Cyber-Physical Systems

Laura Nenzi

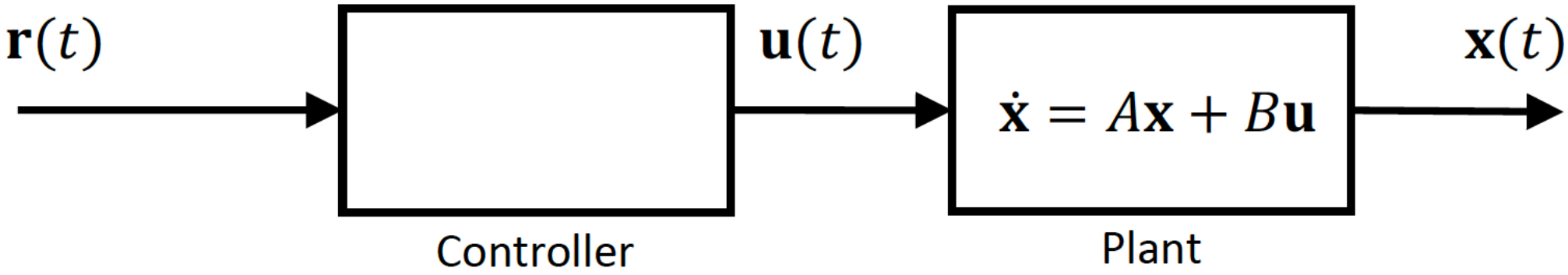
Università degli Studi di Trieste
II Semestre 2019

Lecture 7: Control

[Many Slides due to J. Deshmukh, USC, LA,USA]

Linear Control

Reference Tracking



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

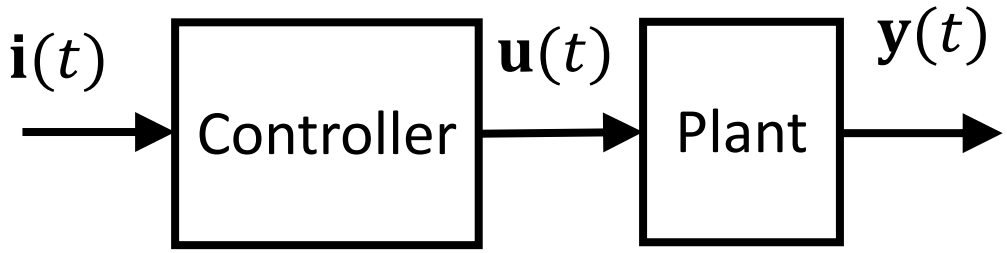
Control objectives:

- 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

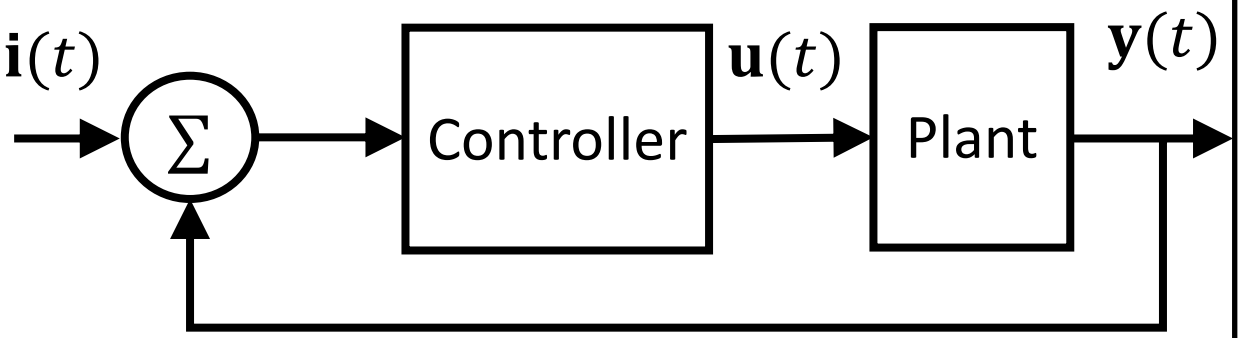
Open-loop or feed-forward control

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

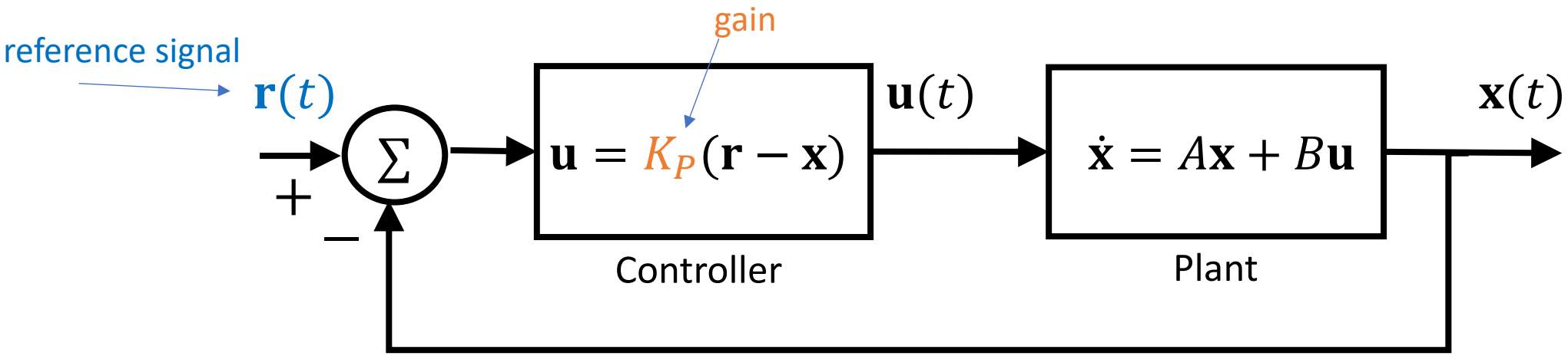


Feed-back control

- 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

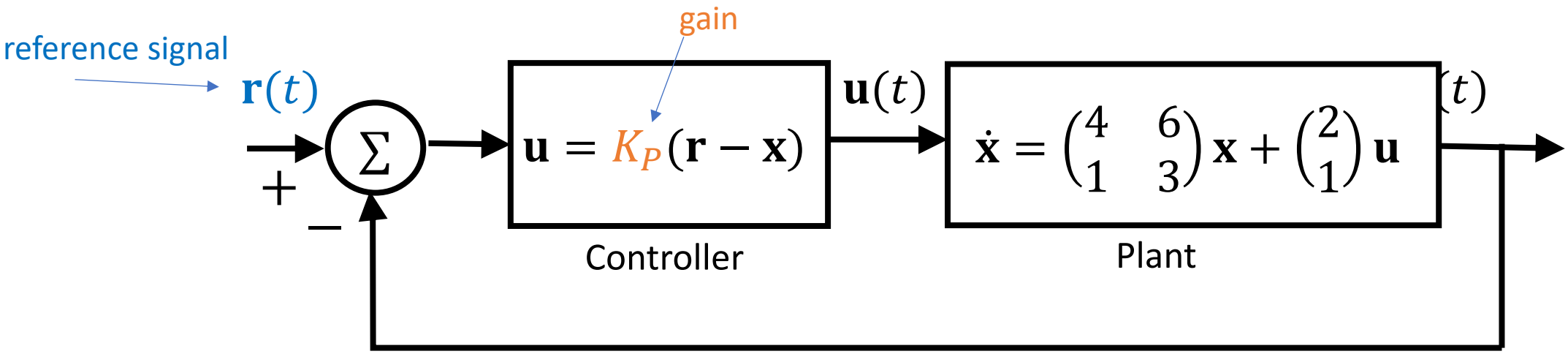


Proportional Controller



- 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

Proportional Controller



- Note $\text{eigs}(A) = 6, 1 \Rightarrow$ unstable plant!
- Let $K = (k_1 \quad k_2)$. Then, $A - BK = \begin{pmatrix} 4 - 2k_1 & 6 - 2k_2 \\ 1 - k_1 & 3 - k_2 \end{pmatrix}$
- Solve the equation: $\det(A - BK - \lambda \boldsymbol{I})$=0, i.e. $\lambda^2 + (2k_1 + k_2 - 7)\lambda + (6 - 2k_2) = 0$
- 2 distinct solution if polynomial of the form $(\lambda - \lambda_1)(\lambda - \lambda_2) = \lambda^2 + (-\lambda_1 - \lambda_2)\lambda + \lambda_1 \lambda_2$
- That means: $2k_1 + k_2 - 7 = (-\lambda_1 - \lambda_2)$ and $6 - 2k_2 = \lambda_1\lambda_2$
- $\lambda_1 = -1, \lambda_2 = -2$ gives $k_1 = 4$, $k_2 = 2$

Controllability Matrix

Can we always choose eigenvalues to find a stabilizing controller?

