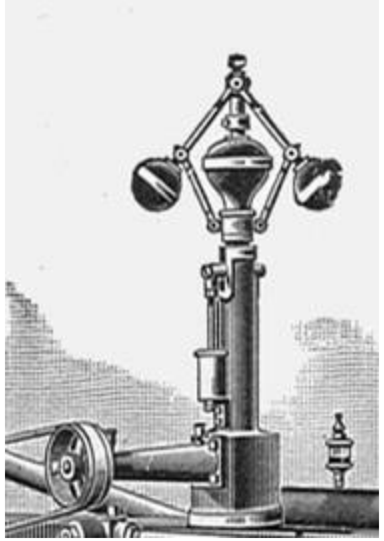
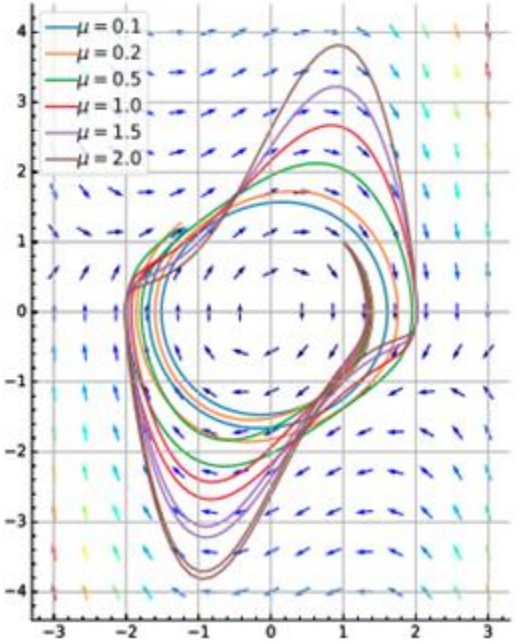
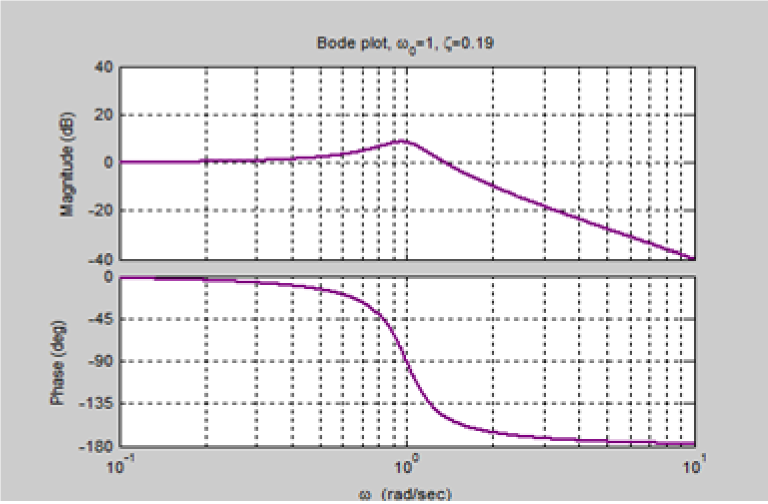


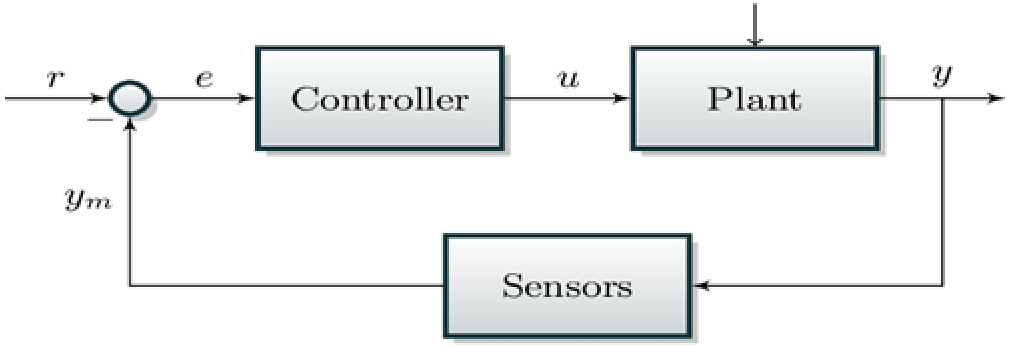


Introduction to **Control Systems**

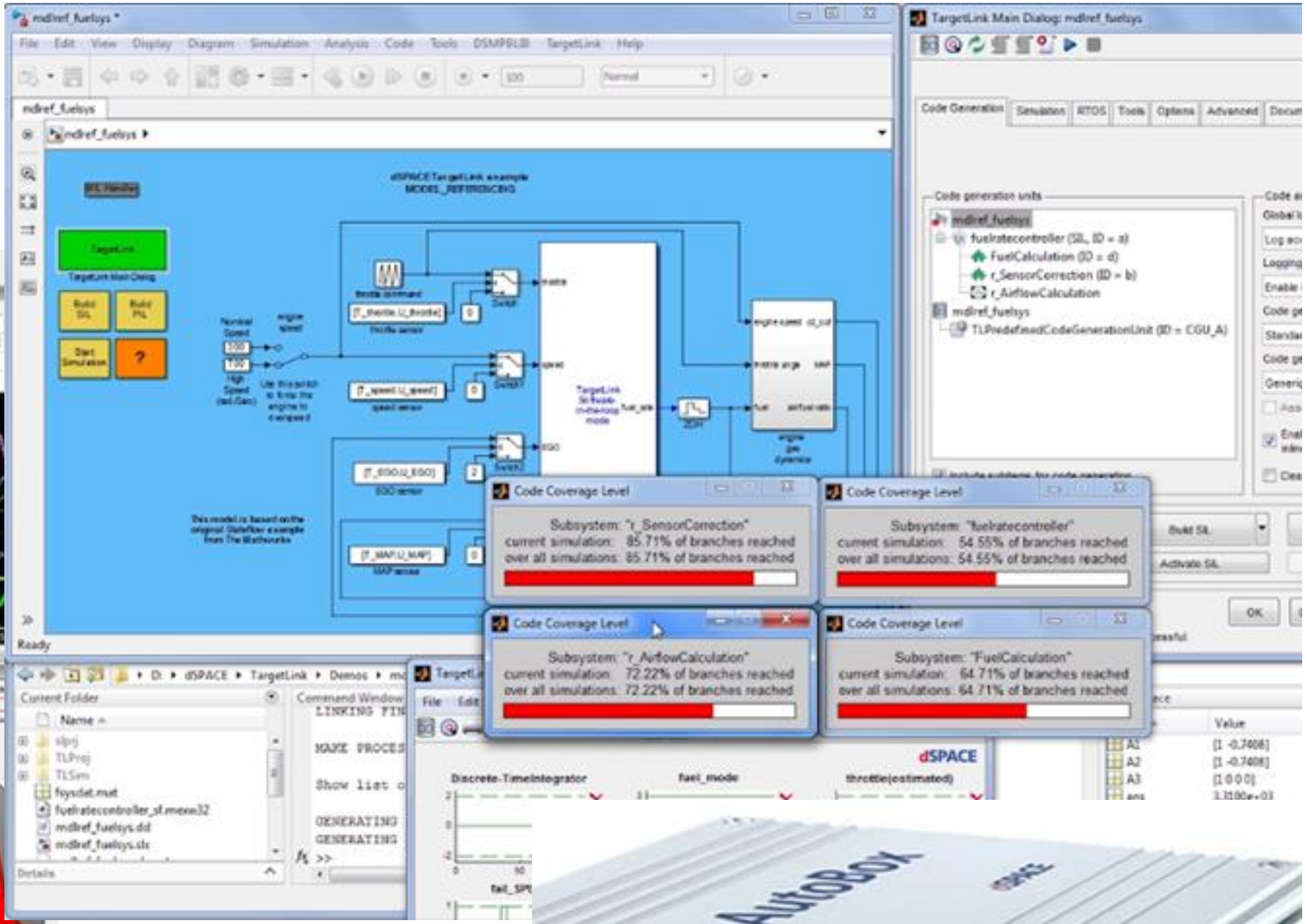
Theory and applications

Enrico Regolin / Laura Nenzi






AVL FUTURE HYBRID
G AVL 1


Code Coverage Level
Subsystem: "r_SensorCorrection"
current simulation: 85.71% of branches reached
over all simulations: 85.71% of branches reached
Code Coverage Level
Subsystem: "fuelratecontroller"
current simulation: 54.55% of branches reached
over all simulations: 54.55% of branches reached
Code Coverage Level
Subsystem: "r_AirflowCalculation"
current simulation: 72.22% of branches reached
over all simulations: 72.22% of branches reached
Code Coverage Level
Subsystem: "FuelCalculation"
current simulation: 64.71% of branches reached
over all simulations: 64.71% of branches reached


MicroAutoBox
dSPACE
ECU
USB
Ethernet
Host-PC

Overview (1)

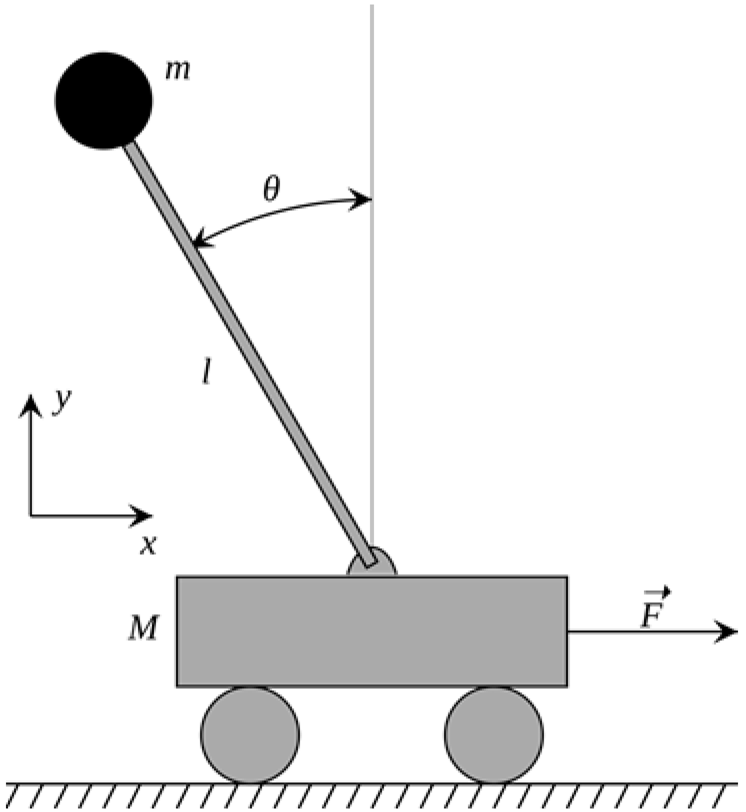
- Linear Control (time domain)
  - Introduction
  - Dynamical Linear Systems
  - Observability & Controllability
  - PID Controllers
  - Luenberger Observer

- Linear Control (frequency domain) NOT IN THIS COURSE

Overview (2)

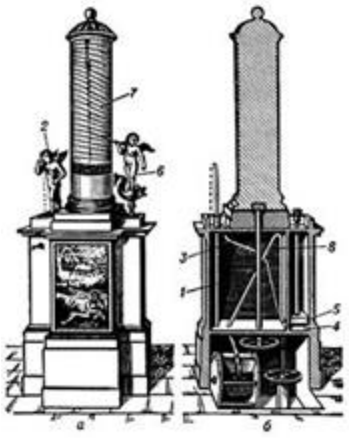
- Optimal Control and KF Estimation
  - Optimal Control (LQR)
  - Model Predictive Control
  - Kalman Filtering

- Control Laboratory
  - Matlab/Simulink
  - Kalman Filtering and Optimal Control
  - Cart-pole



Control Systems History

- Water Clock
  - Alexandria
    (Ctesibius, 3rd century BC)



- Centrifugal Governor
  - Windmills
    (C. Huygeens, 17th century)
  - Steam Engine
    (J. Watt, 1788)

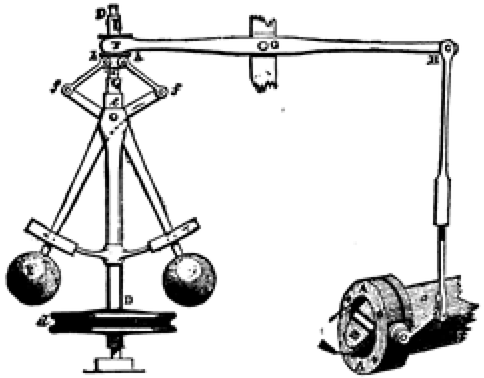
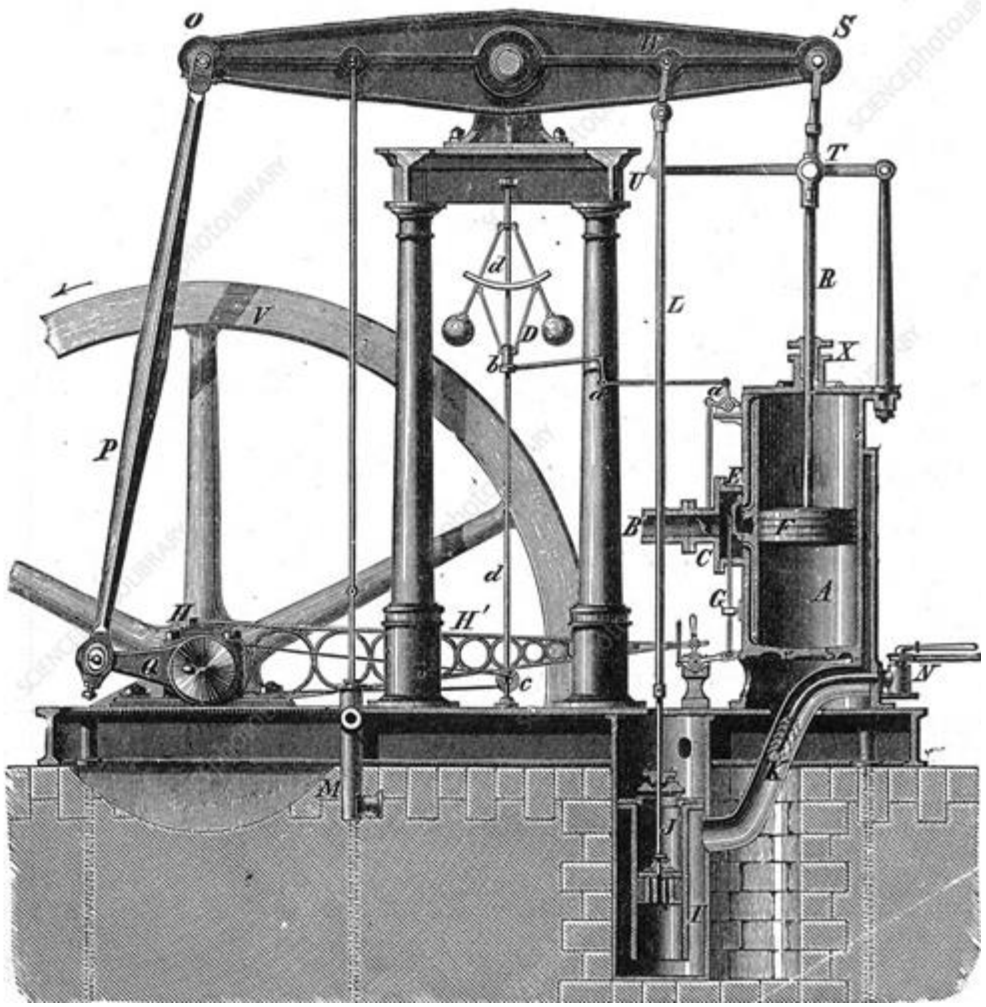
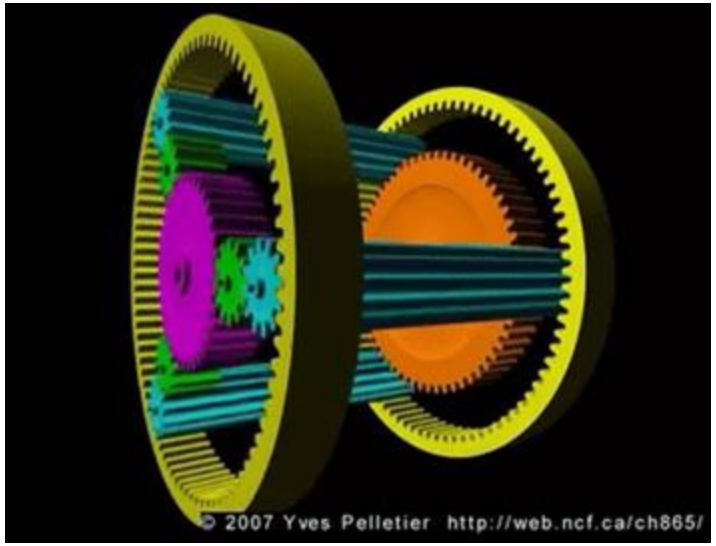


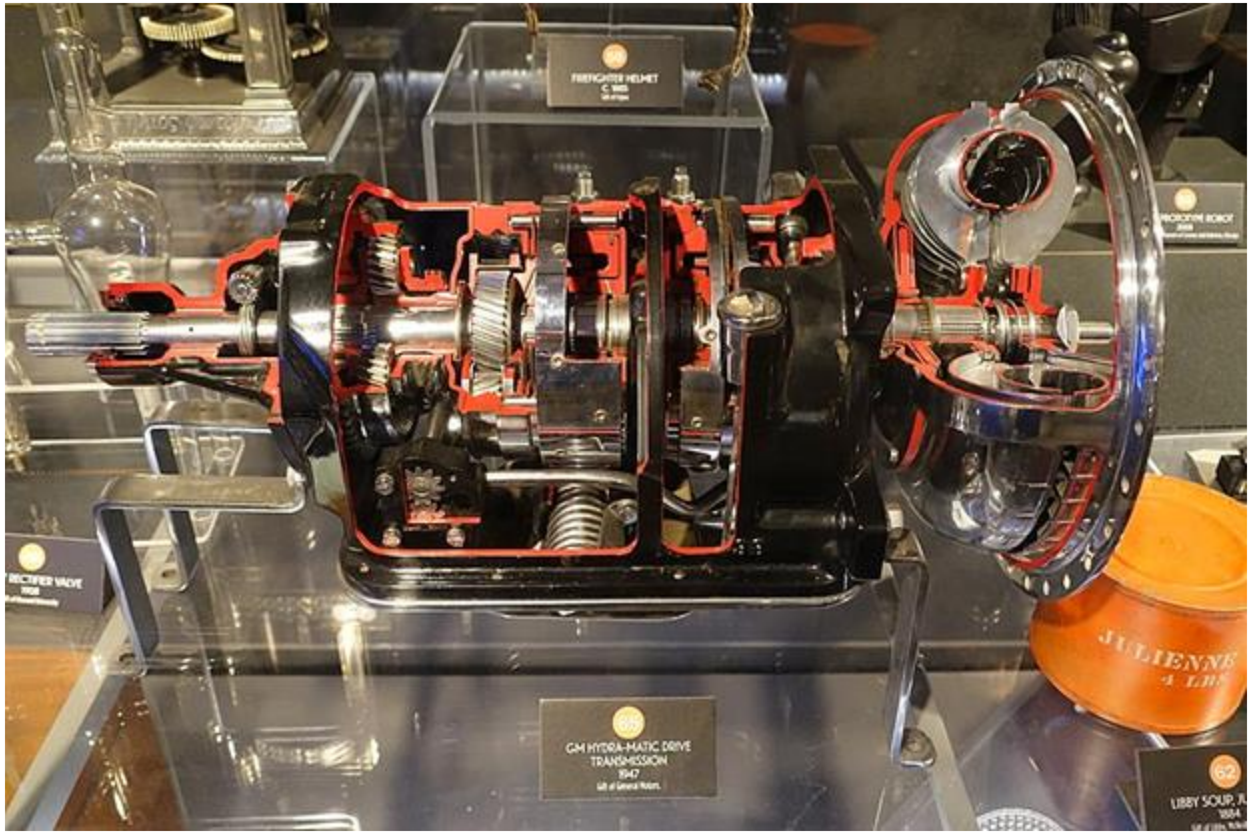
FIG. 4.—Governor and Throttle-Valve.



