IDENTIFICATION, ANALYSIS AND CONTROL OF DYNAMICAL SYSTEMS

Alberto Bemporad

`cse.lab.imtlucca.it/~bemporad/intro_control_course.html`

Academic year 2020-2021

IMT SCHOOL FOR ADVANCED STUDIES LUCCA

COURSE CONTENTS

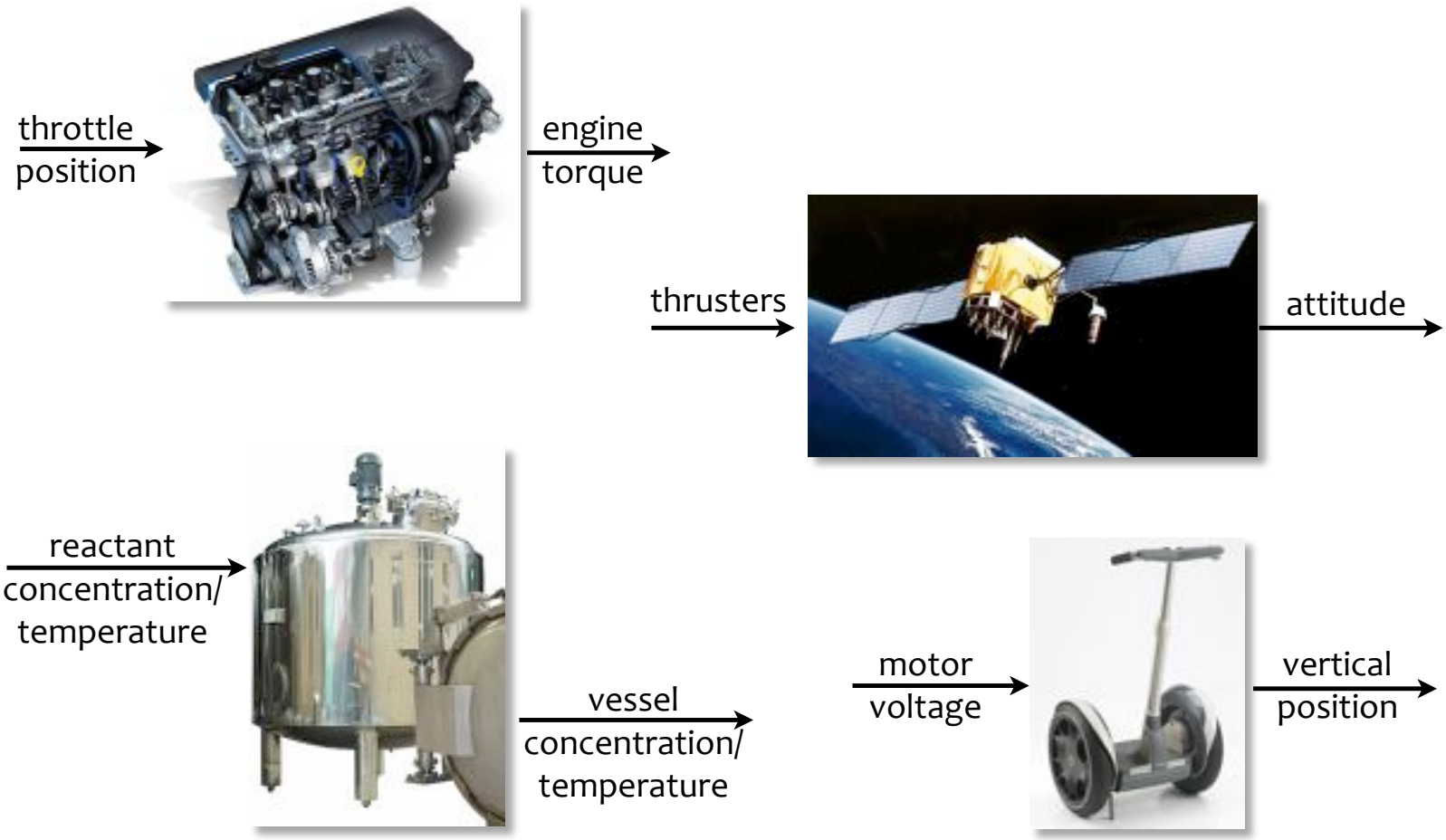
1. Linear dynamical systems in continuous and discrete-time
2. Linearization and discretization, stability analysis
3. Controllability and observability analysis
4. Synthesis of feedback controllers and state estimators
5. System identification (=learn dynamical models from data)

DYNAMICAL SYSTEMS

DYNAMICAL SYSTEMS

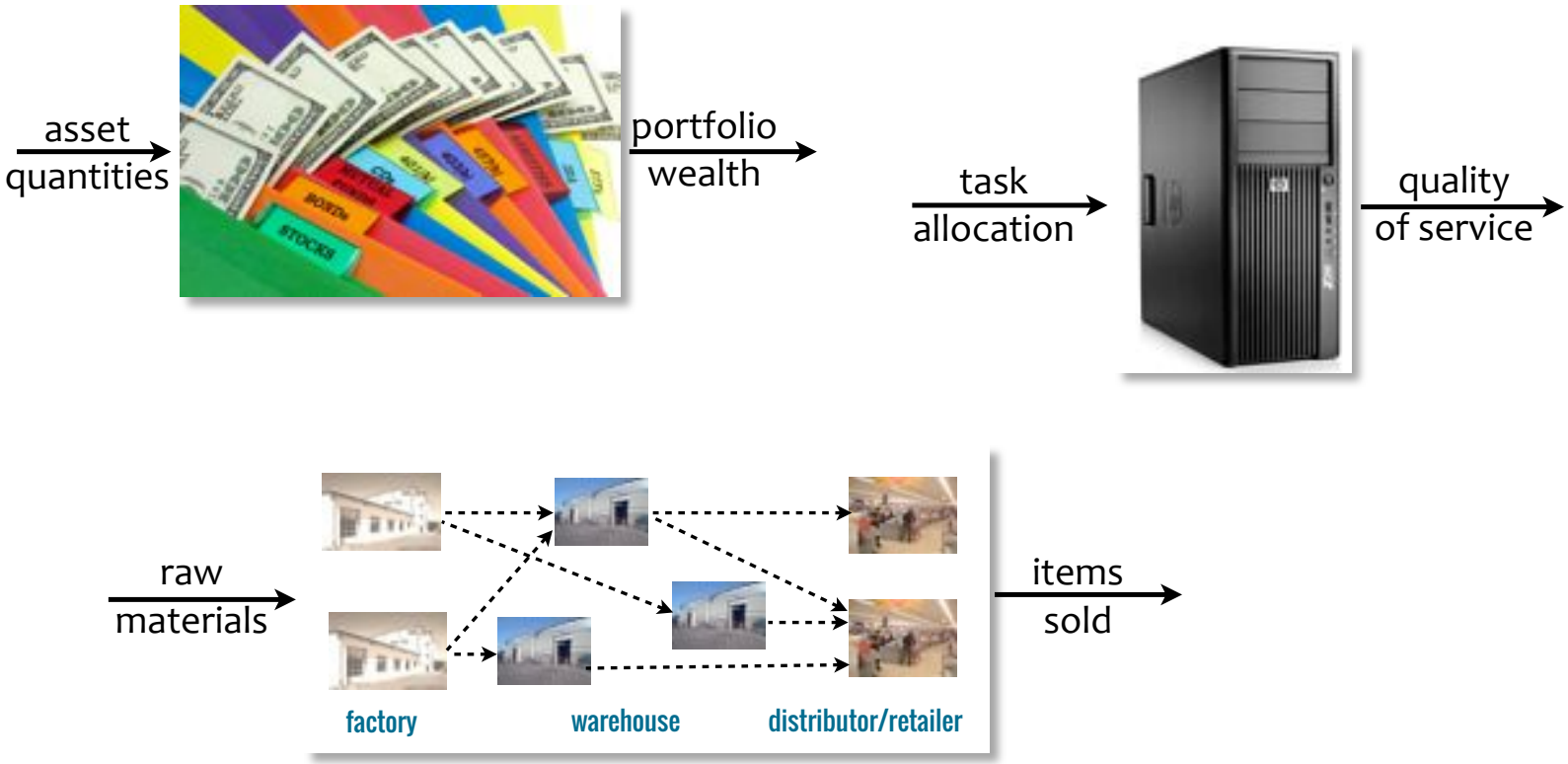
- A **dynamical system** is an object (or a set of objects) that evolves over time, possibly under external excitations.

- Examples: an engine, a satellite, a tank reactor, a human transporter, ...



DYNAMICAL SYSTEMS

- ... a supply chain, a portfolio, a computer server



- The way the system evolves over time is called the **dynamics** of the system.

DYNAMICAL MODELS

- A **dynamical model** of a system is a set of mathematical laws that explain how the system evolves over time, usually under the effect of external excitations, in a **quantitative** way.

- What is the purpose of a dynamical model ?

  1. Understand the system ("How does X influence Y ?")

  2. Simulation ("What happens if I apply action Z on the system ?")

  3. Estimate ("How to estimate variable X from measuring Y ?")

  4. Control ("How to make the system behave autonomously the way I want ?")

LINEAR SYSTEMS

CONTINUOUS-TIME LINEAR SYSTEMS

- System of n first-order linear ordinary differential equations (ODEs) with inputs

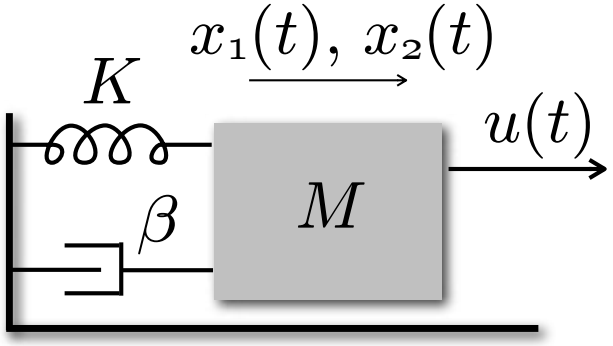
$$\begin{cases} \dot{x}_1(t) &= a_{11}x_1(t) + \ldots + a_{1n}x_n(t) &+b_1u(t) \\ \dot{x}_2(t) &= a_{21}x_1(t) + \ldots + a_{2n}x_n(t) &+b_2u(t) \\ \vdots & \vdots & \vdots \\ \dot{x}_n(t) &= a_{n1}x_1(t) + \ldots + a_{nn}x_n(t) &+b_nu(t) \\ x_1(0) = x_{10}, & \ldots & x_n(0) = x_{n0} \end{cases} \qquad \left[\dot{x} = \frac{dx}{dt}\right]$$

- Set $x = [x_1 \ \ldots \ x_n]' \in \mathbb{R}^n$. The equivalent **matrix form** of the linear ODE system is the so-called **linear system**