How do we determine for a given A, B whether there is a controller?

Controllabitlity Matrix: $C(A,B) = (B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B)$, where n is the state-dim

The pair (A,B) is controllable if the rank of $C(A,B)$ is full (i.e. rows are linearly independent)

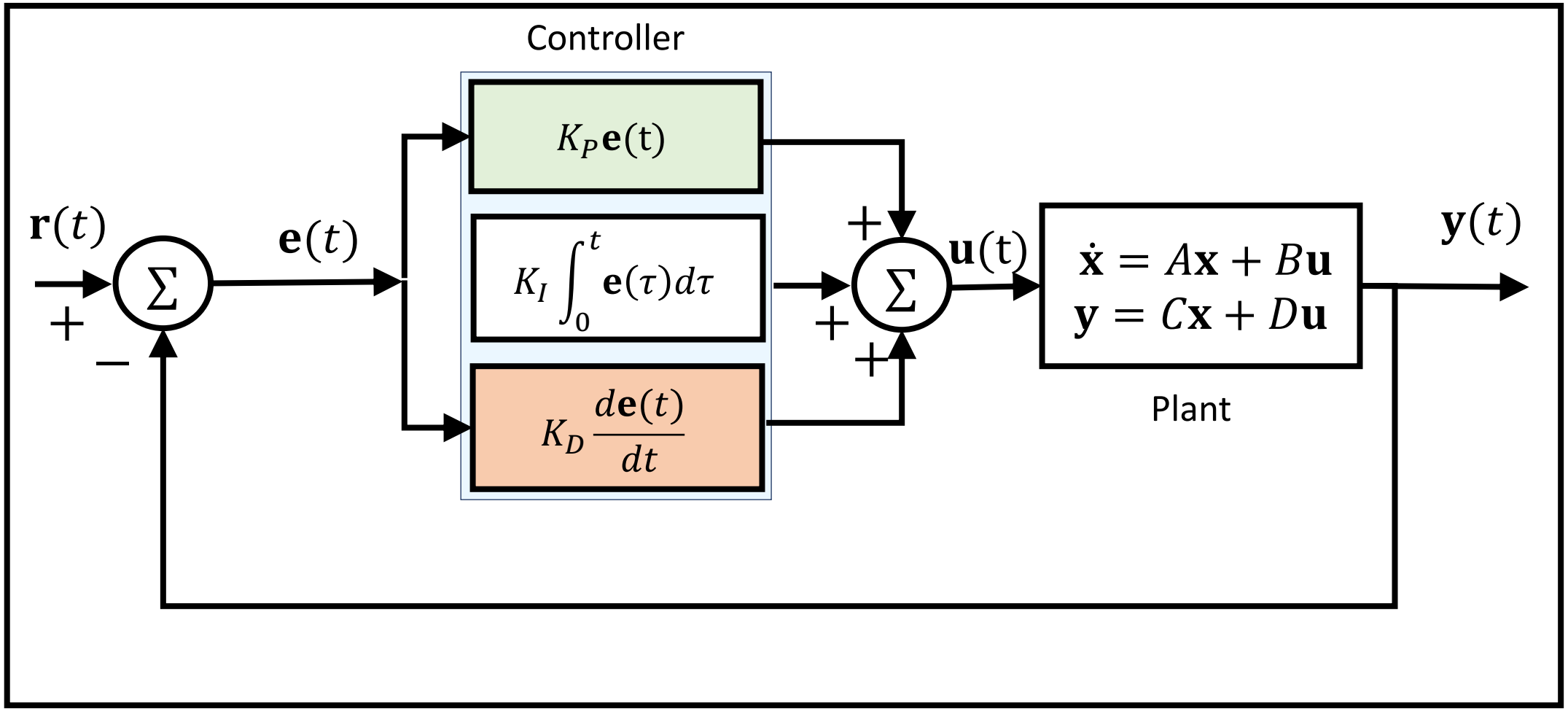
Optimality

- **Optimality**: chose best controller, i.e. driving faster to the equilibrium. E.g. using a Linear Quadratic Regulator (LQR) controller, optimizing the cost function:

$$J_{LQR} = \int_0^\infty [\mathbf{x}(t)^T Q\mathbf{x}(t) + \mathbf{u}(t)^T R\mathbf{u}(t)]\, dt$$

- $\mathbf{x}(t)^T Q\mathbf{x}(t)$ is called state cost, $\mathbf{u}(t)^T R\mathbf{u}(t)$ is called control cost
- Given a feedback law: $\mathbf{u}(t) = -K_{\mathrm{lqr}}\mathbf{x}(t)$, K_{lqr} can be found precisely
- In Matlab, there is a simple one-line function `lqr(A,B,Q,R)` to do this!

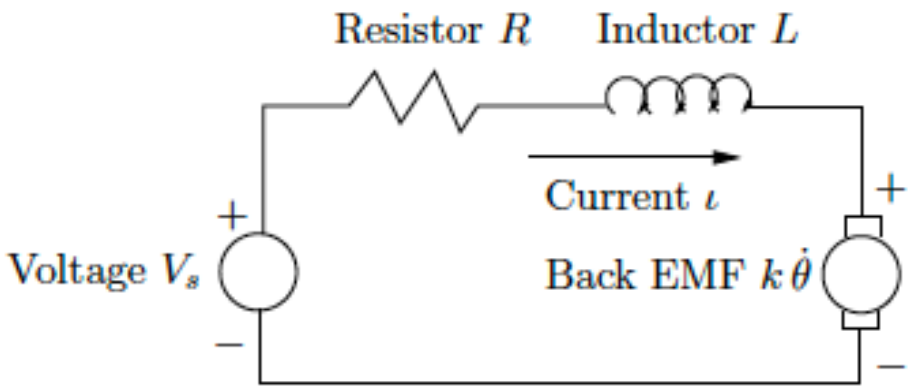
Proportional Integral Derivative (PID) controllers