Control Systems History

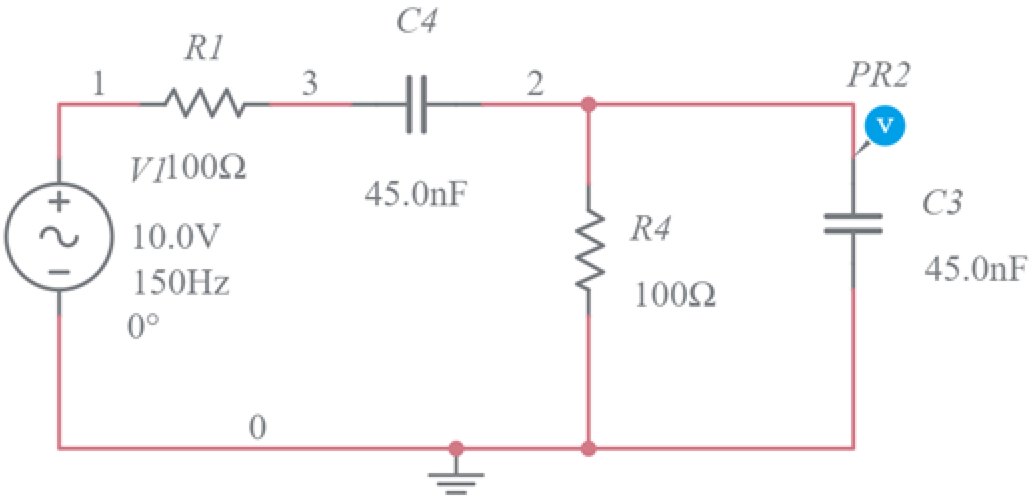


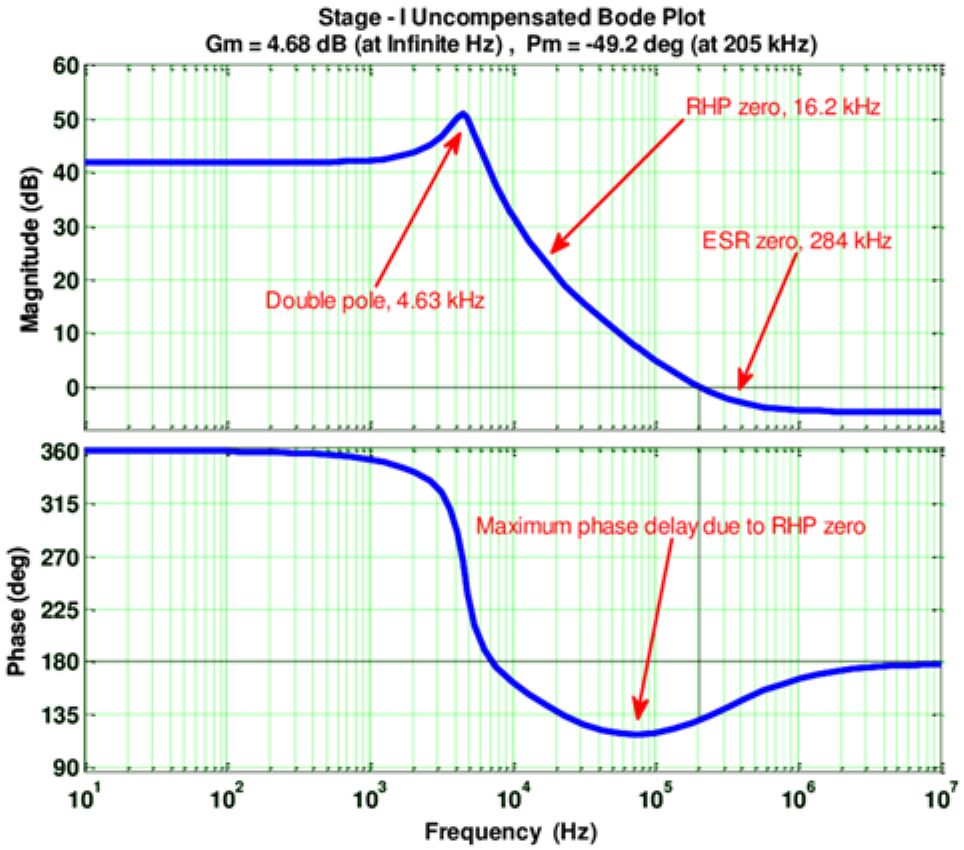
- First Automatic Transmission (Hydramatic, General Motors, 1939)



Control Systems History

- Classical control theory formalized from circuits theory





[Tacoma Bridge Collapse]

Linear Control (time domain)

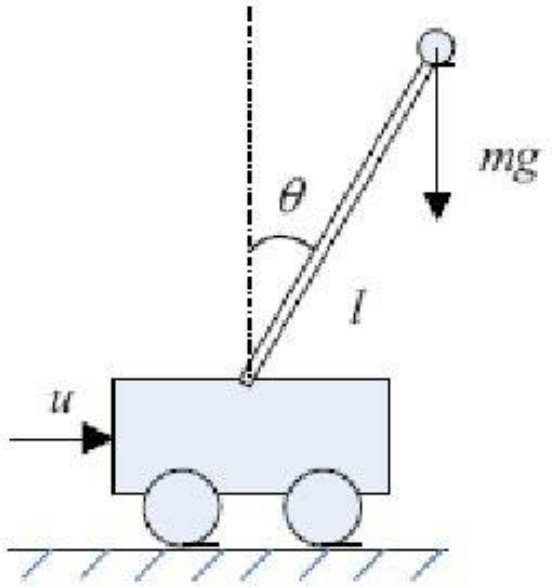
Control Systems Fundamentals

REQUIRED

•Dynamical System MODEL

•Control Input

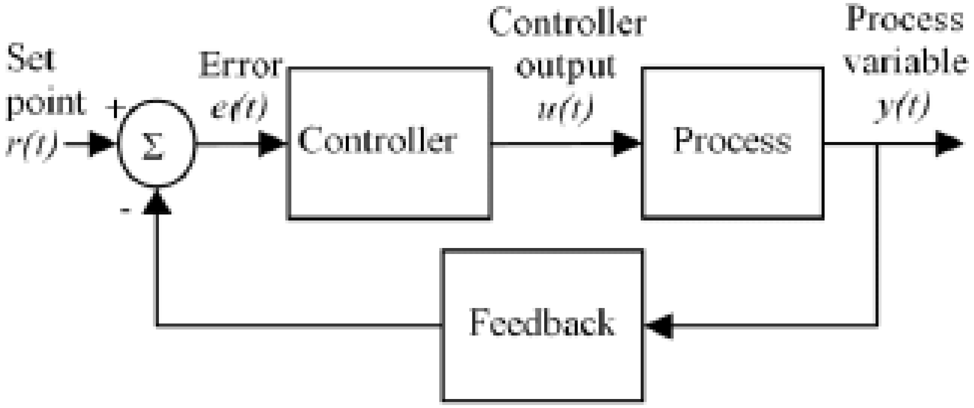
•Reference Signal



CHALLANGES

•Missing/Noisy Information

•Physical limitations



Dynamical Systems (1)

Past history (state) influences future output

- **Continuous Time** **vs.** **Discrete Time**

$$\dot{x} = f(x), \quad t \in [0, \infty) \qquad\qquad x(k+1) = f(x(k)), \quad k = 0, 1, 2, \ldots$$

- **Autonomous** **vs.** **Non-autonomous**

$$\dot{x} = f(x) \qquad\qquad \dot{x} = f(x, u)$$

- **Linear** **vs.** **Non-linear**

$$\begin{aligned} \dot{x}_1 &= -2x_2 \\ \dot{x}_2 &= 0.5x_1 + x_2 + 0.4u \end{aligned} \qquad\qquad \begin{aligned} \dot{x}_1 &= -x_1 x_2 \\ \dot{x}_2 &= 0.5x_1^2 + \sin(x_2) + \frac{0.4}{u} \end{aligned}$$

Dynamical Systems (2)

- **SISO** **vs.** **MIMO**

$$\dot{x} = Ax + b \cdot u$$
$$y = Cx (= 0.5x_1)$$

$$\dot{x} = Ax + B\mathbf{u}$$
$$\mathbf{y} = Cx$$

- **Time Invariant** **vs.** **Time Variant**

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

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

- **Deterministic** **vs.** **Non-Deterministic (Stochastic, noisy, etc.)**

$$\dot{x} = -x^2 - x + u$$
$$y = 0.5x$$

$$x(k+1) = -(2+\nu)x(k)^2 - x(k) + u(k)$$
$$y(k) = 0.5x(k) + \eta$$
$$\nu \sim N(\mu, \sigma), \eta \sim U(0, 1)$$

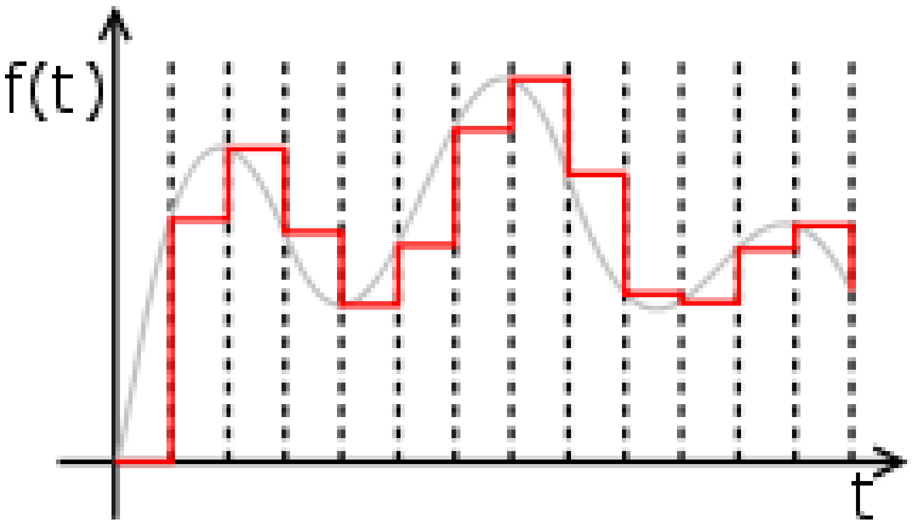
Dynamical Systems (3)

- **LTI systems --- State-Space representation** $x(0) = x_0, \ x \in \mathbb{R}^n$

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$A_d = e^{A\Delta T}$$
$$B_d = A^{-1}(e^{A\Delta T} - 1)B$$

$$x(k+1) = A_d x(k) + B_d u(k)$$
$$y(k) = Cx(k) + Du(k)$$



Dynamical Systems (3)

- **LTI systems --- State-Space representation** $x(0) = x_0, \ x \in \mathbb{R}^n$

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$A_d = e^{A\Delta T}$$
$$B_d = A^{-1}(e^{A\Delta T} - 1)B$$

$$x(k+1) = A_d x(k) + B_d u(k)$$
$$y(k) = Cx(k) + Du(k)$$

- **Output response (continuous time)**

$$y(t) = \boxed{Ce^{At}x_0} + \boxed{C\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau} + Du(t)$$

Free Response (homogeneous solution)

Effect of input

- **Output response (discrete time)**

$$y(k) = CA_d^k x_0 + C\sum_{i=0}^{k-1} A_d^{k-1-i} B_d u(i) + Du(k)$$

Stability condition (Hurwitz)
$x(t) = e^{at}$ $a < 0$ $a > 0$ $real(eig(A)) < 0$
$x(k) = a^k$ $\lvert a\rvert < 1$ $\lvert a\rvert > 1$ $\lvert eig(A_d)\rvert < 1$

State-Space Realizations

Similarity Transformations

- The choice of a state-space model for a given system is not unique.
- For example, let T be an invertible matrix, and consider a coordinate transpormation $x = T\tilde{x}$, i.e., $\tilde{x} = T^{-1}x$. This is called a similarity transformation.
- The standard state-space model can be written as

$$\begin{cases} \dot{x} &= Ax + Bu, \\ y &= Cx + Du. \end{cases} \Rightarrow \begin{cases} T\dot{\tilde{x}} &= AT\tilde{x} + Bu, \\ y &= CT\tilde{x} + Du. \end{cases}$$

i.e.,

$$\begin{aligned} \dot{\tilde{x}} &= (T^{-1}AT)\tilde{x} + (T^{-1}B)u = \tilde{A}\tilde{x} + \tilde{B}u \\ y &= (CT)\tilde{x} + Du = \tilde{C}\tilde{x} + \tilde{D}u. \end{aligned}$$

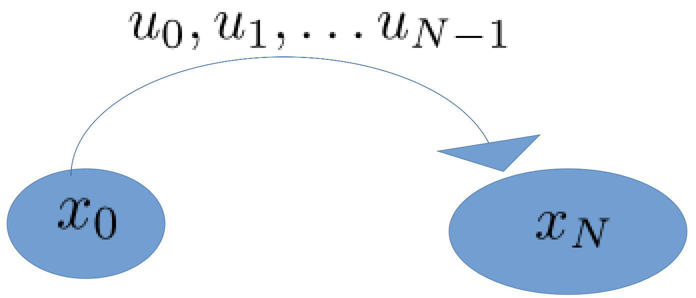
- You can check that the time response is exactly the same for the two models (A, B, C, D) and $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$!

LTI Systems Properties

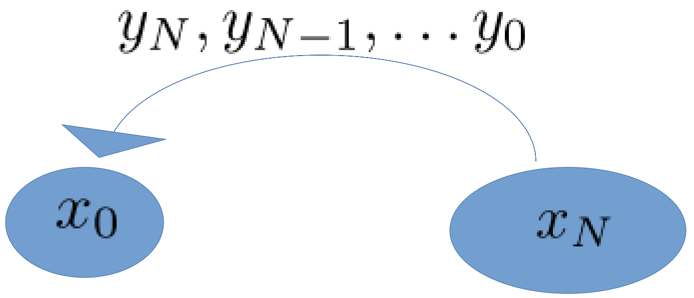
Discrete case

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k)$$

Reaching a state



"Observing" the initial state



LTI Systems Properties

Conditions for all LTI systems:

- <u>Controllability</u> $\iff rank(\mathcal{C}) = n$

$$\mathcal{C} = \left[B, AB, A^2B, \ldots, A^{n-1}B\right]$$

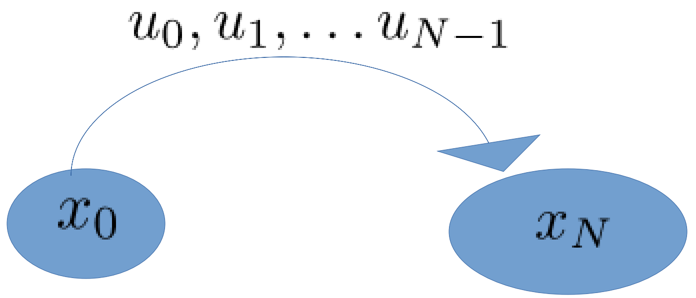
- <u>Observability</u> $\iff rank(\mathcal{O}) = n$

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \cdots \\ CA^{n-1} \end{bmatrix}$$

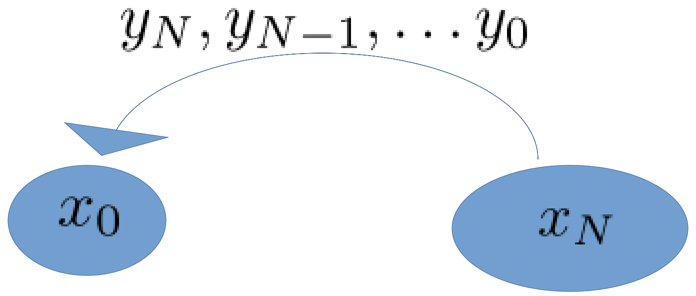
Discrete case

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k)$$

Reaching a state



"Observing" the initial state



LTI Systems Properties

- Pair (A,B) is "Controllable" $\Leftrightarrow rank(\mathcal{C}) = n$
- Pair (A,C) is "Observable" $\Leftrightarrow rank(\mathcal{O}) = n$
- LTI System $\mathcal{S} : \{A, B, C\}$ is a "**minimal state-space realization**" if it is both observable and controllable.
- Example:

$$\mathcal{S}_0 : \{A_0, B, C\}\,, \quad \mathcal{S}_1 : \{A_1, B, C\}$$

$$B = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$A_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix}$$

$$\mathcal{C}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathcal{O}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$$rank(\mathcal{C}_0) = 1 \qquad rank(\mathcal{O}_0) = 2$$

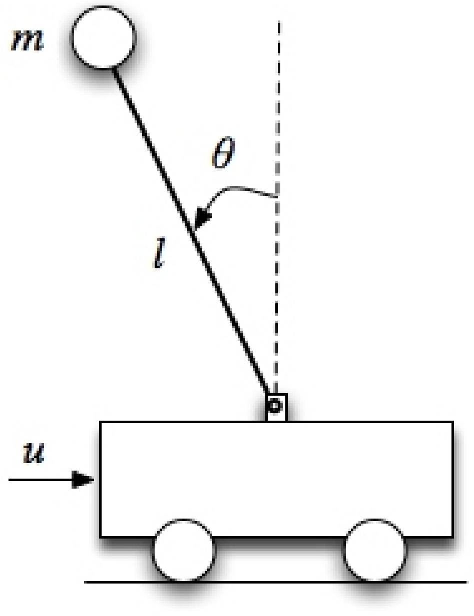
$$\mathcal{C}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad \mathcal{O}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$rank(\mathcal{C}_1) = 3 \qquad rank(\mathcal{O}_1) = 3$$

non-LTI Systems (example)

Is the inverted pendulum (cartpole) controllable?