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

with initial condition $x(0) = x_0$, with vector $x_0 = [x_{10} \ \ldots \ x_{n0}]' \in \mathbb{R}^n$

EXAMPLE: MASS-SPRING-DAMPER SYSTEM



$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ M\dot{x}_2(t) = u - \beta x_2(t) - K x_1(t) \end{cases}$$

velocity = derivative of traveled space

Newton's law

Rewrite as the 2nd order linear system

$$\begin{cases} \frac{dx_1(t)}{dt} = x_2(t) \\ \frac{dx_2(t)}{dt} = -\frac{\beta}{M} x_2(t) - \frac{K}{M} x_1(t) + \frac{1}{M} u(t) \end{cases}$$

or in matrix form

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{\beta}{M} \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix}}_{B} u(t)$$

n^{th}-ORDER LINEAR ODE WITH INPUT

$$\frac{dy^{(n)}(t)}{dt^n} + a_{n-1}\frac{dy^{(n-1)}(t)}{dt^{n-1}} + \cdots + a_1\dot{y}(t) + a_0 y(t)$$
$$= b_{n-1}\frac{du^{(n-1)}(t)}{dt} + b_{n-2}\frac{du^{(n-2)}(t)}{dt} + \cdots + b_1\dot{u}(t) + b_0 u(t)$$

By inspection, the n^{th}-order ODE = 1^{st}-order linear system of ODEs

$$\begin{cases} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= x_3(t) \\ \vdots & \vdots \\ \dot{x}_n(t) &= -a_0 x_1(t) + \ldots - a_{n-1} x_n(t) + u(t) \\ y(t) &= b_0 x_1(t) + \ldots + b_{n-1} x_n(t) \end{cases}$$

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

$$A = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{n-1} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} b_0 & b_1 & b_2 & \ldots & b_{n-1} \end{bmatrix}, D = 0$$

The linear system of 1^{st}-order ODEs is called the **state-space realization** of the n^{th}-order ODE. There are infinitely many state-space realizations.

LAGRANGE'S FORMULA

- Starting from the initial condition $x(0) = x_0$, the continuous-time linear system $\dot{x} = Ax + Bu$ has the unique solution $x(t)$

$$x(t) = \underbrace{e^{At}x_0}_{\text{natural response}} + \underbrace{\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau}_{\text{forced response}}$$

- The **exponential matrix** is defined as

$$e^{At} \triangleq I + At + \frac{A^2t^2}{2} + \ldots + \frac{A^nt^n}{n!} + \ldots$$

(Moler, Van Loan, 2003)

MATLAB
E=expm(A*t)

Python
from scipy.linalg import expm E=expm(A*t)

STATE VECTOR

- Given $x(0)$ and $u(t), \forall t \in [0, T]$, Lagrange's formula allows us to compute $x(t)$ and $y(t), \forall t \in [0, T]$

- Generally speaking, the **state** of a dynamical system is a set of variables that completely summarizes the past history of the system. It allows us to predict its future motion

- Therefore, by knowing the initial state $x(0)$ we can neglect all past history $u(-t), x(-t), \forall t \geq 0$

- The dimension n of the state $x(t) \in \mathbb{R}^n$ is called the **order** of the system

EIGENVALUES AND EIGENVECTORS

- Let us recall some basic concepts of linear algebra:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \text{square matrix of order } n,\ A \in \mathbb{R}^{n \times n}$$

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad \text{identity matrix of order } n$$

- **Characteristic equation** of A:

$$\det(\lambda I - A) = 0$$

- **Characteristic polynomial** of A:

$$P(\lambda) = \det(\lambda I - A) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0$$

EIGENVALUES AND EIGENVECTORS

- The **eigenvalues** of $A \in \mathbb{R}^{n\times n}$ are the roots $\lambda_1, ..., \lambda_n$ of its characteristic polynomial

$$\det(\lambda_i I - A) = 0, \quad i = 1, 2, \ldots, n$$

- An **eigenvector** of A is any vector $v_i \in \mathbb{R}^n$ such that $Av_i = \lambda_i v_i$ for some $i = 1, 2, \ldots, n$.
- The **diagonalization** of A is $A = T\Lambda T^{-1}$, where

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix} = T^{-1}AT,\ T = [v_1|v_2|\ldots|v_n]$$

(not all matrices A are diagonalizable, see **Jordan normal form**)

- **Algebraic multiplicity** of λ_i = number of coincident roots λ_i of $\det(\lambda I - A)$
- **Geometric multiplicity** of λ_i = number of linearly independent eigenvectors v_i such that $Av_i = \lambda_i v_i$.

EIGENVALUES AND MODES

- Let $u(t) \equiv 0$ and assume A diagonalizable
- The state trajectory is the natural response

$$\begin{aligned} x(t) &= e^{At}x(0) = Te^{\Lambda t}\underbrace{T^{-1}x_0}_{\alpha} = [v_1 \dots v_n]\begin{bmatrix} e^{\lambda_1 t} & \dots & 0 \\ & \ddots & \\ 0 & \dots & e^{\lambda_n t} \end{bmatrix}\alpha \\ &= \begin{bmatrix} v_1 e^{\lambda_1 t} & \dots & v_n e^{\lambda_n t} \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i \end{aligned}$$

  where v_i=eigenvector of A, λ_i=eigenvalue of A, $\alpha = T^{-1}x(0) \in \mathbb{R}^n$

- The evolution of the system depends on the eigenvalues λ_i of A, called **modes** of the system (sometimes we also refer to $e^{\lambda_i t}$ as the i-th mode)
- A mode λ_i is called **excited** if $\alpha_i \neq 0$

SOME CLASSES OF DYNAMICAL SYSTEMS

- **Causality**: a dynamical system is **causal** if $y(t)$ does not depend on future inputs $u(\tau)\ \forall \tau > t$ (**strictly causal** if $\forall \tau \geq t$)
- A linear system is always causal, and strictly causal iff $D = 0$
- **Linear time-varying (LTV) systems**:

$$\begin{cases} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{cases}$$

- When A, B, C, D are constant, the system is said **linear time-invariant** (LTI)
- A generalization of LTV systems are **linear parameter-varying (LPV)** systems

$$\begin{cases} \dot{x}(t) &= A(p(t))x(t) + B(p(t))u(t) \\ y(t) &= C(p(t))x(t) + D(p(t))u(t) \end{cases}$$

SOME CLASSES OF DYNAMICAL SYSTEMS

- **Multivariable systems**: more generally, a system can have m inputs ($u(t) \in \mathbb{R}^m$) and p outputs ($y(t) \in \mathbb{R}^p$). For linear systems, we still have

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

  with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, $D \in \mathbb{R}^{p \times m}$

- **Nonlinear systems**

$$\begin{cases} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)) \end{cases}$$

  where $f : \mathbb{R}^{n+m} \to \mathbb{R}^n$, $g : \mathbb{R}^{n+m} \to \mathbb{R}^p$ are (arbitrary) nonlinear functions

- **Time-varying nonlinear systems**

$$\begin{cases} \dot{x}(t) &= f(t, x(t), u(t)) \\ y(t) &= g(t, x(t), u(t)) \end{cases}$$

STABILITY

EQUILIBRIUM

- Consider the continuous-time nonlinear system

$$\begin{cases} \dot{x}(t) &= f(x(t),u(t)) \\ y(t) &= g(x(t),u(t)) \end{cases}$$

Definition

A state $x_r \in \mathbb{R}^n$ and an input $u_r \in \mathbb{R}^m$ are an **equilibrium pair** if for initial condition $x(0) = x_r$ and constant input $u(t) \equiv u_r$ the state remains constant: $x(t) \equiv x_r, \forall t \geq 0$.

- Equivalent definition: (x_r, u_r) is an equilibrium pair if $f(x_r, u_r) = 0$
- x_r is called **equilibrium state**, u_r **equilibrium input**
- The definition generalizes to time-varying nonlinear systems

STABILITY

- Consider the nonlinear system

$$\begin{cases} \dot{x}(t) &= f(x(t), u_r) \\ y(t) &= g(x(t), u_r) \end{cases}$$

and let x_r be an equilibrium state, $f(x_r, u_r) = 0$

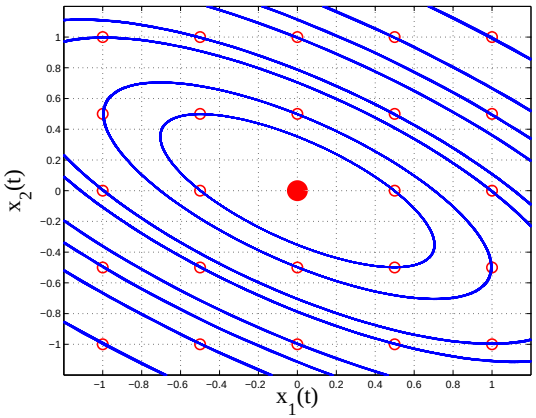
Definition

The equilibrium state x_r is **stable** if for each initial conditions $x(0)$ "close enough" to x_r, the corresponding trajectory $x(t)$ remains near x_r for all $t \geq 0$.

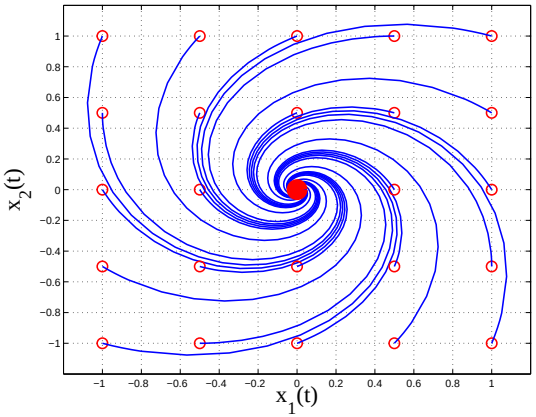
Math definition: $\forall \epsilon > 0 \, \exists \delta > 0 : \forall x(0)$ such that $\|x(0) - x_r\| < \delta \Rightarrow \|x(t) - x_r\| < \epsilon, \forall t \geq 0$.