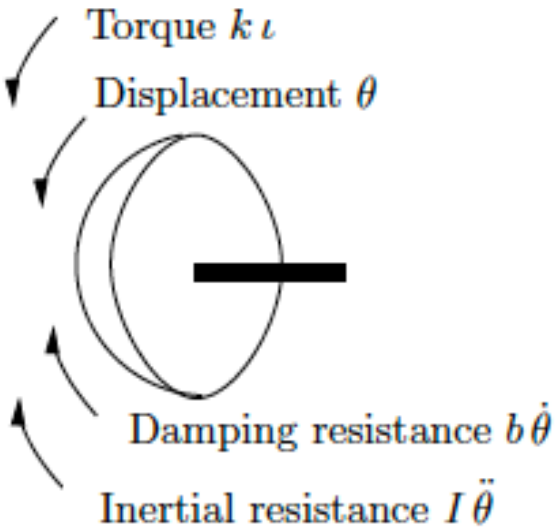


P-only controller

- 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

DC Motor

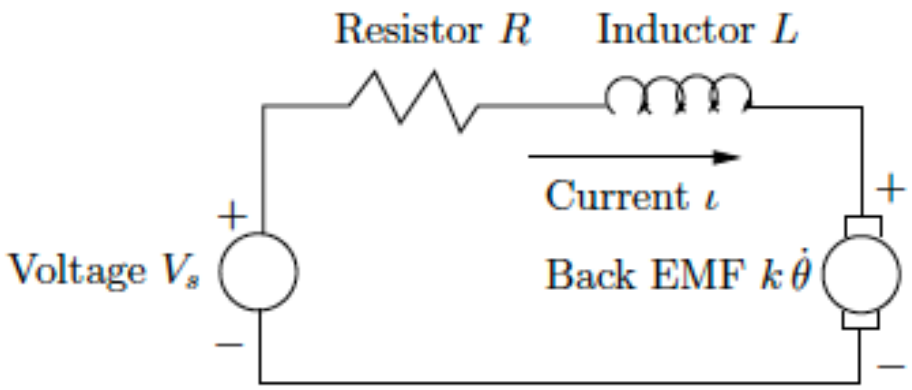


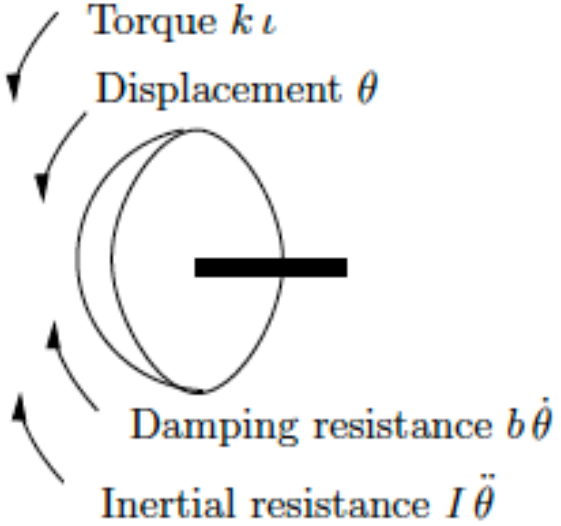


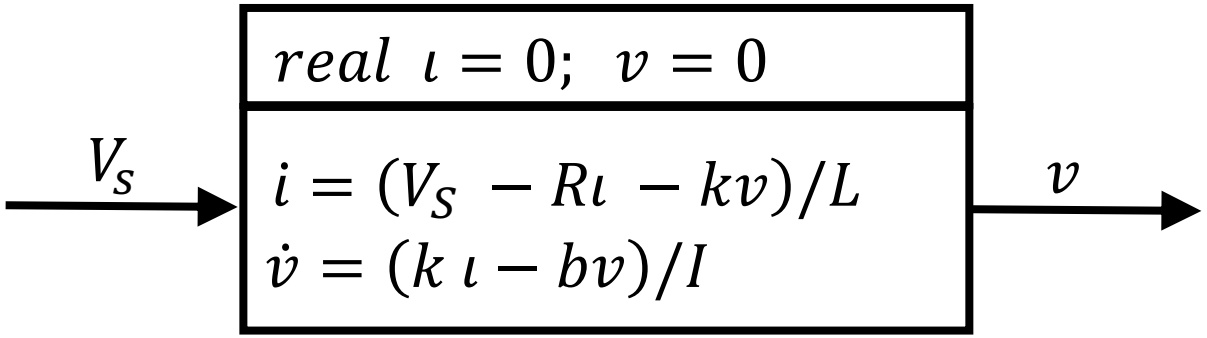
$$V_S = R\iota + L\dot{\iota} + k\dot{\theta}$$

$$I\ddot{\theta} + b\dot{\theta} = k\,\iota$$

DC Motor

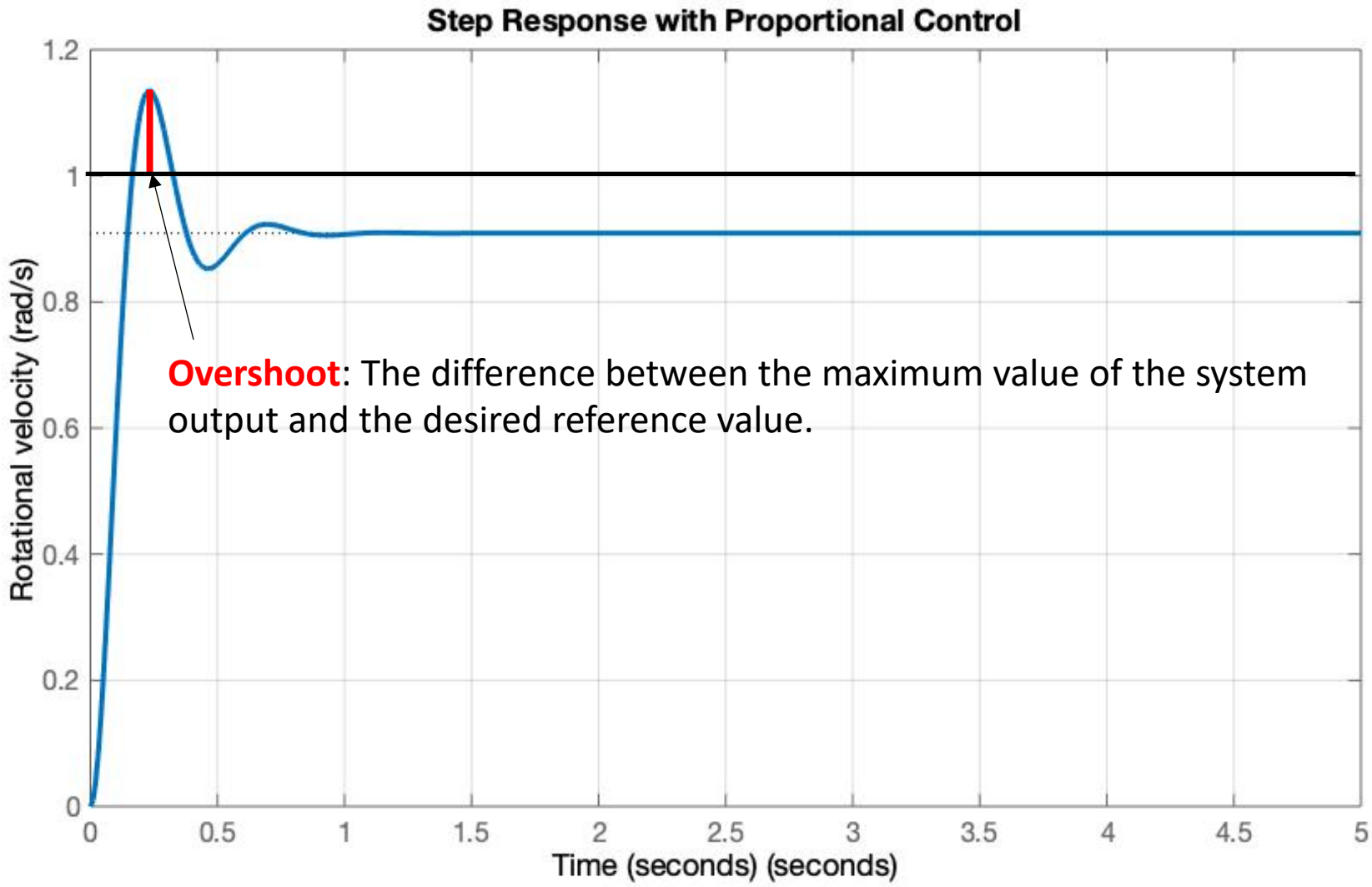

Resistor R
Inductor L
Current ι
Voltage V_s
Back EMF $k\dot{\theta}$
+
−
+
−


Torque $k\,\iota$
Displacement θ
Damping resistance $b\,\dot{\theta}$
Inertial resistance $I\,\ddot{\theta}$


$real\ \ \iota = 0;\ \ v = 0$
V_s
$i = (V_S - R\iota - kv)/L$
$\dot{v} = (k\,\iota - bv)/I$
v