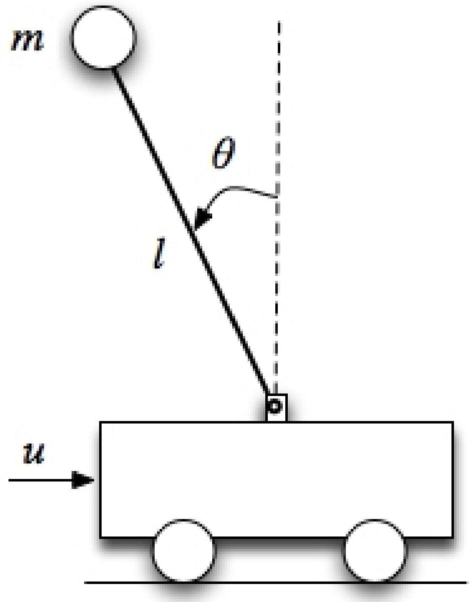
$$\begin{cases} \ddot{p} & = \dfrac{u + m\, l\, \dot{\theta}^2\ \sin\theta - m\, g\ \cos\theta \sin\theta}{M + m \sin\theta^2} \\ \ddot{\theta} & = \dfrac{g\ \sin\theta - \cos\theta \ddot{p}}{l} \end{cases}$$



In non-linear systems Controllability and Observability Matrices represent LOCAL properties.

non-LTI Systems (example)

Is the inverted pendulum (cartpole) controllable?



$$\begin{cases} \ddot{p} & = \frac{u + m\,l\,\dot{\theta}^2\ \sin\theta - m\,g\ \cos\theta\sin\theta}{M + m\sin\theta^2} \\ \ddot{\theta} & = \frac{g\ \sin\theta - \cos\theta\ddot{p}}{l} \end{cases}$$

In non-linear systems Controllability and Observability Matrices represent LOCAL properties.

$$\dot{x} = f(x,u), \quad \text{eq.point } x_0, u_0$$
$$\dot{x} = \underline{A}x + \underline{B}u$$

$$\underline{A} = \frac{\partial f(x,u)}{\partial x}|_{x=x_0, u=u_0}$$
$$\underline{B} = \frac{\partial f(x,u)}{\partial u}|_{x=x_0, u=u_0}$$

$$x = \left[p,\ \dot{p},\ \theta,\ \dot{\theta}\right]^T$$
$$\frac{\partial f}{\partial u} = \left[0,\ \frac{1}{(M + m(1-\cos^2(\theta))},\ 0,\ \frac{-\cos(\theta)}{l\,(M + m(1-\cos^2(\theta))}\right]^T$$

non-LTI Systems (example)

Linearization

$$\dot{x} = f(x,u), \quad \text{eq.point } x_0, u_0$$
$$\dot{x} = \underline{A}x + \underline{B}u$$

$$\underline{A} = \frac{\partial f(x,u)}{\partial x}_{|x=x_0,u=u_0}$$
$$\underline{B} = \frac{\partial f(x,u)}{\partial u}_{|x=x_0,u=u_0}$$

$$(\dot{x} = 0,\ \theta_0 = 0,\ \dot{\theta}_0 = 0,\ u_0 = 0)$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -gm/M & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \alpha & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1/M \\ 0 \\ -1/(Ml) \end{bmatrix}$$

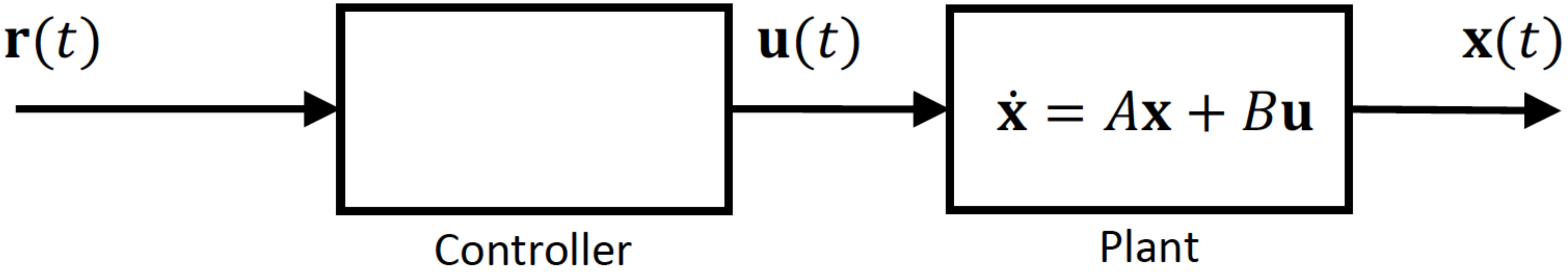
$$\alpha = \frac{(m+M)g}{Ml}$$

$$M = 1,\ m = 0.1,\ g = 9.81,\ l = 0.5$$

$$\mathcal{C} \approx \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \\ 0 & -2 & 0 & -43 \\ -2 & 0 & -43 & 0 \end{bmatrix}$$

$$rank(\mathcal{C}) = 4$$

Reference Tracking



Given a reference trajectory r(t), design u(t) such that x(t) closely follows r(t)

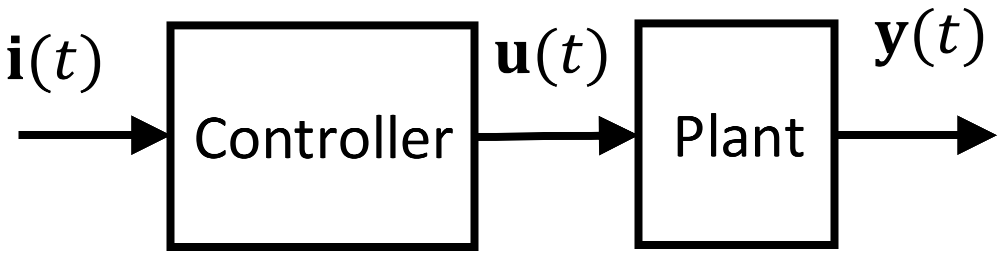
Control objectives:

- Reject disturbances (if there is some perturbation in state, making it get back to initial state)
- Follow reference trajectories (if we want the system to have a certain $\boldsymbol{x_{ref}}$)
- Make system follow some other “desired behavior”

Open-loop vs. Closed-loop control

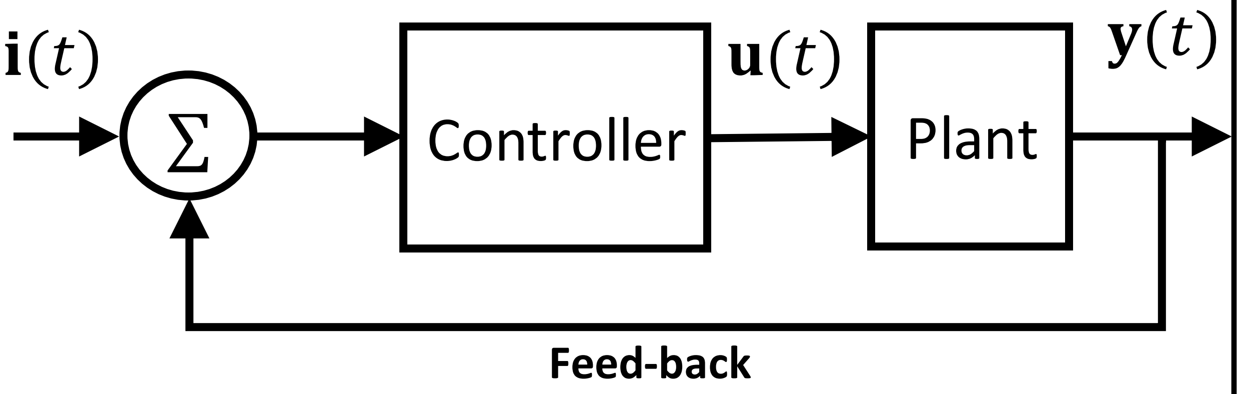
Open-loop or feed-forward control

- Control action does not depend on plant output
- Cheaper, no sensors required.
- Quality of control generally poor without human intervention

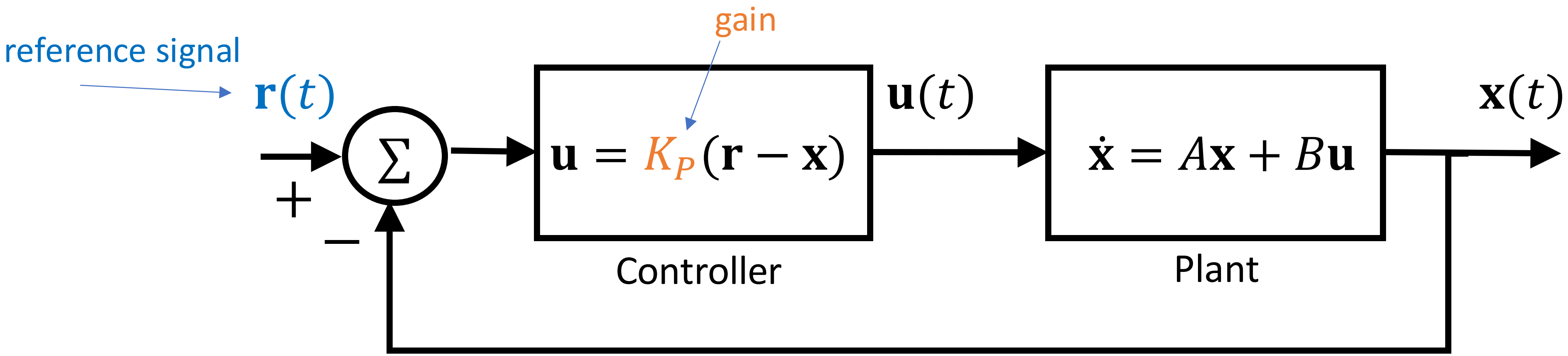


Feed-back control

- Controller adjusts controllable inputs in response to observed outputs
- Can respond better to variations in disturbances
- Performance depends on how well outputs can be sensed, and how quickly controller can track changes in output

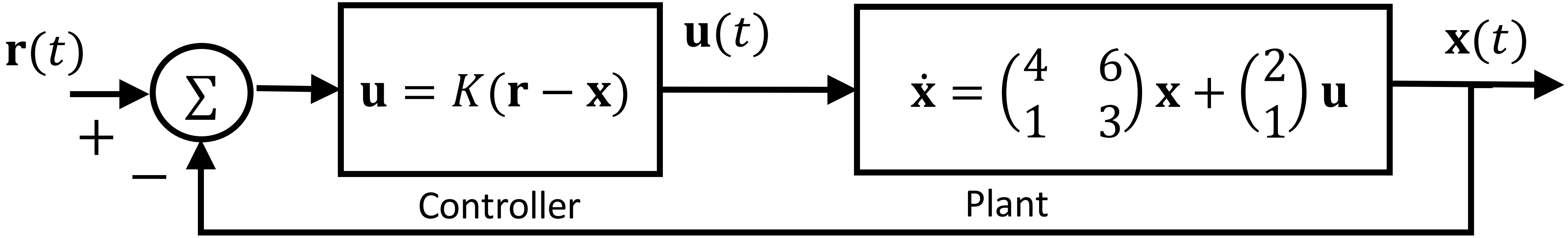


Proportional Controller



- Common objective: make plant state *track* the reference signal $\mathbf{r}(t)$
- $e = r - x$ is the error signal
- Closed-loop dynamics: $\dot{\mathbf{x}} = A\mathbf{x} + BK_P(\mathbf{r} - \mathbf{x}) = (A - BK_P)\mathbf{x} + BK_P\mathbf{r}$
- pick K_P s.t. the composite system is asymptotically stable, i.e. pick K_P such that eigenvalues of $(A - BK)$ have negative real-parts

Designing a pole placement controller

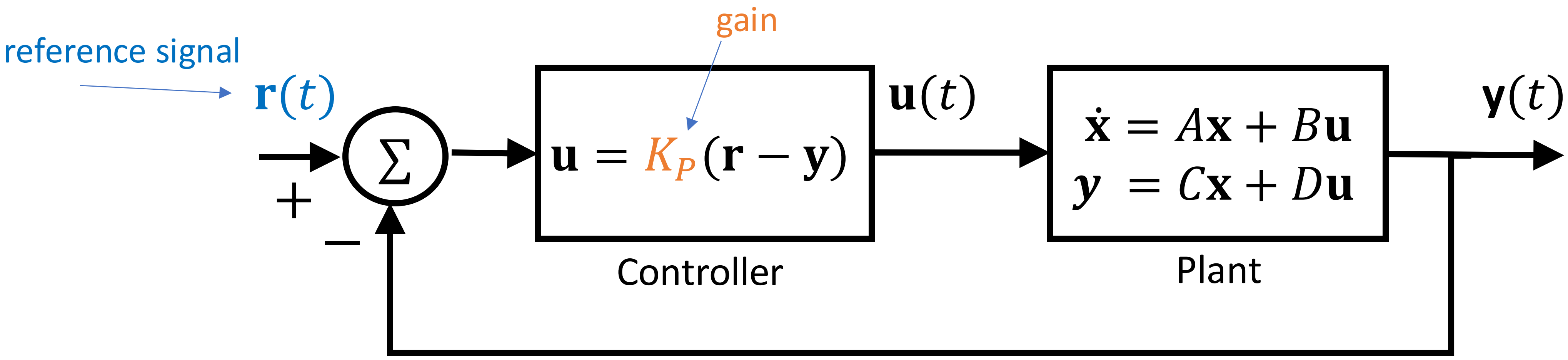


- $\text{eigs}(A)$ are values of λ that satisfy the equation $\det(A - \lambda I) = 0$
- Note $\text{eigs}(A) = 6, 1 \Rightarrow$ unstable plant!

- Let $K = (k_1 \quad k_2)$. Then, $A - BK = \begin{pmatrix} 4 - 2k_1 & 6 - 2k_2 \\ 1 - k_1 & 3 - k_2 \end{pmatrix}$

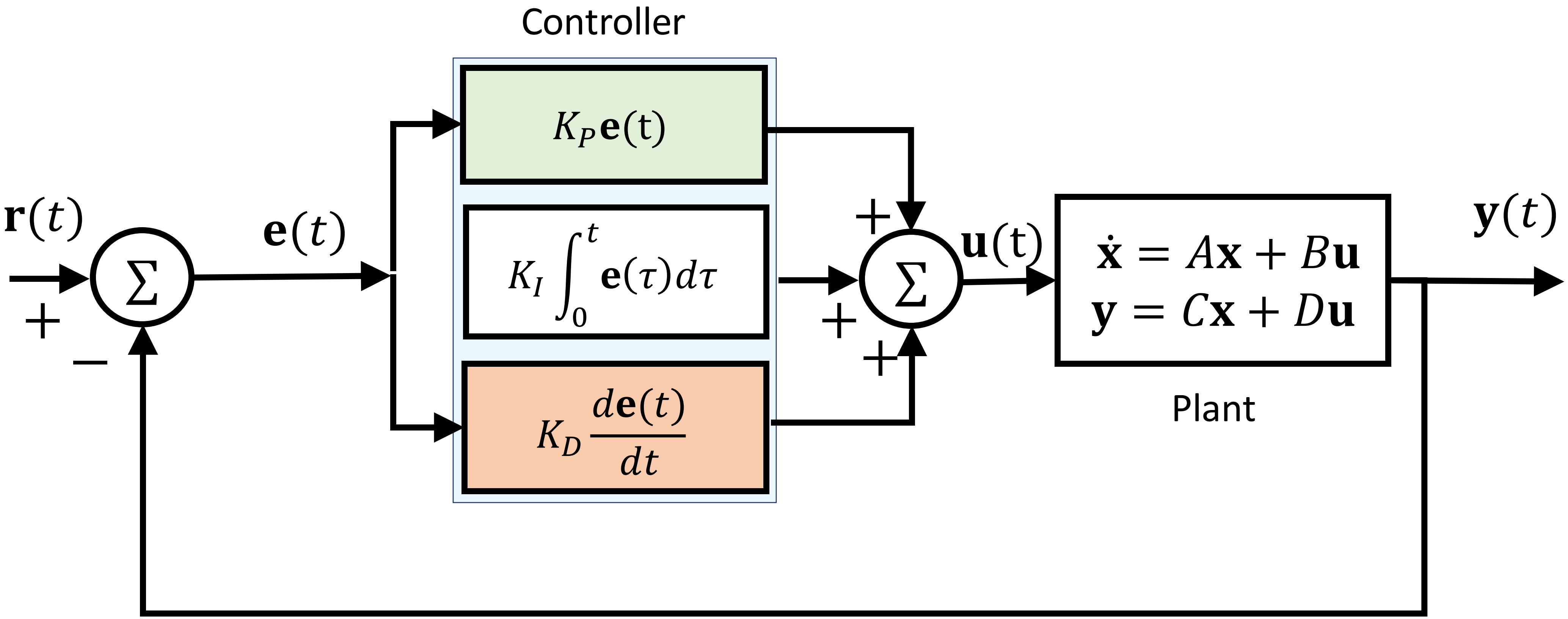
- $\text{eigs}(A - BK)$ satisfy equation $\lambda^2 + (2k_1 + k_2 - 7)\lambda + (6 - 2k_2) = 0$
  - two distinct solutions λ_1, λ_2 if $(\lambda - \lambda_1)\,(\lambda - \lambda_2) = \lambda^2 + (-\lambda_1 - \lambda_2)\lambda + \lambda_1\lambda_2$
  - That means $2k_1 + k_2 - 7 = -\lambda_1 - \lambda_2$ and $6 - 2k_2 = \lambda_1\lambda_2$
  - E.g. $\lambda_1 = -1$ and $\lambda_2 = -2$ gives $k_1 = 4, k_2 = 2$. Thus controller with $K = (4 \quad 2)$ stabilizes the plant!

