Real-World Multi-Agent Systems

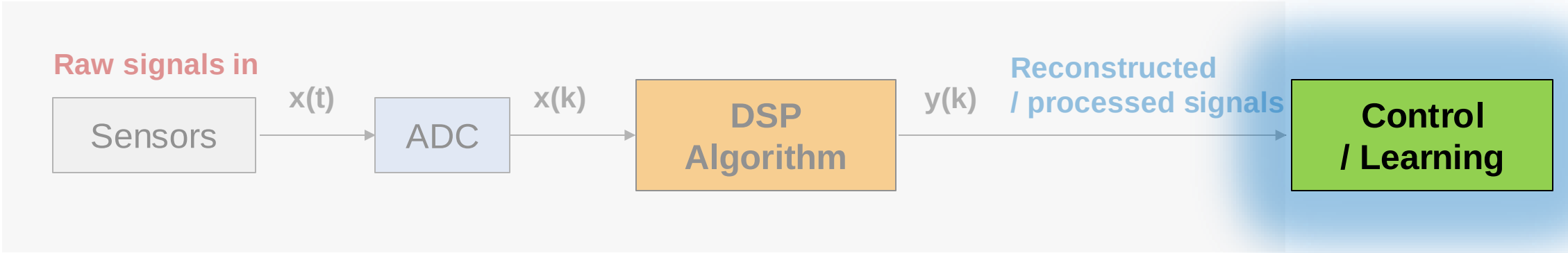
UCL

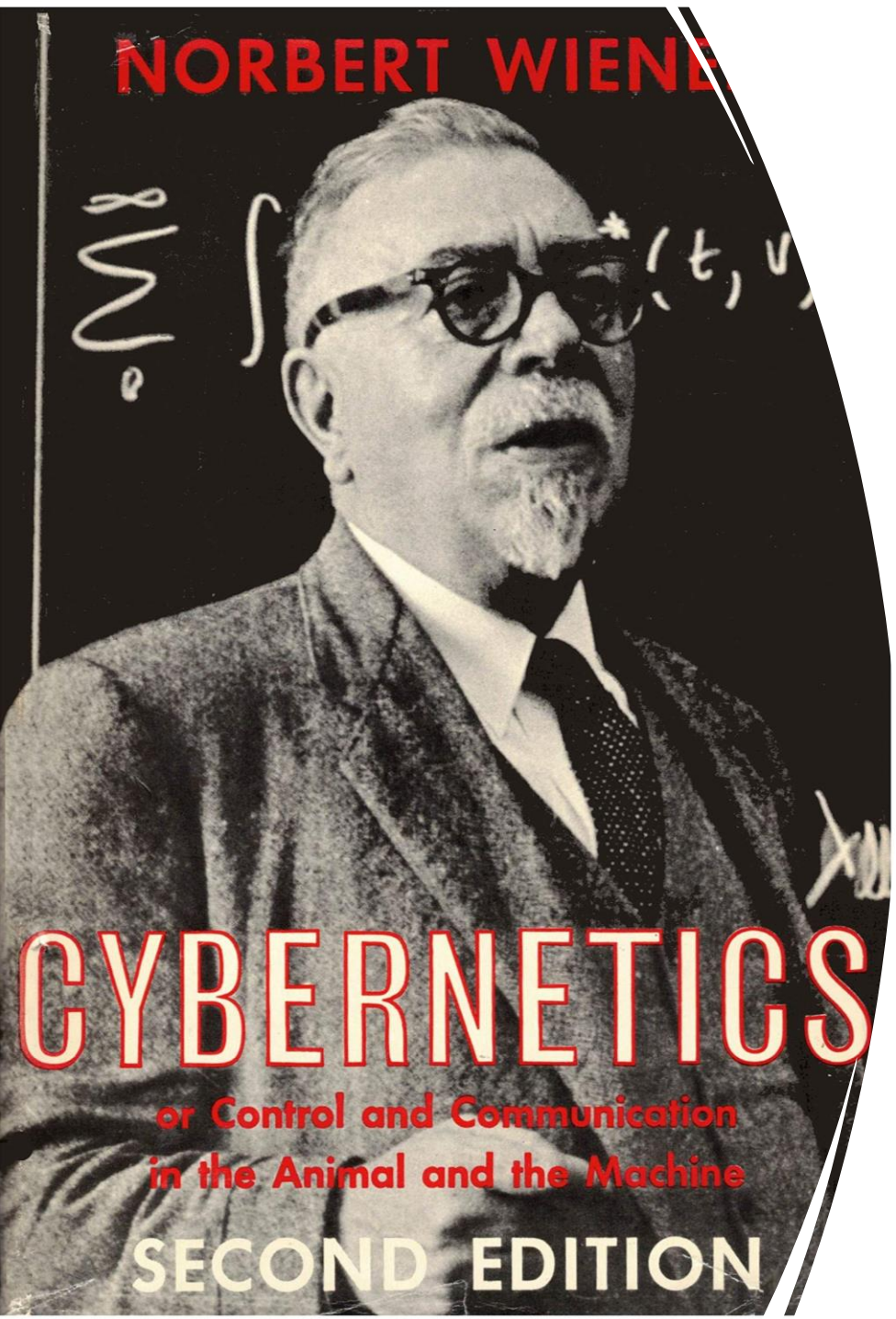
Control of agents

Akin Delibasi
Lecturer in Distributed Systems
Computer Science
University College London, UK

Control of physical agents – robots!

- Background and origin of control theories (Its relationship with AI)
- Concept feedback control
- Common control techniques
- Physics simulation
- Code examples and exercise





Cybernetics

The beginning of the era, and the Dawn of Cybernetics:

- **Cybernetics**: or Control and Communication in the Animal and the Machine, published in 1948, Norbert Wiener, an MIT mathematician.

- Cybernetics: Study of regulatory systems, structural similarities between machines and living beings.

Cybernetics

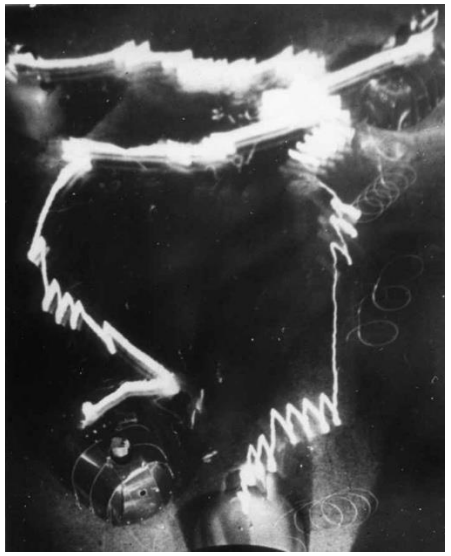
- How Cybernetics Paved the Way for AI:
  - ❑ Shared principles of feedback, adaptation, and autonomous regulation.
  - ❑ Influence on early AI research, particularly in creating adaptive systems.

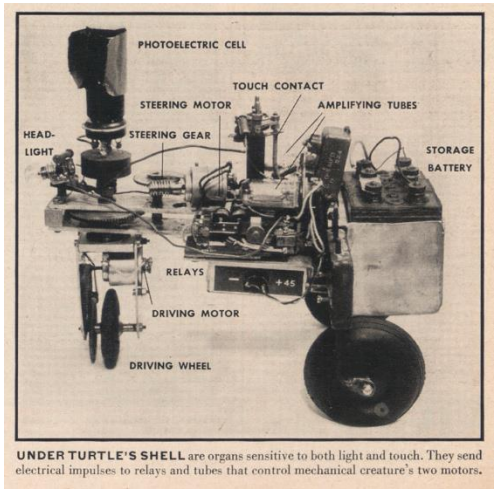
- If you read original book of Cybernetics, you will found out that the description of the “control” or behavior in the book is actually an AI system and the multi-agent systems or artificial lives.

- The fact that Cybernetics evolve into the classical control theory we see today, because there's no such tool or theory as neural networks or any computing power at early stage. Hence, using mathematical equation/formulation is only means to develop automated behaviors.

First Cybernetic Turtles

- William Grey Walter: a neurophysiologist and roboticist who became famous for creating some of the first autonomous robots in the late 1940s.
- These robots, affectionately known as "turtles," were named Elmer and Elsie, which stood for "ELectroMEchanical Robot" and "ELectroSensitive Robot" respectively.
- These robots could simulate simple behaviors that were, until then, seen only in living organisms.