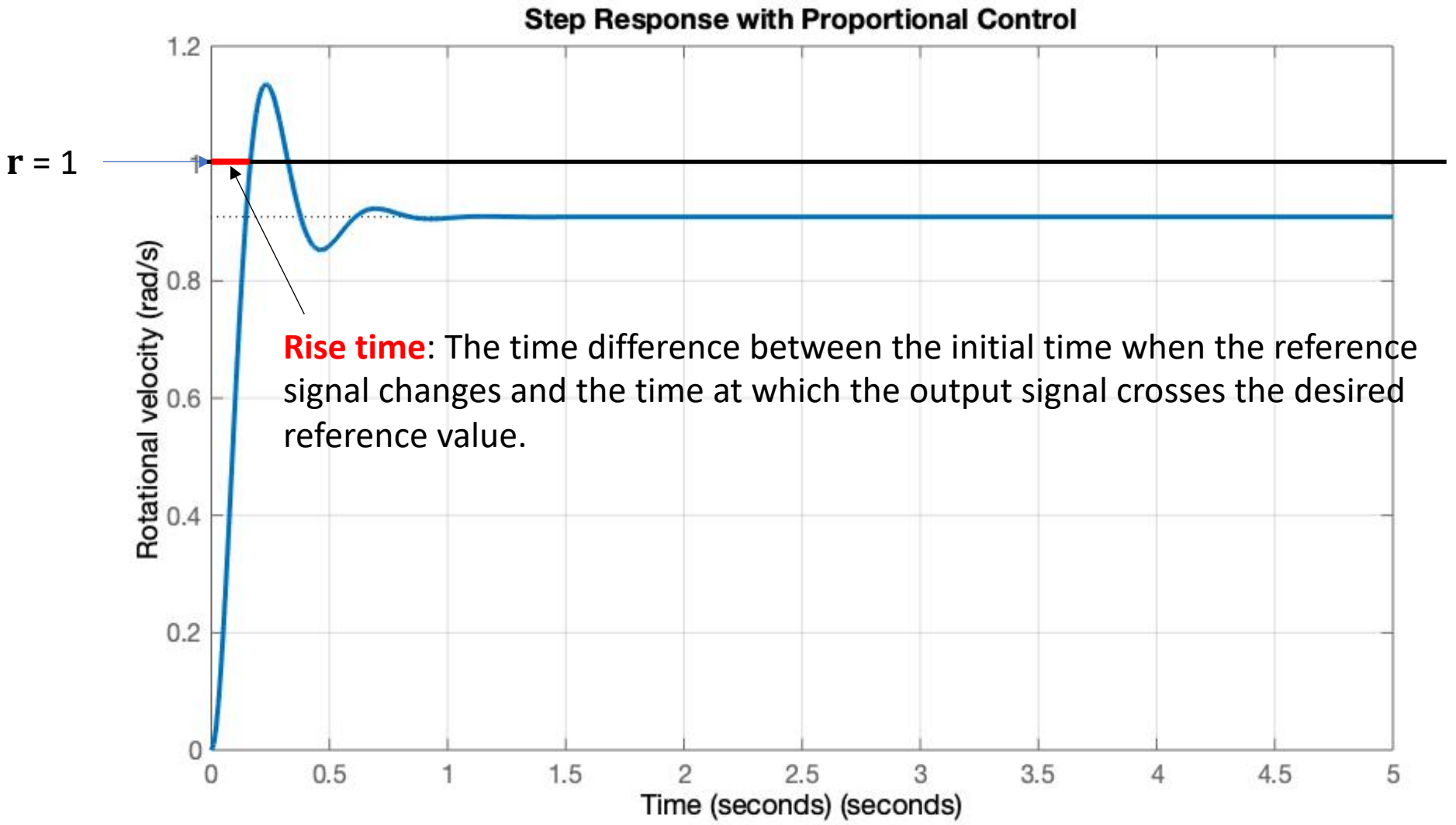
Measuring control performance

$\mathbf{r} = 1$



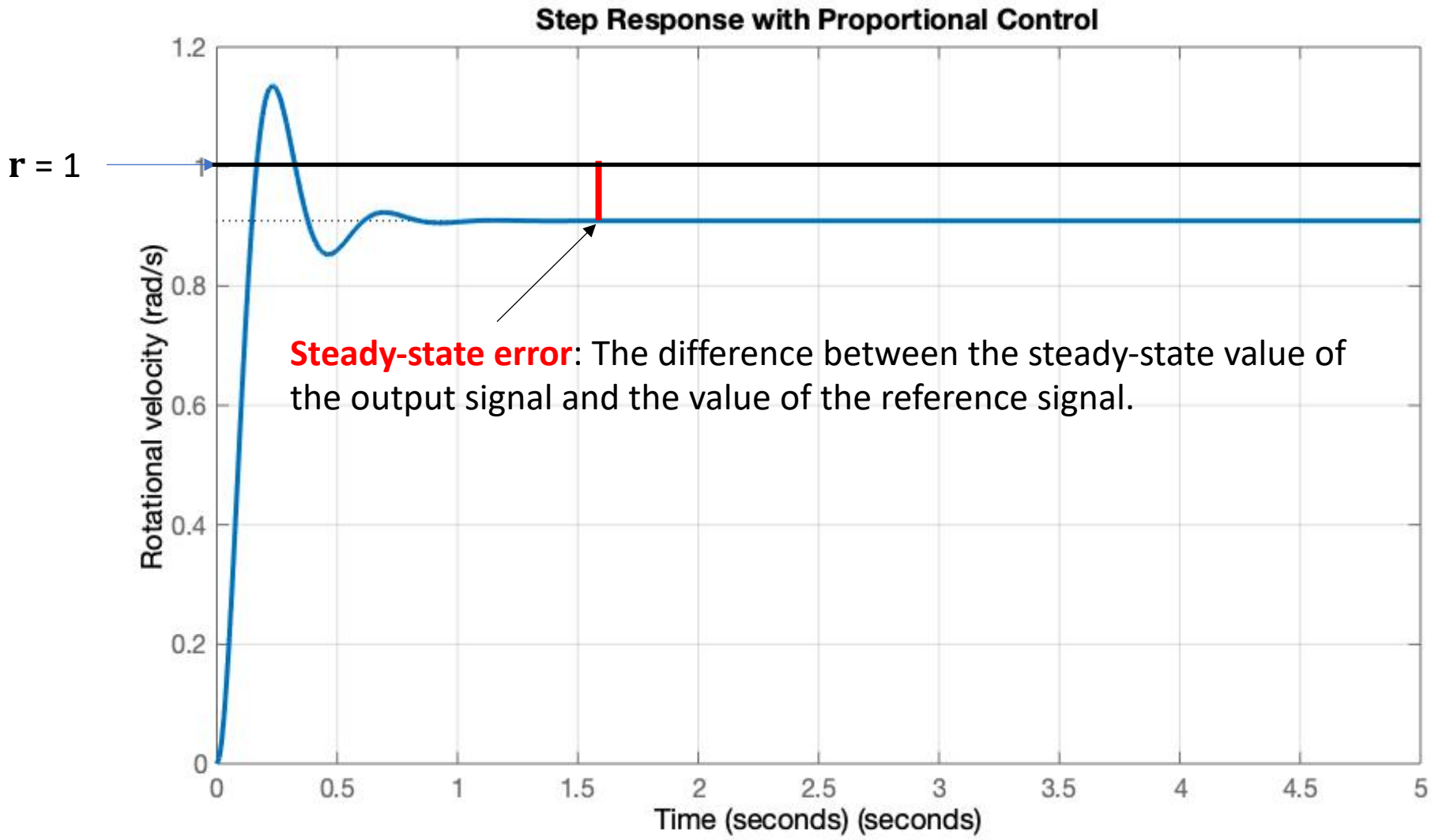
Overshoot: The difference between the maximum value of the system output and the desired reference value.

