MSAI Robotics course

Lecture 7

Control algorithms

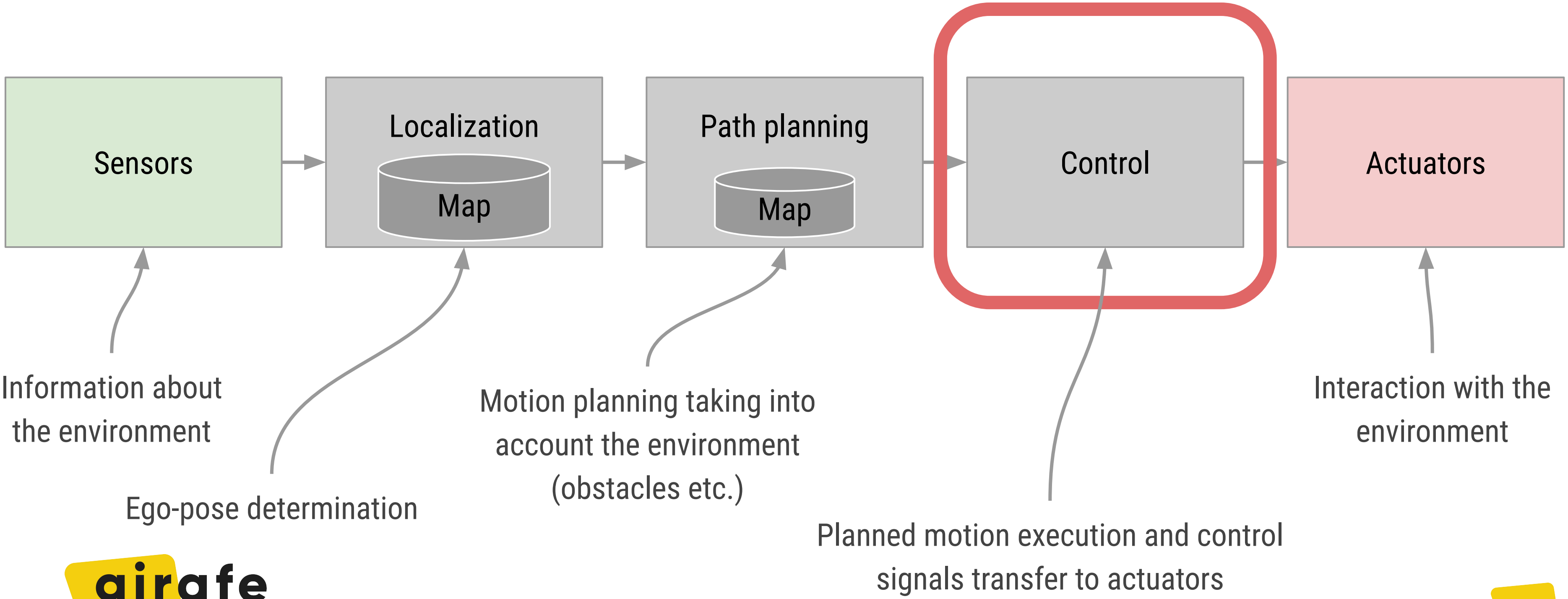
Vladislav Goncharenko
Materials by Oleg Shipitko
MIPT, 2022

girafe ai

Outline

1. Control problems of wheeled robots
   a. Speed control
   b. Trajectory control
2. (very) Short intro into the control theory
3. PID-regulators
4. LQR-regulators
5. Model-predictive control

(SIMPLIFIED) CONTROL SCHEME OF MODERN MOBILE ROBOT



Control problems of wheeled robots

girafe
ai

01

CONTROL PROBLEMS OF WHEELED ROBOTS

- The are two major control problems in wheeled robotics
  - Speed control
  - Trajectory control

Control problems of wheeled robots

girafe
ai

02

CONTROL THEORY IN 5 MINUTES

- The **automatic control theory** is a scientific discipline that studies the processes of automatic control of various objects, while considering the objects themselves as **“converters” of the input signal into the output**.
- **Control** is a purposeful impact on a control object (**plant**).
- **The purpose of control** is to ensure the desired operating mode of the plant (desired output, when acting on the input).
- **Control unit** / **controller** / **regulator** - generates control actions on the plant in accordance with the control law. The controller input is a control error ***e(t)***. The output is the control signal ***u(t)***.

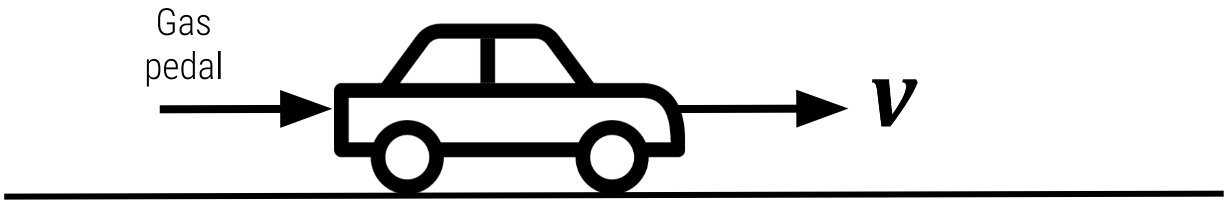
CONTROL THEORY IN 5 MINUTES

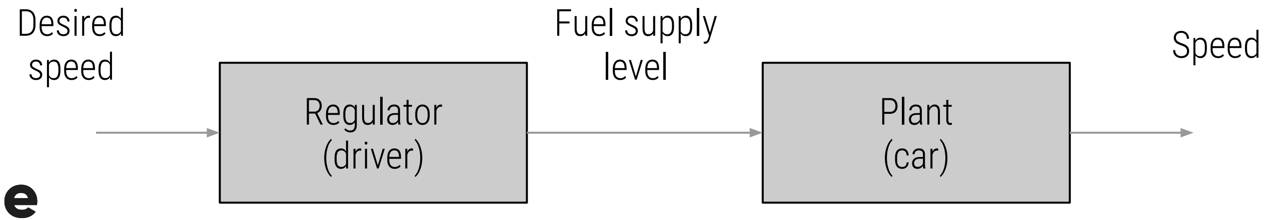
- **Control error *e(t) = g(t) - y(t)*** is the difference between the required value of the controlled variable ***g(t)*** and its current value ***y(t)***.

- **Disturbing signal *f(t)*** is a process at the input of a control object, which leads to disturbances. It may be caused by control signal transmission noise, controlled variable measurement error, environmental influences, etc.