- The equilibrium point x_r is called **asymptotically stable** if it is stable and $x(t) \to x_r$ for $t \to \infty$
- Otherwise, the equilibrium point x_r is called **unstable**

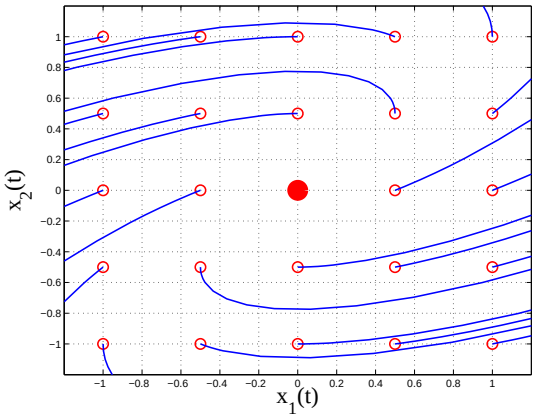
STABILITY OF EQUILIBRIA - EXAMPLES



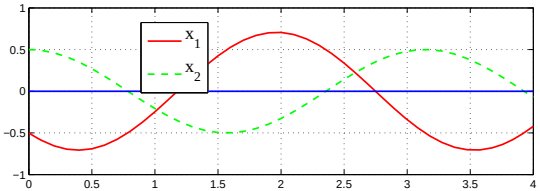
stable equilibrium

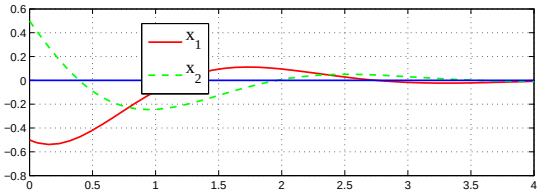


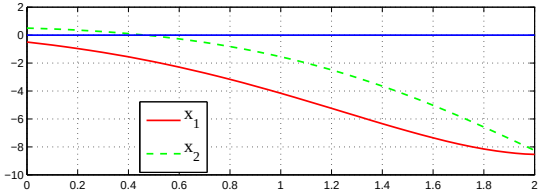
asymptotically stable equilibrium



unstable equilibrium







$$\frac{dx}{dt} = \begin{bmatrix} -2x_1(t) - 4x_2(t) \\ 2x_1(t) + 2x_2(t) \end{bmatrix}$$

$$\frac{dx}{dt} = \begin{bmatrix} -x_1(t) - 2x_2(t) \\ 2x_1(t) - x_2(t) \end{bmatrix}$$

$$\frac{dx}{dt} = \begin{bmatrix} 2x_1(t) - 2x_2(t) \\ x_1(t) \end{bmatrix}$$

STABILITY OF FIRST-ORDER LINEAR SYSTEMS

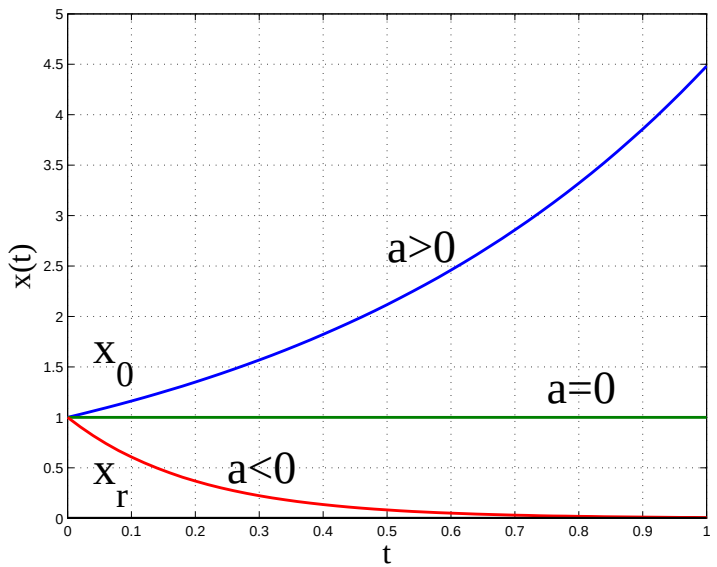
- Consider the first-order linear system

$$\dot{x}(t) = ax(t) + bu(t)$$

- $x_r = 0, u_r = 0$ is an equilibrium pair
- For $u(t) \equiv 0, \forall t \geq 0$, the solution is

$$x(t) = e^{at} x_0$$

- The origin $x_r = 0$ is
  - unstable if $a > 0$
  - stable if $a \leq 0$
  - asymptotically stable if $a < 0$



STABILITY OF CONTINUOUS-TIME LINEAR SYSTEMS

Since the natural response of $\dot{x} = Ax + Bu$ is $x(t) = e^{At}x_0$, the stability properties depend only on A. We can therefore talk about **system stability** of a linear system (A, B, C, D)

Theorem

Let $\lambda_1, \ldots, \lambda_m, m \leq n$ be the eigenvalues of $A \in \mathbb{R}^{n \times n}$.

The system $\dot{x} = Ax + Bu$ is

- asymptotically stable iff $\Re\lambda_i < 0, \forall i = 1, \ldots, m$
- (marginally) stable if $\Re\lambda_i \leq 0, \forall i = 1, \ldots, m$, and the eigenvalues with null real part have equal algebraic and geometric multiplicity
- unstable otherwise (in particular, if $\exists\, i$ such that $\Re\lambda_i > 0$).

The stability properties of a linear system only depend on the **real part** of the eigenvalues of matrix A

STABILITY OF CONTINUOUS-TIME LINEAR SYSTEMS

Proof:

- The natural response is $x(t) = e^{At}x_0$ ($e^{At} \triangleq I + At + \frac{A^2t^2}{2} + \ldots + \frac{A^nt^n}{n!} + \ldots$)
- If matrix A is diagonalizable[1] , $A = T\Lambda T^{-1}$,

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix} \Rightarrow e^{At} = T \begin{bmatrix} e^{\lambda_1 t} & 0 & \ldots & 0 \\ 0 & e^{\lambda_2 t} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & e^{\lambda_n t} \end{bmatrix} T^{-1}$$

- Take any eigenvalue $\lambda = a + jb$:

$$|e^{\lambda t}| = e^{at}|e^{jbt}| = e^{at}$$

- A is always diagonalizable if algebraic multiplicity = geometric multiplicity

□

[1] If A is not diagonalizable, it can be transformed to Jordan form. In this case the natural response $x(t)$ contains modes $t^j e^{\lambda t}$, $j = 0, 1, \ldots,$ alg. multiplicity - geom. multiplicity

LINEARIZATION OF NONLINEAR SYSTEMS

- Consider the nonlinear system

$$\begin{cases} \dot{x}(t) &= f(x(t),u(t)) \\ y(t) &= g(x(t),u(t)) \end{cases}$$

- Let (x_r, u_r) be an equilibrium, $f(x_r, u_r) = 0$
- Objective: investigate the dynamic behaviour of the system for small perturbations $\Delta u(t) \triangleq u(t) - u_r$ and $\Delta x(0) \triangleq x(0) - x_r$.
- The evolution of $\Delta x(t) \triangleq x(t) - x_r$ is given by

$$\begin{aligned} \dot{\Delta} x(t) &= \dot{x}(t) - \dot{x}_r = f(x(t),u(t)) \\ &= f(\Delta x(t) + x_r, \Delta u(t) + u_r) \\ &\approx \underbrace{\frac{\partial f}{\partial x}(x_r,u_r)}_{A}\Delta x(t) + \underbrace{\frac{\partial f}{\partial u}(x_r,u_r)}_{B}\Delta u(t) \end{aligned}$$

LINEARIZATION OF NONLINEAR SYSTEMS

- Similarly

$$\Delta y(t) \approx \underbrace{\frac{\partial g}{\partial x}(x_r, u_r)}_{C} \Delta x(t) + \underbrace{\frac{\partial g}{\partial u}(x_r, u_r)}_{D} \Delta u(t)$$

where $\Delta y(t) \triangleq y(t) - g(x_r, u_r)$ is the perturbation of the output from its equilibrium

- The perturbations $\Delta x(t)$, $\Delta y(t)$, and $\Delta u(t)$ are (approximately) ruled by the **linearized system**

$$\begin{cases} \dot{\Delta} x(t) &= A\Delta x(t) + B\Delta u(t) \\ \Delta y(t) &= C\Delta x(t) + D\Delta u(t) \end{cases}$$

LYAPUNOV'S STABILITY

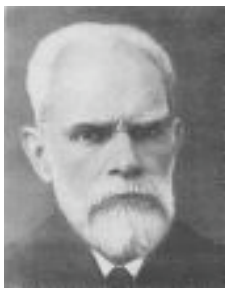
LYAPUNOV'S INDIRECT METHOD

- Consider the nonlinear system $\dot{x} = f(x)$, with f differentiable, and assume $x = 0$ is equilibrium point ($f(0) = 0$)
- Consider the linearized system $\dot{x} = Ax$, with $A = \left.\frac{\partial f}{\partial x}\right|_{x=0}$

1. If $x = 0$ is an asymptotically stable equilibrium for $\dot{x} = Ax \Rightarrow$ it is (locally) asymptotically stable for the nonlinear system

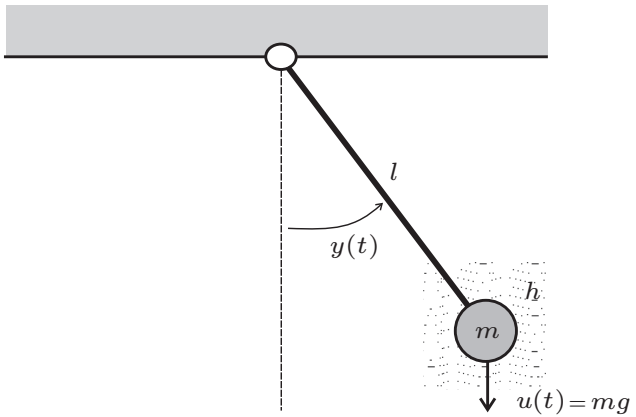
2. If $x = 0$ is an unstable equilibrium for $\dot{x} = Ax \Rightarrow$ it is unstable for the nonlinear system

3. If $x = 0$ is marginally stable for $\dot{x} = Ax \Rightarrow$ nothing can be said about its stability for the nonlinear system



Aleksandr Mikhailovich Lyapunov
(1857-1918)

EXAMPLE: PENDULUM



$y(t)$ = angular displacement

$\dot{y}(t)$ = angular velocity

$\ddot{y}(t)$ = angular acceleration

$u(t)$ = mg gravity force

$h\dot{y}(t)$ = viscous friction torque

l = pendulum length

ml^2 = pendulum rotational inertia

- mathematical model

$$ml^2\ddot{y}(t) = -lmg\sin y(t) - h\dot{y}(t)$$

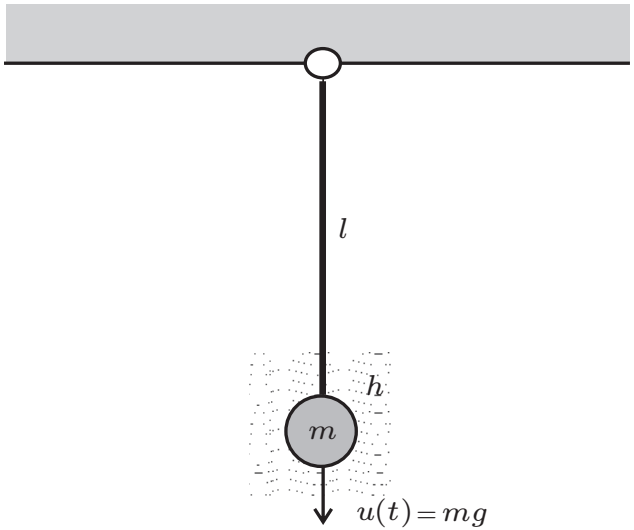
- in state-space form ($x_1 = y, x_2 = \dot{y}$)