Measuring control performance



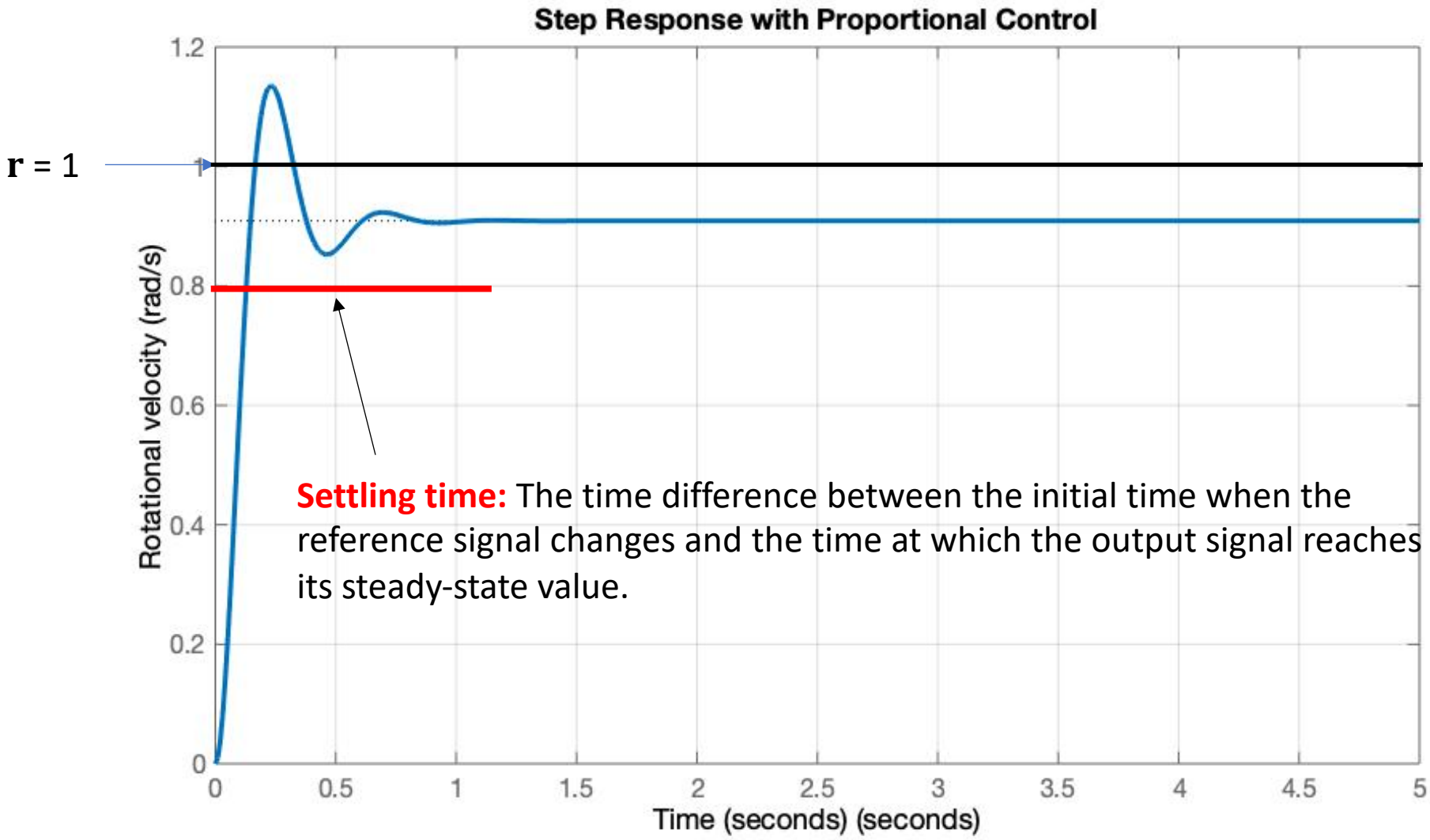
Rise time: The time difference between the initial time when the reference signal changes and the time at which the output signal crosses the desired reference value.

Measuring control performance



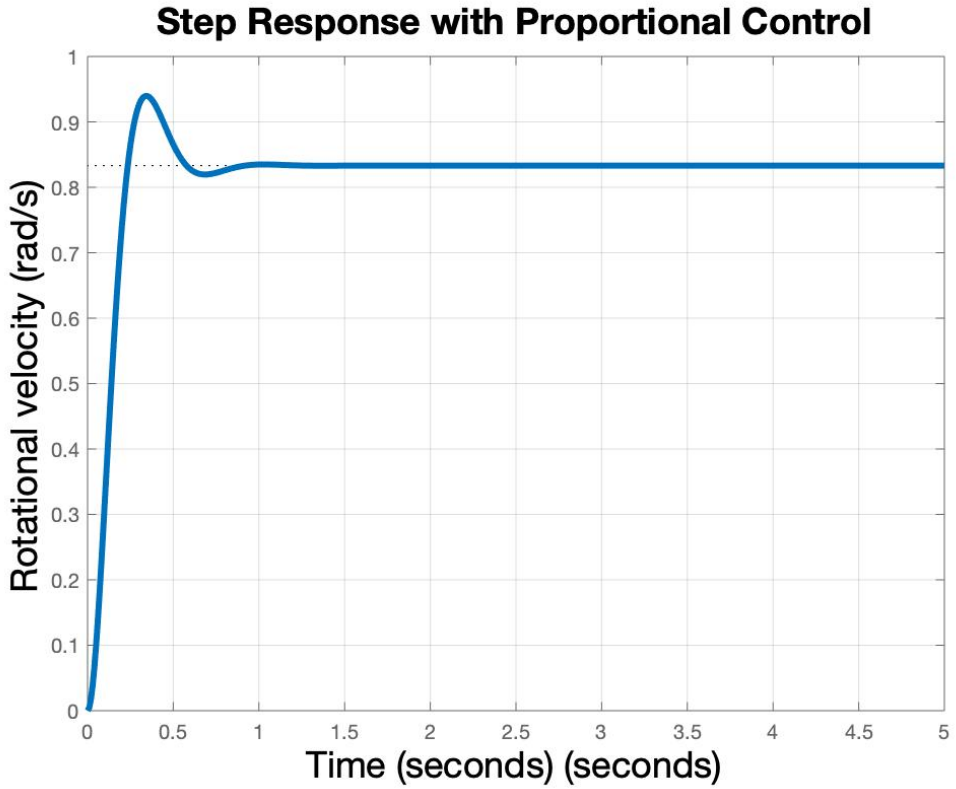
Steady-state error: The difference between the steady-state value of the output signal and the value of the reference signal.

Measuring control performance

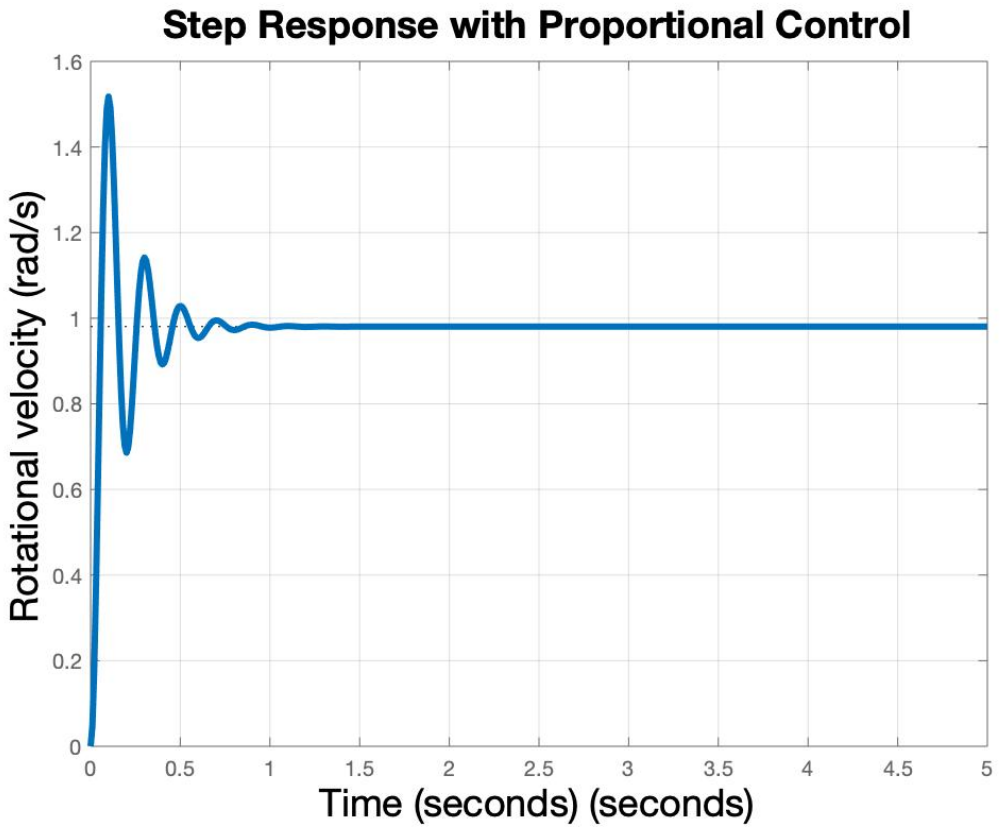


Settling time: The time difference between the initial time when the reference signal changes and the time at which the output signal reaches its steady-state value.