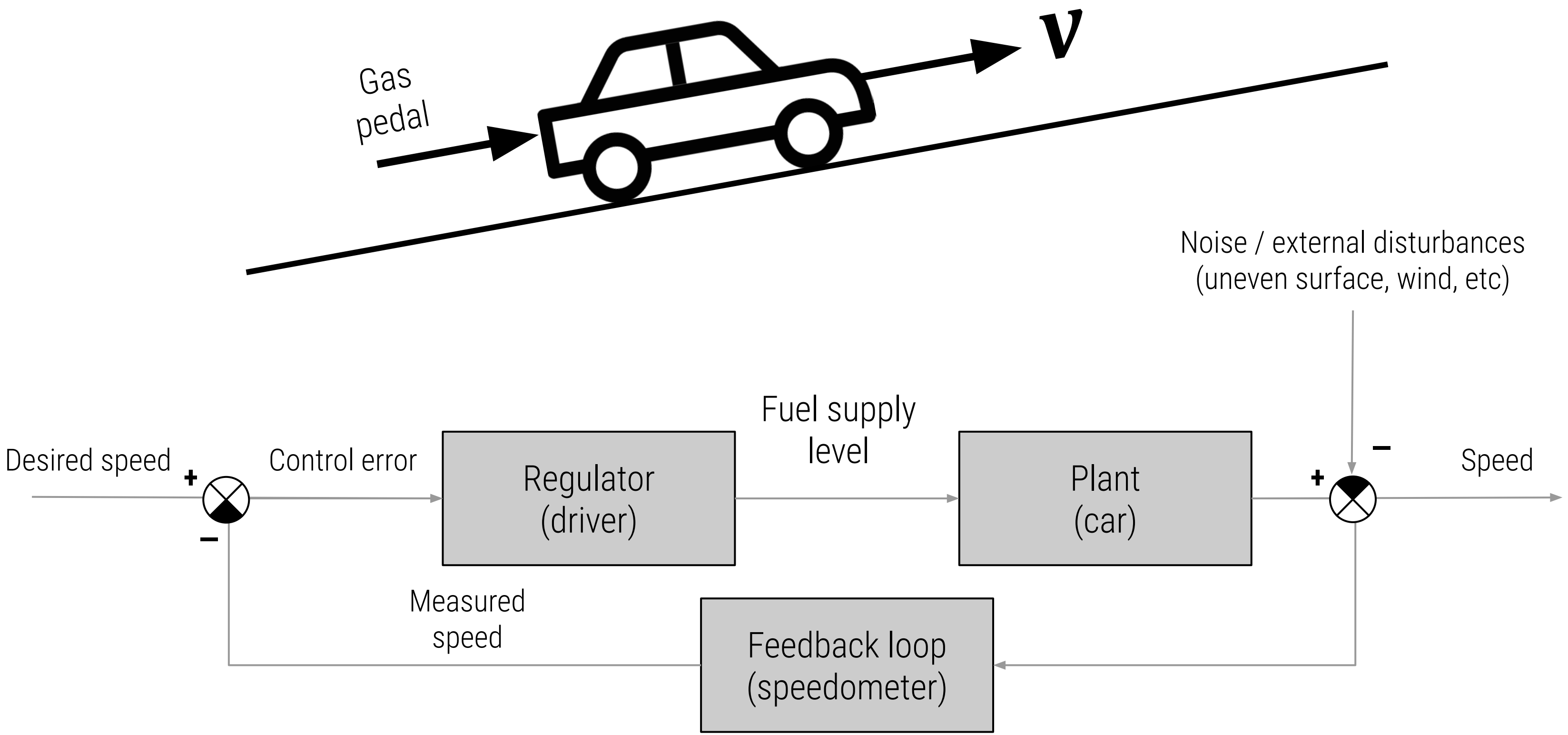
girafe ai

EXAMPLE OF OPEN-LOOP CONTROL SYSTEM





EXAMPLE OF CLOSED-LOOP CONTROL SYSTEM



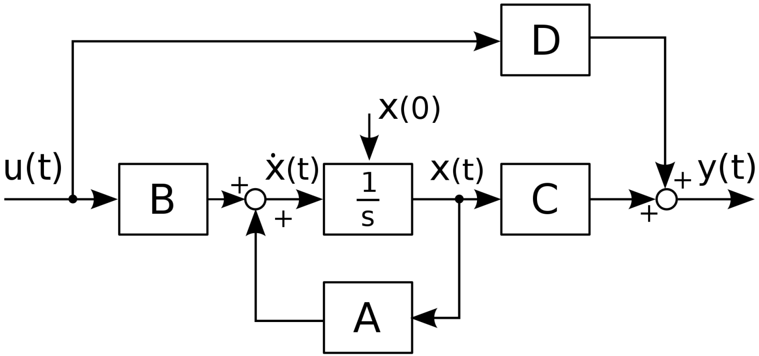
STATE-SPACE REPRESENTATION

- **The state space** is one of the main ways of representing dynamical systems in Control Theory

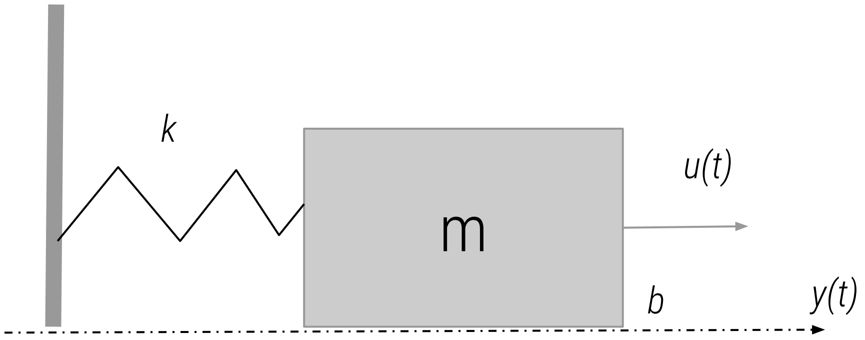
$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$

$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

$\mathbf{x}(\cdot)$ is called the "state vector", $\mathbf{x}(t) \in \mathbb{R}^n$;
$\mathbf{y}(\cdot)$ is called the "output vector", $\mathbf{y}(t) \in \mathbb{R}^q$;
$\mathbf{u}(\cdot)$ is called the "input (or control) vector", $\mathbf{u}(t) \in \mathbb{R}^p$;
$\mathbf{A}(\cdot)$ is the "state (or system) matrix", $\dim[\mathbf{A}(\cdot)] = n \times n$,
$\mathbf{B}(\cdot)$ is the "input matrix", $\dim[\mathbf{B}(\cdot)] = n \times p$,
$\mathbf{C}(\cdot)$ is the "output matrix", $\dim[\mathbf{C}(\cdot)] = q \times n$,
$\mathbf{D}(\cdot)$ is the "feedthrough (or feedforward) matrix"



STATE SPACE. EXAMPLE



$$m\ddot{y}(t) = u(t) - b\dot{y}(t) - ky(t)$$

y(t) – load position
u(t) – applied force
b – coefficient of friction
k – spring elasticity
m – weight

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \dot{\mathbf{x}}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \mathbf{u}(t)$$

$x_1(t)$ – object position
$x_2(t)$ – object velocity

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \end{bmatrix}$$

SYSTEM TRANSFER FUNCTION

- **Transfer function** — an alternative way to describe a dynamic systems

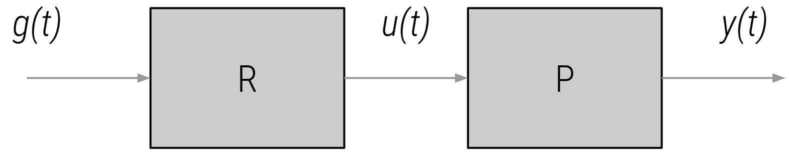
- Let *u(t) — input signal, y(t)* — output

- Transfer function **W(s)** can be written as:

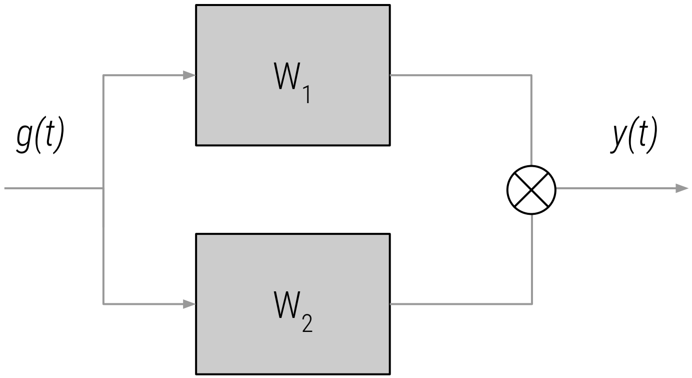
$$W(s) = \frac{Y(s)}{U(s)}$$

- where s = jw [rad/s] — transfer function operator; *U(s)* and *Y(s)* — results of Laplace transform of *u(t)* and *y(t)* correspondingly