UNDER TURTLE'S SHELL are organs sensitive to both light and touch. They send electrical impulses to relays and tubes that control mechanical creature's two motors.

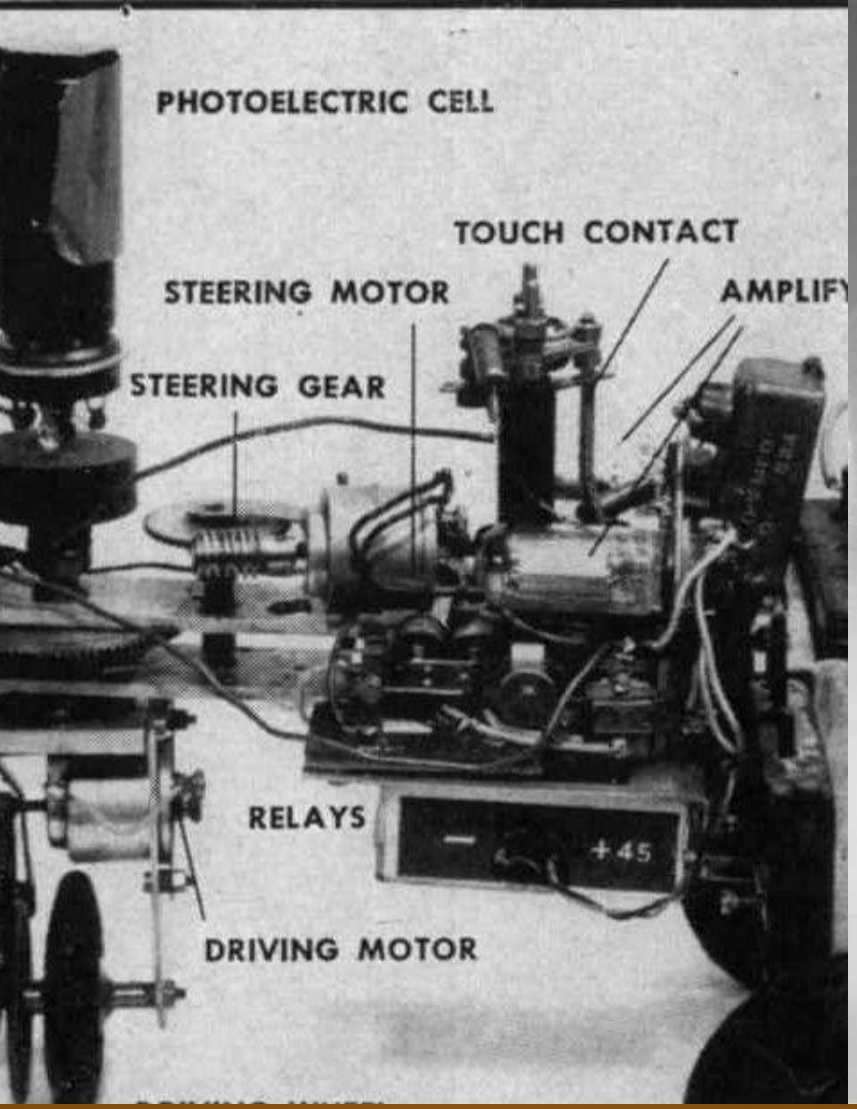
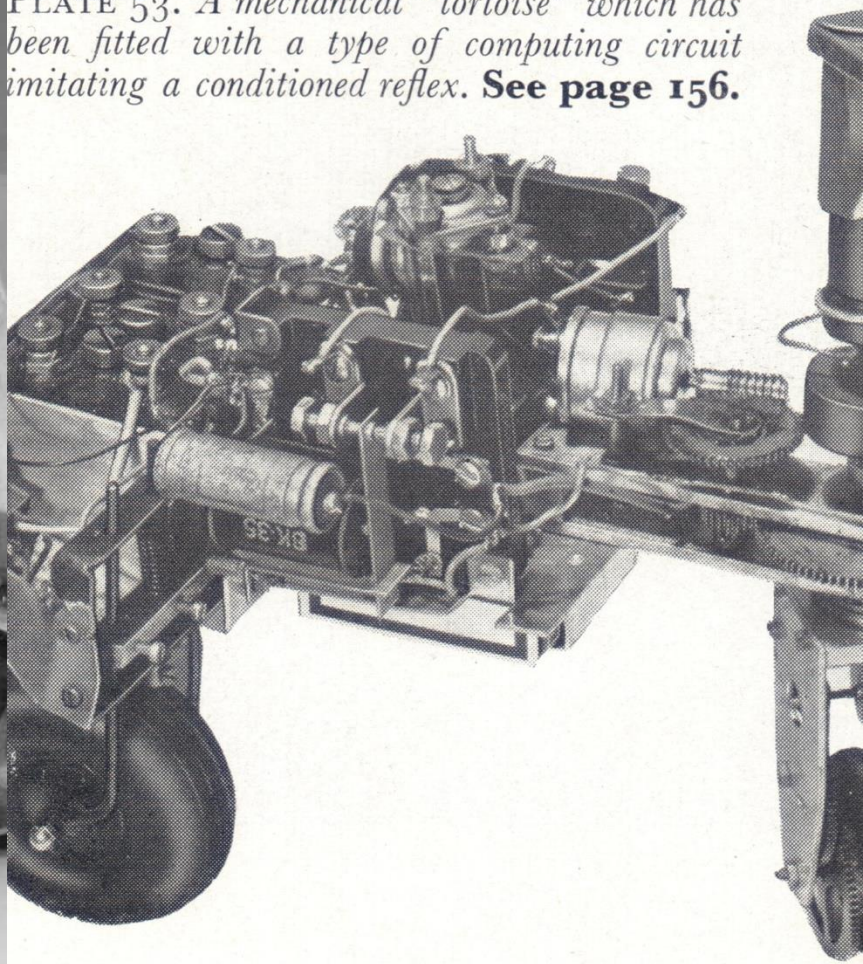


PLATE 53. *A mechanical "tortoise" which has been fitted with a type of computing circuit imitating a conditioned reflex.* **See page 156.**



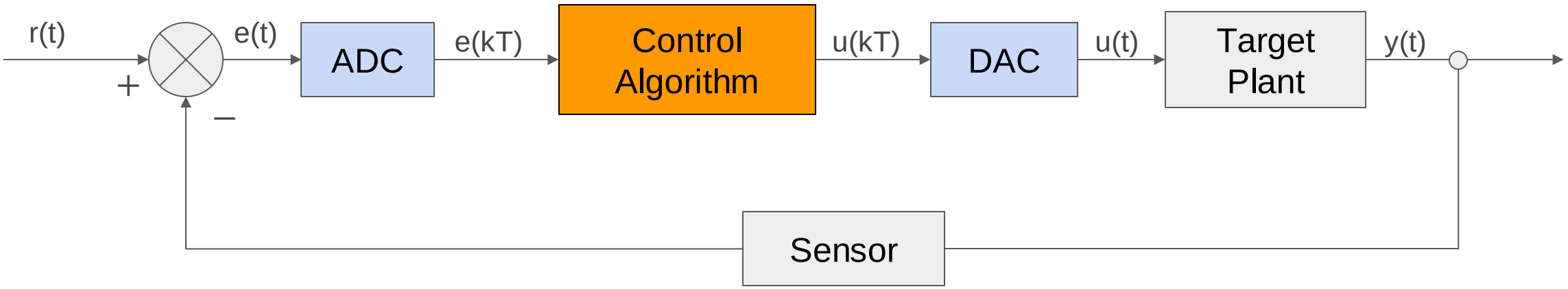
The “turtlebot” in 1940s

Concept of Feedback Control

Feedback Control

Feedback Control: A system that adjusts its actions based on the observed deviation from a desired goal.