Proportional Controller

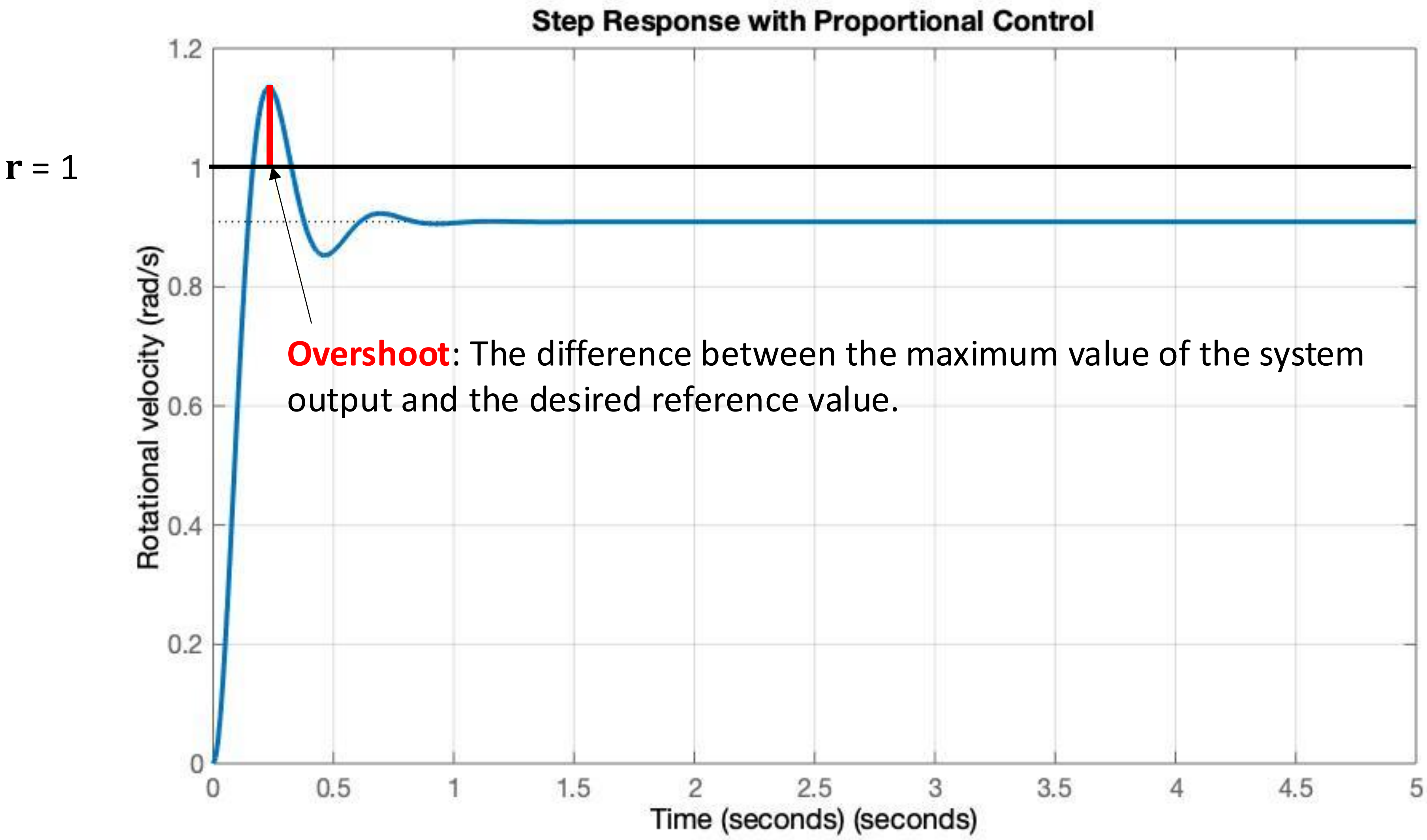


Proportional Integral Derivative (PID) controllers

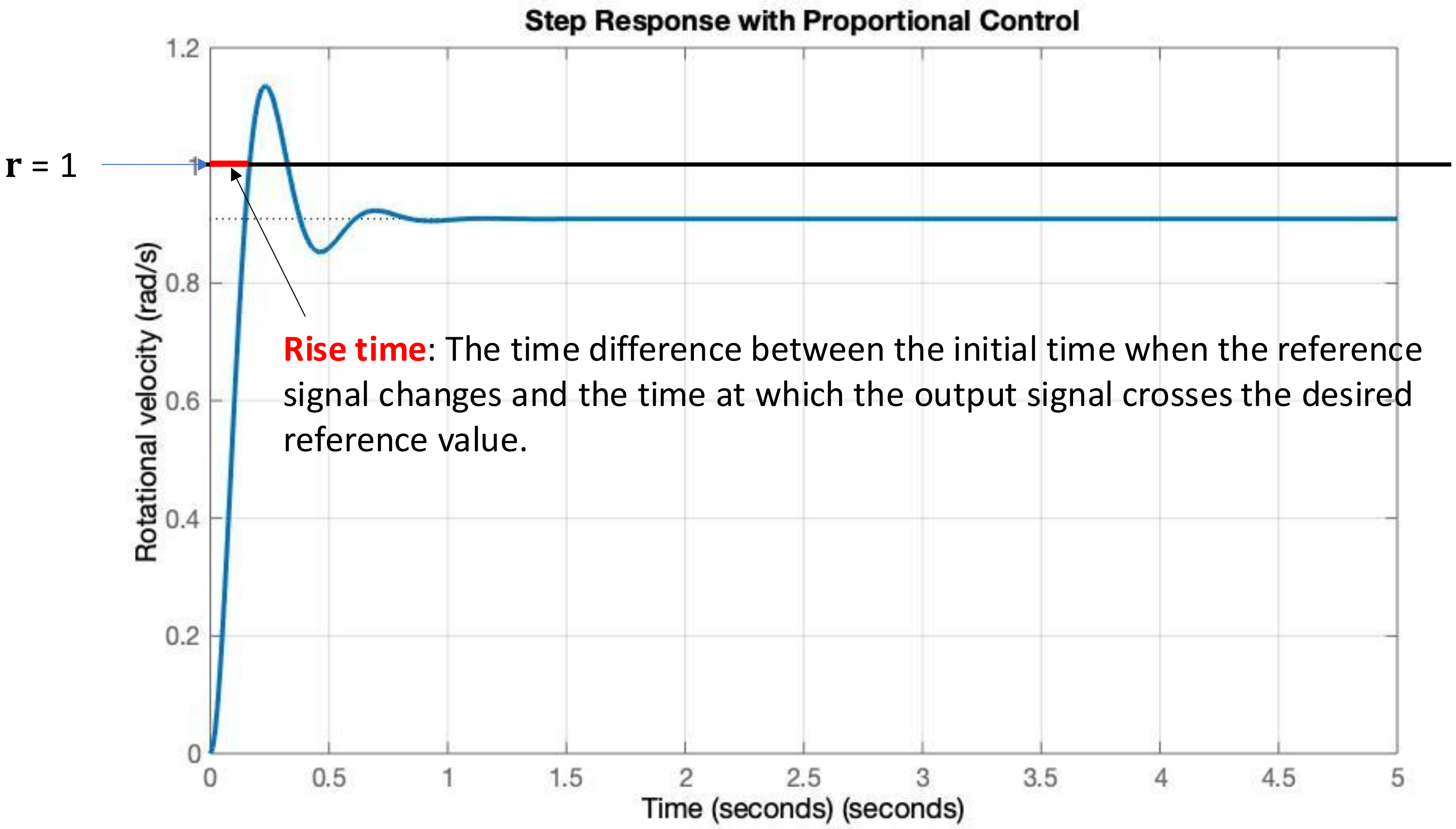
eigs(A) are values of λ that satisfy the equation $\det(A - \lambda I) = 0$
Note eigs$(A) = 6, 1 \Rightarrow$ unstable plant!



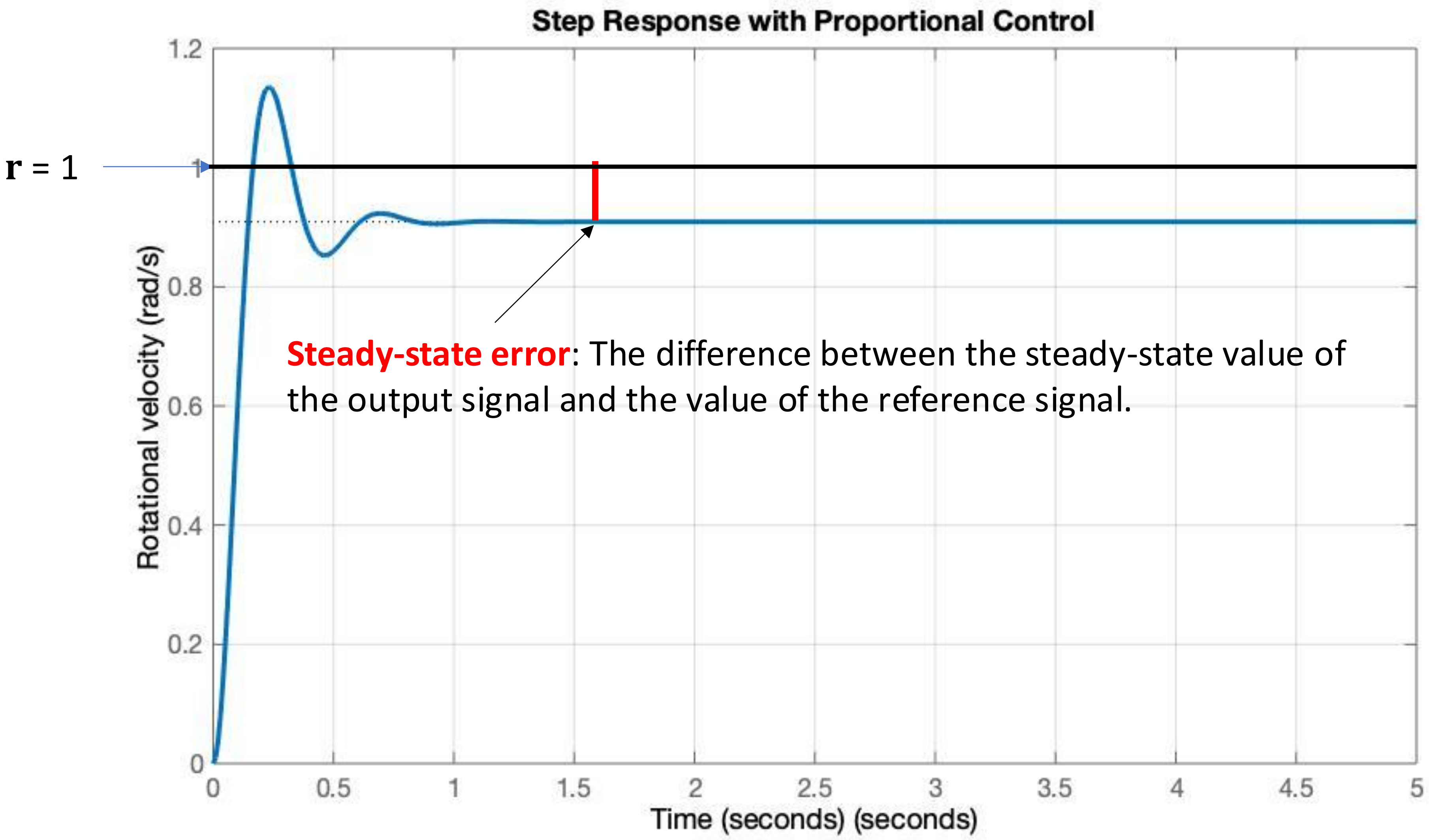
Measuring control performance



Measuring control performance

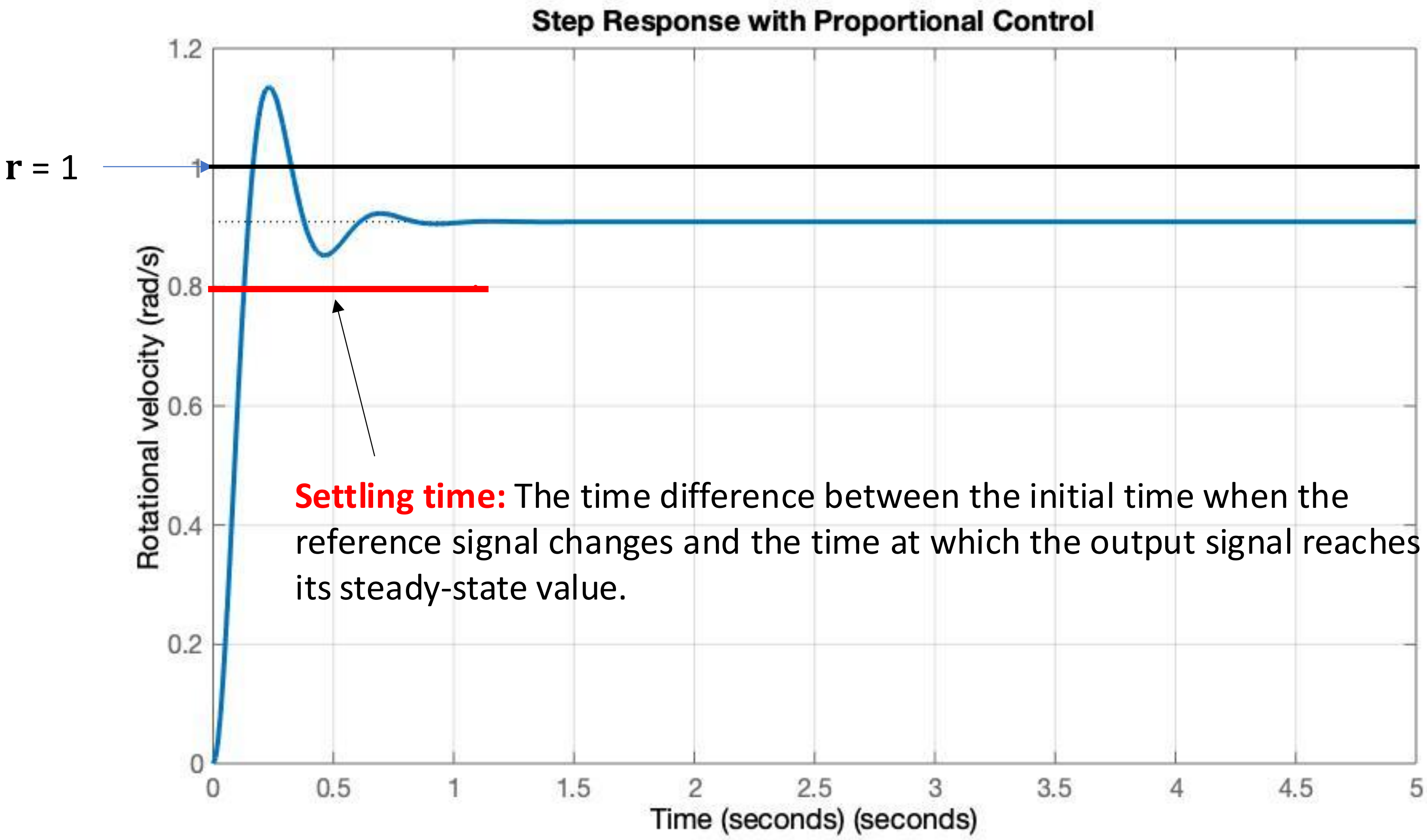


Measuring control performance



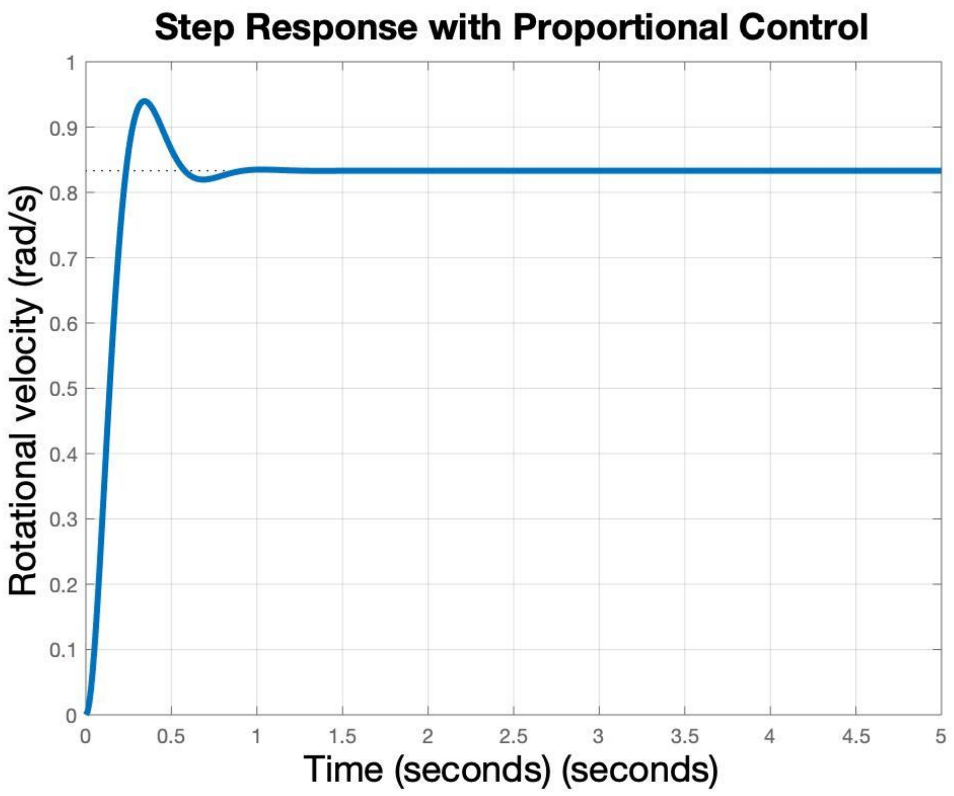
Steady-state error: The difference between the steady-state value of the output signal and the value of the reference signal.

Measuring control performance

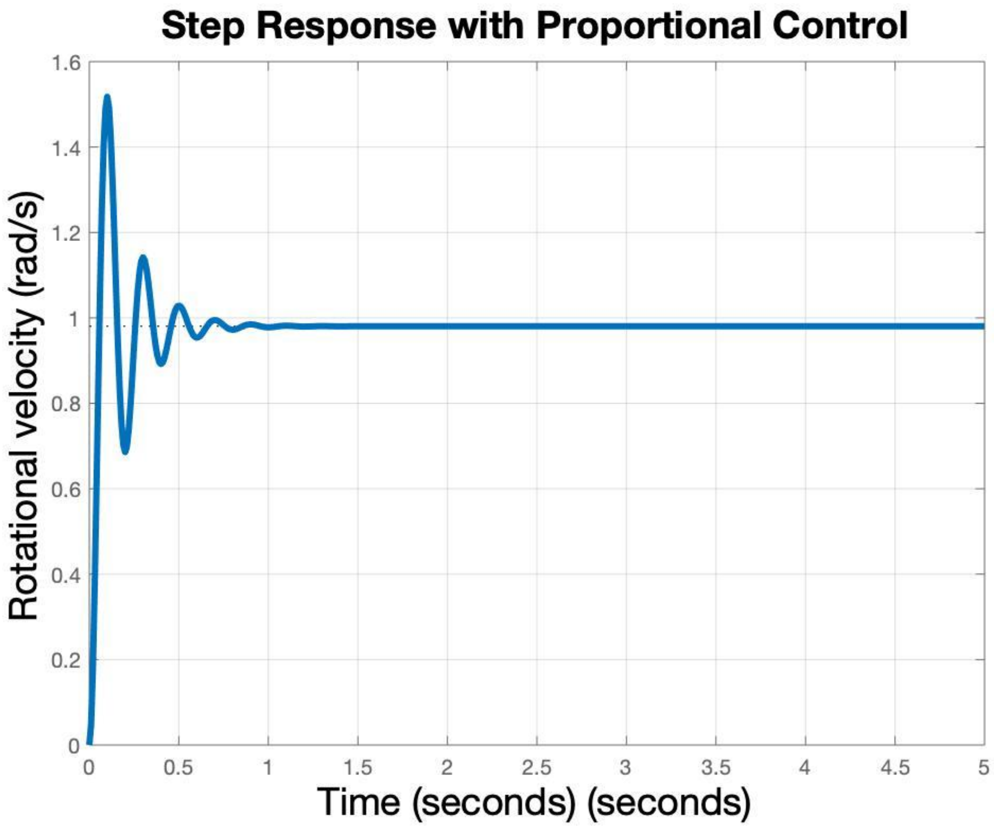


Settling time: The time difference between the initial time when the reference signal changes and the time at which the output signal reaches its steady-state value.