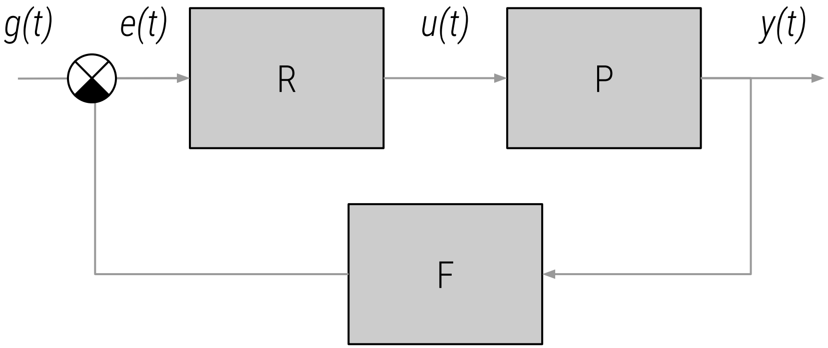
HOW TO READ STRUCTURAL DIAGRAMS



$$W = RP$$

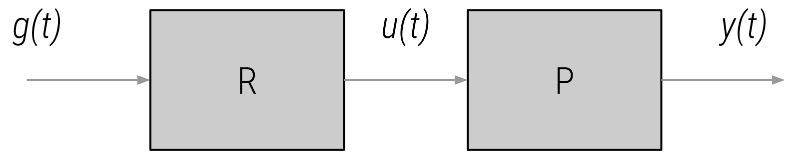


$$W = W_1 + W_2$$

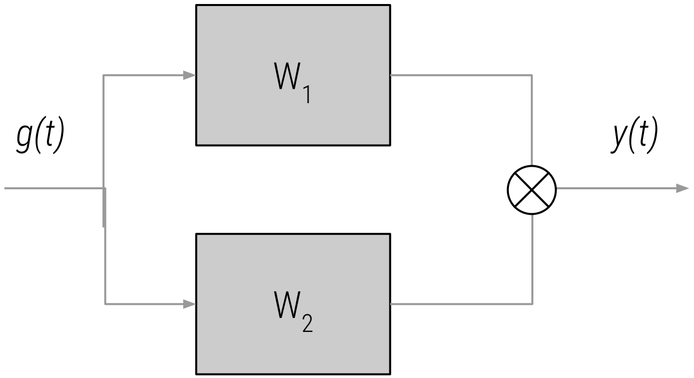


$$W = \frac{RP}{1 + FRP}$$

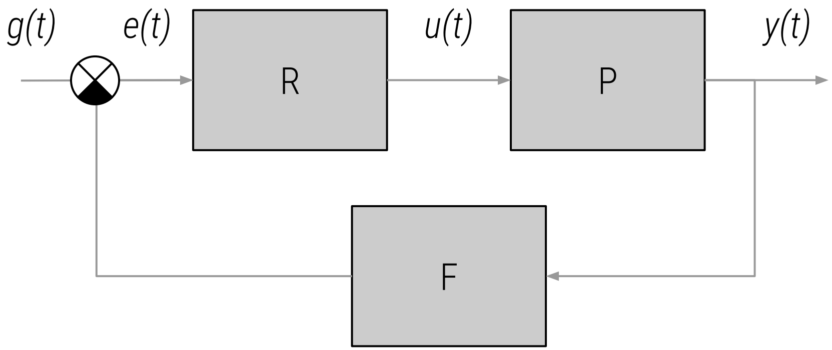
HOW TO READ STRUCTURAL DIAGRAMS



$$W = RP$$

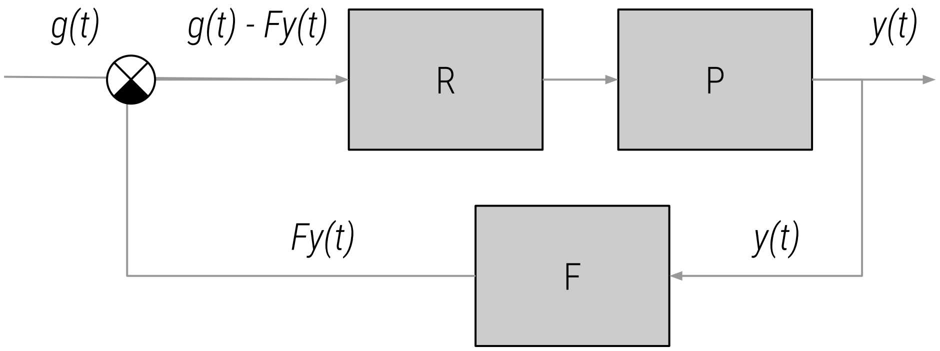


$$W = W_1 + W_2$$



$$W = \frac{RP}{1 + FRP}$$

CLOSED-LOOP SYSTEM TRANSFER FUNCTION



W – open-loop system transfer function

$$W = \frac{out}{in} = \frac{y(t)}{g(t)} = RP$$

$$Wg(t) = y(t)$$

$$W\left[g(t) - Fy(t)\right] = y(t)$$

$$Wg(t) - WFy(t) = y(t)$$

$$Wg(t) = y(t)\left[1 + FW\right]$$

$$W_{closed} = \frac{y(t)}{g(t)} = \frac{RP}{1 + FRP}$$

PROS AND CONS OF CONTROL SCHEMES

Opened-loop system

Cons

- ❏ Sensitive to parameter changes
- ❏ Sensitive to disturbances
- ❏ Needs periodic adjustments

Pros

- ❏ Easy to develop (cheap)
- ❏ Does not affect stability
- ❏ High reaction speed

Closed-loop system

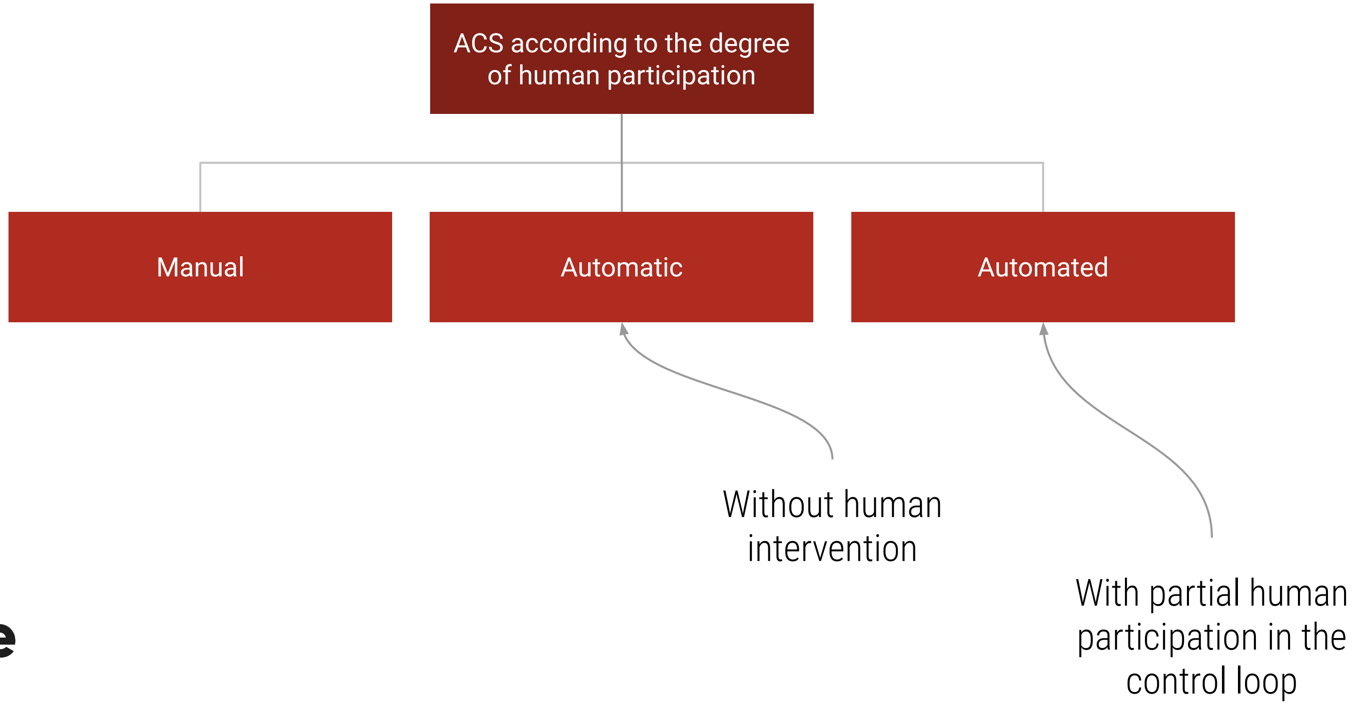
Cons

- ❏ Difficult to develop (expensive)
- ❏ Potential to stability loss
- ❏ Processing speed reduction

Pros

- ❏ Insensitive to parameter changes (within certain limits)
- ❏ Not sensitive to disturbances (within some limits)

TYPES OF AUTOMATIC CONTROL SYSTEMS (ACS)



TYPES OF AUTOMATIC CONTROL SYSTEMS (ACS)