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{l}\sin x_1 - Hx_2, \quad H \triangleq \frac{h}{ml^2} \end{cases}$$

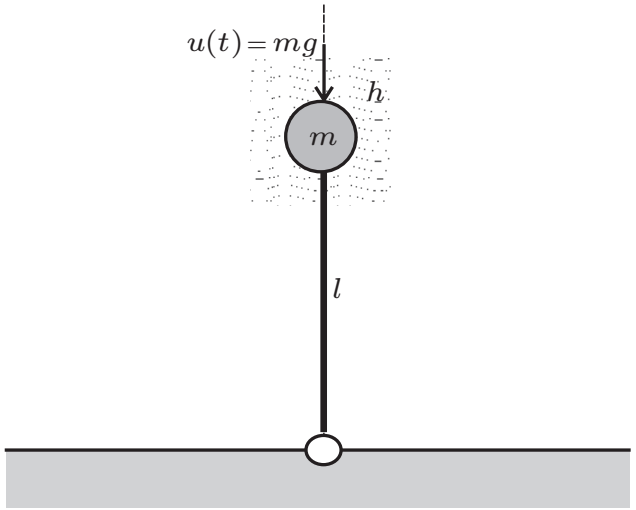
EXAMPLE: PENDULUM

Look for equilibrium states:

$$\begin{bmatrix} x_{2r} \\ -\frac{g}{l}\sin x_{1r} - H x_{2r} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{cases} x_{2r} = 0 \\ x_{1r} = \pm k\pi,\ k = 0, 1, \ldots \end{cases}$$



$x_{2r} = 0, x_{1r} = 0, \pm 2\pi, \ldots$



$x_{2r} = 0, x_{1r} = \pm\pi, \pm 3\pi, \ldots$

EXAMPLE: PENDULUM

- Linearize the system around $x_{1r} = 0, x_{2r} = 0$

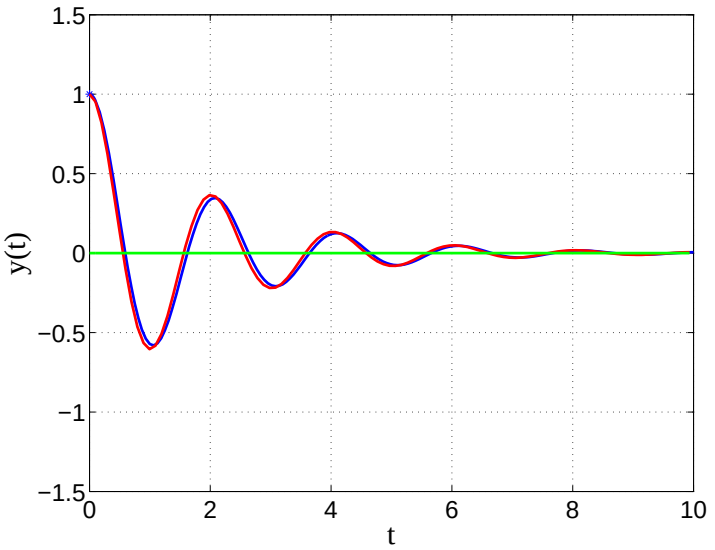
$$\Delta\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -H \end{bmatrix}}_{A} \Delta x(t)$$

- find the eigenvalues of A

$$\det(\lambda I - A) = \lambda^2 + H\lambda + \frac{g}{l} = 0 \Rightarrow \lambda_{1,2} = \frac{1}{2}\left(-H \pm \sqrt{H^2 - 4\frac{g}{l}}\right)$$

- $\Re\lambda_{1,2} < 0 \Rightarrow \dot{x} = Ax$ asymptotically stable

- by Lyapunov's indirect method $x_r = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ is also an asymptotically stable equilibrium for the pendulum



EXAMPLE: PENDULUM

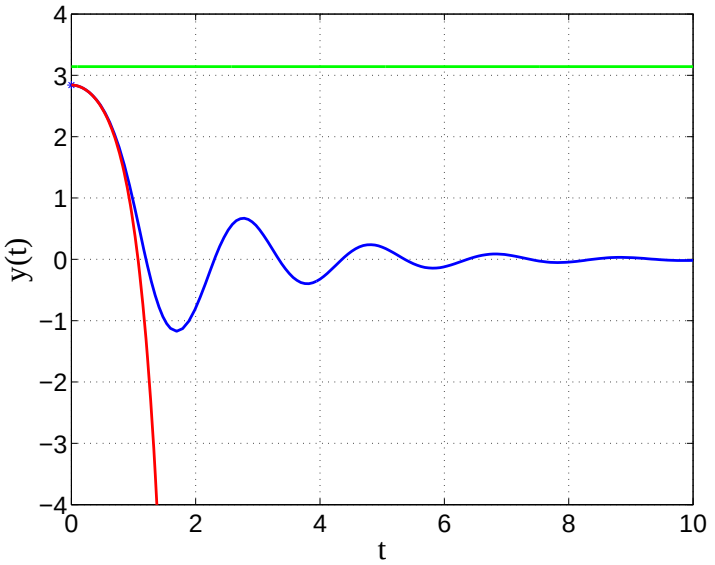
- Linearize the system around $x_{1r} = \pi, x_{2r} = 0$

$$\Delta\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ \frac{g}{l} & -H \end{bmatrix}}_{A} \Delta x(t)$$

- find the eigenvalues of A

$$\det(\lambda I - A) = \lambda^2 + H\lambda - \frac{g}{l} = 0 \Rightarrow \lambda_{1,2} = \frac{1}{2}\left(-H \pm \sqrt{H^2 + 4\frac{g}{l}}\right)$$

- $\lambda_1 < 0, \lambda_2 > 0 \Rightarrow \dot{x} = Ax$ unstable
- by Lyapunov's indirect method $x_r = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$ is also an unstable equilibrium for the pendulum



LYAPUNOV'S DIRECT METHOD

- A second method exists to analyze **global** stability of nonlinear systems, based on the concept of **Lyapunov functions**
- **Key idea**: if the energy of a system dissipates over time, the system asymptotically reaches a minimum-energy configuration
- Assumptions: consider the autonomous nonlinear system $\dot{x} = f(x)$, with $f(\cdot)$ differentiable, and let $x = 0$ be an equilibrium ($f(0) = 0$)
- Some definitions of positive definiteness of a function $V : \mathbb{R}^n \mapsto \mathbb{R}$
  - V is **locally positive definite** if $V(0) = 0$ and there exists a **ball** $B_\epsilon = \{x : \|x\|_2 \leq \epsilon\}$ around the origin such that $V(x) > 0 \, \forall x \in B_\epsilon \setminus 0$
  - V is **globally positive definite** if $B_\epsilon = \mathbb{R}^n$ (i.e. $\epsilon \to \infty$)
  - V is **negative definite** if $-V$ is positive definite
  - V is **positive semi-definite** if $V(x) \geq 0 \, \forall x \in B_\epsilon$
  - V is **negative semi-definite** if $-V$ is positive semi-definite

LYAPUNOV'S DIRECT METHOD

- Example: let $x = [x_1\ x_2]', V : \mathbb{R}^2 \to \mathbb{R}$
  - $V(x) = x_1^2 + x_2^2$ is globally positive definite
  - $V(x) = x_1^2 + x_2^2 - x_1^3$ is locally positive definite
  - $V(x) = x_1^4 + \sin^2(x_2)$ is locally positive definite and globally positive semi-definite

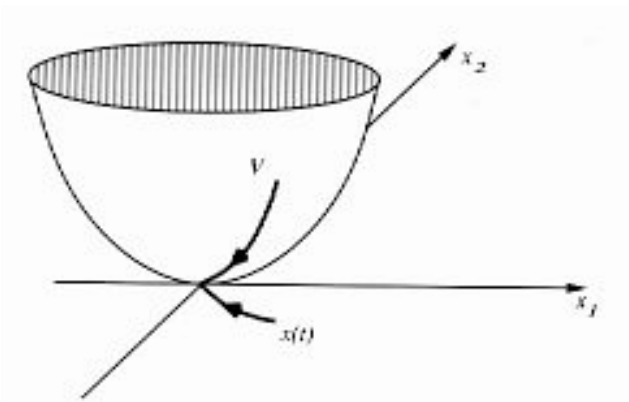
LYAPUNOV'S DIRECT METHOD

Theorem

Given the nonlinear system $\dot{x} = f(x)$, $f(0) = 0$, let $V : \mathbb{R}^n \mapsto \mathbb{R}$ be positive definite in a ball B_ϵ around the origin, $\epsilon > 0$, $V \in C^1(\mathbb{R})$. If the function

$$\dot{V}(x) = \nabla V(x)' \dot{x} = \nabla V(x)' f(x)$$

is negative definite on B_ϵ, then the origin is an asymptotically stable equilibrium point. If $\dot{V}(x)$ is only negative semi-definite on B_ϵ, then the the origin is a stable equilibrium point.