Measuring control performance



$\mathbf{K_P}$ = 50



$\mathbf{K_P}$ = 500

P-only controller

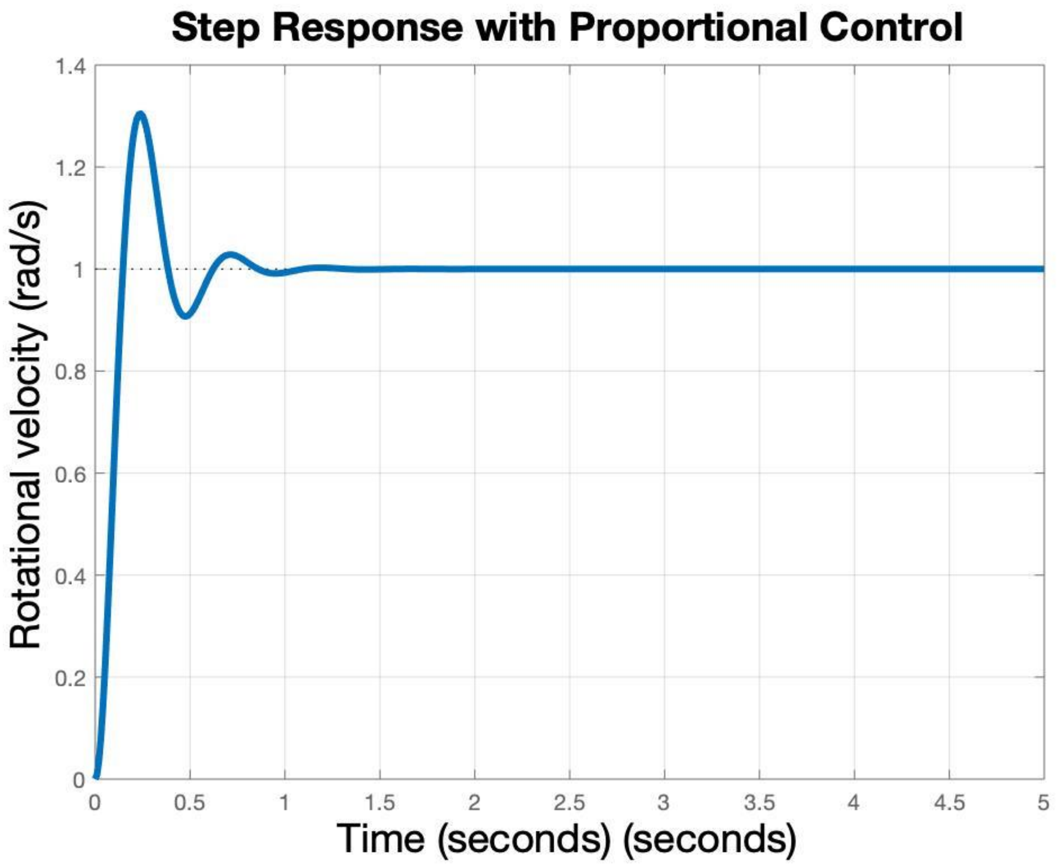
- Compute error signal $\mathbf{e} = \mathbf{r} - \boldsymbol{y}$
- Proportional term $K_p\mathbf{e}$:
  - K_p proportional gain;
  - Feedback correction proportional to error
- Cons:
  - If K_p is small, error can be large! [undercompensation]
  - If K_p is large,
    - system may oscillate (i.e. unstable) [overcompensation]
    - may not converge to set-point fast enough
  - P-controller always has steady state error or offset error

PI-controller

Compute error signal $\mathbf{e} = \mathbf{r} - \mathbf{y}$

Integral term: $K_I \int_0^t \mathbf{e}(\tau)d\tau$

- K_I integral gain;
- Feedback action proportional to cumulative error over time
- If a small error persists, it will add up over time and push the system towards eliminating this error): **eliminates offset/steady-state error**

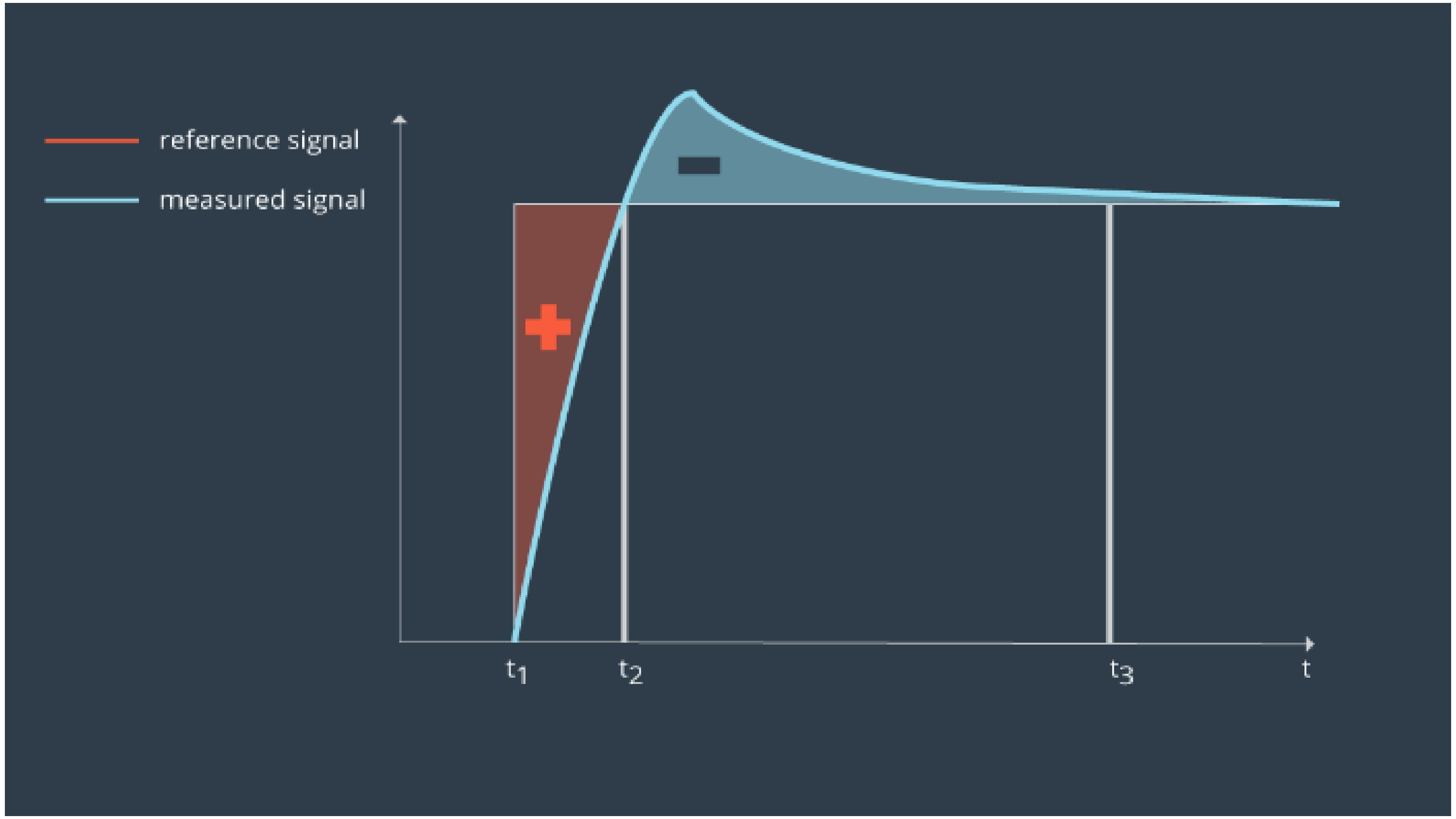


Disadvantages:

- Integral action by itself can increase instability
- Integrator term can accumulate error and suggest corrections that are not feasible for the actuators (integrator windup)
    - Real systems “saturate” the integrator beyond a certain value

PI-controller

Integrator windup



PD-controller

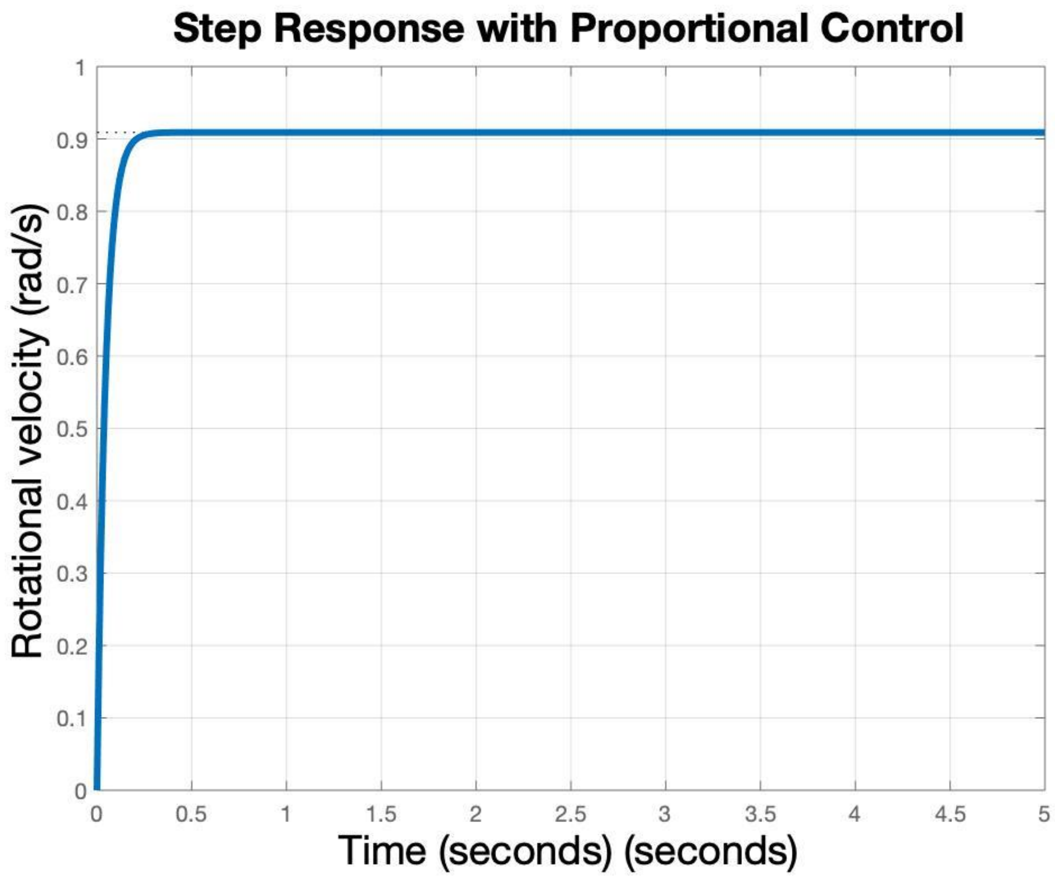
Compute error signal $\mathbf{e} = \mathbf{r} - \mathbf{y}$

Derivative term $K_d \dot{\mathbf{e}}$:

- K_d derivative gain;
- Feedback proportional to how fast the error is increasing/decreasing

Purpose:

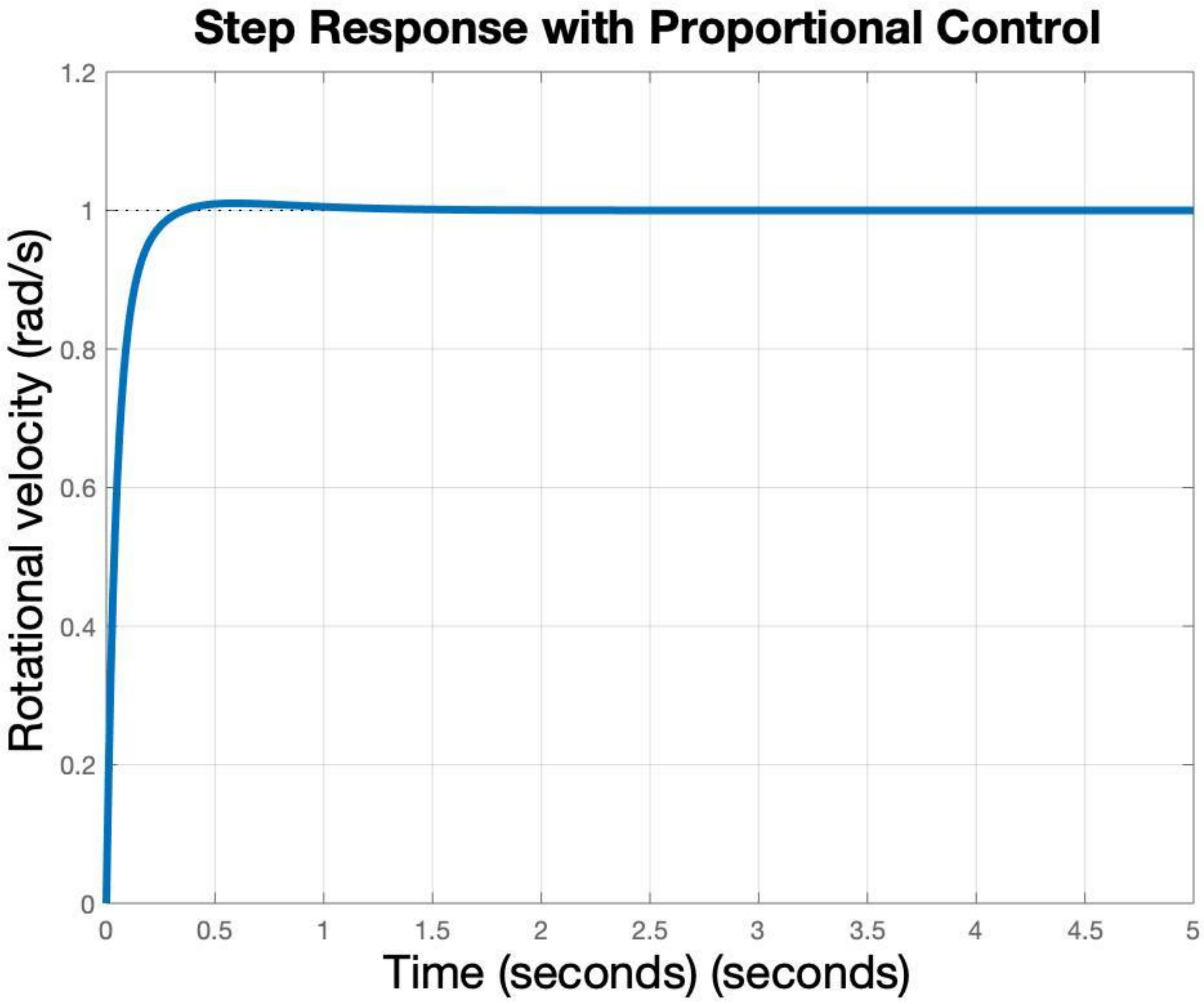
- “Predictive” term, can reduce overshoot: if error is decreasing slowly, feedback is slower
- Can improve tolerance to disturbances



Disadvantages:

- Still cannot eliminate steady-state error
- High frequency disturbances can get amplified

PID-controller

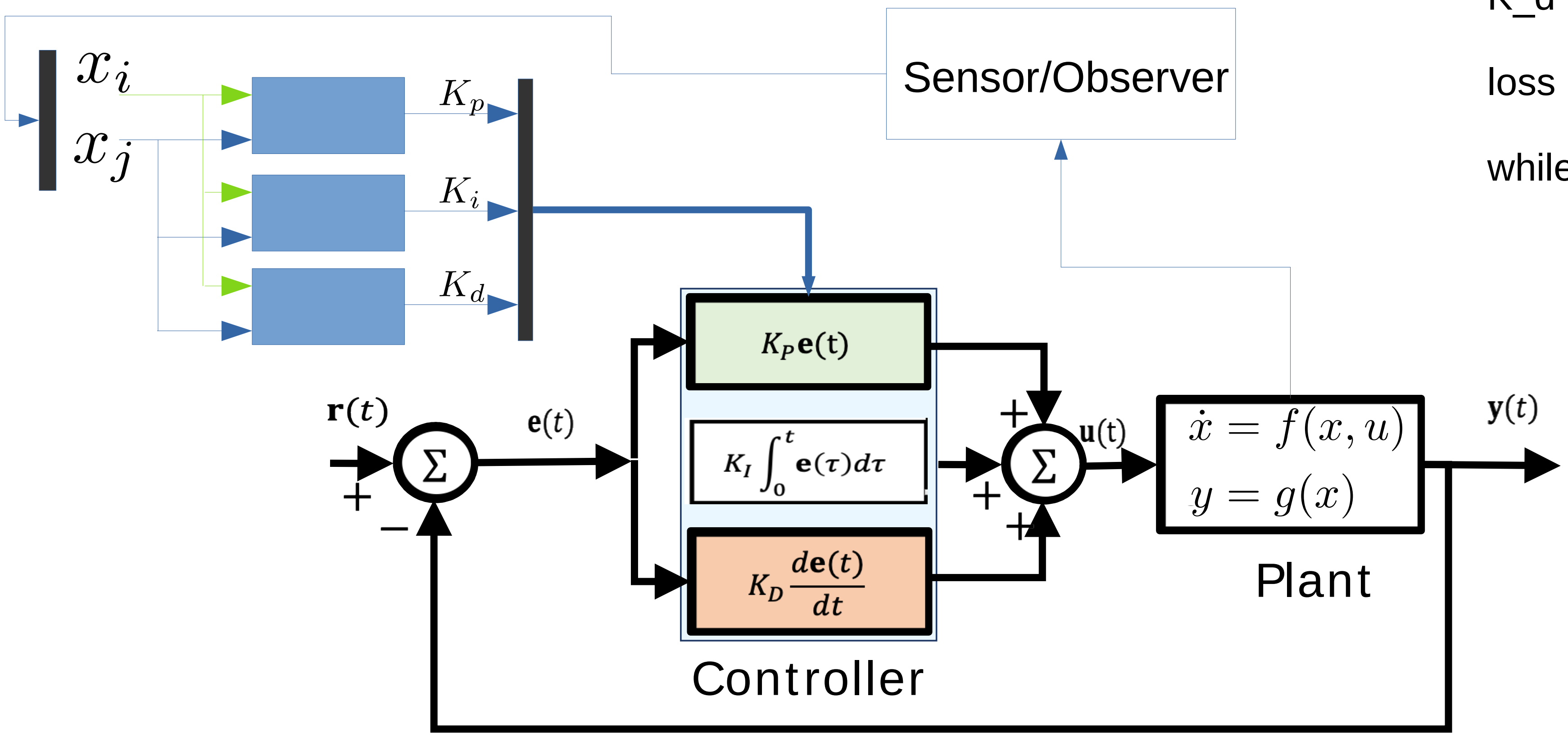


PID controller in practice

- May often use only PI or PD control
- Many heuristics to *tune* PID controllers, i.e., find values of K_P, K_I, K_D
- Several *recipes* to tune, usually rely on designer expertise
- E.g. *Ziegler-Nichols* method: increase K_P till system starts oscillating with period T (say till $K_P = K^*$), then set $K_P = 0.6K^*, K_I = \frac{1.2K^*}{T}, K_D = \frac{3}{40}K^*T$
- Matlab/Simulink has PID controller blocks + PID auto-tuning capabilities
- Work well with linear systems or for small perturbations,
- For non-linear systems use "gain-scheduling"
  - (i.e. using different K_P, K_I, K_D gains in different operating regimes)