- ACS by the signals nature
  - Continuous
  - Discrete
  - Relay

With a smooth change in the input signal, the output changes abruptly

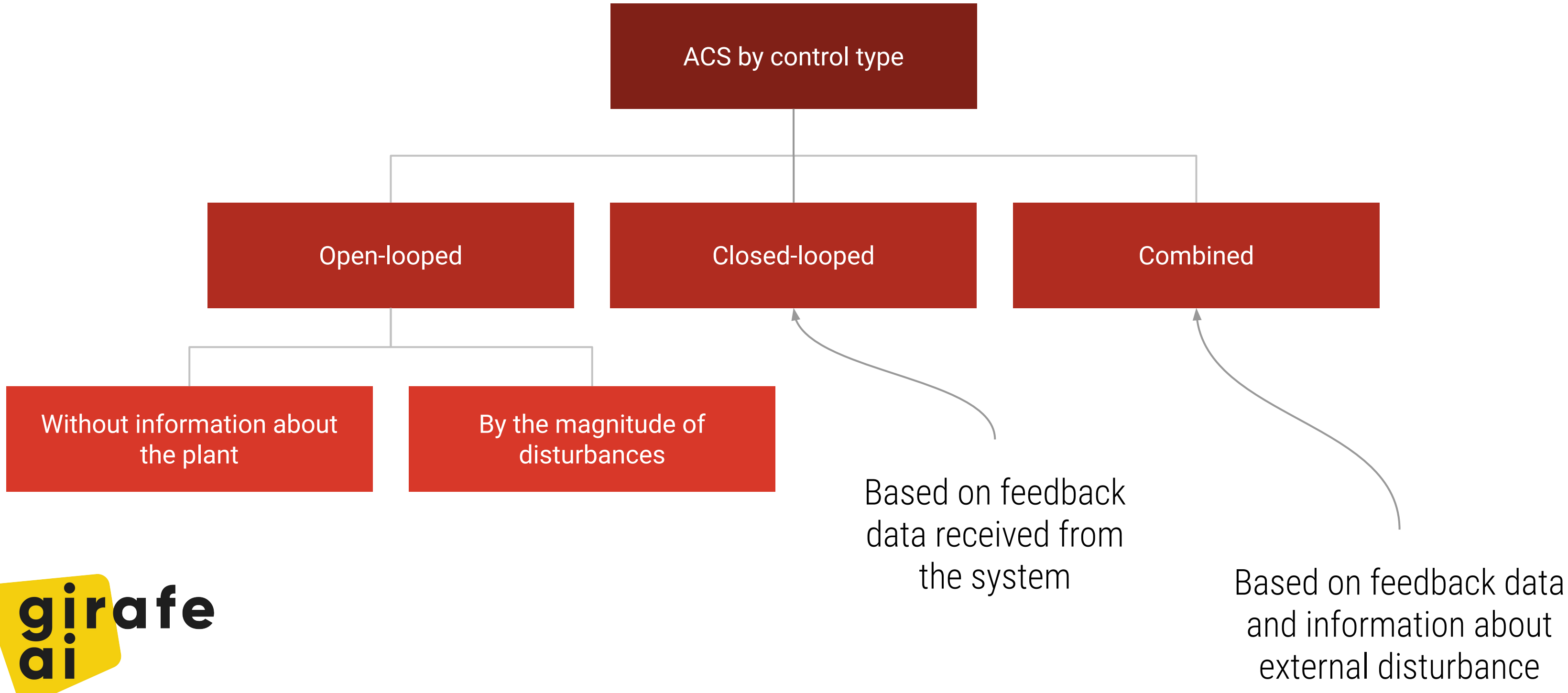
TYPES OF AUTOMATIC CONTROL SYSTEMS (ACS)

ACS by the type of transformation mathematical model

- Linear
- Nonlinear

$$W[\alpha g_1(t) + \beta g_2(t)] = \alpha W[g_1(t)] + \beta W[g_2(t)]$$

TYPES OF AUTOMATIC CONTROL SYSTEMS (ACS)

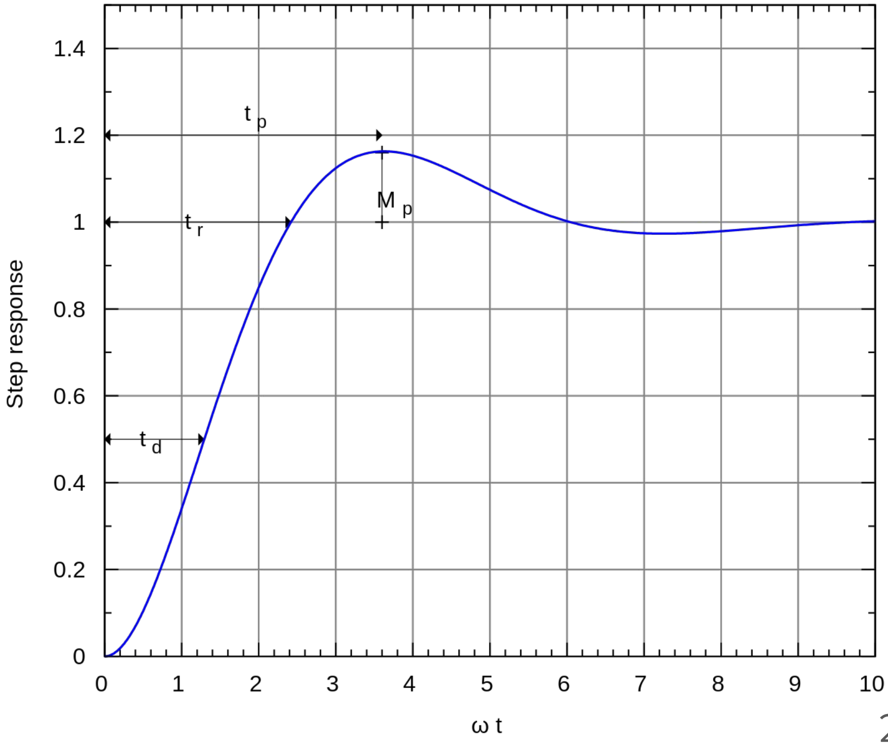


TRANSITION PROCESS AND ITS CHARACTERISTICS

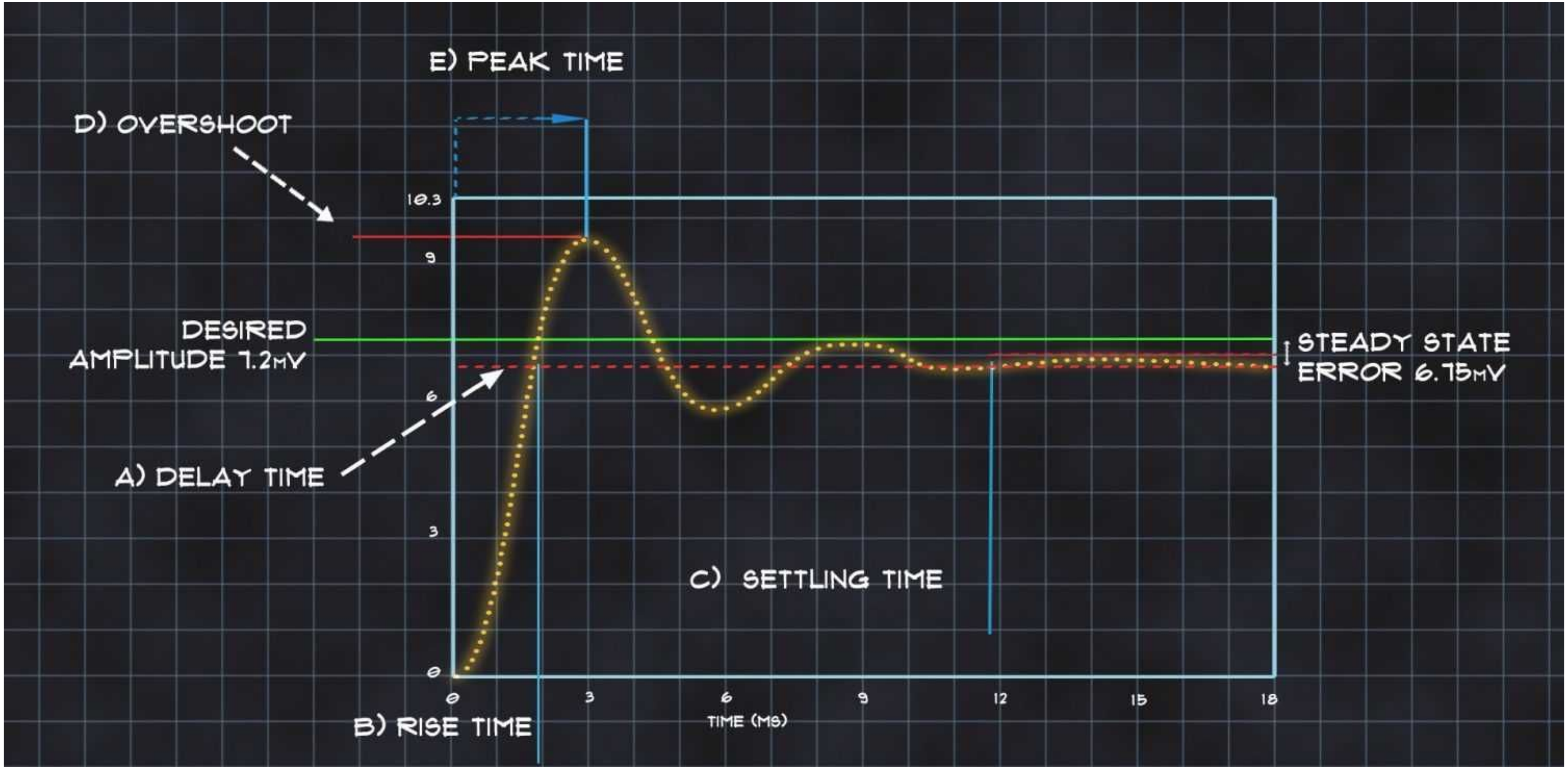
The transition process arises as a reaction of the system to a change in external conditions (input signals, external disturbances, etc.)