Measuring control performance



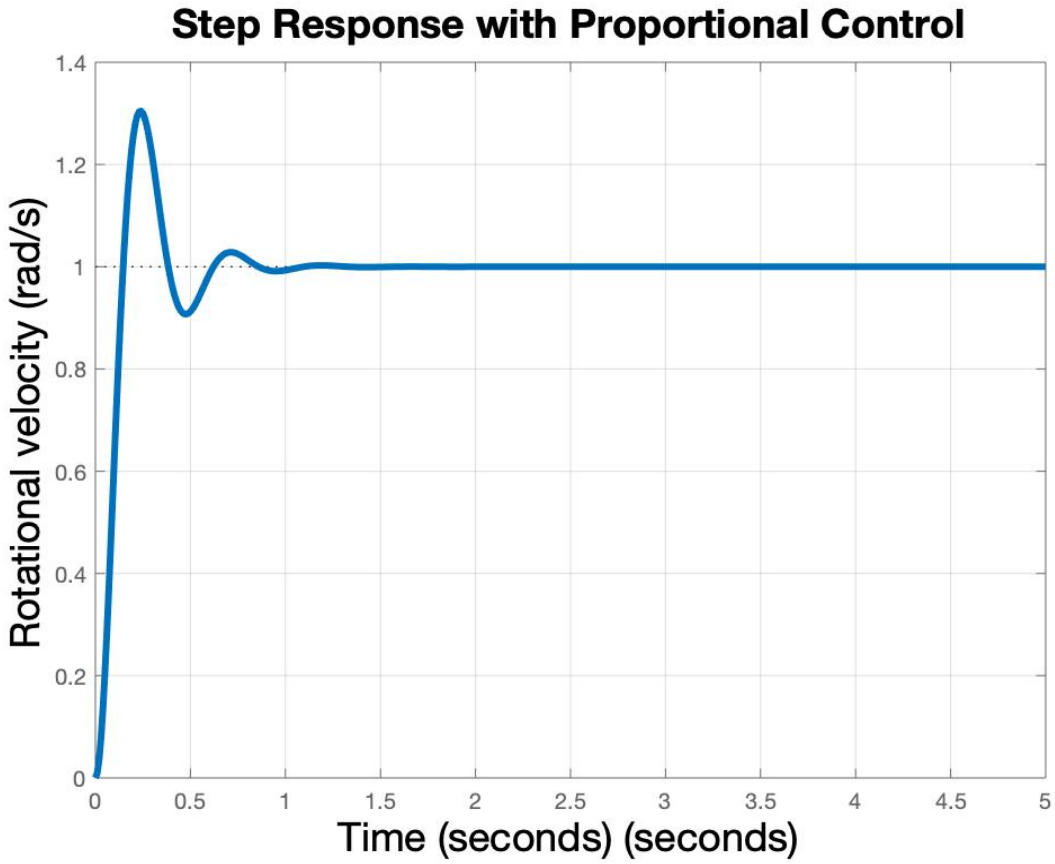
$\mathbf{K_P}$ = 50



$\mathbf{K_P}$ = 500

PI-controller

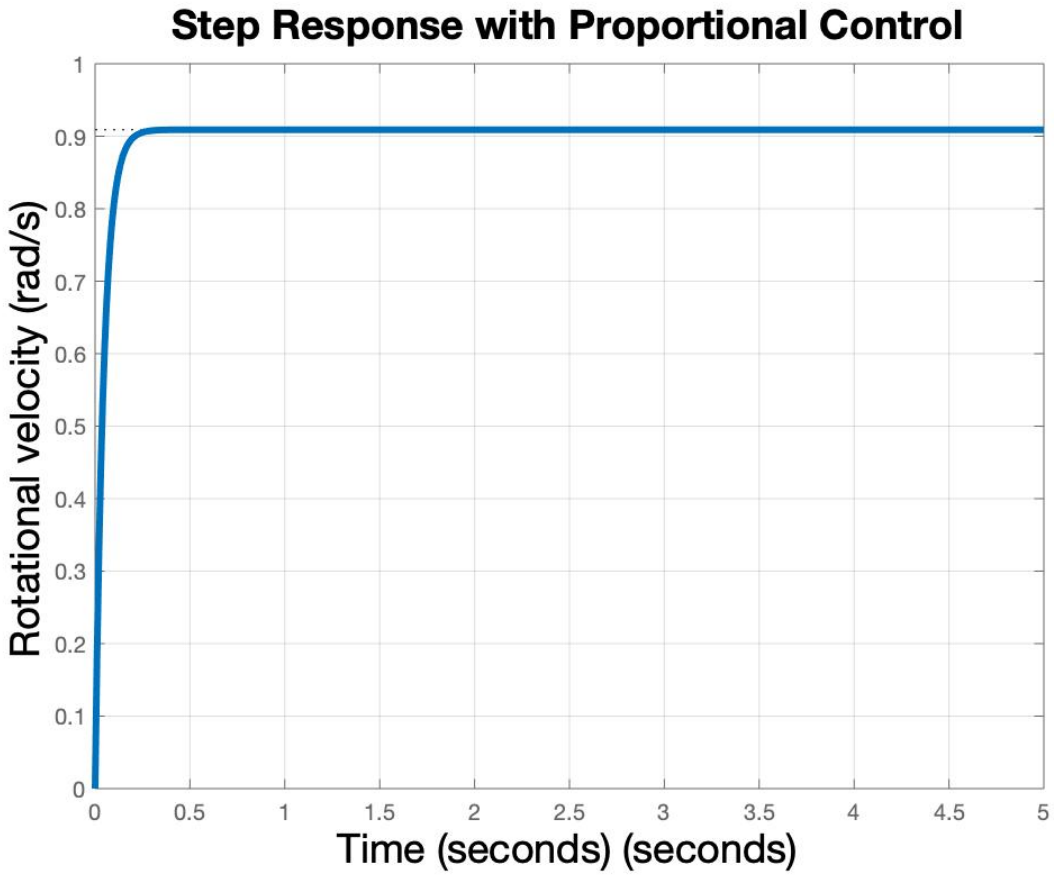
- Compute error signal $\mathbf{e} = \mathbf{r} - \mathbf{y}$
- Integral term: $K_I \int_0^t \mathbf{e}(\tau) d\tau$
  - K_I integral gain;
  - Feedback action proportional to cumulative error over time
  - If a small error persists, it will add up over time and push the system towards eliminating this error): eliminates offset/steady-state error