Gain Scheduling Example

Used for NONLINEAR / unknown systems



<u>Calibration Routine Example</u>

K_p = f_p (state, param_set)
K_i = f_i (state, param_set)
K_d = f_d (state, param_set)

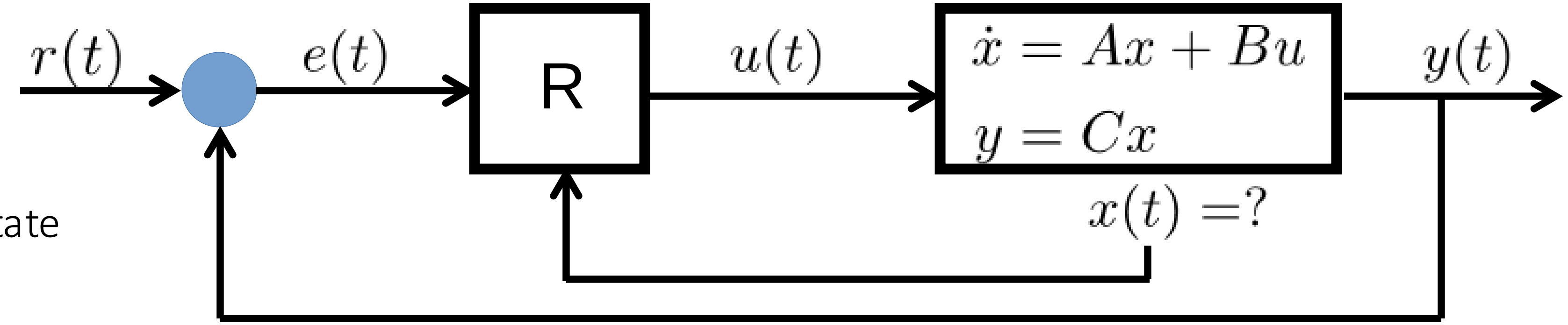
loss = g(stability, risetime, overshoot, etc.)

while not (end condition):

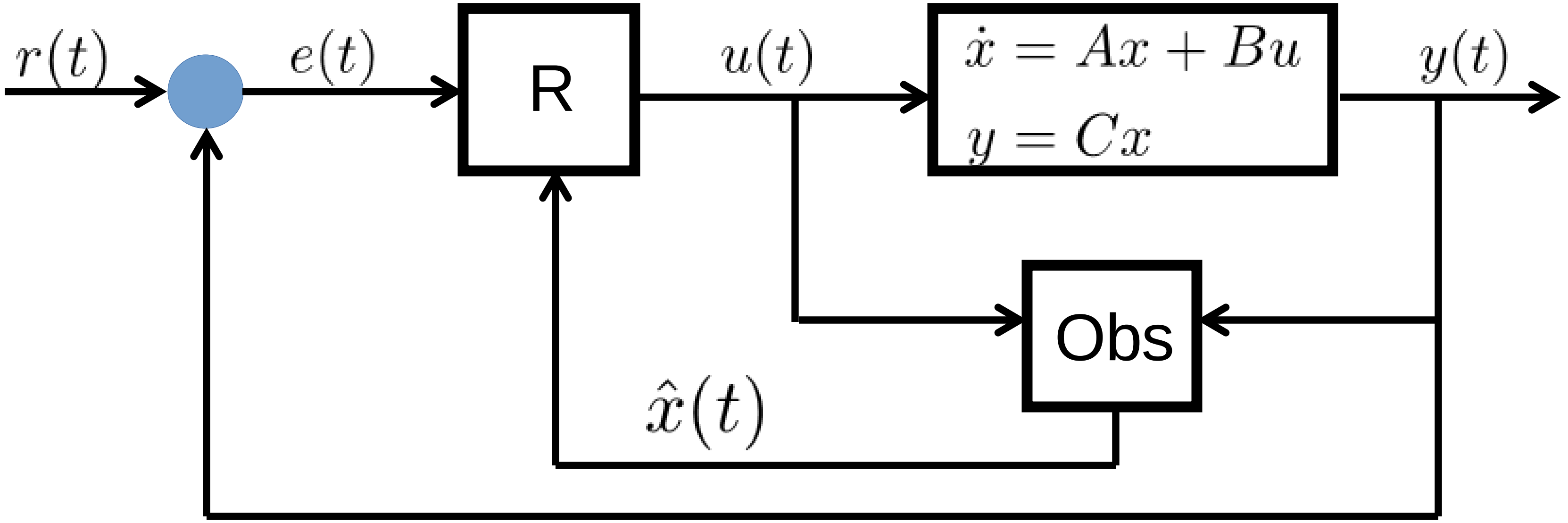
    loss = run_system (param_set)
    optimization_step(param_set)

Observation

- Problem:Control
    - design with (partially) unknown state



- Solution:
    - Luenberger Observer



Luenberger Observer

- State-space representation
- $$\dot{x} = Ax + Bu$$
  $$y = Cx$$

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y})$$
$$\hat{y} = C\hat{x}$$
$$u = K(x_{ref} - \hat{x})$$

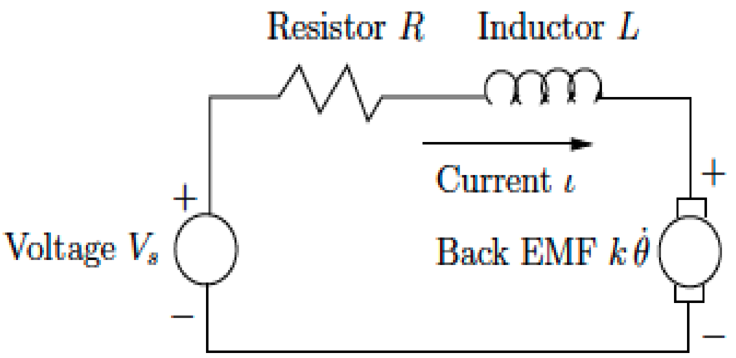
Control design parameters

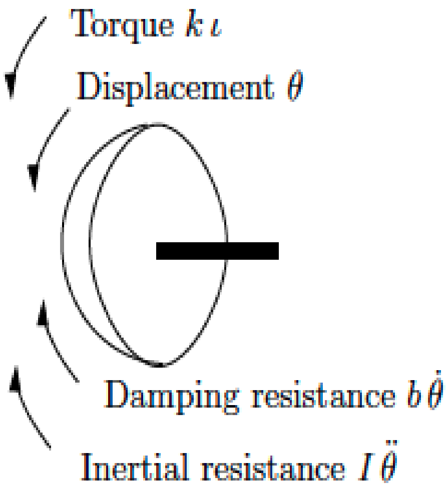
- Observer Error satisfies: $\dot{e} = (A - LC)e$
- Required: Observability, Controllability
- Pole Placement

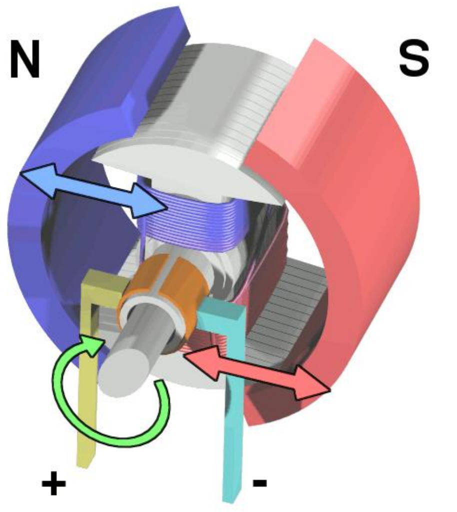
$$K : eig(A - BK) = \{\lambda_{c1}, \dots, \lambda_{cn}\}$$
$$L : eig(A^T - LC) = \{\lambda_{o1}, \dots, \lambda_{on}\}$$

Overall system is stable iff both observer and controller are stable

Example - DC Motor







b = 0.1 # friction coefficient (Nm/(rad/sec))
I = 0.01 # mechanical inertia (Kg*m^2)
k = 0.01 # motor torque constant (Nm/A)
R = 1   # armature resistance (Ohm)
L = 0.5 # armature inductance (H)

$$V_s = Ri + L\frac{di(t)}{dt} + k\theta_v$$
$$I\frac{d\theta_v}{dt} + b\theta_v = ki$$

State-space representation
$\dot{x} = Ax + Bu$

$$A = \begin{bmatrix} -b/I & k \\ -k/L & -R \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$x = \begin{bmatrix} \theta_v \\ i \end{bmatrix} \qquad u = V_s$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Modern Control Theory:

Optimal Control, MPC

- **Optimal Control / LQR**
- MPC

(Nonlinear) Optimal Control

$$\dot{x} = f(x, u, t)$$
$$x \in \mathbb{R}^n, u \in \mathbb{R}^m$$
$$x(t_0) = x_0$$

- Minimization of cost function $J[u(t)]$ over time interval $[t_0, t_1]$

$$J[u(t)] = \underbrace{S(x(t_1), t_1)}_{\text{Final State Rating}} + \underbrace{\int_{t_0}^{t_1} L(x, u, t)dt}_{\text{Integral Cost}}$$

- Find solution $\quad \underline{x} := \begin{bmatrix} x \\ u \end{bmatrix}$

LQR Control (finite time, discrete)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$J = x(T)'Qx(T) + \sum_{k=0}^{T-1} [x(k)'Qx(k) + u(k)'Ru(k)], \quad Q, R > 0$$

- Solution

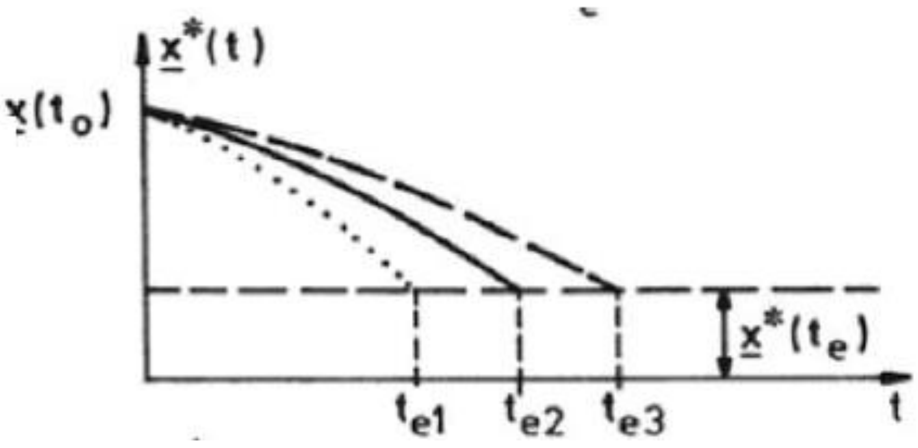
LQR Control (finite time, discrete)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$J = x(T)'Qx(T) + \sum_{k=0}^{T-1} [x(k)'Qx(k) + u(k)'Ru(k)], \quad Q, R > 0$$

- Solution
- $u(k) = -K(k)x(k)$

- Depends on final time T



LQR Control (finite time, discrete)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$J = x(T)'Qx(T) + \sum_{k=0}^{T-1} [x(k)'Qx(k) + u(k)'Ru(k)], \quad Q, R > 0$$

$$J = \phi(A, B, Q, R, x_0, u_0, \ldots, u_{T-1})$$

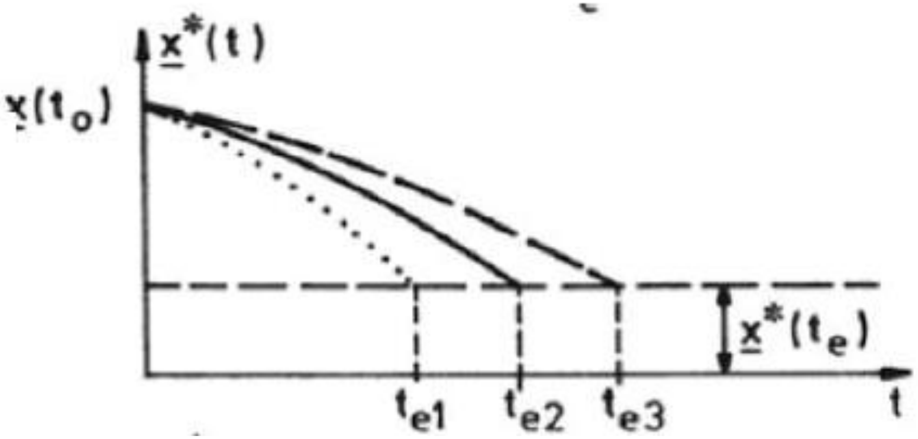
- Solution
- $u(k) = -K(k)x(k)$

- Depends on final time T

$$P(T) = Q$$

$$K(k) = (R + B'P(k+1)B)^{-1}(B'P(k+1)A)$$

$$P(k-1) = A'P(k)A - (A'P(k)B)(R + B'P(k)B)^{-1}(B'P(k)A) + Q$$



LQR Control (finite time, discrete)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$J = x(T)'Qx(T) + \sum_{k=0}^{T-1} [x(k)'Qx(k) + u(k)'Ru(k)], \quad Q, R > 0$$

$$J = \phi(A, B, Q, R, x_0, u_0, \dots, u_{T-1})$$

- Solution

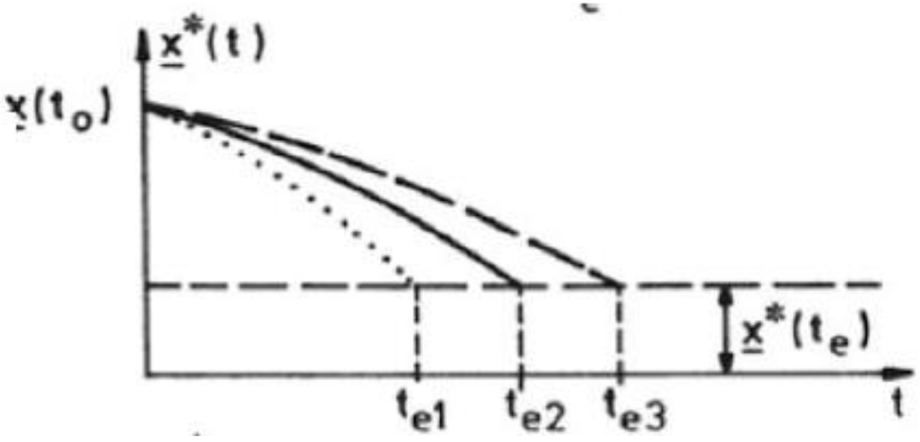
$$u(k) = -K(k)x(k)$$

  - Does not depend on initial condition!

$$P(T) = Q$$

$$K(k) = (R + B'P(k+1)B)^{-1}(B'P(k+1)A)$$

$$\boxed{P(k-1) = A'P(k)A - (A'P(k)B)(R + B'P(k)B)^{-1}(B'P(k)A) + Q}$$



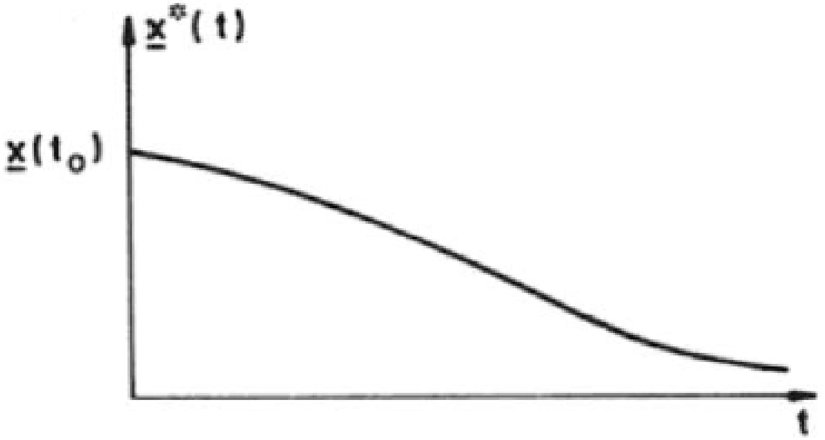
LQR Control (infinite time, discrete)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$J = \sum_{k=0}^{\infty} [x(k)'Qx(k) + u(k)'Ru(k)], \quad Q, R > 0$$

- Solution

$$u(k) = -Kx(k)$$



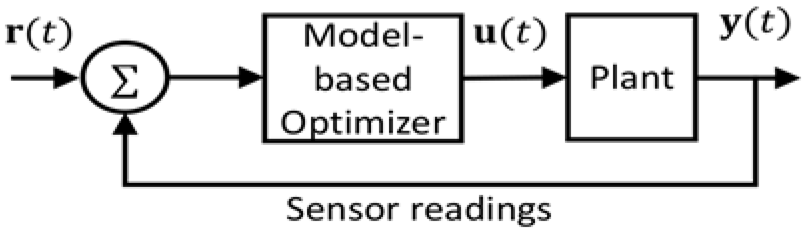
$$K = (R + B'PB)^{-1}(B'PA)$$

$$\boxed{P = A'PA - (A'PB)(R + B'PB)^{-1}(B'PA) + Q \qquad \text{ARE}}$$

- Optimal Control / LQR
- <u>**MPC**</u>

Model Predictive Control

Main idea: Use a dynamical model of the plant (inside the controller) to predict the plant's future evolution, and optimize the control signal over possible futures



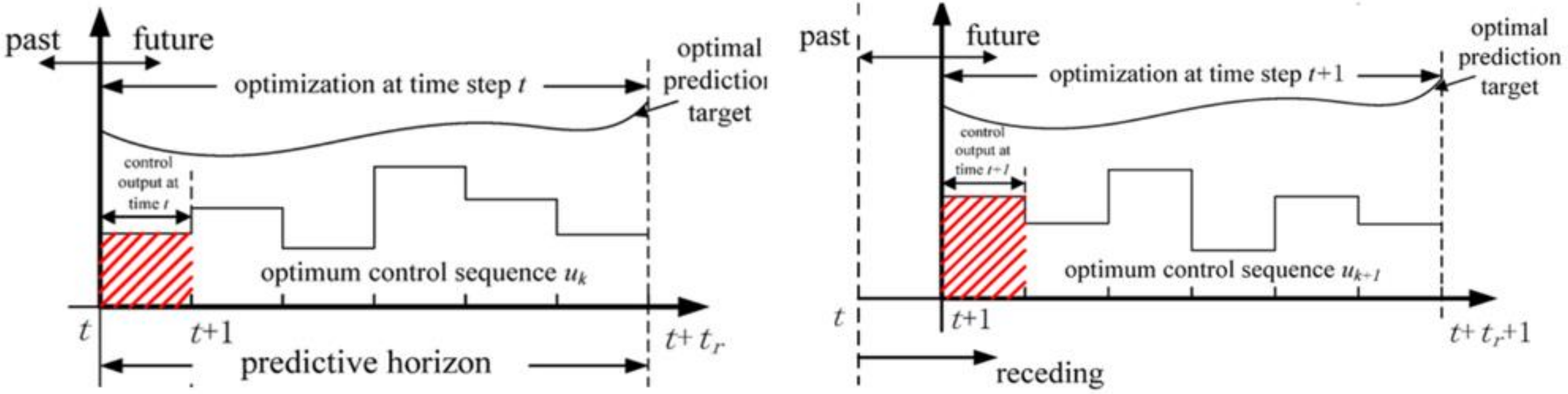


Image from: https://tinyurl.com/yaej43x5

Why MPC?