Transition process characteristics

- **Transition time** — the time during which the output signal approaches the steady-state value (with some delta 1-5%)
- **Overshoot** — ratio of difference max. value of the output signal and its steady-state value
- **Oscillation** — the absolute value of the amplitudes ratio of the of the first and second oscillations
- **Steady-state error** — the difference between the desired and real value of the output at infinity
- ...



TRANSITION PROCESS AND ITS CHARACTERISTICS

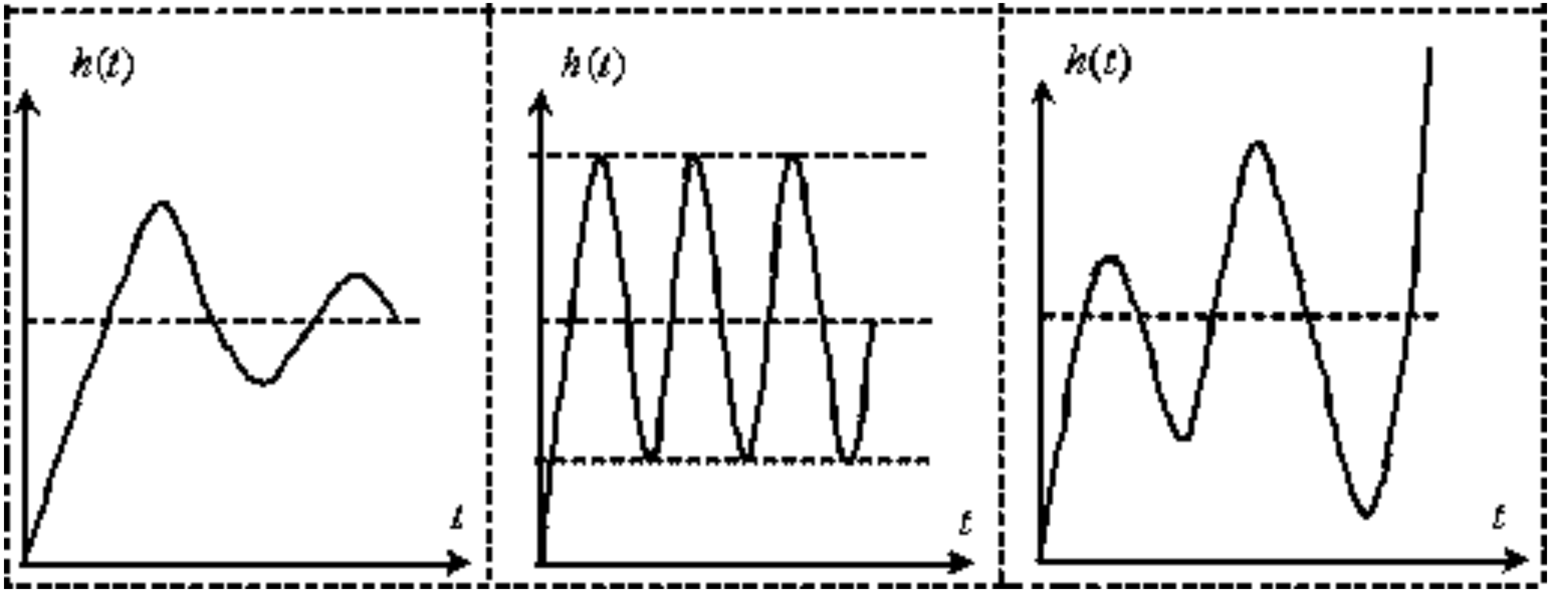


SYSTEM STABILITY

Stable system

System at the stability boundary

Unstable system

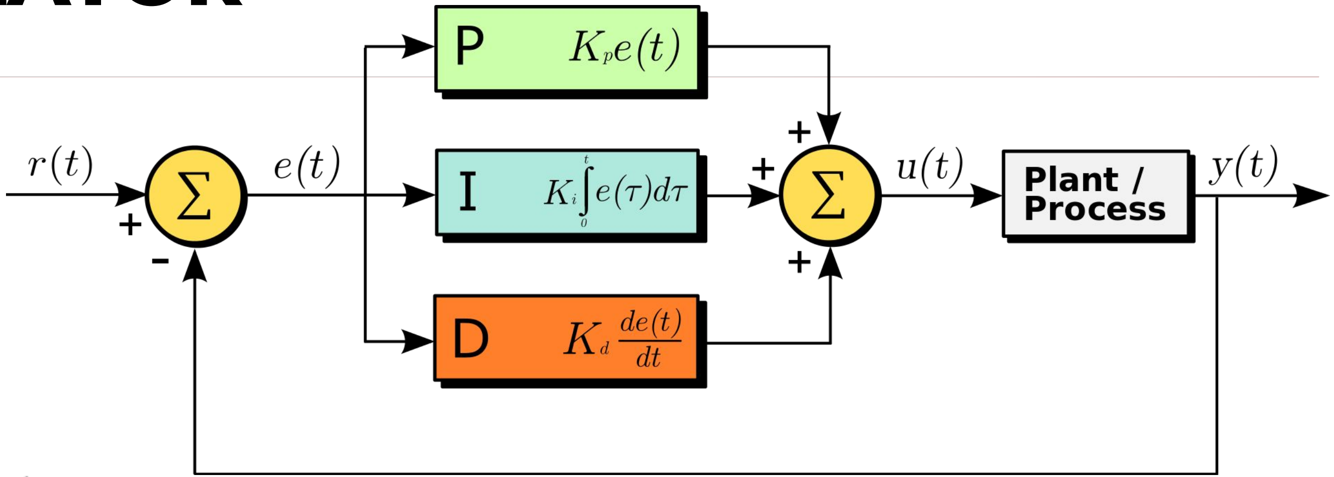


PID-regulators

girafe
ai

03

PID-REGULATOR

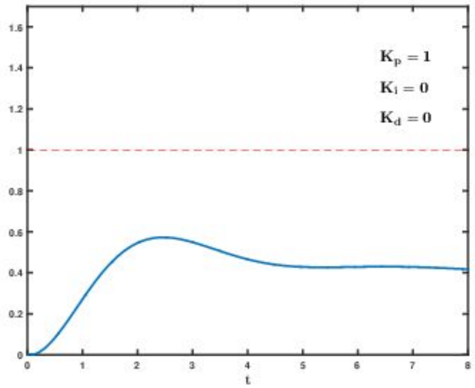


Continuous case:

$$u(t) = P + I + D = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

Discrete case:

$$U(n) = K_p E(n) + K_p K_{ip} T \sum_{k=0}^{n} E(k) + \frac{K_p K_{dp}}{T} (E(n) - E(n-1))$$