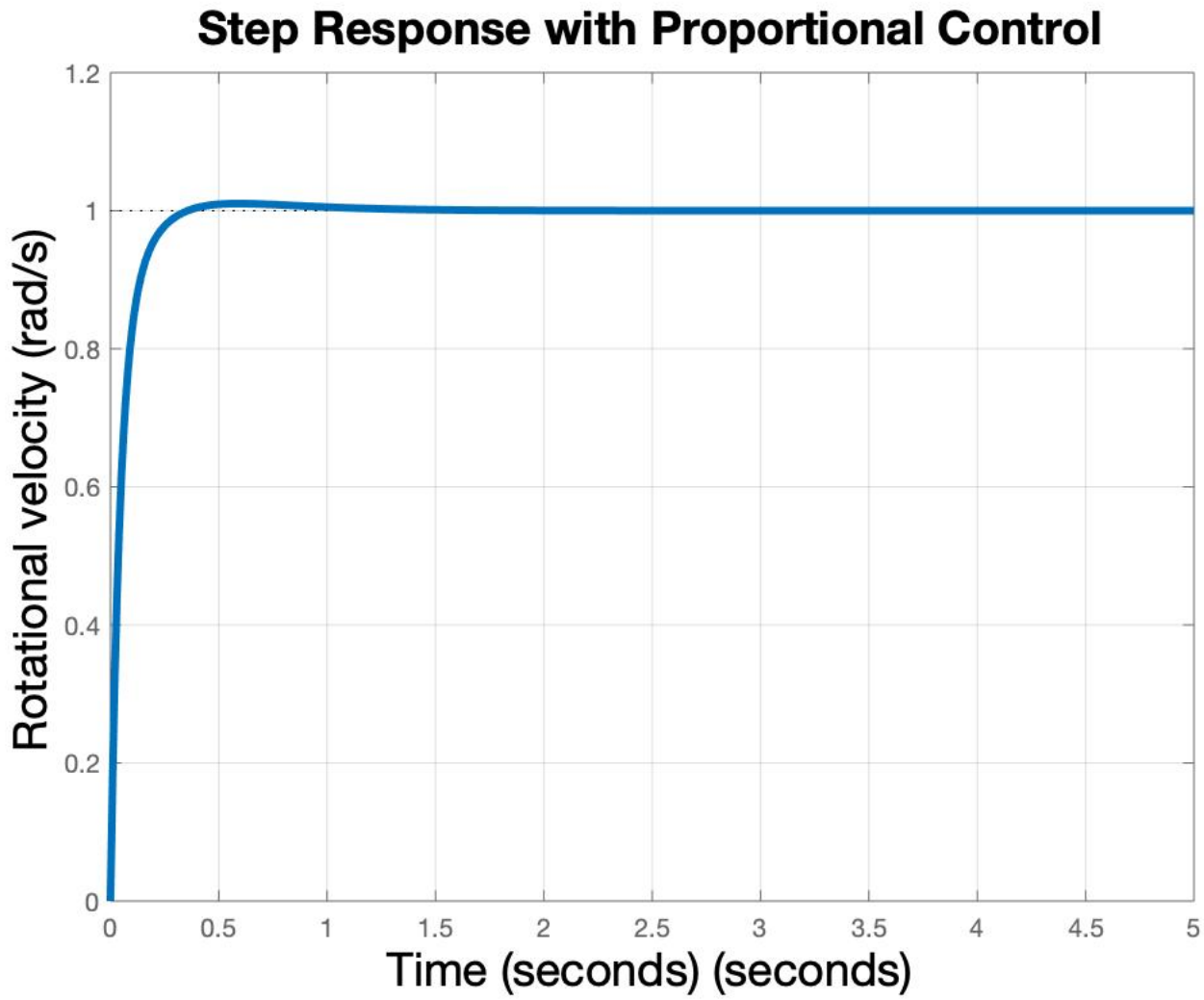
- Disadvantages:
  - Integral action by itself can increase instability
  - Integrator term can accumulate error and suggest corrections that are not feasible for the actuators (integrator windup)
    - Real systems “saturate” the integrator beyond a certain value

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



PI-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)

Nonlinear Control

Feedback Linearization

- Main idea: Try to choose control such the nonlinear system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ becomes linear

Input Transformation

- This operation is called input transformation, which leads to exact cancellation of a nonlinearity, giving rise to a linear equation
- Also known as exact feedback linearization or dynamic inversion
- Using feedback ***to linearize*** the system
- Example:
  - $\dot{x_1} = x_2$
  - $\dot{x_2} = (-x_2 - \cos x_1) + bu$
  - Let's define a new control input v such that, $u = \frac{1}{b}(v + x_2 + \cos x_1)$
    - $\dot{x_1} = x_2$
    - $\dot{x_2} = v$

State Transformation

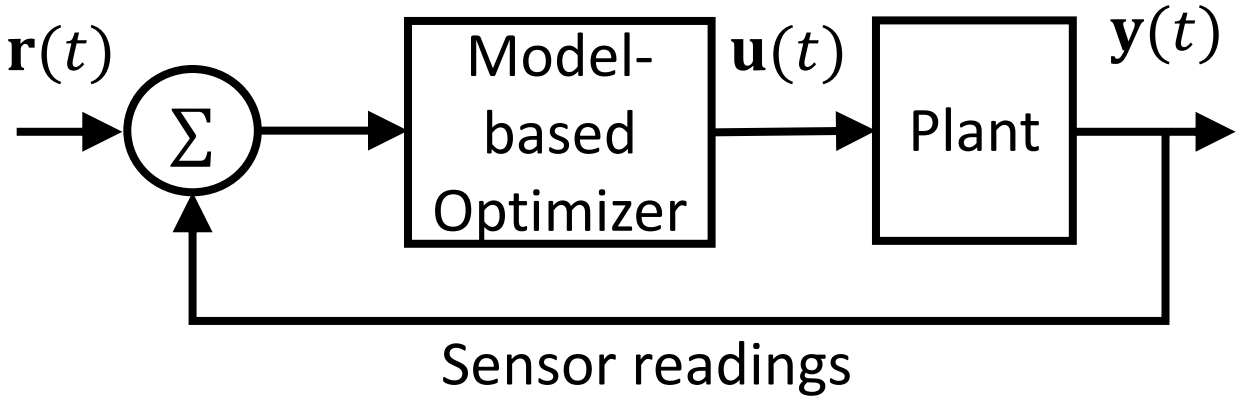
- Consider system:
  - $\dot{x_1} = a \ \sin x_2$
  - $\dot{x_2} = -x_1^2 + u$
- How do we cancel out $\sin x_2$?
- We can first change variables by a nonlinear transformation:
  - $z_1 = x_1, z_2 = a \sin x_2$
- Now, $\dot{z_1} = z_2$, and
  - $\dot{z_2} = \dot{x_2}\, a \cos x_2 = a(-x_1^2 + u) \cos x_2 = a(-z_1^2 + u) \cos \sin^{-1} \frac{z_2}{a}$