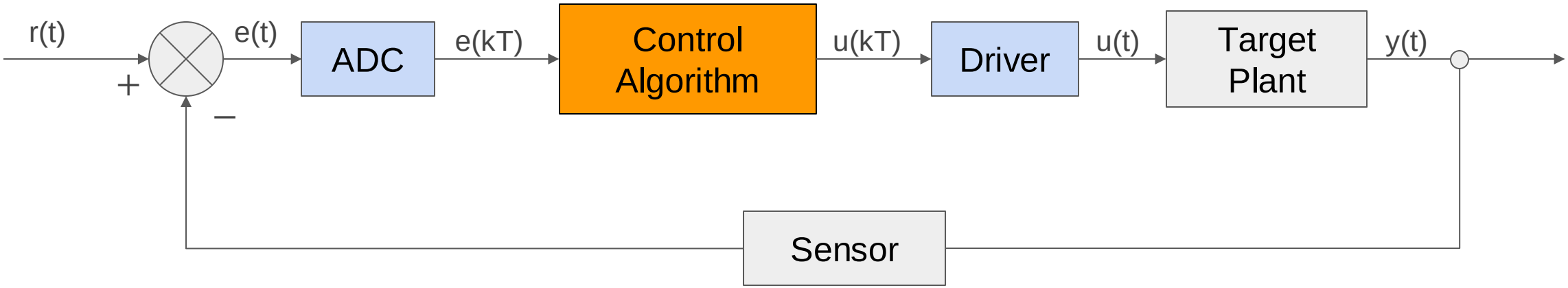
Goal: Enables systems to autonomously adjust to changes, errors, or unexpected events.



Feedback Control

Components:

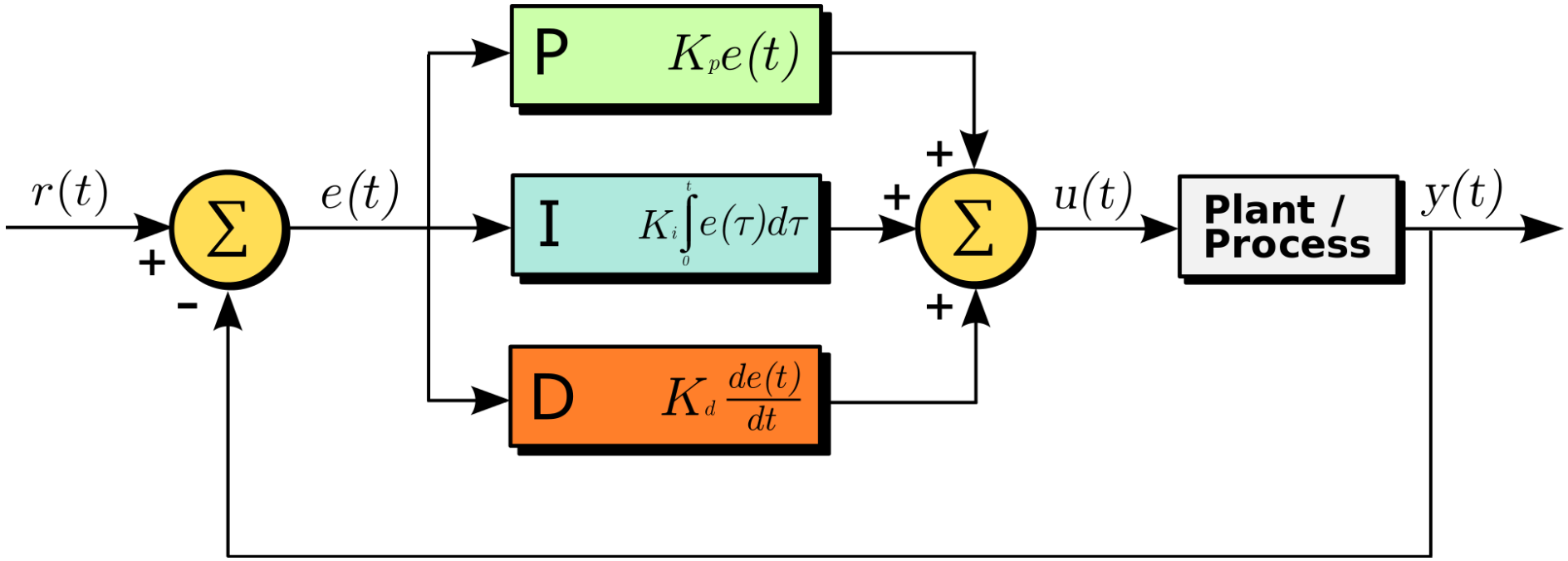
- Controller: Computes the desired action.
- Actuator: Implements the action.
- Sensor: Measures the outcome.
- Feedback Loop: Information path that connects the outcome to the input.



Common control techniques

PID Control

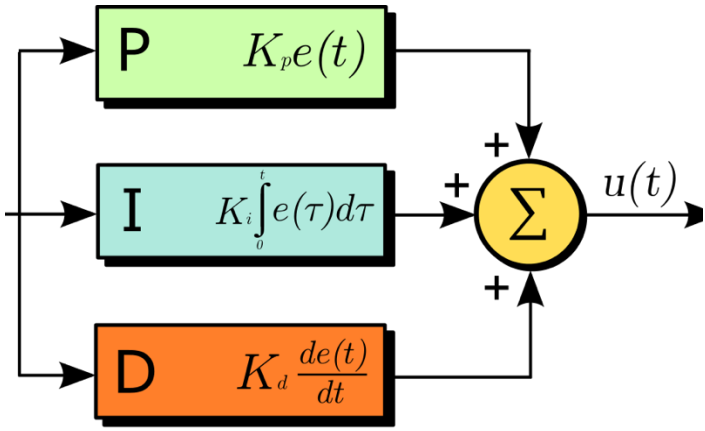
- PID Control: Proportional, Integral, Derivative - the three components of this control system.
- PID Control is a single-input and single-output (SISO) controller: a simple single-variable control system with one input and one output.



PID Control

Definition of control gains (yes, we call them “gains”):

- Proportional (P): Produces an action proportional to the current error (how far off the system is from the desired state).
- Integral (I): use accumulated past errors to produce a counter-action.
- Derivative (D): apply action based on the rate of change of errors (proportional to the derivative of errors).



$$\boldsymbol{u}(t) = k_p e + k_d \dot{e} + k_I \int e \quad dt$$

PID controller

Here, we directly go to the design a **discrete** PID controller, and learn how a discrete controller can be written/coded from an analog form.

Let error be $e = x_{ref} - x$, PID controller in continuous time

$$\boldsymbol{u}(t) = k_p e + k_d \dot{e} + k_I \int e \quad dt$$

PID stands for

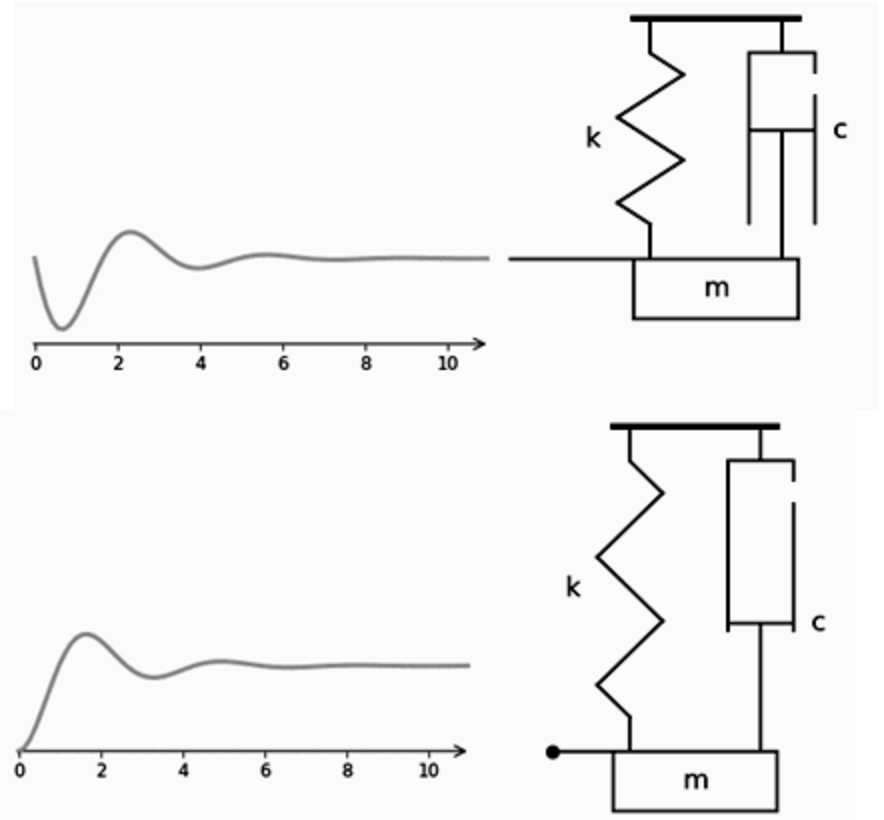
1. P: proportional control, control effort is linearly proportional to the system error;
2. I : integral control, control effort is linearly proportional to the integral of error over a period of time;
3. D: derivative control, control effort is linearly proportional to the rate of change of error, which gives a sharp response to a sudden change of signals.