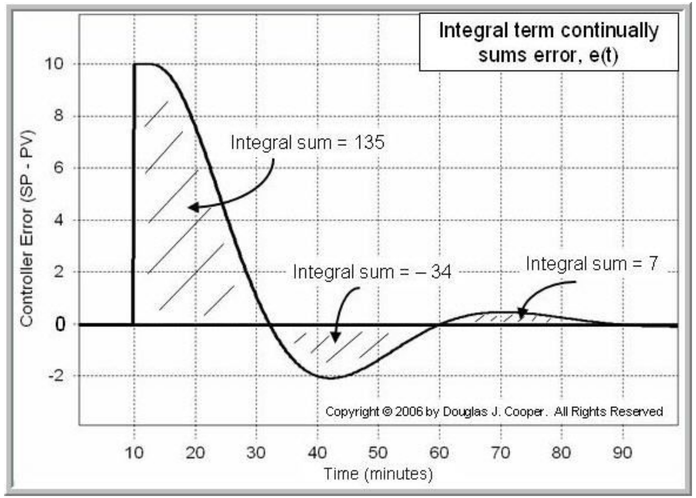
P-REGULATOR

- The output is **proportional to the deviation of the output signal from the set value** (setpoint)
  - If the input signal is equal to the set value, then the output is equal to zero
- Due to static (steady-state) error, the output of the proportional controller never stabilizes at the setpoint
- The larger the proportional coefficient (gain), the smaller the static error, however, if the gain is too large, **self-oscillations** may begin in the system, and with a further increase in the coefficient, the system may become unstable.

The heater cannot reach the set temperature because in this case, the error (and the power of the heater) will become equal to zero -> the kettle will start to cool down

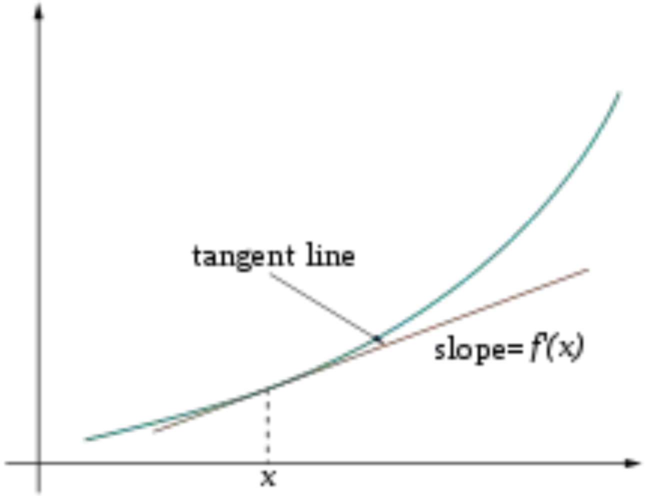
INTEGRAL COMPONENT

- The output is **proportional to the integral of the control error** over time
- Allows the regulator to account for the static error over time
- If there are no external disturbances, then the controlled variable will stabilize at the set value, the signal of the proportional component will be equal to zero, and the output signal will be fully provided by the integrating component
- The integral component can lead to self-oscillations if the coefficient is selected incorrectly

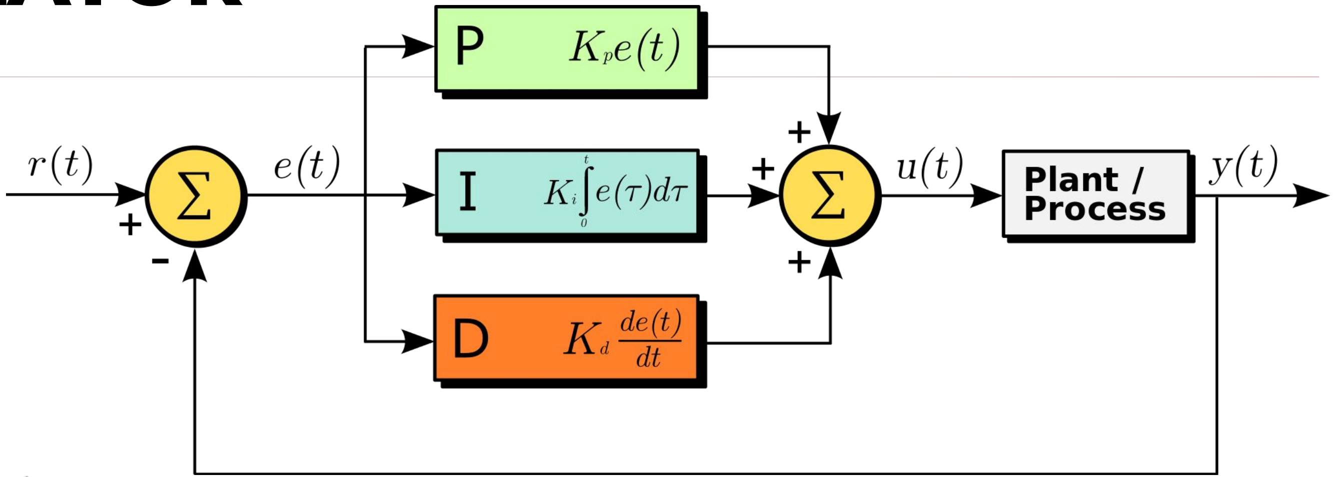


DIFFERENTIAL COMPONENT

- **Proportional to the rate of change of the control deviation**
- Designed to counteract future deviations from the set value
  - Deviations can be caused by external disturbances or delayed action of the regulator on the system



PID-REGULATOR

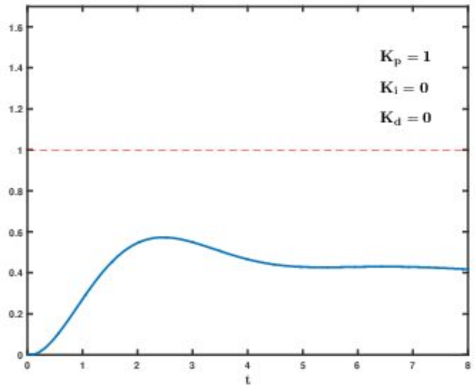


Continuous case:

$$u(t) = P + I + D = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

Discrete case:

$$U(n) = K_p E(n) + K_p K_{ip} T \sum_{k=0}^{n} E(k) + \frac{K_p K_{dp}}{T} (E(n) - E(n-1))$$



PID-REGULATOR. TUNUNG

- Regulator tuning methods
  - By method type
    - Manual
    - Heuristic-based
    - Optimization-based
  - By operation mode
    - Online methods
    - Offline methods

LQR-regulators

girafe
ai

04

LINEAR–QUADRATIC REGULATOR (LQR)

- One of the types of **optimal** regulators
- Uses a **quadratic** quality functional
- **Optimal control** is a control that provides for a given control object a control law that provides a maximum or minimum of a given quality functional

Continuous case:

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt$$

$$u = -R^{-1} B^T P x$$

Discrete case:

$$x_{k+1} = A x_k + B u_k$$

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k\right)$$

$$u_k = -F x_k$$

QUALITY FUNCTIONAL

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

Penalty for the failure of the system to achieve the desired state

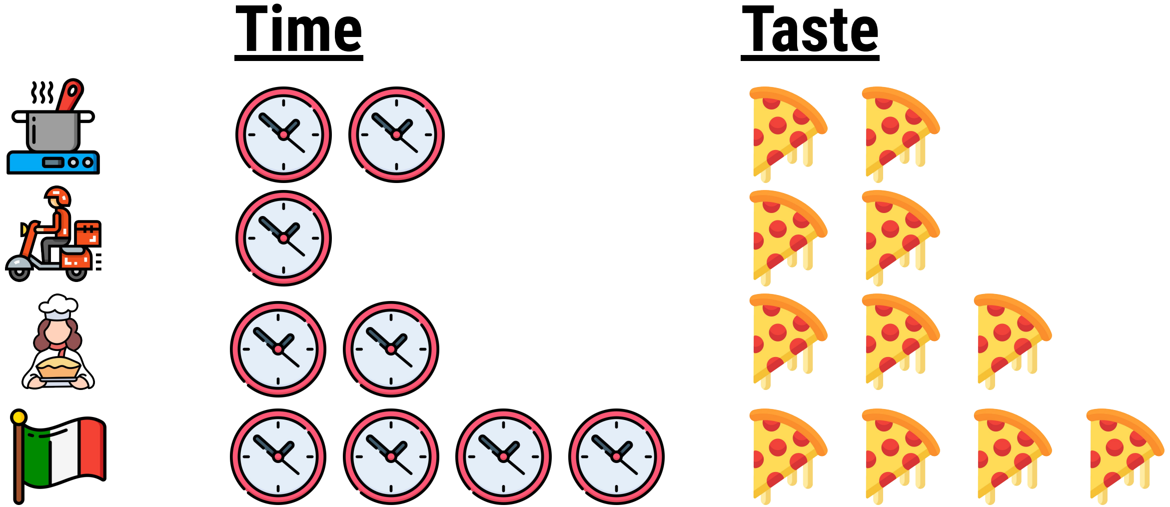
Penalty for resource consumption

Sum for all time moments (the penalty will increase until the system has reached the desired state and / or continues to consume resources)

