cos(\omega_d t) + B\sin(\omega_d t))$$

$$\omega_d = \omega_0\sqrt{1 - \zeta^2}$$



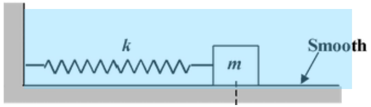
- Critically damped: ζ=1
- Overdamped: ζ >1
- Underdamped: ζ <1

Recap

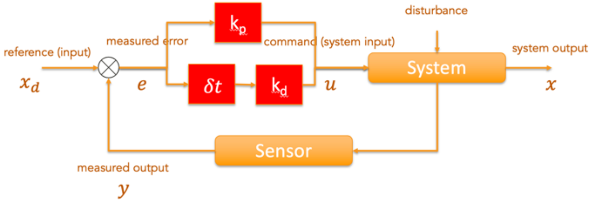
- What is a dynamical system?
  - a system that changes its state (x) over time
    $\dot{x} = F(x, u) \neq 0$ or $x_{t+1} = F(x_t, u_t) \neq 0$
- What is the goal of control?
  - Control is a mechanism to <u>produce inputs</u> (u) to the dynamical system to try to guide its state towards a desired state
  - Our goal is that $e = x - x_d = 0$
- Types of control → What "u" we generate depending on "e"
  - Bang-bang: depending on the sign of "e" we output one of two possible "u"s
  - Proportional: $u = -k_p e + u_b$
  - Proportional-Integral: $u = -k_p e - k_I \int e\, dt + u_b$
  - Proportional-Derivative: $u = -k_p e - k_d \dot{e} + u_b$

Why do we look at a spring-damped mass?

- The behavior of the spring-damped mass is equivalent to the behavior of our controller:
  - In a PD controller, we have control over the <u>stiffness</u> of the spring and the <u>viscosity</u> of the damping
  - With these two values, we can vary the behavior of the system: from oscillating to converging fast and without overshoot, to converging too slow.
  - Our ideal case: critically damped!



≈



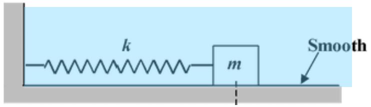
Critically damped: $\zeta = \frac{k_{vf}}{2\sqrt{mk_{spring}}} = 1$

$$k_{vf} = 2\sqrt{mk_{spring}}$$

https://pollev.com/robertomartinmartin739

Exercise

- We have an existing mass-spring-damper system (uncontrollable!)
- Unit mass m=1
- k_{spring} =10000, k_{vf} =10
- What is ω_0, ζ and the behavior of the system?
- We connect the mass-spring-damper system to a PD controller (we control it!)
- What should be the gains of the controller to change the frequency to 200 rad/s and make it critically damped?
  - The gains of the combined system result from adding the natural system and the PD gains



$$\omega_0 = \sqrt{\frac{k_{spring}}{m}}$$

$$\zeta = \frac{k_{vf}}{2\sqrt{m k_{spring}}}$$

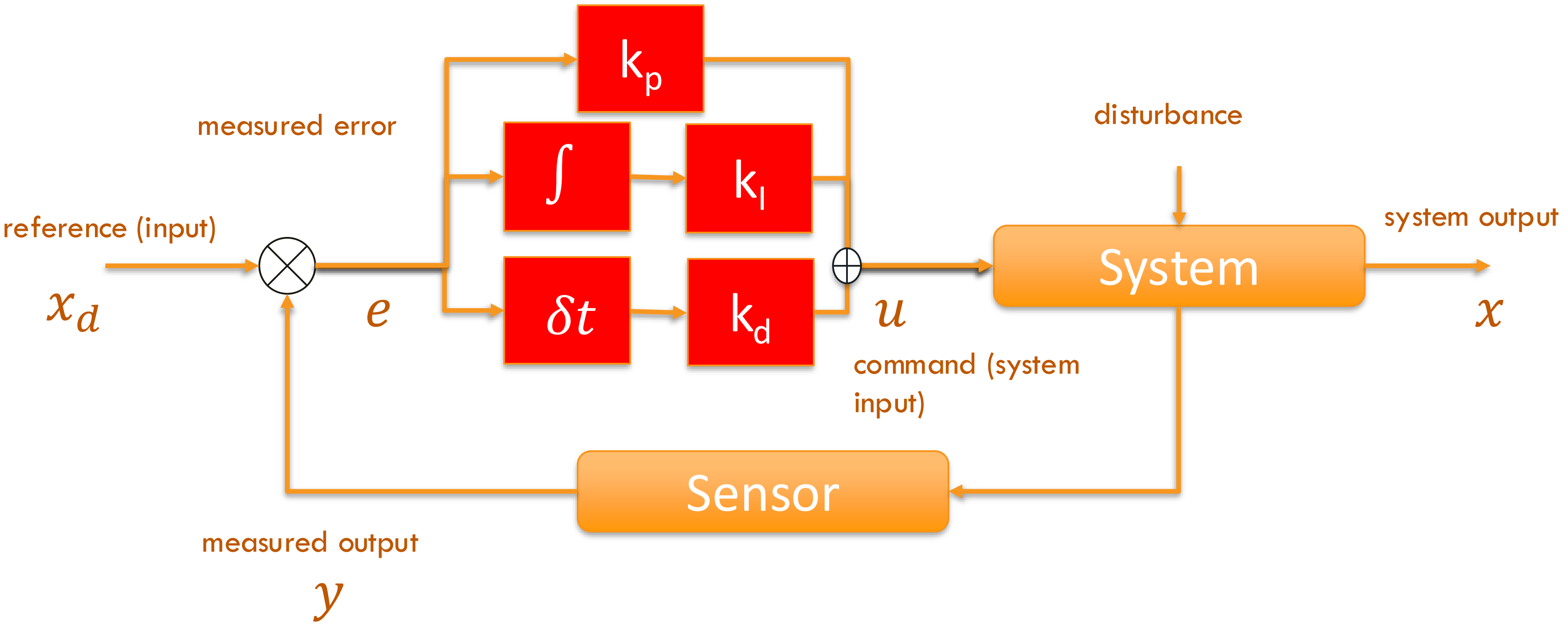
- https://www.matthewpeterkelly.com/tutorials/pdControl/index.html