Such a function $V : \mathbb{R}^n \mapsto \mathbb{R}$ is called a **Lyapunov function** for the system $\dot{x} = f(x)$

EXAMPLE OF LYAPUNOV'S DIRECT METHOD

- Consider the following nonlinear system $\dot{x} = f(x)$ (Slotine, Li, 1991 - Example 3.8)

$$\begin{cases} \dot{x}_1 &= x_1(x_1^2 + x_2^2 - 2) - 4x_1x_2^2 \\ \dot{x}_2 &= 4x_1^2x_2 + x_2(x_1^2 + x_2^2 - 2) \end{cases}$$

- The state $x = 0$ is an equilibrium because $\dot{x} = f(0) = 0$
- Consider the candidate Lyapunov function

$$V(x_1, x_2) = x_1^2 + x_2^2$$

which is globally positive definite. Its time derivative $\dot{V}$ is

$$\dot{V}(x_1, x_2) = 2(x_1^2 + x_2^2)(x_1^2 + x_2^2 - 2)$$

- It is easy to check that $\dot{V}(x_1, x_2)$ is negative definite if $\|x\|_2^2 = x_1^2 + x_2^2 < 2$
- Since for any B_ϵ with $0 < \epsilon < \sqrt{2}$ the hypotheses of Lyapunov's theorem are satisfied, $x = 0$ is an asymptotically stable equilibrium
- Any B_ϵ with $0 < \epsilon < \sqrt{2}$ is a also a **domain of attraction**

EXAMPLE OF LYAPUNOV'S DIRECT METHOD (CONT'D)

- Cf. Lyapunov's indirect method: the linearization around $x = 0$ is

$$\frac{\partial f(0,0)}{\partial x} = \left[\begin{array}{cc} 3x_1^2 - 3x_2^2 - 2 & -6x_1x_2 \\ 10x_1x_2 & 5x_1^2 + 3x_2^2 - 2 \end{array}\right]\Bigg|_{x=0} = \left[\begin{array}{cc} -2 & 0 \\ 0 & -2 \end{array}\right]$$

  which is an asymptotically stable matrix
- Lyapunov's indirect method tells us that the origin is locally asymptotically stable
- Lyapunov's direct method also tells us that B_ϵ is a domain of attraction for all $0 < \epsilon < \sqrt{2}$

- Consider this other example: $\dot{x} = -x^3$. The origin as an equilibrium. But $\frac{df(0)}{dx} = -3 \cdot 0^2 = 0$, so Lyapunov indirect method is useless.
- Lyapunov's direct method with $V = x^2$ provides $\dot{V} = -2x^4$, and therefore we can conclude that $x = 0$ is (globally) asymptotically stable

CASE OF CONTINUOUS-TIME LINEAR SYSTEMS

- Let us apply Lyapunov's direct method to linear systems $\dot{x} = Ax$ and choose $V(x) = x'Px$, with $P = P' \succ 0$ (P=positive definite and symmetric matrix)
- The derivative $\dot{V}(x) = \dot{x}'Px + x'P\dot{x} = x'(A'P + PA)x$
- $\dot{V}(x)$ is negative definite if and only if the **Lyapunov equation**

$$A'P + PA = -Q$$

is satisfied for some $Q \succ 0$ (for example, $Q = I$)

Theorem

The autonomous linear system $\dot{x} = Ax$ is asymptotically stable $\Leftrightarrow \forall Q \succ 0$ the Lyapunov equation $A'P + PA = -Q$ has one and only one solution $P \succ 0$

MATLAB

```
P=lyap(A',Q)
```

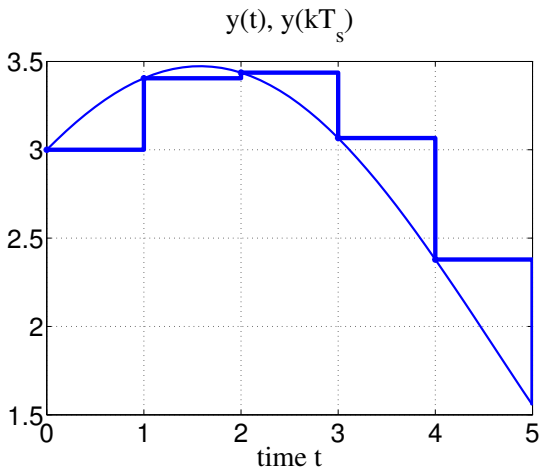
Python

```
import control as ctrl
P=lyap(A.T.copy(),Q)
```

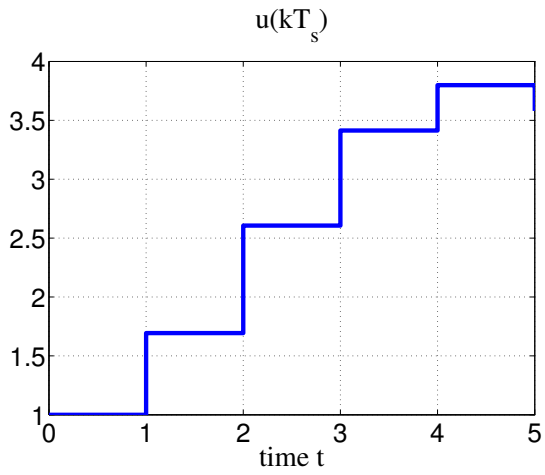
(note transposition of matrix A !)

DISCRETE-TIME SYSTEMS

DISCRETE-TIME MODELS



sampled continuous-time signal



discrete-time signal

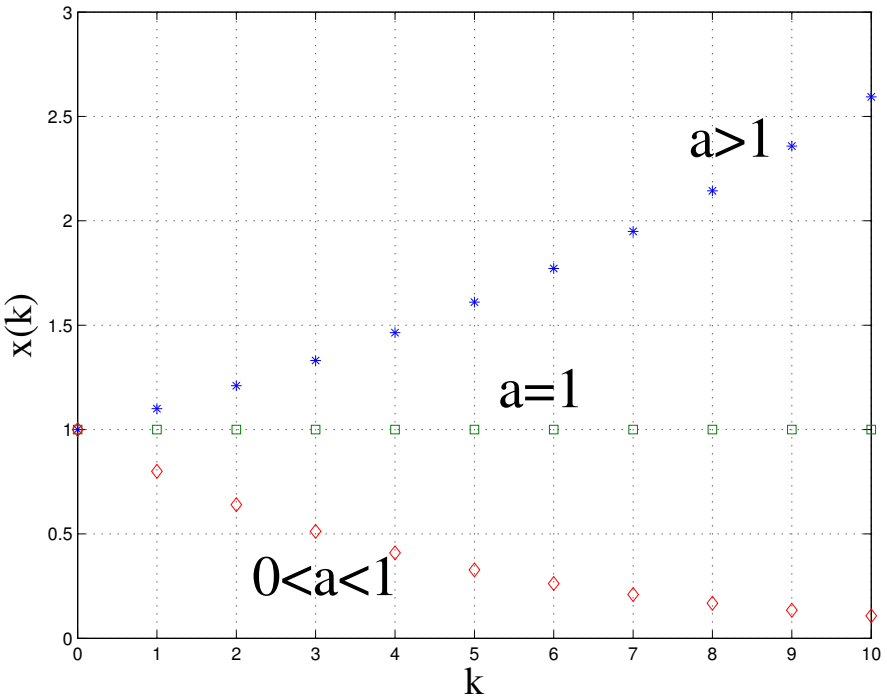
- Discrete-time models describe relationships between **sampled** variables $x(kT_s), u(kT_s), y(kT_s), k = 0, 1, \ldots$
- The value $u(kT_s)$ is kept constant during the **sampling interval** $[kT_s, (k+1)T_s)$
- A discrete-time signal can either represent the **sampling** of a **continuous-time** signal, or be an intrinsically discrete signal
- Discrete-time signals are at the basis of **digital controllers** (as well as of digital filters in signal processing)

DIFFERENCE EQUATION

- Consider the first-order **difference equation** (autonomous system)

$$\begin{cases} x(k+1) &= ax(k) \\ x(0) &= x_0 \end{cases}$$

- The solution is $x(k) = a^k x_0$



LINEAR DISCRETE-TIME SYSTEM

- Consider the set of n first-order linear difference equations forced by the input $u(k) \in \mathbb{R}$

$$\begin{cases} x_1(k+1) &= a_{11}x_1(k) + \ldots + a_{1n}x_n(k) &+b_1u(k) \\ x_2(k+1) &= a_{21}x_1(k) + \ldots + a_{2n}x_n(k) &+b_2u(k) \\ \vdots & \vdots & \vdots \\ x_n(k+1) &= a_{n1}x_1(k) + \ldots + a_{nn}x_n(k) &+b_nu(k) \\ x_1(0) = x_{10}, & \ldots & x_n(0) = x_{n0} \end{cases}$$

- In compact matrix form:

$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) \\ x(0) &= x_0 \end{cases}$$

where $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$.

LINEAR DISCRETE-TIME SYSTEM

- The solution is

$$x(k) = \underbrace{A^k x_0}_{\text{natural response}} + \underbrace{\sum_{i=0}^{k-1} A^i B u(k-1-i)}_{\text{forced response}}$$

- If matrix A is diagonalizable, $A = T\Lambda T^{-1}$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \Rightarrow A^k = T \begin{bmatrix} \lambda_1^k & 0 & \dots & 0 \\ 0 & \lambda_2^k & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^k \end{bmatrix} T^{-1}$$

where $T = [v_1 \dots v_n]$ collects n independent eigenvectors.

MODAL RESPONSE

- Assume input $u(k) = 0, \forall k \geq 0$
- Assume A is diagonalizable, $A = T\Lambda T^{-1}$
- The state trajectory (natural response) is

$$x(k) = A^k x_0 = T\Lambda^k T^{-1} x_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^k v_i$$

where

  - λ_i = eigenvalues of A
  - v_i = eigenvectors of A
  - α_i = coefficients that depend on the initial condition $x(0)$

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = T^{-1} x(0),\ T = [v_1 \dots v_n]$$

- The system modes depend on the eigenvalues of A, as in continuous-time

EXAMPLE - WEALTH OF A BANK ACCOUNT

- k= year counter
- ρ= interest rate
- $x(k)$= wealth at the beginning of year k
- $u(k)$= money saved at the end of year k
- x_0= initial wealth in bank account

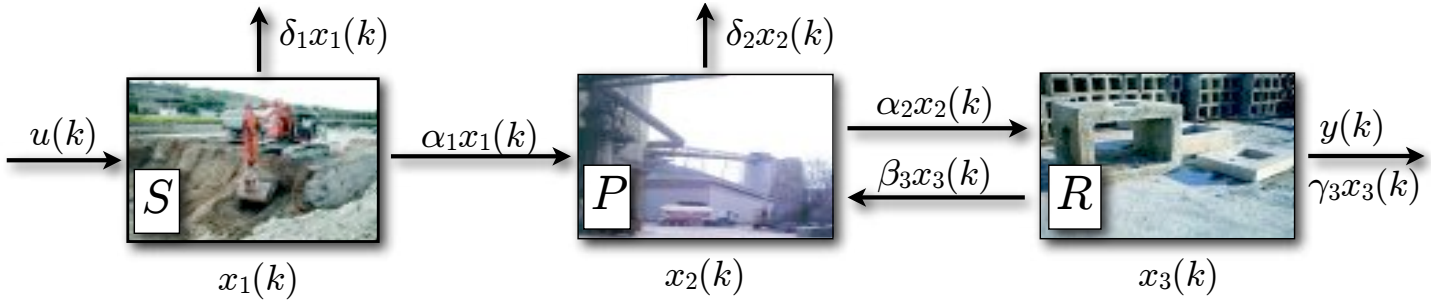
Discrete-time model:

$$\begin{cases} x(k+1) &= (1+\rho)x(k)+u(k) \\ x(0) &= x_0 \end{cases}$$

x_0	10 k€
$u(k)$	5 k€
ρ	10 %

$$x(k) = (1.1)^k \cdot 10 + \frac{1-(1.1)^k}{1-1.1} 5 = 60(1.1)^k - 50$$

EXAMPLE - SUPPLY CHAIN



- Problem statement:
  - At each month k, S purchases the quantity $u(k)$ of raw material
  - A fraction δ_1 of raw material is discarded, a fraction α_1 is shipped to producer P
  - A fraction α_2 of product is sold by P to retailer R, a fraction δ_2 is discarded
  - Retailer R returns a fraction β_3 of defective products every month and sells a fraction γ_3 to customers
- Mathematical model:

$$\begin{cases} x_1(k+1) &= (1-\alpha_1-\delta_1)x_1(k) + u(k) \\ x_2(k+1) &= \alpha_1 x_1(k) + (1-\alpha_2-\delta_2)x_2(k) \\ & \quad +\beta_3 x_3(k) \\ x_3(k+1) &= \alpha_2 x_2(k) + (1-\beta_3-\gamma_3)x_3(k) \\ y(k) &= \gamma_3 x_3(k) \end{cases}$$

k	month counter
$x_1(k)$	raw material in S
$x_2(k)$	products in P
$x_3(k)$	products in R
$u(k)$	raw material purchased by S
$y(k)$	products sold to customers