PD controller

Let's drop the integral ***I*** term, then what it replicates?

$$\boldsymbol{u}(t) = k_p e + k_d \dot{e}$$

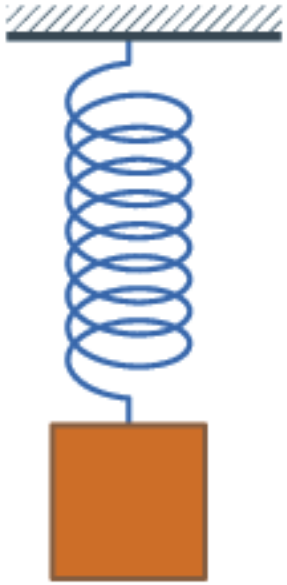
$$\boldsymbol{u}(k) = k_p(0 - x(k)) + k_d(0 - \dot{x}(k))$$



A PD controller resembles a spring-damper system

A restoring force of a spring is proportional to the displacement x:

$$F_S = -kx$$



where k is the spring stiffness.

A viscous/damping force is proportional to the velocity with opposite sign,

$$F_d = -c\dot{x}$$

where c is the damping coefficient.

Recap: spring-damper system

Assume system mass is m, apply Newton's 2nd law,

$$m\ddot{x} = -kx - c\dot{x} \quad \Leftarrow \quad ma = f$$

$$\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} \quad \Leftarrow \quad a = \frac{f}{m}$$

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

where $\omega_0 = \sqrt{\frac{k}{m}}$ is the natural frequency (resonance) of system,

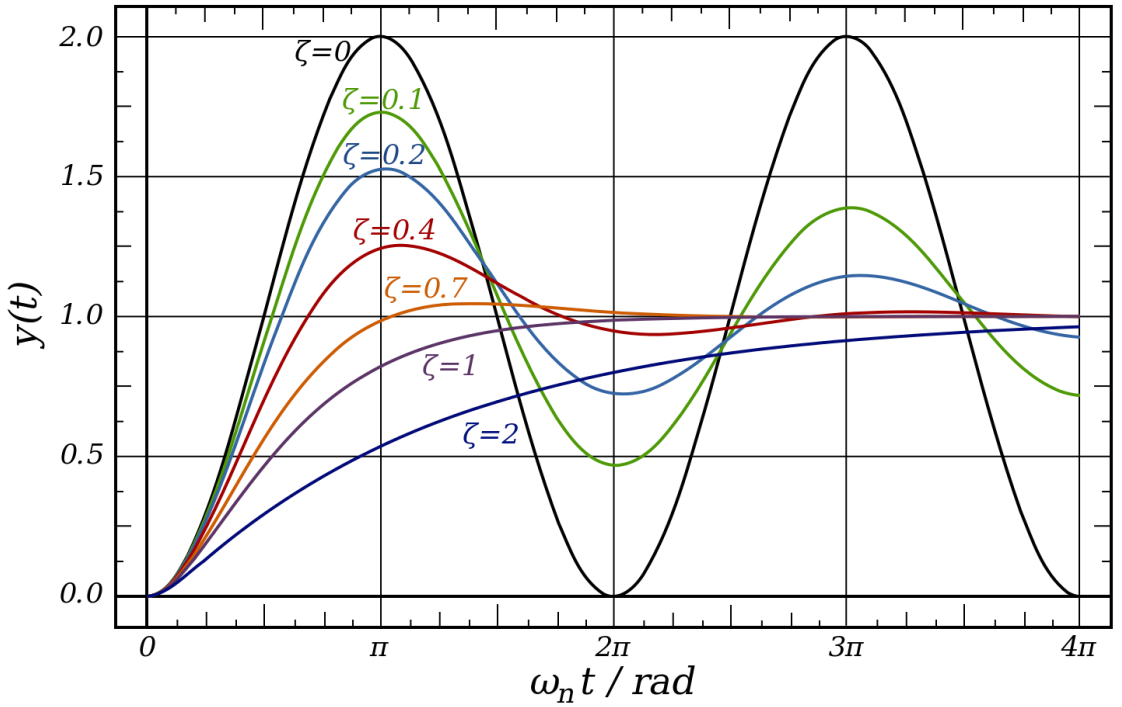
$\zeta = \frac{c}{2\sqrt{km}}$ is the 'damping ratio'.

**Refer to lecture "Dynamics" as well.*

Properties of a spring-damper system

ζ, the 'damping ratio', determines the system to be:

1. under-damped, $\zeta < 1$
2. critical-damped, $\zeta = 1$
3. over-damped, $\zeta > 1$



See more: en.wikipedia.org/wiki/Damping_ratio

$$\zeta = \frac{c}{2\sqrt{km}} \quad \Rightarrow \quad c = 2\zeta\sqrt{km}$$

So, theoretically, you can always know what viscous coefficient ***c*** is need to make the system critically damped.

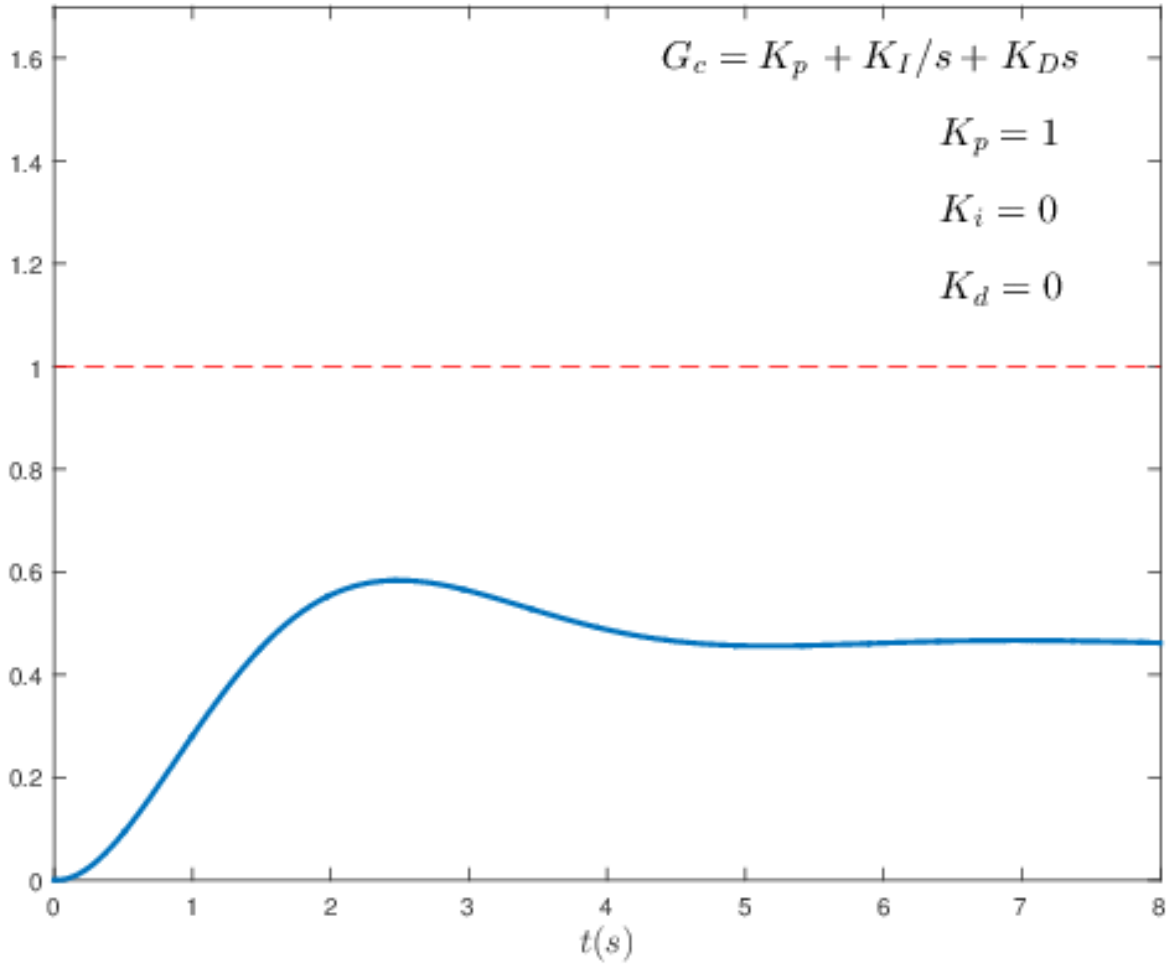
Effects of gains

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	Slightly increase	Decrease	Decrease	No effect	Improve if velocity signal is good (not noisy, little delay)

Note:

- k_d term predicts system behavior in *one tick*, which gives a control effort with the anticipation of the change during the next sampling time.
- In theory, given any k_p gain, there is always a k_d gain that can ensure critical damping of the response. However, due to the noise and delay of velocity, k_d cannot to be too large otherwise noise in amplified. Therefore, k_p gain can't be too large neither.