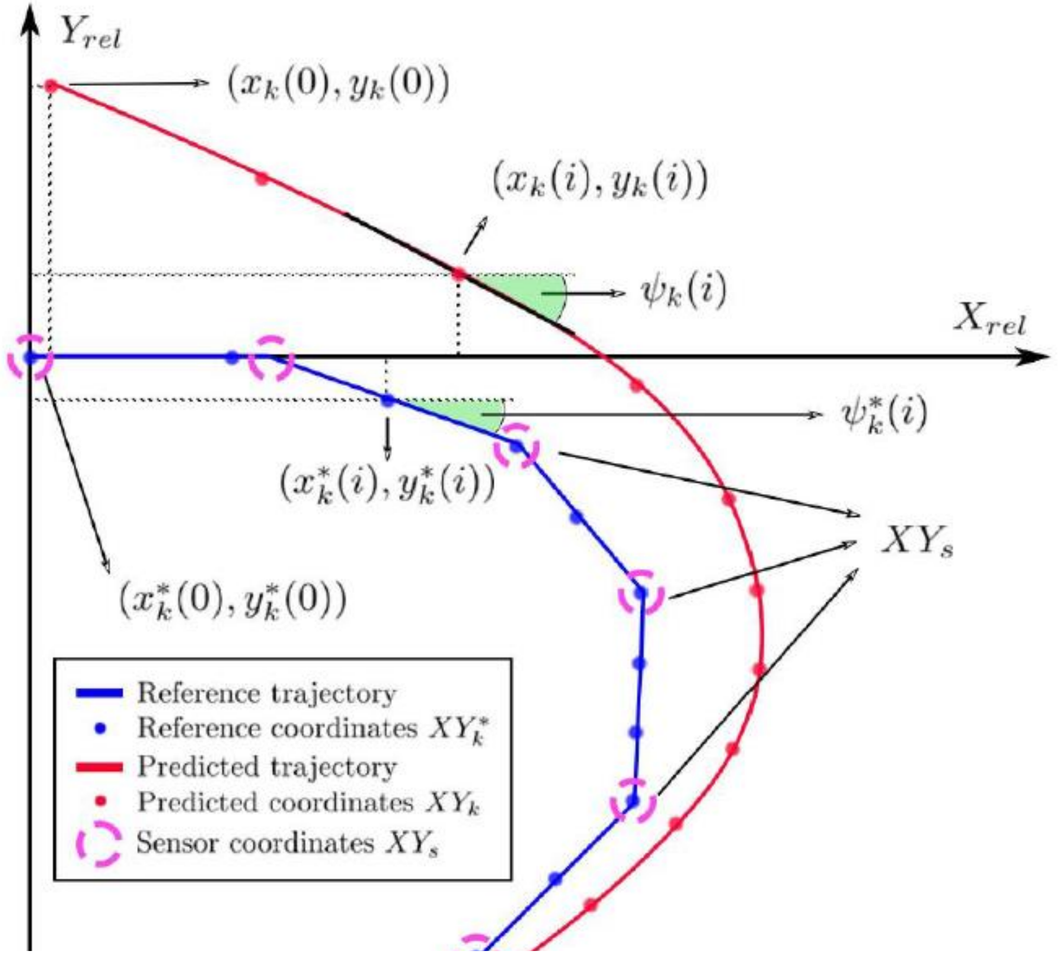
- Optimal control with constraints (input, output and states)
- ideal for MIMO (Multi Input Multi Output) systems
- linear and nonlinear models

- RECEDING HORIZON PRINCIPLE

"At any time instant k, based on the available process information, solve the optimization problem with respect to the future control sequence $[u(k), ..., u(k+N-1)]$ and apply only its first element $u^o(k)$. Then, at next time instant $k+1$, a new optimization problem is solved, based on the process information available at time $k+1$, along the prediction horizon $[k+1, k+N]$." (Camacho)

Receding Horizon Principle

- Closed Loop solution (no constraints, LQR)
- Open Loop solution (constraints)



$$J(x(k), u(\cdot), k) = \sum_{i=0}^{N-1} \left(||x(k+i)||_Q^2 + ||u(k+i)||_R^2 \right) + ||x(k+N)||_S^2$$

Linear MPC (1)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$x(k+i) = A^i x(k) + \sum_{j=0}^{i-1} A^{i-j-1} Bu(k+j), \quad i > 0$$

$$X(k) = \mathcal{A}x(k) + \mathcal{B}U(k) \quad \Rightarrow \quad A\underline{x} = b$$

$$X(k) = \begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N-1) \\ x(k+N) \end{bmatrix}, \quad U(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-2) \\ u(k+N-1) \end{bmatrix}, \quad \mathcal{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N-1} \\ A^N \end{bmatrix},$$

Linear MPC (2)

$$x(k+1) = Ax(k) + Bu(k), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$x(k+i) = A^i x(k) + \sum_{j=0}^{i-1} A^{i-j-1} Bu(k+j), \quad i > 0$$

$$X(k) = \mathcal{A}x(k) + \mathcal{B}U(k) \qquad \Rightarrow \qquad A\underline{x} = b$$

$$\mathcal{B} = \begin{bmatrix} B & 0 & 0 & \cdots & 0 & 0 \\ AB & B & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A^{N-2}B & A^{N-3}B & A^{N-4}B & \cdots & B & 0 \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \cdots & AB & B \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} I^{(nN)} \ , \ -\mathcal{B} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} X(k) \\ U(k) \end{bmatrix}}_{\underline{x}} = \underbrace{\mathcal{A}x(k)}_{b}$$

(Non-)Linear MPC

$$s = [x, u, \Delta u]^T$$

$$J_{MPC} = \sum_{i=1}^{N} (||x(i) - x^*(i)||_Q^2 + ||u(i) - u^*(i)||_R^2 + ||\Delta u(i) - \Delta u^*(i)||_{\Delta R}^2)$$

- Linear formulation:

$$\begin{aligned} \underset{s}{\text{minimize}} \quad & J_{MPC}(s) \\ \text{subject to} \quad & A_{eq} s = b_{eq}, \\ & A_{ineq} s \leq b_{ineq} \end{aligned}$$

- Nonlinear formulation:

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & J_{MPC}(x, u) \\ \text{subject to} \quad & \\ & x(k+1) = f(x(k), u(k)), \\ & h(x(k), u(k)) \leq 0 \end{aligned}$$

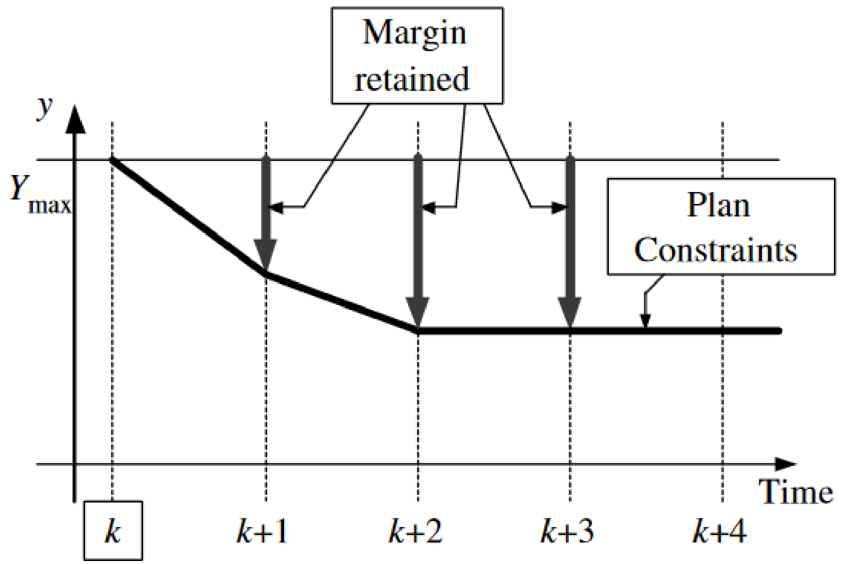
Issues with MPC

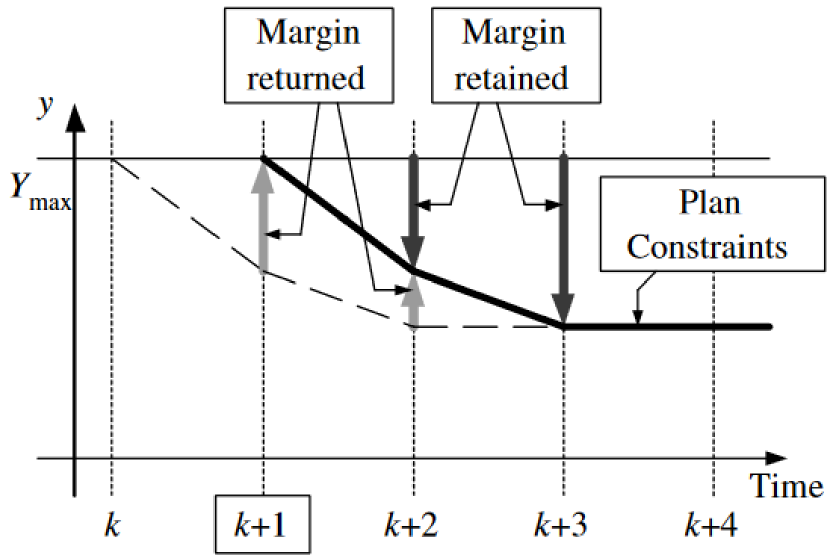
- Feasibility
- Stability
- Computation

Conflicting Requirements
(several solutions depending on needs)

- Robustness formulation: system affected by process and measurement noise

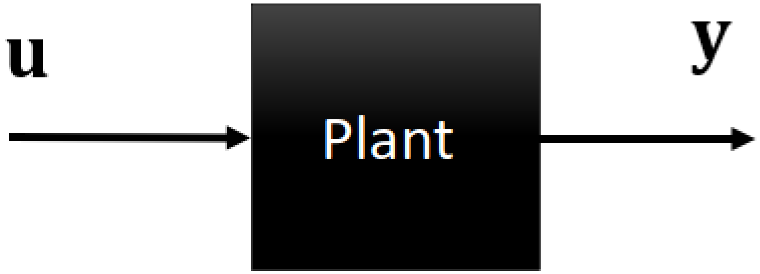
Constraints tightening





Kalman Filtering

What is state estimation?



- Given a “black box” component, we can try to use a linear or nonlinear system to model it (maybe based on physics, or data-driven)
- Model may posit that the plant has internal states, but we typically have access only to the outputs of the model (whatever we can measure using a sensor)
- May need internal states to implement controller: how do we estimate them?
- State estimation: Problem of determining internal states of the plant

Deterministic vs. Noisy case

Typically sensor measurements are noisy (manufacturing imperfections, environment uncertainty, errors introduced in signal processing, etc.)

In the absence of noise, the model is deterministic: for the same input you always get the same output

Can use a simpler form of state estimator called an observer (e.g. a Luenberger observer)

- $\frac{d\hat{\mathbf{x}}}{dt} = A\hat{\mathbf{x}} + B\mathbf{u} + L(\mathbf{y} - \hat{\mathbf{y}})$
- $\hat{\mathbf{y}} = C\hat{\mathbf{x}} + D\mathbf{u}$
- $\mathbf{u}(t) = -K_{\text{lqr}}\hat{\mathbf{x}}(t),$

$\Longleftrightarrow$ $\dot{e} = (A - LC)e$

In the presence of noise, we use a state estimator, such as a Kalman Filter

Kalman Filter is one of the most fundamental algorithm that you will see in autonomous systems, robotics, computer graphics, …

Random variables and statistics refresher

- For random variable w, $\mathbb{E}[w]$: expected value of w, also known as mean
- Suppose $\mathbb{E}[x] = \mu$: then $\mathrm{var(w)}$: variance of w, is $\mathbb{E}[(w-\mu)^2]$
- For random variables x and y, $\mathrm{cov}(x, y)$: covariance of x and y
  - $\mathrm{cov}(x, y) = \mathbb{E}[(x - \mathbb{E}(x)(y - \mathbb{E}(y)]$
- For random ***vector*** $\mathbf{x}$, $\mathbb{E}[\mathbf{x}]$is a vector
- For random vectors, $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$, cross-covariance matrix is $m \times n$ matrix: $\mathrm{cov}(\mathbf{x}, \mathbf{y}) = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{y} - \mathbb{E}[\mathbf{y}])^{\mathrm{T}}]$
- $w \sim N(\mu, \sigma^2)$: w is a normally distributed variable with mean μ and variance σ

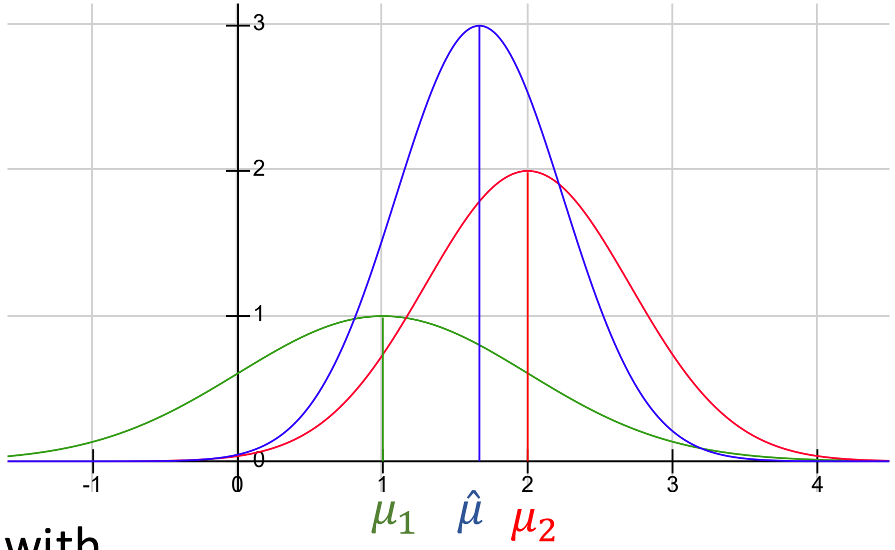
Data fusion example

- Using radar and a camera to estimate the distance to the lead car:
  - Measurement is never free of noise
  - Actual distance: x
  - Measurement with radar: $z_1 = x + v_1$ ($v_1 \sim N(\mu_1, \sigma_1^2)$ is radar noise)
  - With camera: $z_2 = x + v_2$ ($v_2 \sim N(\mu_2, \sigma_2^2)$ is camera noise)
  - How do you combine the two estimates?
- Use a weighted average of the two estimates, prioritize more likely measurement
  - $\hat{\mu} = \frac{(z_1/\sigma_1^2) + (z_2/\sigma_2^2)}{(1/\sigma_1^2) + (1/\sigma_2^2)} = kz_1 + (1-k)z_2$, where $k = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$
  - $\hat{\sigma}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$
- Observe: uncertainty reduced, and mean is closer to measurement with lower uncertainty

$\mu_1 = 1, \sigma_1^2 = 1$

$\mu_2 = 2, \sigma_2^2 = 0.5$

$\hat{\mu} = 1.67, \sigma_2^2 = 0.33$



Multi-variate sensor fusion

- Instead of estimating one quantity, we want to estimate n quantities, then:
- Actual value is some vector $\mathbf{x}$
- Measurement noise for i^{th} sensor is $v_i \sim N(\boldsymbol{\mu}_i, \Sigma_i)$, where $\boldsymbol{\mu}_i$ is the mean vector, and Σ_i is the covariance matrix
- $\Lambda = \Sigma^{-1}$ is the **information matrix**
- For the two-sensor case:
  - $\hat{\mathbf{x}} = (\Lambda_1 + \Lambda_2)^{-1}(\Lambda_1 \mathbf{z}_1 + \Lambda_2 \mathbf{z}_2)$, and $\hat{\Sigma} = (\Lambda_1 + \Lambda_2)^{-1}$

Motion makes things interesting

- What if we have one sensor and making repeated measurements of a moving object?
- Measurement differences are not all because of sensor noise, some of it is because of object motion
- Kalman filter is a tool that can include a motion model (or in general a dynamical model) to account for changes in internal state of the system
- Combines idea of ***prediction*** using the system dynamics with ***correction*** using weighted average (Bayesian inference)

Stochastic Difference Equation Models

- We assume that the plant (whose state we are trying to estimate) is a stochastic discrete dynamical process with the following dynamics:

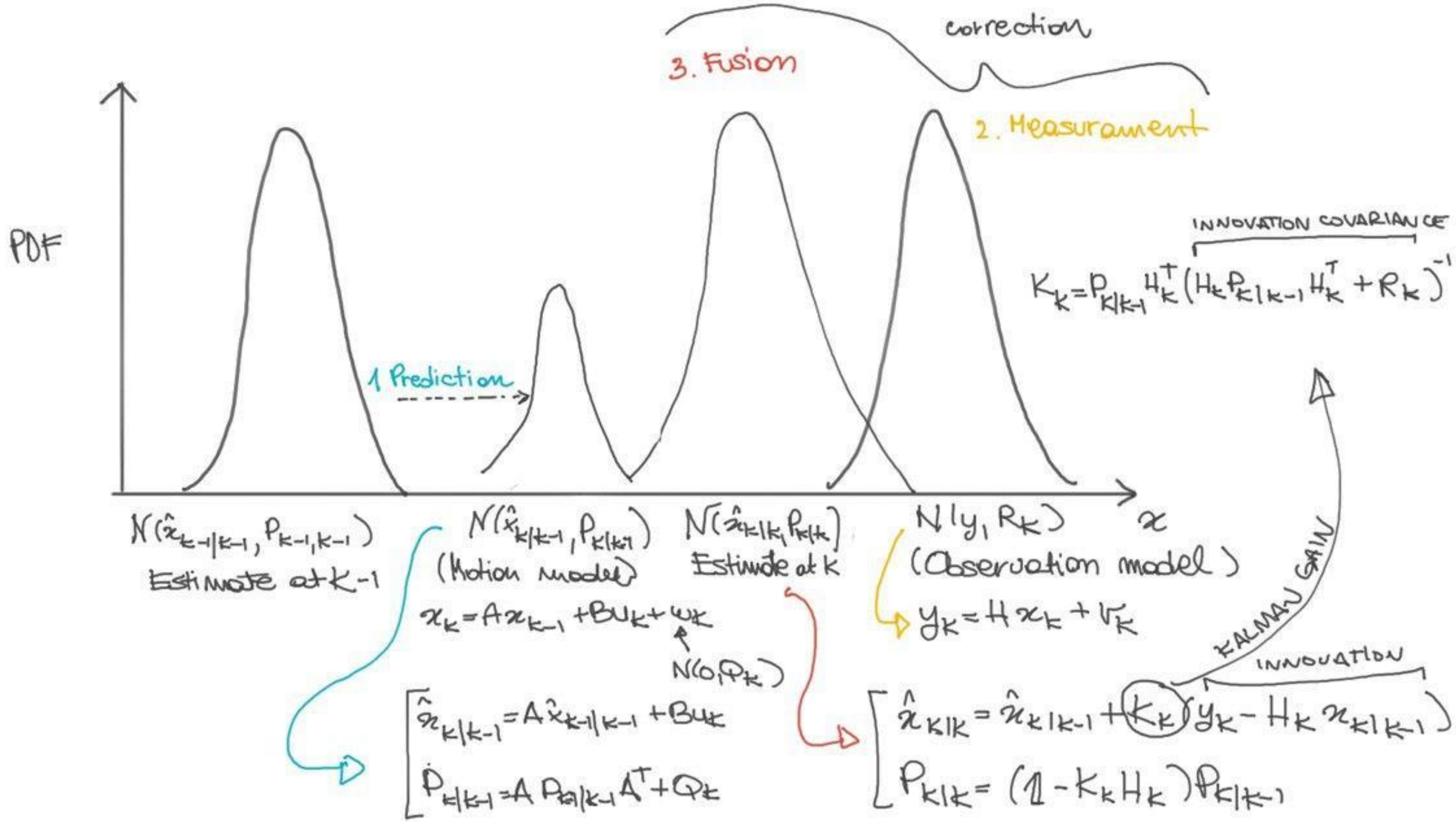
$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k \text{ (Process Model)}$$

$$\boldsymbol{y}_k = H\mathbf{x}_k + \mathbf{v}_k \text{ (Measurement Model)}$$

$\mathbf{x}_k, \mathbf{x}_{k-1}$	State at time k,$k-1$
$\mathbf{u}_k$	Input at time k
$\mathbf{w}_k$	Random vector representing noise in the plant, $\mathbf{w} \sim N(\mathbf{0}, Q_k)$
$\mathbf{v}_k$	Random vector representing sensor noise, $\mathbf{v} \sim N(\mathbf{0}, R_k)$
$\mathbf{z}_k$	Output at time k

n	Number of states
m	Number of inputs
p	Number of outputs
A	$n \times n$ matrix
B	$n \times m$ matrix
H	$p \times n$ matrix

Kalman Filter



Step I: Prediction

- We assume an estimate of $\mathbf{x}$ at time $k-1$, fusing information obtained by measurements till time $k-1$: this is denoted $\hat{\mathbf{x}}_{k-1|k-1}$
- We also assume that the error between the estimate $\hat{\mathbf{x}}_{k-1|k-1}$ and the actual $\mathbf{x}_{k-1}$ has 0 mean, and covariance $P_{k-1|k-1}$
- Now, we use these values and the state dynamics to predict the value of $\mathbf{x}_k$
- Because we are using measurements only up to time $k-1$, we can denote this predicted value as $\hat{\mathbf{x}}_{k|k-1}$, and compute it as follows:

$$\hat{\mathbf{x}}_{k|k-1} \coloneqq A\hat{\mathbf{x}}_{k-1|k-1} + B\mathbf{u}_k$$

Step I: Prediction

$$P_{k|k-1} = \mathrm{cov}\left(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}\right) = \mathrm{cov}\left(A\mathbf{x}_{k-1} + B\mathbf{u}_k + w_k - A\hat{\mathbf{x}}_{k-1|k-1} - B\mathbf{u}_k\right)$$

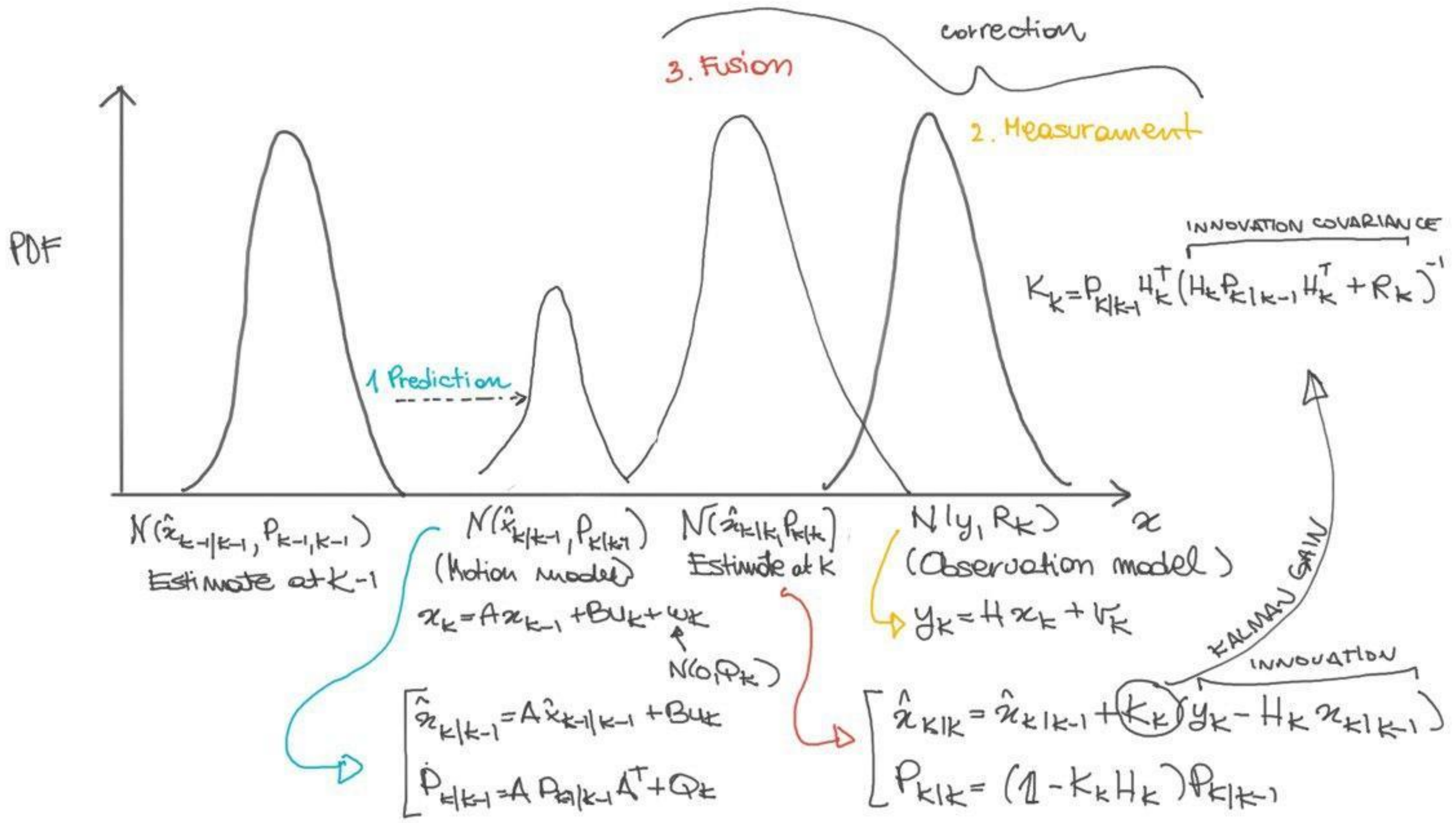
$$= A\mathrm{cov}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}\right)A^T + cov(w_k)$$

$$= AP_{k-1|k-1}A^T + Q_k$$

- Thus, the state and error covariance prediction are:

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &\coloneqq A\hat{\mathbf{x}}_{k-1|k-1} + B\mathbf{u}_k \\ P_{k|k-1} &\coloneqq AP_{k-1|k-1}A^T + Q_k \end{aligned}}$$

Kalman Filter



Step II: Correction

- This is where we basically do data fusion between new measurement and old prediction to obtain new estimate
- Note that data fusion is not straightforward like before because we don't really observe/measure $\mathbf{x}_k$ directly, but we get measurement $\boldsymbol{y}_k$, for an observable output!
- Idea remains similar: Do a weighted average of the prediction $\hat{\mathbf{x}}_{k|k-1}$ and new information
- We integrate new information by using the difference between the predicted output and the observation

Step II: Correction

- Predicted output: $\hat{\boldsymbol{y}}_k = H_k \hat{\mathbf{x}}_{k|k-1}$
- We denote the error in predicted output as the *innovation*

$$\mathbf{z}_k := \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1}$$

- Covariance of innovation

$$S_k = \mathrm{cov}\,(\mathbf{z}_k) = cov(H_k \mathbf{x}_k + \mathbf{v}_k - H_k \hat{\mathbf{x}}_{k|k-1}) = R_k + H_k P_{k|k-1} H_k^T$$

- Then to do data fusion is given by:

$$\hat{\boldsymbol{x}}_{k|k} := \hat{\boldsymbol{x}}_{k|k-1} + K_k z_k$$

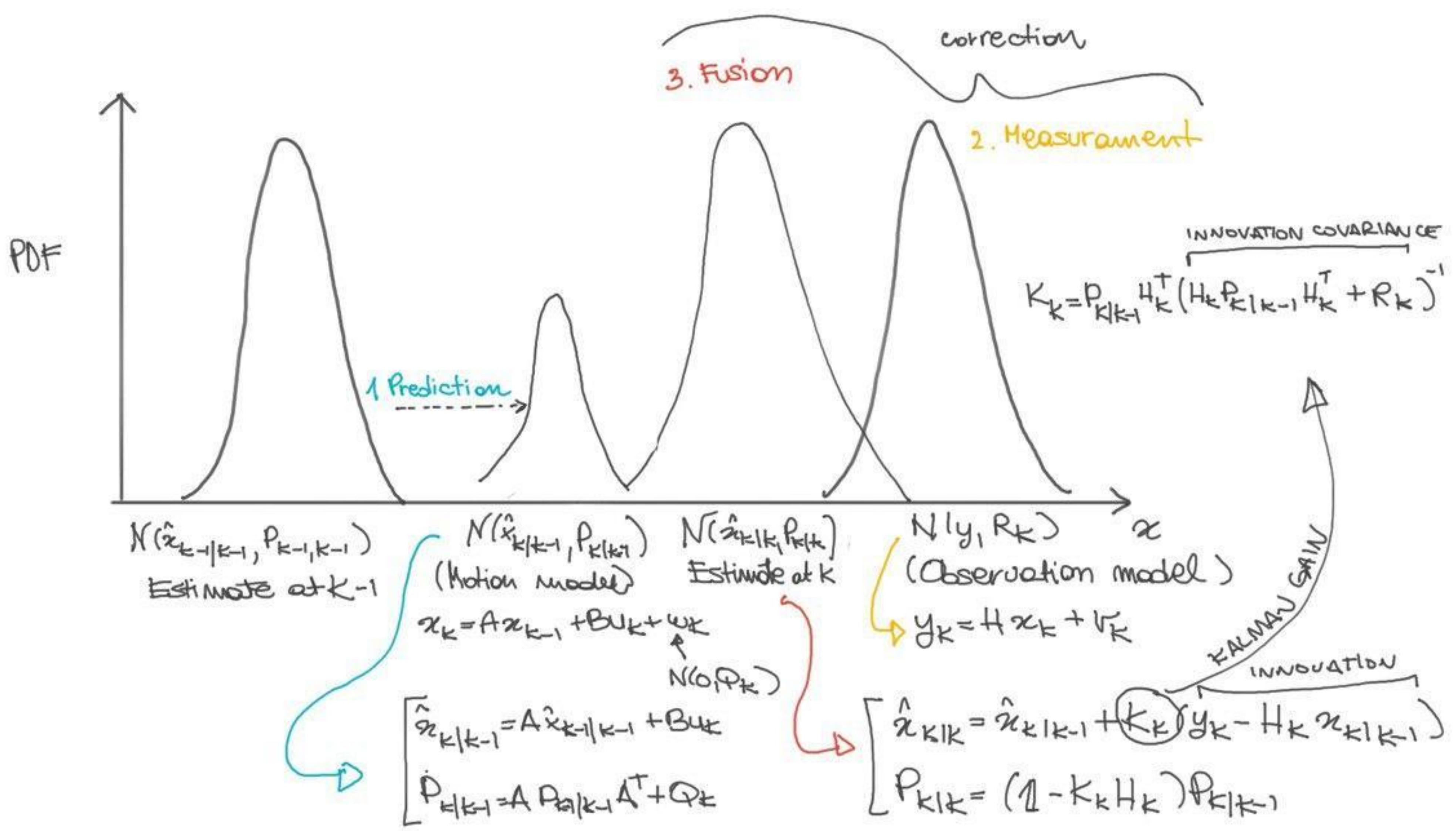
- Where, $K_k = P_{k|k-1} H_k^T S_k^{-1}$ is the (optimal) Kalman gain. It minimizes the least square error
- Finally, the updated error covariance estimate is given by:

$$P_{k|k} := (I - K_k H_k)\, P_{k|k-1}$$

Step II: Correction

Innovation	$\mathbf{z}_k := \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k\|k-1}$
Innovation Covariance	$S_k := R_k + H_k P_{k\|k-1} H_k^T$
Optimal Kalman Gain	$K_k := P_{k\|k-1} H_k^T S_k^{-1}$
State estimate at time k	$\hat{\boldsymbol{x}}_{k\|k} := \hat{\boldsymbol{x}}_{k\|k-1} + K_k \mathbf{z}_k$
Covariance estimate at time k	$P_{k\|k} := (I - K_k H_k) P_{k\|k-1}$

Kalman Filter



one-dimensional example

- Let's take a simple one-dimensional example
- Kalman filter prediction equations become:
  - $\hat{x}_{k|k-1} \coloneqq a\hat{x}_{k-1|k-1} + bu$; $\sigma^2_{k|k-1} \coloneqq \underbrace{a^2\sigma^2_{k-1|k-1}}_{\text{prior uncertainty in estimate}} + \underbrace{\sigma^2_q}_{\text{uncertainty in process}}$
- Also, the correction equations become:
  - Innovation: $z_k \coloneqq y_k - \hat{x}_{k|k-1}$, $\mathrm{S_k} = \sigma^2_r + \sigma^2_{k|k-1}$
  - Optimal gain: $k = \sigma^2_{k|k-1}/(\sigma^2_r + \sigma^2_{k|k-1})$,
  - Updated state estimate: $\hat{x}_{k|k} \coloneqq \hat{x}_{k|k-1} + k(y_k - \hat{x}_{k|k-1})$
  - I.e. updated state estimate: $\hat{x}_{k|k} \coloneqq (1-k)\ \hat{x}_{k|k-1} + ky_k$ (Weighted average!)

Extended Kalman Filter

- We skipped derivations of equations of the Kalman filter, but a fundamental property assumed is that the process model and measurement model are both linear.
- Under linear models and Gaussian process/measurement noise, a Kalman filter is an *optimal* state estimator (minimizes mean square error between estimate and actual state)
- In an EKF, state transitions and observations need not be linear functions of the state, but can be any differentiable functions
- I.e., the process and measurement models are as follows:

$$\mathbf{x}_k = f(x_{k-1}, u_k) + w_k$$
$$y_k = h(x_k) + v_k$$

EKF updates

- Functions f and h can be used directly to compute state-prediction, and predicted measurement, but cannot be directly used to update covariances
- So, we instead use the Jacobian of the dynamics at the predicted state
- This linearizes the non-linear dynamics around the current estimate
- Prediction updates:

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &:= f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \\ P_{k|k-1} &:= F_k P_{k-1|k-1} F_k^T + Q_k \end{aligned}}$$

$$\boxed{F_k := \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}=\mathbf{u}_k}}$$

EKF updates

- Correction updates:

$$\boxed{H_k \; := \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}}}$$

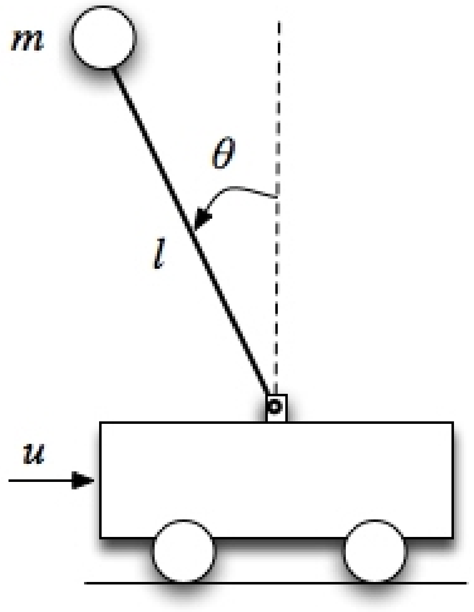
Innovation	$\mathbf{z}_k \; := \mathbf{y}_k - h(\hat{\mathbf{x}}_{k\|k-1})$
Innovation Covariance	$S_k := R_k + H_k P_{k\|k-1} H_k^T$
Near-Optimal Kalman Gain	$K_k := P_{k\|k-1} H_k^T S_k^{-1}$
A posteriori state estimate	$\hat{\boldsymbol{x}}_{k\|k} \; := \hat{\boldsymbol{x}}_{k\|k-1} + K_k \mathbf{y}_k$
A posteriori error covariance estimate	$P_{k\|k} \; := P_{k\|k-1}(I \; - K_k H_k)$

Simulink Example - Cartpole

$$\begin{cases} \ddot{p} & = \dfrac{u + m\, l\, \dot{\theta}^2\ \sin\theta - m\, g\ \cos\theta \sin\theta}{M + m \sin\theta^2} \\ \ddot{\theta} & = \dfrac{g\ \sin\theta - \cos\theta \ddot{p}}{l} \end{cases}$$

$$x = \left[p,\ \dot{p},\ \theta,\ \dot{\theta}\right]^T$$

$$y = [p, \theta]$$



- Full-state estimation (Luenberger, Kalman)
- Optimal Control