EXAMPLE - STUDENT POPULATION DYNAMICS

- Problem statement:
  - 3-years course
  - percentage of promoted, repeaters, and dropouts are roughly constant
  - direct enrollment in 2nd and 3rd academic year is not allowed
  - students cannot enroll for more than 3 years

k	Year
$x_i(k)$	Number of students enrolled in year i at year k, $i = 1, 2, 3$
$u(k)$	Number of freshmen at year k
$y(k)$	Number of graduates at year k
α_i	promotion rate during year i, $0 \leq \alpha_i \leq 1$
β_i	failure rate during year i, $0 \leq \beta_i \leq 1$
γ_i	dropout rate during year i, $\gamma_i = 1 - \alpha_i - \beta_i \geq 0$

- 3^{rd}-order linear discrete-time system:

$$\begin{cases} x_1(k+1) &= x_1(k) - \alpha_1 x_1(k) - \gamma_1 x_1(k) + u(k) = \beta_1 x_1(k) + u(k) \\ x_2(k+1) &= \alpha_1 x_1(k) + \beta_2 x_2(k) \\ x_3(k+1) &= \alpha_2 x_2(k) + \beta_3 x_3(k) \\ y(k) &= \alpha_3 x_3(k) \end{cases}$$

EXAMPLE - STUDENT POPULATION DYNAMICS

- In matrix form

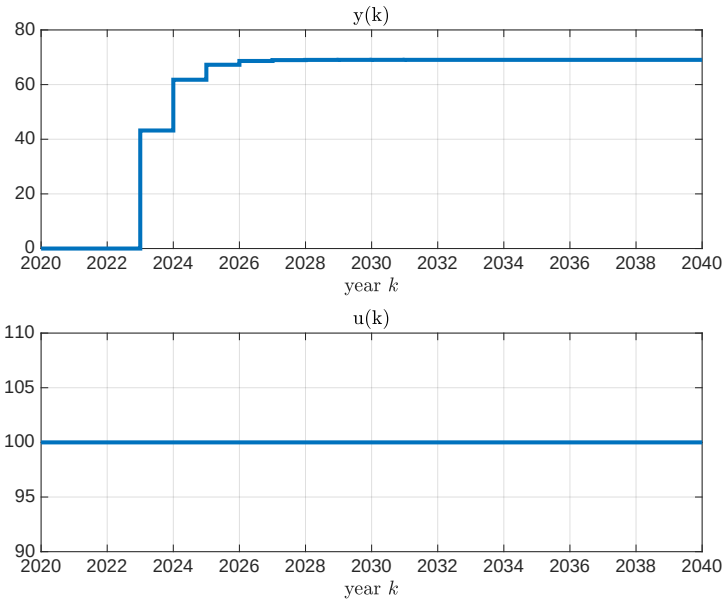
$$\begin{cases} x(k+1) &= \begin{bmatrix} \beta_1 & 0 & 0 \\ \alpha_1 & \beta_2 & 0 \\ 0 & \alpha_2 & \beta_3 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(k) \\ \\ y(k) &= \begin{bmatrix} 0 & 0 & \alpha_3 \end{bmatrix} x(k) \end{cases}$$

- Simulation

$\alpha_1 = .60$	$\beta_1 = .20$
$\alpha_2 = .80$	$\beta_2 = .15$
$\alpha_3 = .90$	$\beta_3 = .08$

$$u(k) \equiv 100, k = 2020, \ldots$$

$$\lim_{k \to \infty} y(k) \approx 69.0537$$



n^{th}-ORDER DIFFERENCE EQUATION

- Consider the n^{th}-**order difference equation** forced by u

$$a_n y(k-n) + a_{n-1} y(k-n+1) + \cdots + a_1 y(k-1) + y(k) = b_n u(k-n) + \cdots + b_1 u(k-1) + b_0 u(k)$$

- Equivalent linear discrete-time system in **canonical state matrix form**

$$\begin{cases} x(k+1) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(k) \\ y(k) = \begin{bmatrix} (b_n - b_0 a_n) & \dots & (b_1 - b_0 a_1) \end{bmatrix} x(k) + b_0 u(k) \end{cases}$$

- There are infinitely many **state-space realizations**

MATLAB
tf2ss

Python
ctrl.tf2ss

- n^{th}-order difference equations are very useful for digital filters, digital controllers, and to reconstruct models from data (**system identification**)

SOME STATE-SPACE REALIZATION METHODS

- The following state-space realization is called **controllable canonical form**)

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \dots & -a_1 \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} b_n & b_{n-1} & \dots & b_1 \end{bmatrix}, \; D = 0 \qquad (b_0 = 0)$$

MATLAB

```
sysc=canon(ss(A,B,C,D),'companion')
```

Python

```
sysc,T=ctrl.canonical_form(
  ctrl.ss(A,B,C,D), form='reachable')
```

- The following state-space realization is called **observable canonical form**

$$A = \begin{bmatrix} -a_1 & 1 & 0 & \dots & 0 \\ -a_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ -a_{n-1} & 0 & 0 & \dots & 1 \\ -a_n & 0 & 0 & \dots & 0 \end{bmatrix}, \; B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}, \; D = 0 \qquad (b_0 = 0)$$

MATLAB

```
sys=canon(ss(A',C',B',D),'companion')
syso=ss(sys.A',sys.C',sys.B',D)
```

Python

```
syso,T=ctrl.canonical_form(
  ctrl.ss(A,B,C,D), form='observable')
```

- We will see later that (A, B) in controllable canonical form is a **reachable** pair, (A, C) in observable canonical form is an **observable** pair

MAPPING PAST I/O PAIRS TO STATE VECTOR

- The observable canonical form of the n^{th}-order difference equation

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=1}^{n} b_i u(k-i) \qquad (b_0 = 0)$$

corresponds to the following definition of the state vector $x(k)$:

$$\left\{\begin{array}{rcl} x_1(k) & = & y(k) \\ x_2(k) & = & -\sum_{i=2}^{n} a_i y(k+1-i) + \sum_{i=2}^{n} b_i u(k+1-i) \\ \vdots & & \vdots \\ x_j(k) & = & -\sum_{i=j}^{n} a_i y(k+j-1-i) + \sum_{i=j}^{n} b_i u(k+j-1-i) \\ \vdots & & \vdots \\ x_{n-1}(k) & = & -a_{n-1} y(k-1) - a_n y(k-2) + b_{n-1} u(k-1) + b_n u(k-2) \\ x_n(k) & = & -a_n y(k-1) + b_n u(k-1) \end{array}\right.$$

- This is easy to verify by inspection, just compute $x(k+1)$ and check

DISCRETE-TIME LINEAR SYSTEM

$$\begin{cases} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k)+Du(k) \\ x(0) &= x_0 \end{cases}$$

- From a given initial condition $x(0)$ and input sequence $\{u(k)\}_{k=0}^{\infty}$ one can predict the entire sequence of states $x(k)$ and outputs $y(k), \forall k \in \mathbb{N}$
- The state $x(0)$ summarizes all the past history of the system
- The dimension n of the state $x(k) \in \mathbb{R}^n$ is called the **order** of the system
- The system is called **proper** (or **strictly causal**) if $D = 0$
- General multivariable case:

$$\begin{aligned} x(k) &\in \mathbb{R}^n \\ u(k) &\in \mathbb{R}^m \\ y(k) &\in \mathbb{R}^p \end{aligned} \qquad \begin{aligned} A &\in \mathbb{R}^{n\times n} \\ B &\in \mathbb{R}^{n\times m} \\ C &\in \mathbb{R}^{p\times n} \\ D &\in \mathbb{R}^{p\times m} \end{aligned}$$

EQUILIBRIUM

- Consider the discrete-time nonlinear system

$$\begin{cases} x(k+1) &= f(x(k),u(k)) \\ y(k) &= g(x(k),u(k)) \end{cases}$$

Definition

A state $x_r \in \mathbb{R}^n$ and an input $u_r \in \mathbb{R}^m$ are an **equilibrium pair** if for initial condition $x(0) = x_r$ and constant input $u(k) \equiv u_r, \forall k \in \mathbb{N}$, the state remains constant: $x(k) \equiv x_r, \forall k \in \mathbb{N}$.