Effects of gains



PID controller

In most robotics applications where the tracking control of actuation is concerned, the native integral term "I" in PID controller is not used, because:

1. reference is usually changing and there is rarely a steady reference
2. integral can bring the error to zero for systems with slow responses
3. a large integral gain would only result in latency in response, overshooting, or unstable oscillations

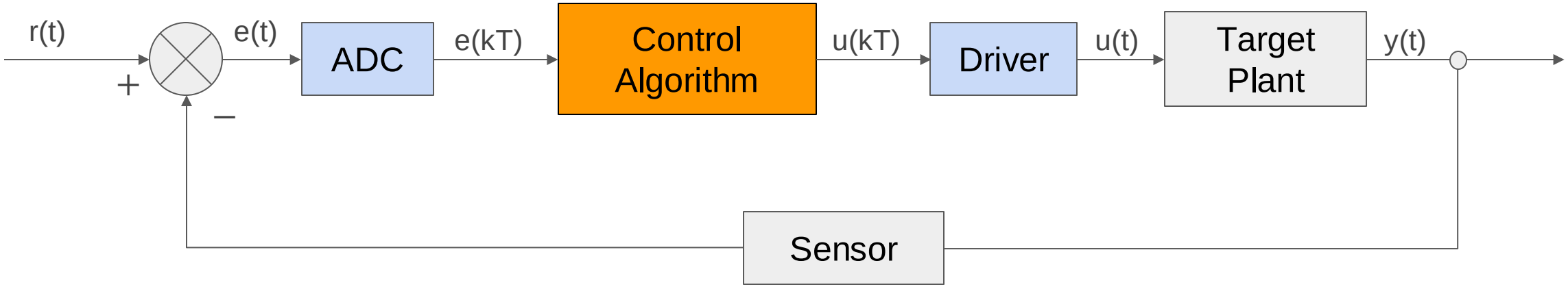
Instead, the problem of steady state error is usually resolved by:

1. model based feedforward control – inverse dynamics (not covered here)
2. high PD gains, may include constrained integral term

Design of digital controller

Digital control

Nowadays, most controllers are implemented in digital computers, or micro-chips.



A digital control system only ‘sees’ the sensory information and command the control action at times, at a constant time interval realized either by hard real-time (interrupter in the CPU or other chips) or “soft” real-time (software timer).

Digital PID controller

Controls are implemented in computer-based systems or by digital computation, ie micro-controller, DSP, FPGA etc.. A digital control system only ‘sees’ the sensory information and command the control action at times, at a constant time interval.

The continuous PID law

$$\boldsymbol{u}(t) = k_p e + k_d \dot{e} + k_I \int e \, dt$$

How the discretized version of PID control looks like?

$$\boldsymbol{u}(k) = k_p e(k) + k_d \dot{e}(k) + k_i \sum e(i), i = 0, \ldots, k$$

Digital PID controller

Using backward Euler method:

$$\dot{e}(k) = \frac{[e(k) - e(k-1)]}{T}$$ (usually, derivative terms are filtered)

$$\int e\, dt = T \sum e(i)$$, note, in k[th] control loop, range of i is: $i = 0, \ldots, k$

PID in continuous time

$$\boldsymbol{u}(t) = k_p e + k_d \dot{e} + k_I \int e\, dt$$

PID in discrete time

$$\boldsymbol{u}(k) = k_p e(k) + k_d \frac{[e(k) - e(k-1)]}{T} + k_I T \sum e(i)$$

$$\boldsymbol{u}(k) = k_p e(k) + k_d \dot{e}(k) + k_i \sum e(i), i = 0, \ldots, k$$

Digital PD controller

A PD controller replicates a spring-damper system

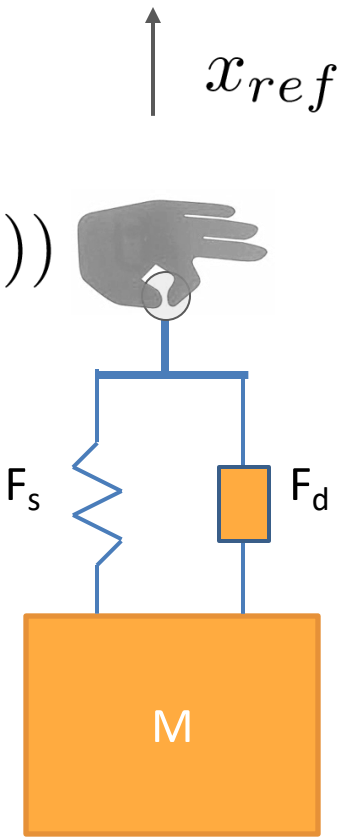
$$\boldsymbol{u}(k) = k_p e(k) + k_d \dot{e}(k)$$

$$= k_p(x_{ref} - x(k)) + k_d(\dot{x}_{ref} - \dot{x}(k))$$

$$\boldsymbol{u}(k) = \boxed{k_p(0 - x(k))} + \boxed{k_d(0 - \dot{x}(k))}$$

Spring

Damper



Digital PD controller

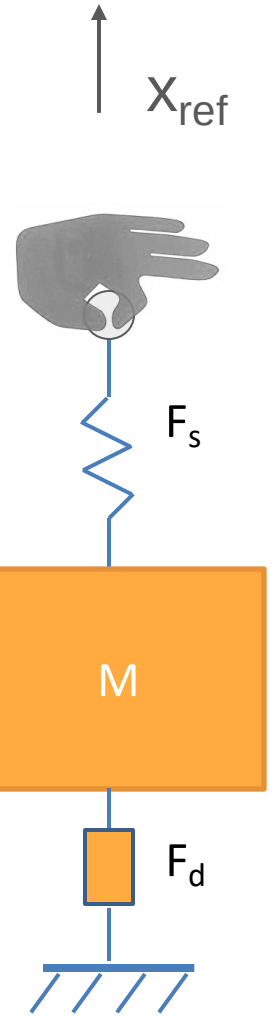
Commonly, another expression is used:

$$\boldsymbol{u}(k) = k_p(x_{ref} - x(k)) - k_d\dot{x}(k)$$

assuming zero reference velocity.

What is the difference between these two?