Squaring so that deviations in the negative direction are not subtracted, but added to the penalty

QUALITY FUNCTIONAL

What if we want to eat pizza?



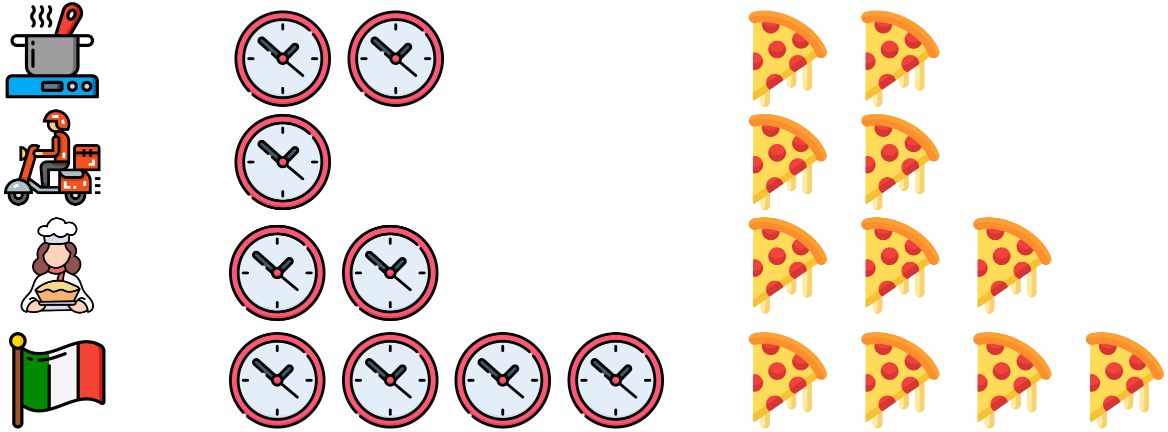
QUALITY FUNCTIONAL

What if we want to eat pizza?

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right)$$

Time

Taste



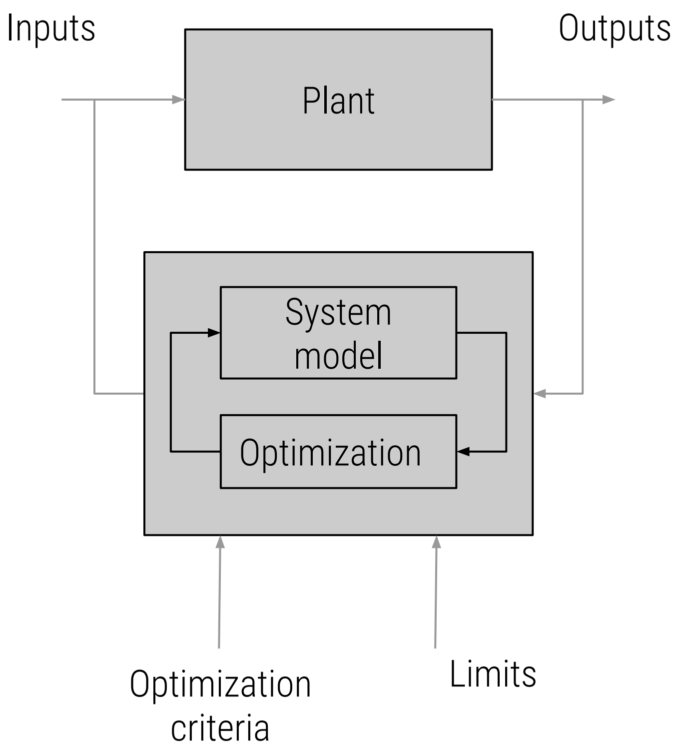
Model-predictive control

girafe
ai

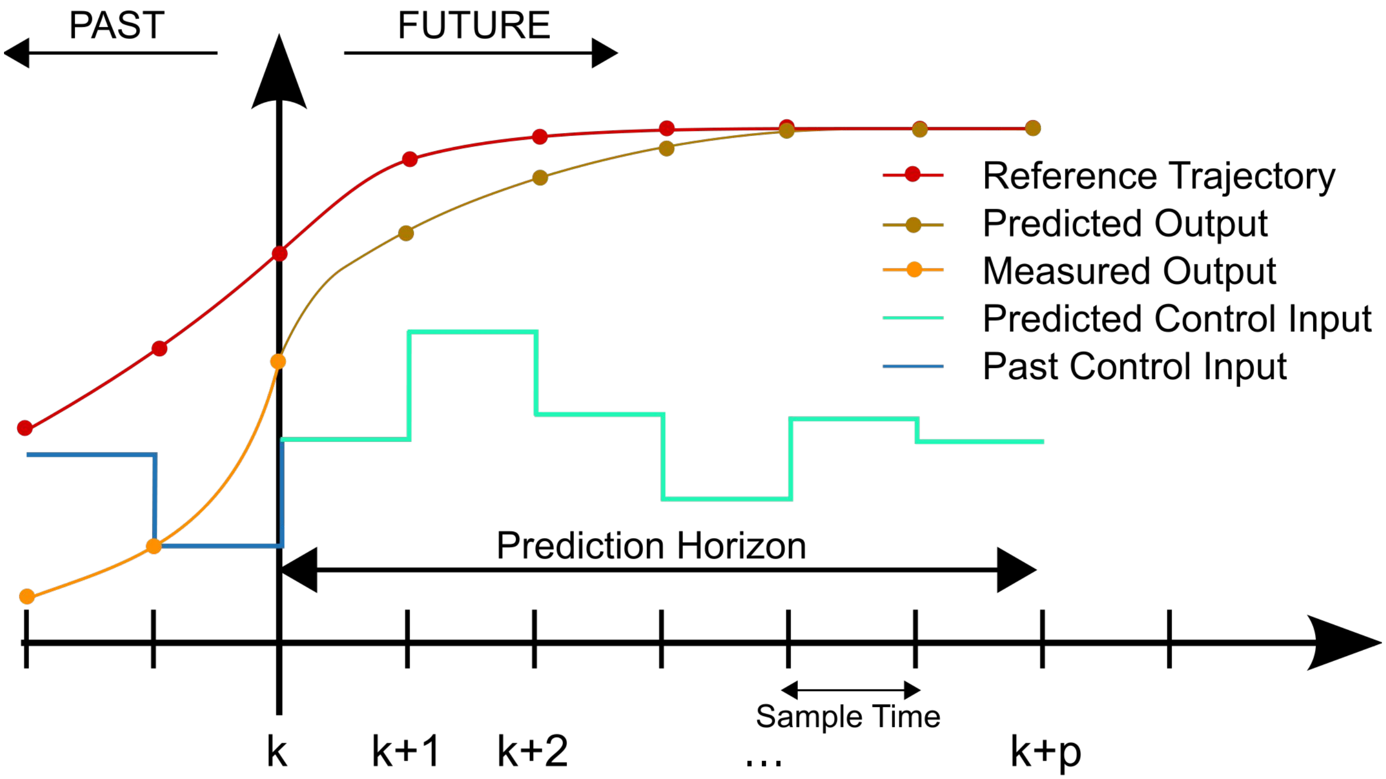
05

MODEL-PREDICTIVE CONTROL (MPC)