State Transformation

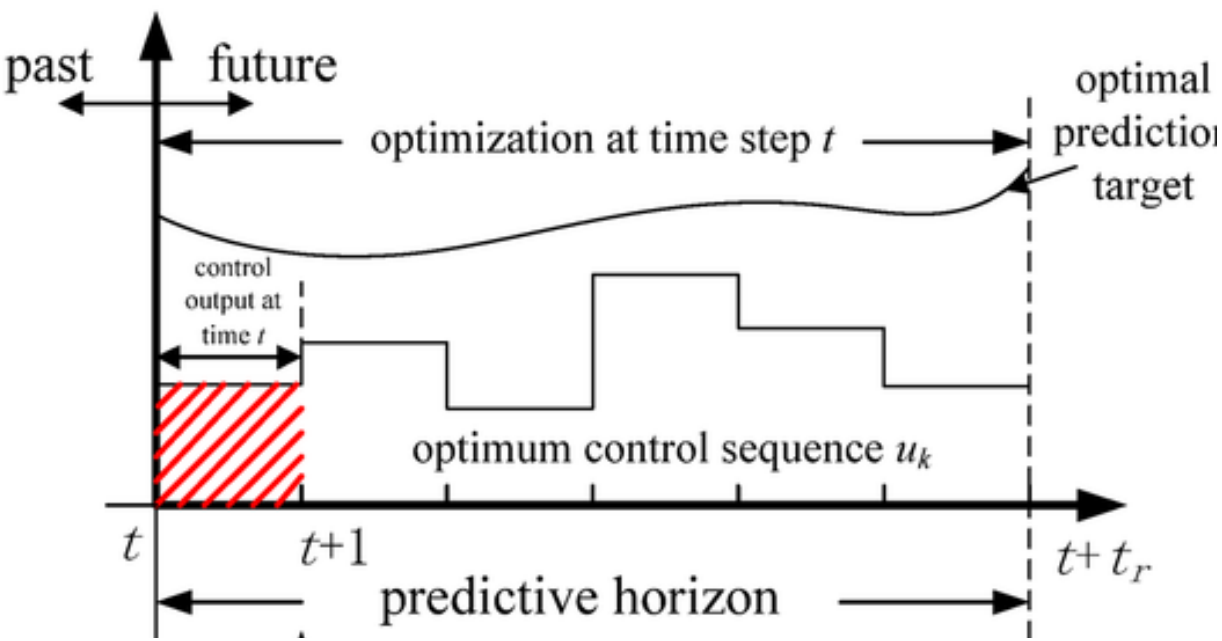
- Equations rewritten:
  - $\dot{z_1} = z_2$
  - $\dot{z_2} = a(-z_1^2 + u)\cos\sin^{-1}\frac{z_2}{a}$
- Now we can pick $u = z_1^2 + \frac{1}{a\cos\sin^{-1}\frac{z_2}{a}}v$
- Rewriting in terms of x's:
  - $u = x_1^2 + \frac{1}{a\cos x_2}v$
- This gives us a linear system $\dot{z_1} = z_2; \dot{z_2} = v$, which we can again stabilize using linear system methods

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



Model Predictive Control



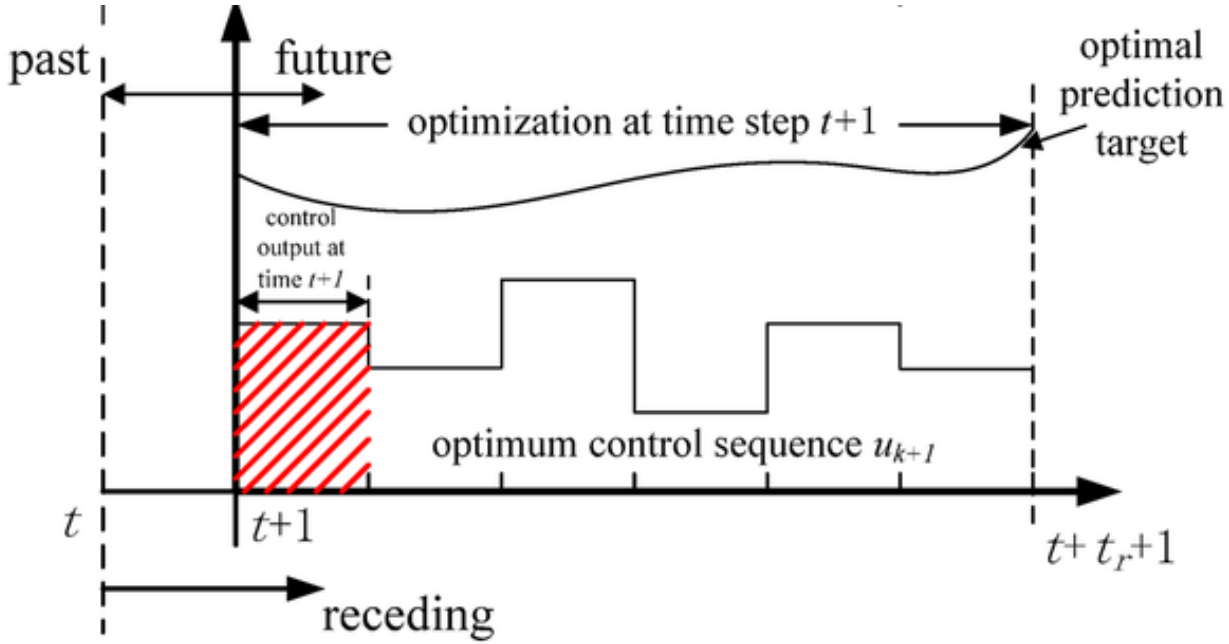


Image from: https://tinyurl.com/yaej43x5

Model Predictive Control

- Create difference equation: $\mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k]); \mathbf{y}[k] = g(\mathbf{x}[k])$
- At time t, solve an optimal control problem over next N steps:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{k=0}^{N-1} \|\mathbf{y}[t+k] - r[t]\|^2 + \rho\|\mathbf{u}[\mathrm{t}+\mathrm{k}]\|^2$$

$$s.t.\ \mathbf{x}[t+k+1] = f(\mathbf{x}[t+k], \mathbf{u}[t+k])$$
$$\mathbf{y}[t+k] = \mathbf{g}(\mathbf{x}[t+k])$$

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}, \mathbf{y}_{\min} \leq \mathbf{y} \leq y_{\max}$$

- Only apply optimal control input value $\mathbf{u}^*$ at time t
- At time $t+1$: get new measurements, repeat optimization

Observer design

Observability

When the controller can observe the state of the system only partially via the output vector, the controller needs to estimate the state of the system based on the observation of the output signal.

Observability Matrix: $W = (C \quad CA \quad CA^2 \quad \dots \quad CA^{n-1})^T$, where n is the state-dim

The pair (A, B) is observable if the rank of $W(A, B)$ is full (i.e. rows are linearly independent)

To reconstruct the internal state:

- For linear systems (with no noise), this is done with the use of state estimators or observers
- For linear systems with noisy measurements and possible “process noise” in the system itself : we use Kalman filter

What is state estimation and why is it needed?

- 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 n 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)
- 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, …

Luenberger observer

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

- $\mathbf{u}(t) = -K_{\mathrm{lqr}}\hat{\mathbf{x}}(t)$,

- The observer error $e = \mathbf{x} - \hat{\mathbf{x}}$ satisfies the equation: $\dot{e} = (A - LC)e$

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 $\mathrm{var}(\mathrm{w})$: 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

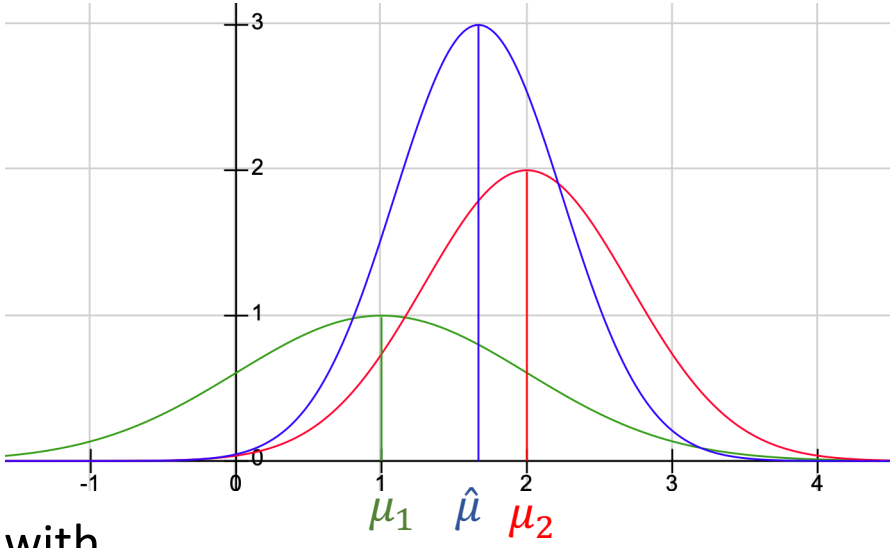
Data fusion example

- 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
  - $\hat{x} = \frac{(z_1/\sigma_1^2) + (z_2/\sigma_2^2)}{(1/\sigma_1^2)+(1/\sigma_2^2)} = kz_1 + (1-k)z_2$, where $k = \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}$
  - $\hat{\mu} = \hat{x}, \hat{\sigma}^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$
- 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:

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

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

$\mathbf{x}_k, \mathbf{x}_{k-1}$	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