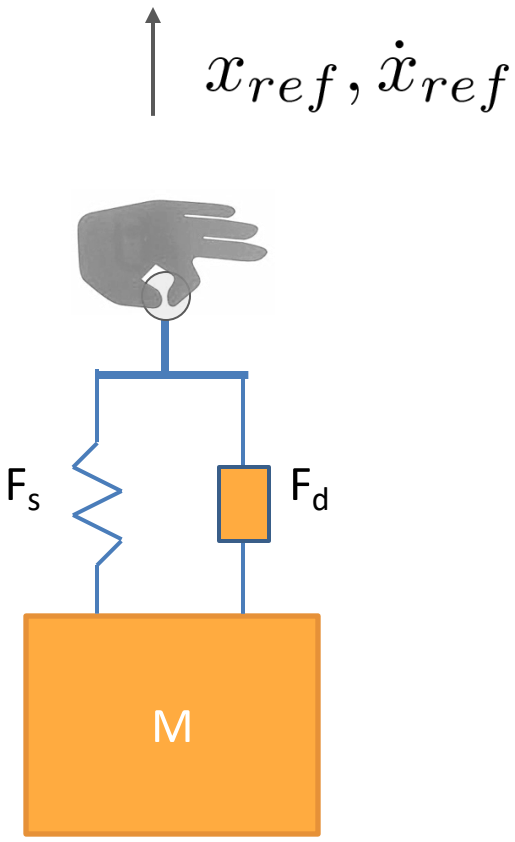
More stable when input has discontinuity



PD pseudo code

Algorithm 1 Example PD controller

```
1:  procedure MoveJoint(Joint, x_ref)
2:      dt ← 1/controlFrequency
3:      ControlCycles ← controlFrequency/updateFrequency
4:      x_old ← 0
5:      e_old ← 0
6:      time ← 0
7:      Disable velocity joint controller
8:      while time < endTime do
9:          x_real ← rotation position of Joint
10:         e ← x_ref − x_real
11:         ė ← (e − e_old)/(dt × ControlCycles)
12:         for each c in [0..ControlCycles] do
13:             τ ← K_p × e + K_d × ė
14:             Send the torque, τ, to Joint
15:         end for
16:         x_old ← x_real
17:         e_old ← e
18:         time ← time + ControlCycles × dt
19:     end while
20: end procedure
```



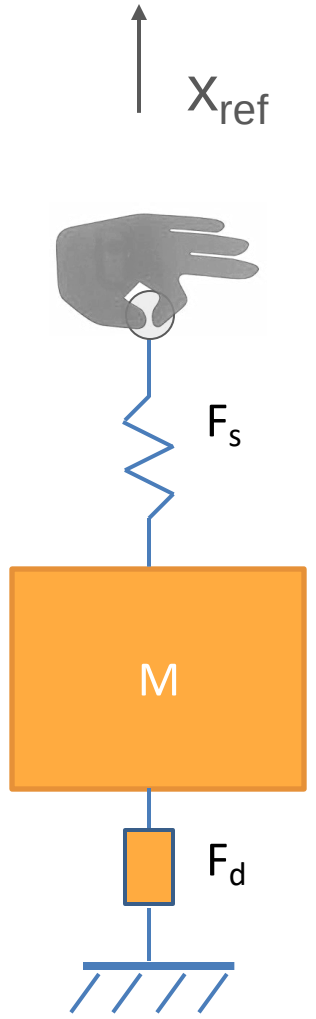
PD pseudo code

$$\boldsymbol{u}(k) = k_p(x_{ref} - x(k)) - k_d\dot{x}(k)$$

$\mathbf{u}=K_p*(x_{ref} - x\_measured) - K_d*dx\_measured$

or

$\mathbf{u}=K_p*(x_{ref} - x\_measured) - K_d*v\_measured$



Key takeaway

PID control:

- Understanding of PID gains
- Like the spring-damper system, the resonance frequency of this virtual spring-damper determines the bandwidth of the PD controller.
- The “resonance frequency” must be several times higher than the frequency of the reference that needs to be tracked.

Summary

What are the features of PID controller?

- A single input single output (SISO) controller
- PID is, by its nature, an instantaneous controller (record of past few ticks and no insight about future).

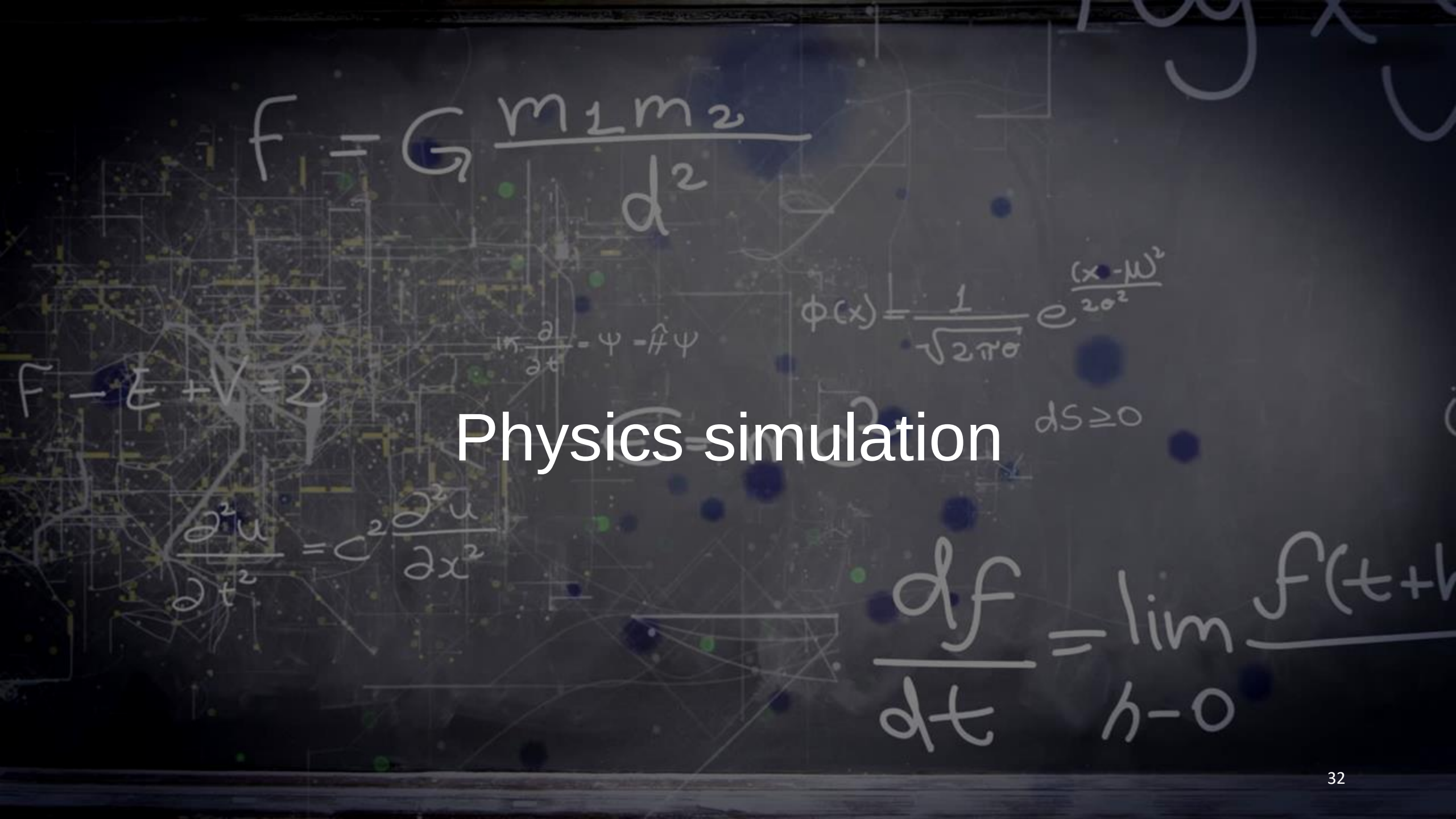
Multiple PID controllers are decoupled, however, for most industrial applications, PID control can achieve decent/reasonable performance.

Summary

Nevertheless, there are always some cases in which it is hard to tune the control gains because: tuning is empirical!

Particularly, the applications where PID can be unsatisfactory or quite difficult are:

- process control (some chemical process), slow dynamics, large latency;
- systems that are fundamentally unstable and has constraints in control effort, eg double inverted pendulum, flight control of aircraft, quadrotor.


Physics simulation

Numerical integration: how to work out?