- The approach is also based on optimization
    - Optimization occurs on the **finite planning horizon**
    - For optimization, a model is used that predicts the behavior of the system
    - The first step of the optimal control law is performed
    - New optimization takes place on the planning horizon shifted by one time interval into the future



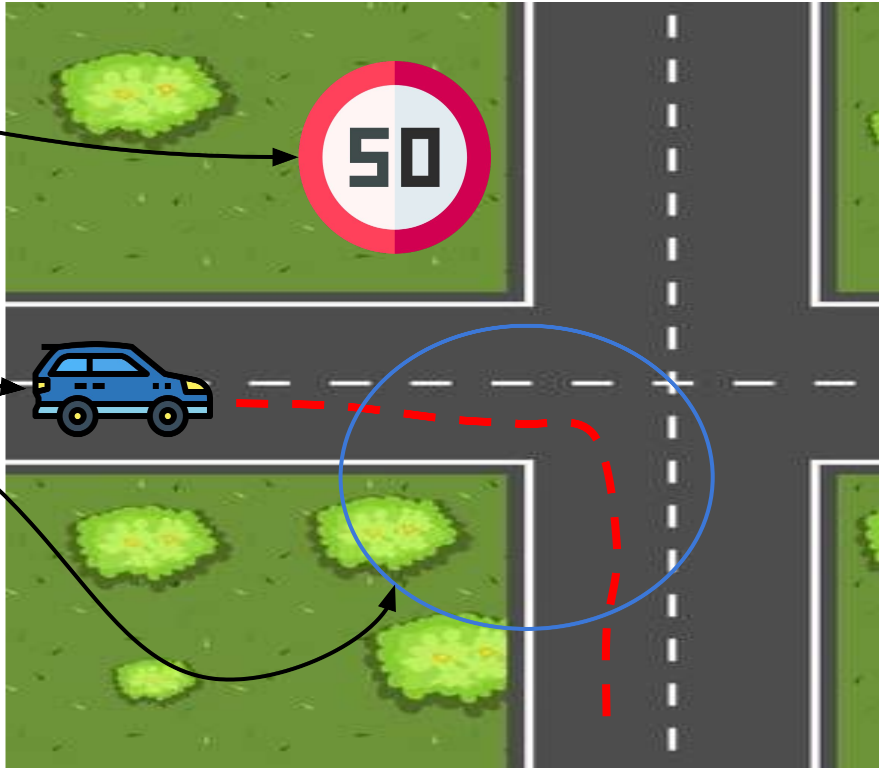
MODEL-PREDICTIVE CONTROL (MPC)



MODEL-PREDICTIVE CONTROL (MPC)

MPC is able to accommodate the constraints of the external environment and system

MPC is able to accommodate future events in the control law



TRAJECTORY CONTROL EXAMPLE. STANLEY

Stanley: The Robot that Won the DARPA Grand Challenge

Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, and Pascal Stang
Stanford Artificial Intelligence Laboratory
Stanford University
Stanford, California 94305

TRAJECTORY CONTROL EXAMPLE. STANLEY

Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing†

Gabriel M. Hoffmann, Claire J. Tomlin
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305, USA
{gabeh,tomlin}@stanford.edu

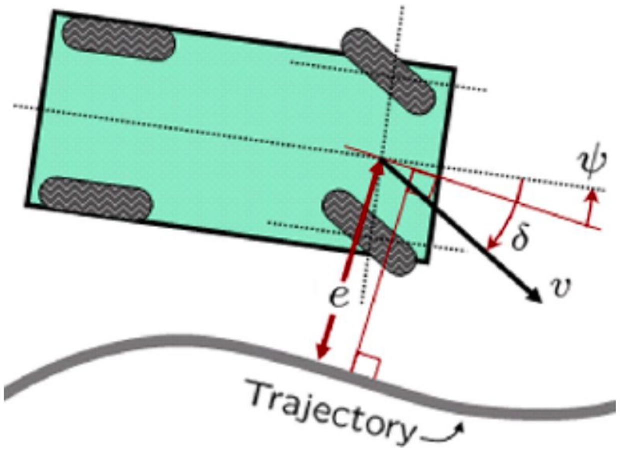
Michael Montemerlo, Sebastian Thrun
Computer Science Department
Stanford University
Stanford, CA 94305, USA
{mmde,thrun}@stanford.edu

***Abstract*— This paper presents a nonlinear control law for an automobile to autonomously track a trajectory, provided in real-time, on rapidly varying, off-road terrain. Existing methods can suffer from a lack of global stability, a lack of tracking accuracy, or a dependence on smooth road surfaces, any one of which could lead to the loss of the vehicle in autonomous off-road driving. This work treats automobile trajectory tracking in a new manner, by considering the orientation of the front wheels – not the vehicle's body – with respect to the desired trajectory, enabling collocated control of the system. A steering control law is designed using the kinematic equations of motion, for which global asymptotic stability is proven. This control law is then augmented to handle the dynamics of pneumatic tires and of the servo-actuated steering wheel. To control vehicle speed, the brake and throttle are actuated by a switching**

Fig. 1. Stanley, the Stanford Racing Team's entry in the DARPA Grand Challenge 2005, under autonomous control, with no human in the vehicle.

TRAJECTORY CONTROL EXAMPLE. STANLEY

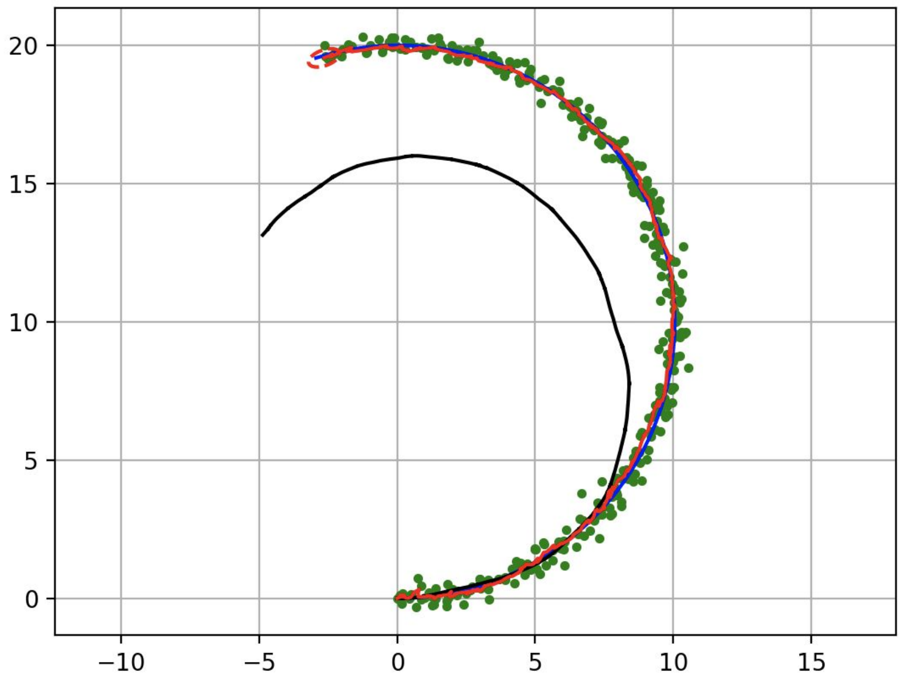
- ❑ $v(t)$ – speed
- ❑ $e(t)$ – crosstrack error
- ❑ $\psi(t)$ – yaw angle relative to the closest reference track segment
- ❑ $\delta(t)$ - steering angle relative to the axis of symmetry
- ❑ $(\psi(t) - \delta(t))$ – steering angle relative to the closest reference track segment



ADDITIONAL RESOURCES

1. Video: Controlling Self Driving Car with PID
2. Video: What is LQR control?
3. Video: Understanding Model Predictive Control
4. Paper: Stanley: The Robot that Won the DARPA Grand Challenge
5. Paper: Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing
6. PID regulator online demo, recorded demo

Practical notice: Python Robotics



Many algorithms implemented in Python in this repo: https://github.com/AtsushiSakai/PythonRobotics

It is very handful for understanding main concepts behind each algorithm, so check it out!

Thanks for attention!

Questions? Additions? Welcome!