- Equivalent definition: (x_r, u_r) is an equilibrium pair if $f(x_r, u_r) = x_r$
- x_r is called **equilibrium state**, u_r **equilibrium input**
- The definition generalizes to time-varying discrete-time nonlinear systems

STABILITY

- Consider the nonlinear system

$$\begin{cases} x(k+1) &= f(x(k), u_r) \\ y(k) &= g(x(k), u_r) \end{cases}$$

and let x_r an equilibrium state, $f(x_r, u_r) = x_r$

Definition

The equilibrium state x_r is **stable** if for each initial conditions $x(0)$ "close enough" to x_r, the corresponding trajectory $x(k)$ remains close to x_r for all $k \in \mathbb{N}$

Analytic definition: $\forall \epsilon > 0 \; \exists \delta > 0 : \|x(0) - x_r\| < \delta \Rightarrow \|x(k) - x_r\| < \epsilon, \forall k \in \mathbb{N}.$

- The equilibrium point x_r is called **asymptotically stable** if it is stable and $x(k) \to x_r$ for $k \to \infty$

- Otherwise, the equilibrium point x_r is called **unstable**

STABILITY OF FIRST-ORDER LINEAR SYSTEMS

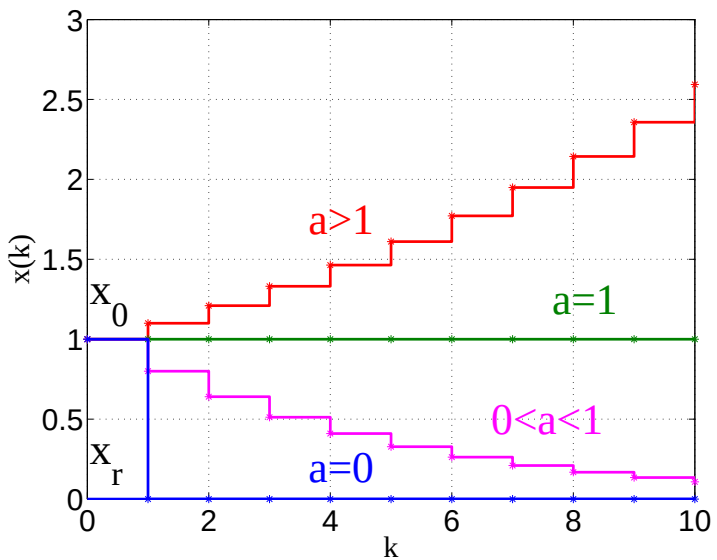
- Consider the first-order linear system

$$x(k+1) = ax(k) + bu(k)$$

- $x_r = 0, u_r = 0$ is an equilibrium pair
- For $u(k) \equiv 0, \forall k = 0, 1, \ldots$, the solution is

$$x(k) = a^k x_0$$

- The origin $x_r = 0$ is
  - unstable if $|a| > 1$
  - stable if $|a| \leq 1$
  - asymptotically stable if $|a| < 1$



STABILITY OF DISCRETE-TIME LINEAR SYSTEMS

The natural response of $x(k+1) = Ax(k) + Bu(k)$ is $x(k) = A^k x_0$, so stability only depend on A. We therefore talk about **system stability**

Theorem

Let $\lambda_1, \ldots, \lambda_m, m \leq n$ be the eigenvalues of $A \in \mathbb{R}^{n \times n}$.

The system $x(k+1) = Ax(k) + Bu(k)$ is

- asymptotically stable iff $|\lambda_i| < 1, \forall i = 1, \ldots, m$
- (marginally) stable if $|\lambda_i| \leq 1, \forall i = 1, \ldots, m$, and the eigenvalues with unit modulus have equal algebraic and geometric multiplicity [a]
- unstable otherwise(in particular, if $\exists\, i$ such that $|\lambda_i| > 1$)

[a] Algebraic multiplicity of λ_i = number of coincident roots λ_i of $\det(\lambda I - A)$. Geometric multiplicity of λ_i = number of linearly independent eigenvectors v_i, $Av_i = \lambda_i v_i$

The stability properties of a discrete-time linear system only depend on the **modulus** of the eigenvalues of matrix A

STABILITY OF DISCRETE-TIME LINEAR SYSTEMS

Proof:

- The natural response is $x(k) = A^k x_0$
- If matrix A is diagonalizable[2] , $A = T\Lambda T^{-1}$,

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \Rightarrow A^k = T \begin{bmatrix} \lambda_1^k & 0 & \dots & 0 \\ 0 & \lambda_2^k & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^k \end{bmatrix} T^{-1}$$

- Take any eigenvalue $\lambda = \rho e^{j\theta}$:

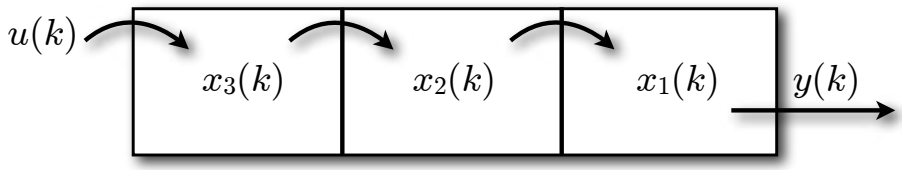
$$|\lambda^k| = \rho^k |e^{jk\theta}| = \rho^k$$

- A is always diagonalizable if algebraic multiplicity - geometric multiplicity $\square$
- Lyapunov theorems also exist for nonlinear discrete-time systems (LaSalle, 1997)

[2] If A is not diagonalizable, it can be transformed to Jordan form. In this case the natural response $x(t)$ contains modes $k^j \lambda^k, j = 0, 1, \dots,$ alg. multiplicity $-$ geom. multiplicity

ZERO EIGENVALUES

- Modes corresponding to λ_i=0 go to zero in finite-time
- This has no continuous-time counterpart, where instead all converging modes tend to zero in infinite time ($e^{\lambda_i t}$)
- Example: dynamics of a buffer



$$\begin{cases} x_1(k+1) = x_2(k) \\ x_2(k+1) = x_3(k) \\ x_3(k+1) = u(k) \\ y(k) = x_1(k) \end{cases} \Rightarrow \begin{cases} x(k+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k) \\ y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x(k) \end{cases}$$

- Natural response: $A^3 x(0) = 0$ for all $x(0) \in \mathbb{R}^3$
- For $u(k) \equiv 0$, the buffer deploys after at most 3 steps !

EXACT SAMPLING

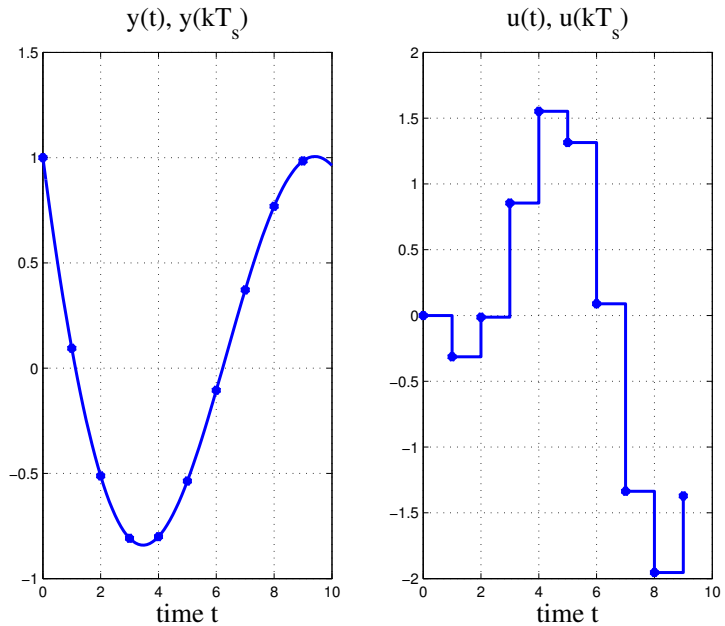
- Consider the continuous-time system

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \\ x(0) &= x_0 \end{cases}$$

- We want to characterize the value of $x(t), y(t)$ at the time instants $t = 0, T_s, 2T_s, \ldots, kT_s, \ldots$, **under the assumption that the input $u(t)$ is constant during each sampling interval (zero-order hold, ZOH)**

$$u(t) = \bar{u}(k),\ kT_s \le t < (k+1)T_s$$

- $\bar{x}(k) \triangleq x(kT_s)$ and $\bar{y}(k) \triangleq y(kT_s)$ are the state and the output samples at the k^{th} sampling instant, respectively



EXACT SAMPLING

- Use Lagrange formula to get the response of the continuous-time system between $t_0 = kT_s$ and $t = (k+1)T_s$ from $x(t_0) = x(kT_s)$:

$$\begin{aligned} x(t) &= e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\sigma)}Bu(\sigma)d\sigma \\ &= e^{A((k+1)T_s - kT_s)}x(kT_s) + \int_{kT_s}^{(k+1)T_s} e^{A((k+1)T_s-\sigma)}Bu(\sigma)d\sigma \end{aligned}$$

- Since the input $u(t)$ is piecewise constant, $u(\sigma) \equiv \bar{u}(k)$, $kT_s \le \sigma < (k+1)T_s$. By setting $\tau = \sigma - kT_s$ we get

$$x((k+1)T_s) = e^{AT_s}x(kT_s) + \left(\int_0^{T_s} e^{A(T_s-\tau)}d\tau\right)Bu(kT_s)$$

and hence

$$\bar{x}(k+1) = e^{AT_s}\bar{x}(k) + \left(\int_0^{T_s} e^{A(T_s-\tau)}d\tau\right)B\bar{u}(k)$$

which is a linear difference relation between $\bar{x}(k)$ and $\bar{u}(k)$!

EXACT SAMPLING

- The discrete-time system

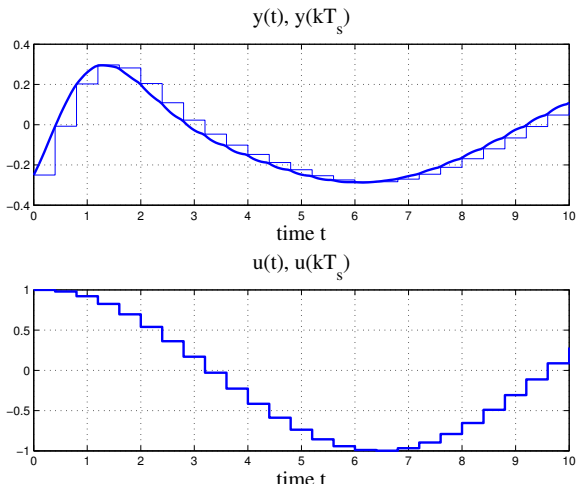
$$\begin{cases} \bar{x}(k+1) &= \bar{A}\bar{x}(k) + \bar{B}\bar{u}(k) \\ \bar{y}(k) &= \bar{C}\bar{x}(k) + \bar{D}\bar{u}(k) \end{cases}$$

depends on the original continuous-time system through the relations

$$\bar{A} \triangleq e^{AT_s}, \quad \bar{B} \triangleq \left(\int_0^{T_s} e^{A(T_s-\tau)} d\tau \right) B, \quad \bar{C} \triangleq C, \quad \bar{D} \triangleq D$$

(if A is invertible then $\bar{B} = (\bar{A} - I)A^{-1}B$)

- If $u(t)$ is piecewise constant, $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ provides the exact evolution of state and output samples at discrete times kT_s



MATLAB

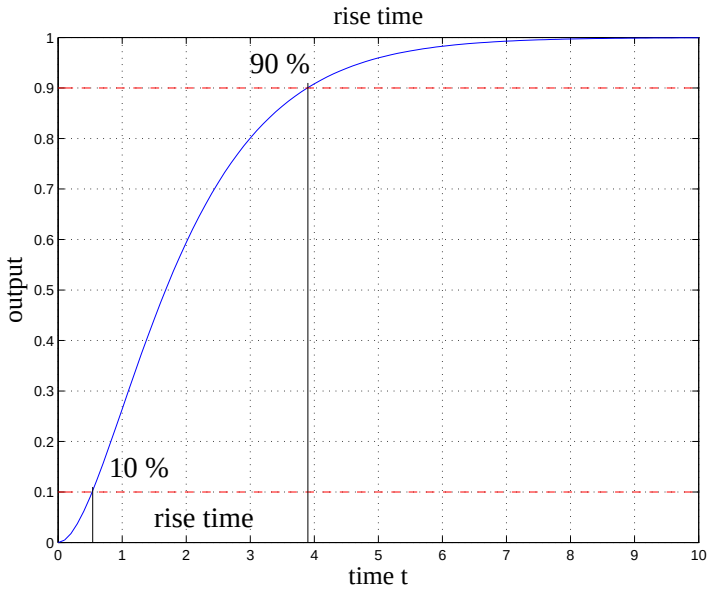
```
sys=ss(A,B,C,D);
sysd=c2d(sys,Ts);
[Ab,Bb,Cb,Db]=ssdata(sysd);
```