A system with unit mass m of 1, given acceleration a, in continuous-time, velocity is the integral of acceleration

$$v = \int a \quad dt$$

Position is the integral of velocity

$$s = \int v \quad dt$$

Numerical integration

A system with unit mass m of 1, given acceleration a, in continuous-time, velocity is the integral of acceleration

$$v = \int a \quad dt$$

Position is the integral of velocity

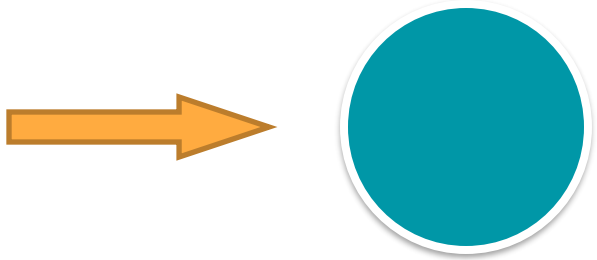
$$s = \int v \quad dt$$

In discrete-time, $v_{n+1} = v_n + a_n \cdot dt$ (we don't know a_{n+1}, acceleration a_n is the most up-to date value)

And also, $s_{n+1} = s_n + [\frac{(v_{n+1} + v_n)}{2}] \cdot dt = s_n + v_n \cdot dt + \frac{1}{2} \cdot a_n \cdot dt^2$

Apply law of physics

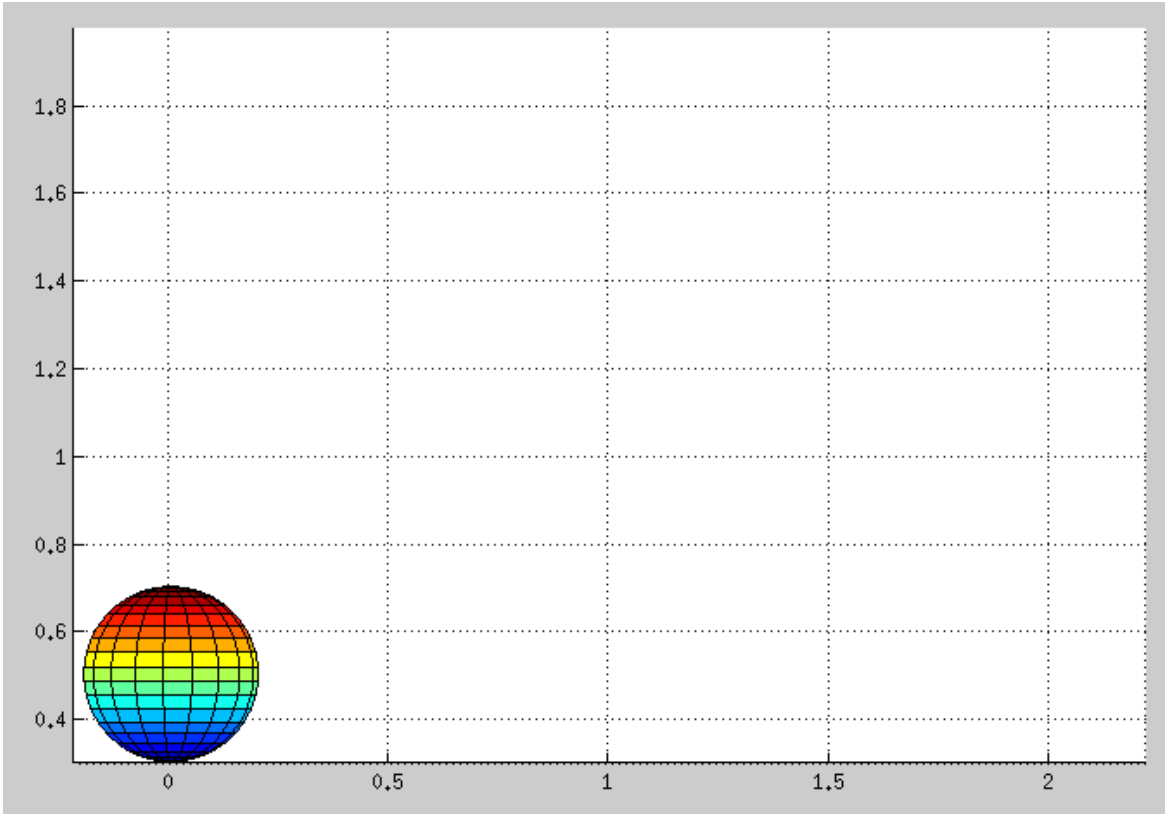
- Newton's laws of motion (a=F/m):



- Numerically integrating acceleration to obtain velocity, and numerically integrating velocity to obtain position

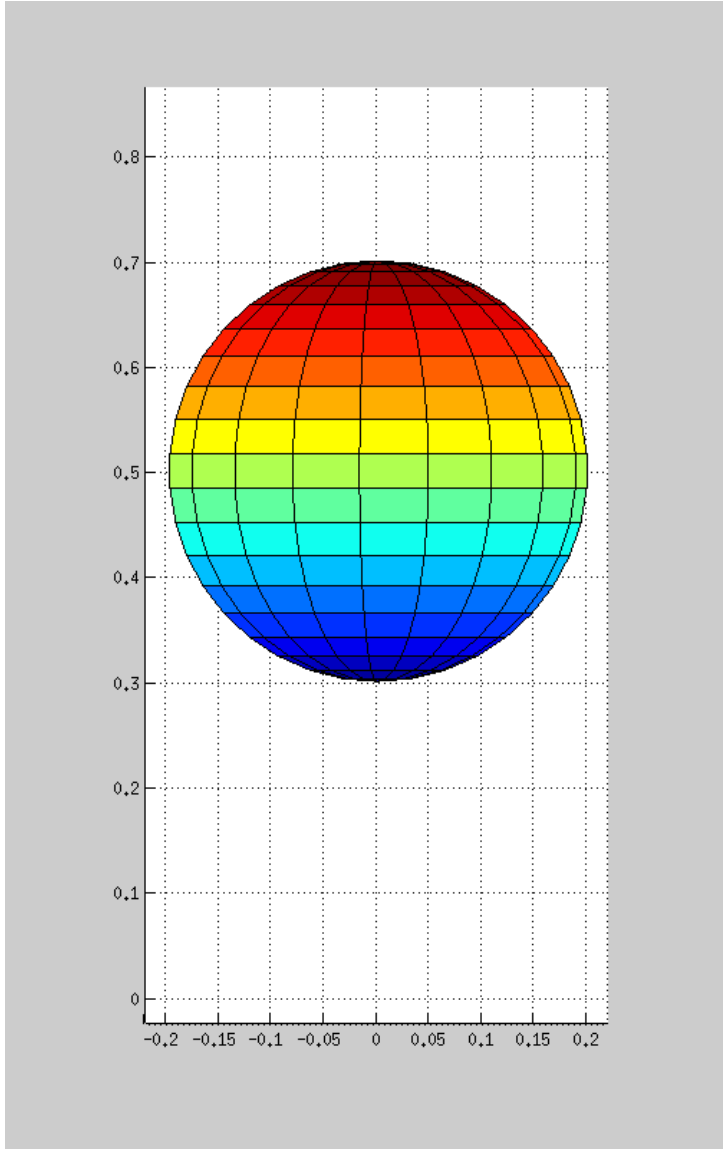
Simulations: projectile motion in free space

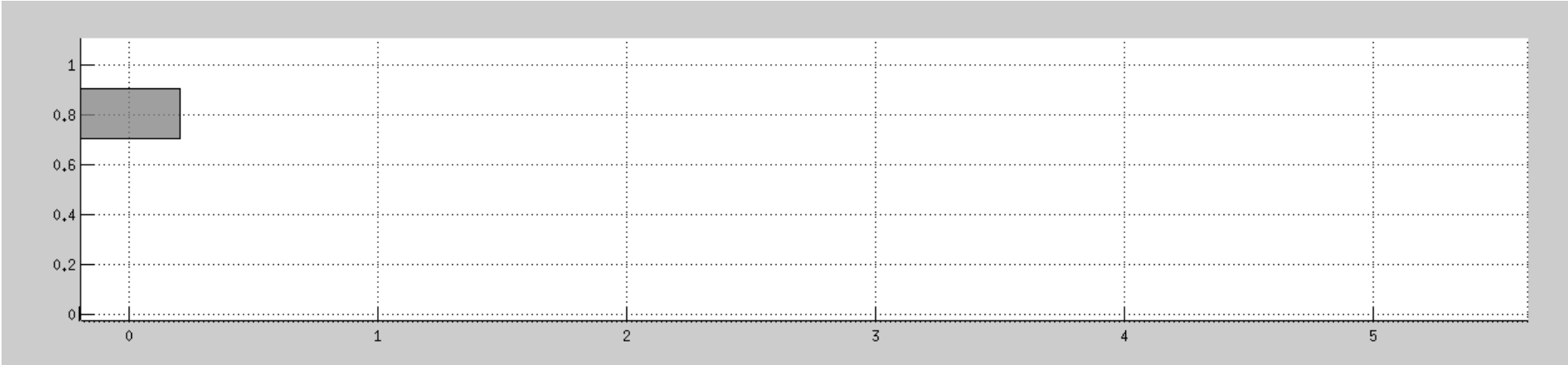
Numerical simulation of physical systems



Simulations: free fall; simple contact model, a bouncing ball.