We can put all together: Proportional-Integral-Derivative (PID) Control

$$u = -k_p e - k_I \int_0^t e \, dt - k_d \dot{x}$$

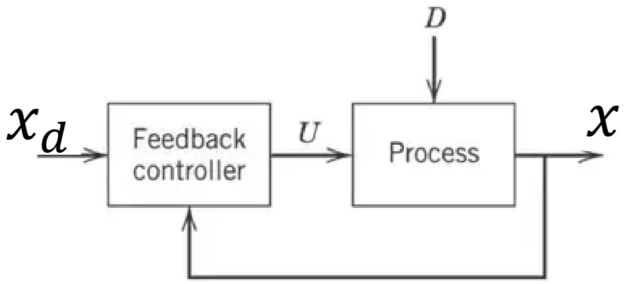
- PD and PID are the most used types of controller everywhere!
  - Industry
  - Research
- Tuning the values of the parameters becomes "an art"
  - Some principled strategies
  - Tends to be trial and error

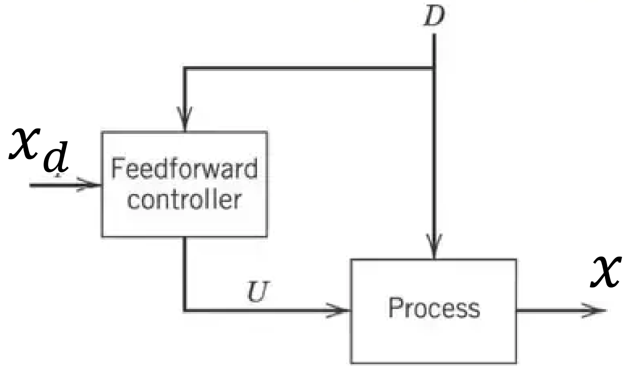
Proportional-Integral-Derivative Control: Diagram



Feedback vs. Feedforward Control

- Feedback control (closed loop):
  - Sense error (due to disturbances), then determine control
  - Reactive
  - Problem: it always goes "behind" the error
- Feedforward control (open loop):
  - Sense directly the disturbance
  - Consider the response from the system (model!) to plan the best signal "u" to compensate the disturbance
  - Problem: Independent of the outcome from the system (does not closes the loop)
- Combined feedforward and feedback
  - We predict the response from the system and compute "u" accordingly but we also compensate for disturbances based on observations





Example of Feedforward

- Autonomous car
- Controls steering
- Feedback controller:
  - If error (for example, distance to middle line) changes, change the steering
  - Reactive (we will move away from the line)
- Feedforward controller:
  - Measure disturbance (for example, the inclination of the road)
  - Compute how much to steer the wheel to compensate for the deviation
  - If well done, we do not need to move away from the line!

Final Recap

- What is control and what is it for?
- Inputs/Outputs
- Types of control: SISO/MIMO
- Open loop vs. closed loop
- State-Space Representation
- Simple controllers
  - Bang-Bang control
  - Proportional control (P)
  - Integral control (I)
  - Derivative control (D)
- PD Controller behavior (over/under/critically damped)
- Type of controller (feedback vs. feedforward)