Numerical simulation of physical systems



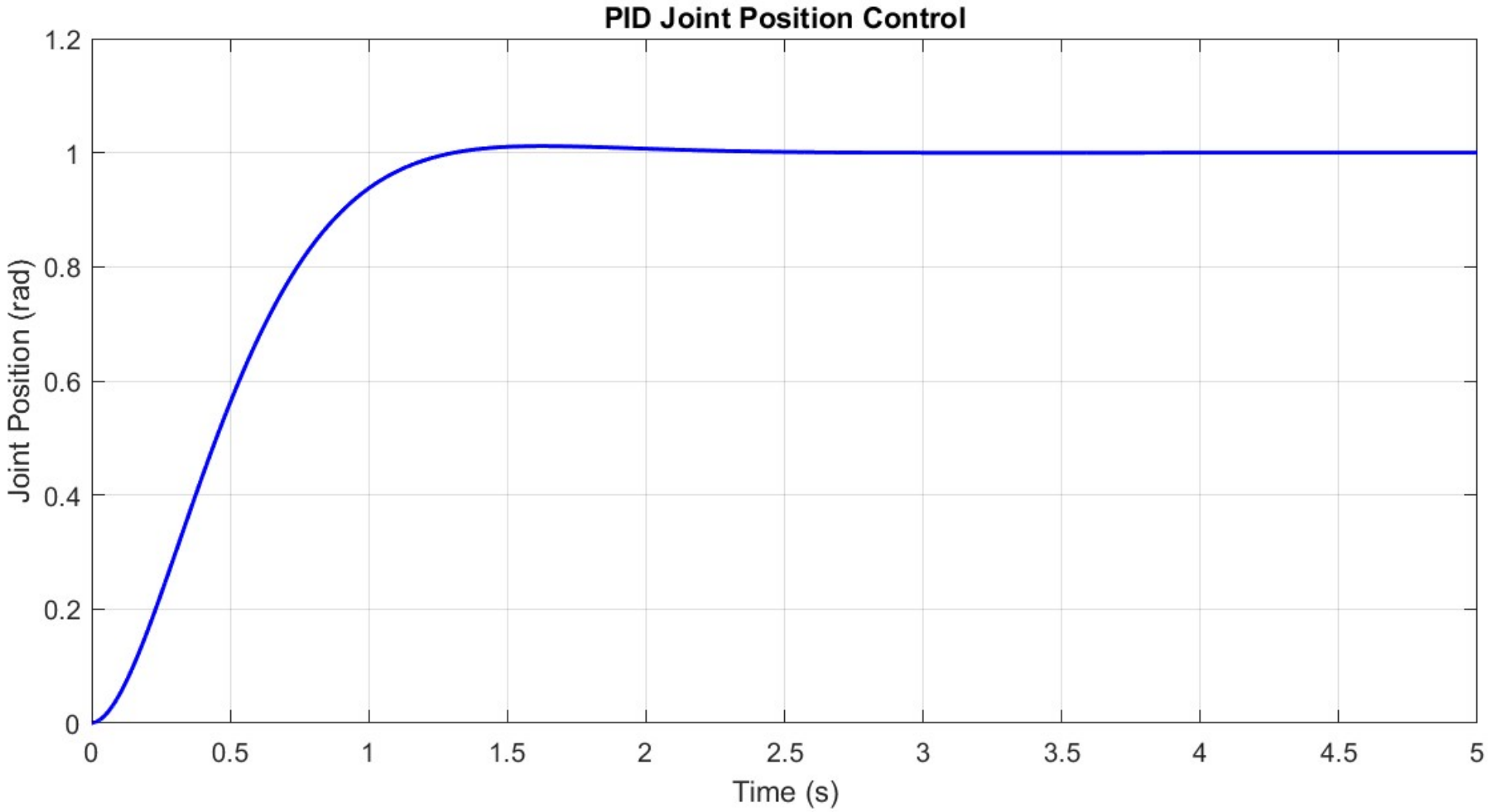


Numerical simulation of physical systems

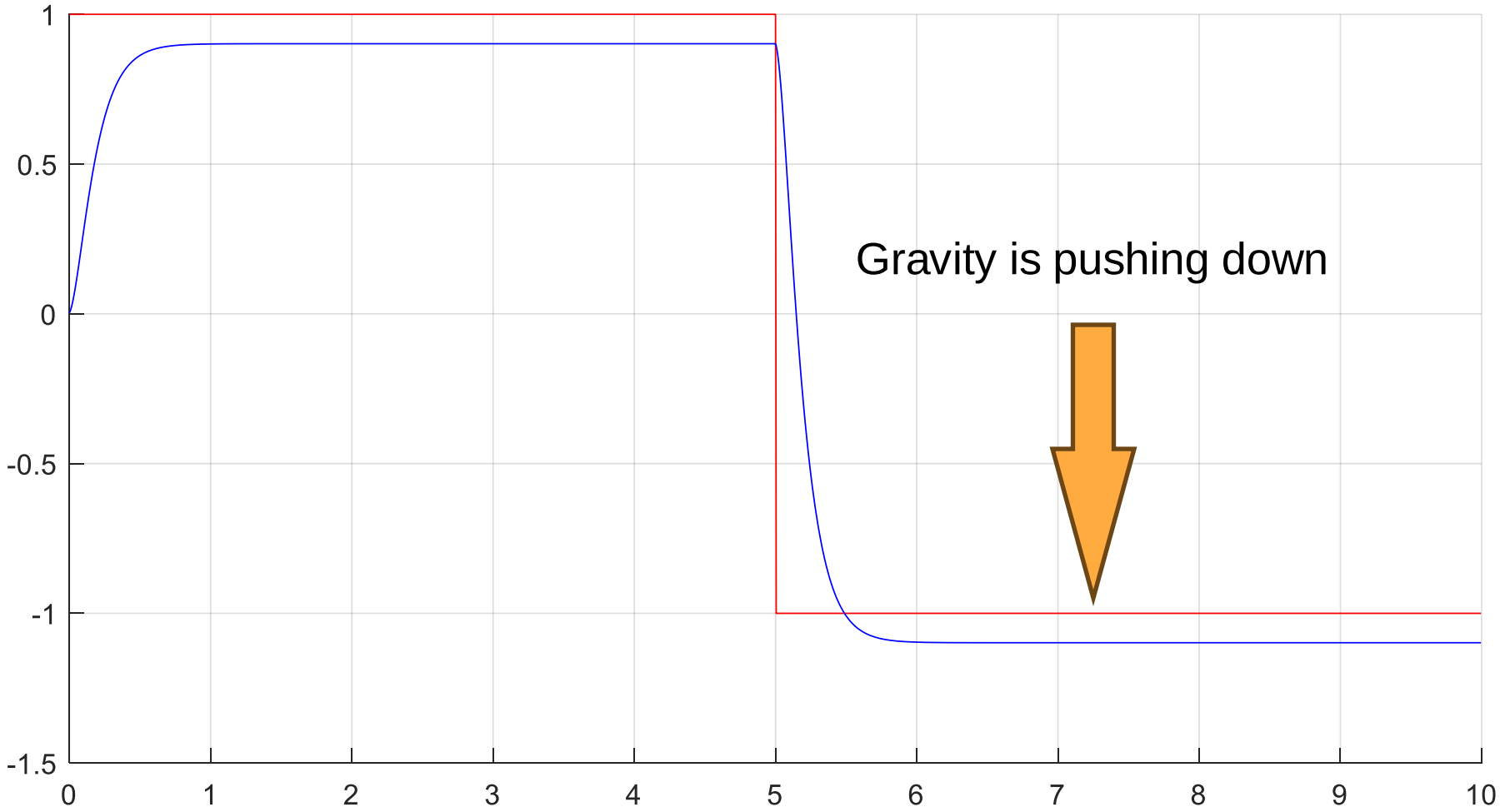
Example of a ballistic motion with ground contact.

Case study of control in simulated physics (code walkthrough)

Tuned PD control response



PD control response in presence of external force



PD with added integral gain (PID)

- Rethink: why response is asymmetric?