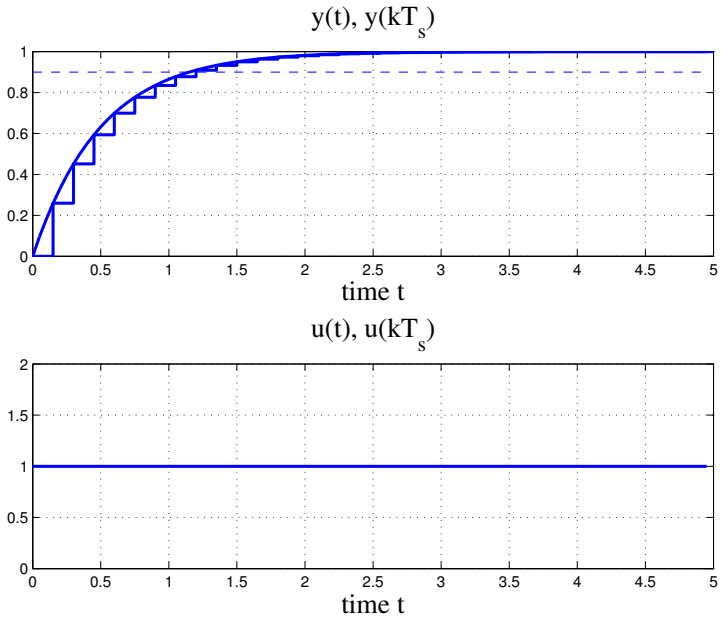
Python

```
sys=ctrl.ss(A,B,C,D)
sysd=ctrl.c2d(sys,Ts)
Ab,Bb,Cb,Db=ctrl.ssdata(sysd)
```

CHOICE OF SAMPLING TIME

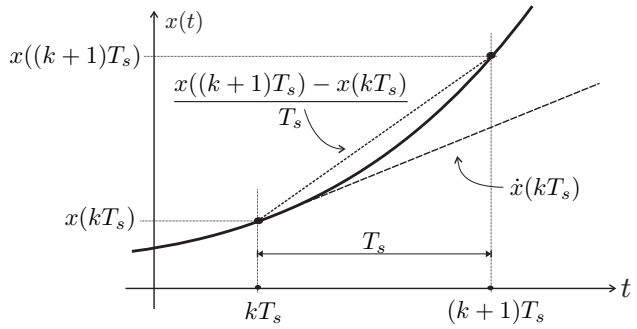
Rule of thumb: $T_s \approx \frac{1}{10}$ of **rise time** = time to move from 10% to 90% of the steady-state value, for input $u(t) \equiv 1, x(0) = 0$





EULER'S FORWARD METHOD

$$\dot{x}(kT_s) \approx \frac{x((k+1)T_s) - x(kT_s)}{T_s}$$



Leonhard Paul Euler
(1707-1783)

- For nonlinear systems $\dot{x}(t) = f(x(t), u(t))$:

$$\bar{x}(k+1) = \bar{x}(k) + T_s f(\bar{x}(k), \bar{u}(k))$$

- For linear systems $\dot{x}(t) = Ax(t) + Bu(t)$:

$$x((k+1)T_s) = (I + T_s A)x(kT_s) + T_s B u(kT_s)$$

$$\bar{A} \triangleq I + T_s A, \quad \bar{B} \triangleq T_s B, \quad \bar{C} \triangleq C, \quad \bar{D} \triangleq D$$

- $e^{T_s A} = I + T_s A + \ldots + \frac{T_s^n A^n}{n!} + \ldots$ Euler's method $\approx$ exact sampling for $T_s \to 0$

EIGENVALUES MAPPING

- Let λ_i = eigenvalues of matrix A (continuous-time system), $i = 1, \ldots, n$.
  Assume A diagonalizable, $A = T\Lambda T^{-1}$

- The eigenvalues of $e^{T_s A} = T e^{T_s \Lambda} T^{-1}$ are $e^{T_s \lambda_i}$
  $\Rightarrow \Re\lambda_i < 0 \rightarrow |e^{T_s \lambda_i}| < 1$

- The eigenvalues of $I + T_s A = T(I + T_s \Lambda)T^{-1}$ are $1 + T_s \lambda_i$
  $\Rightarrow \Re\lambda_i < 0 \not\rightarrow |1 + T_s \lambda_i| < 1$!

Euler's forward method can make an asymptotically stable continuous-time system unstable if T_s is not small enough!

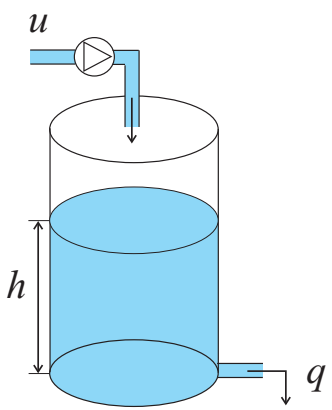
EXAMPLE - HYDRAULIC SYSTEM

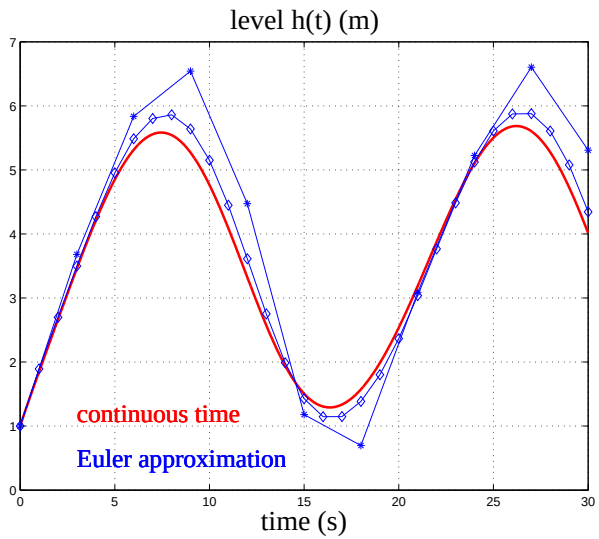
Continuous-time model

$$\begin{cases} \frac{d}{dt}h(t) &= -\frac{a\sqrt{2g}}{A}\sqrt{h(t)} + \frac{1}{A}u(t) \\ q(t) &= a\sqrt{2g}\sqrt{h(t)} \end{cases}$$

Discrete-time model

$$\begin{cases} \bar{h}(k+1) &= \bar{h}(k) - \frac{T_s a\sqrt{2g}}{A}\sqrt{\bar{h}(k)} + \frac{T_s}{A}\bar{u}(k) \\ \bar{q}(k) &= a\sqrt{2g}\sqrt{\bar{h}(k)} \end{cases}$$ (Torricelli's law)





N-STEPS EULER METHOD

- We can obtain the matrices A, B of the discrete-time linearized model while integrating the nonlinear continuous-time dynamic equations $\dot{x} = f(x, u)$

- **N-steps explicit forward Euler method**: given $\bar{x}(k), \bar{u}(k)$, execute the following steps

  1. $x = \bar{x}(k), \bar{A} = I, \bar{B} = 0$
  2. for n=1,2,...,N do
     - $\bar{A} \leftarrow (I + \frac{T_s}{N} \frac{\partial f}{\partial x}(x, \bar{u}(k))\bar{A}$
     - $\bar{B} \leftarrow (I + \frac{T_s}{N} \frac{\partial f}{\partial x}(x, \bar{u}(k))\bar{B} + \frac{T_s}{N} \frac{\partial f}{\partial u}(x, \bar{u}(k))$
     - $x \leftarrow x + \frac{T_s}{N} f(x, \bar{u}(k))$
  3. end
  4. return $\bar{x}(k+1) \approx x$ and matrices $\bar{A}, \bar{B}$ such that $\bar{x}(k+1) \approx A\bar{x}(k) + B\bar{u}(k)$.

- Property: the difference between the state $\bar{x}(k+1)$ and its approximation x computed by the above iterations satisfies $\|\bar{x}(k+1) - x)\| = O\left(\frac{T_s}{N}\right)$

- Explicit forward Runge-Kutta 4 method also available

TUSTIN'S DISCRETIZATION METHOD

- Assume $u(t)$ constant within the sampling interval. Given the linear system $\dot{x}(t) = Ax(t) + Bu(t)$, apply the trapezoidal rule to approximate the integral

$$\bar{x}(k+1) - \bar{x}(k) = \int_{kT_s}^{(k+1)T_s} \dot{x}(t)dt = \int_{kT_s}^{(k+1)T_s} (Ax(t) + Bu(t))dt$$

$$\approx \frac{T_s}{2}\left(A\bar{x}(k) + B\bar{u}(k) + A\bar{x}(k+1) + B\bar{u}(k)\right) \text{ (trapezoidal rule)}$$

and therefore

$$(I - \frac{T_s}{2}A)\bar{x}(k+1) = (I + \frac{T_s}{2})\bar{x}(k) + T_sB\bar{u}(k)$$

$$\bar{x}(k+1) = \left(I - \frac{T_s}{2}A\right)^{-1}\left(I + \frac{T_s}{2}A\right)\bar{x}(k) + \left(I - \frac{T_s}{2}A\right)^{-1}T_sB\bar{u}(k)$$

- Advantage: simpler to compute than exponential matrix, without too much loss of approximation quality

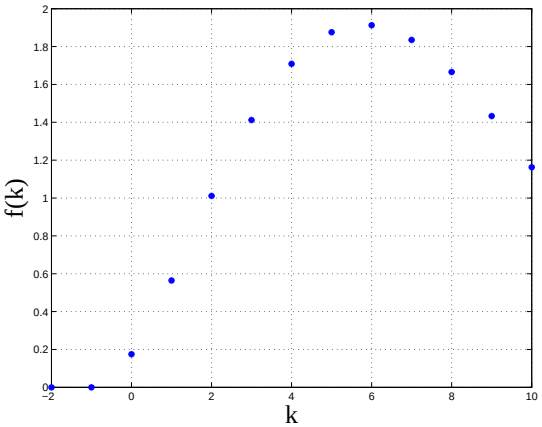
Z-TRANSFORM

Consider a function $f(k)$, $f : \mathbb{Z} \to \mathbb{R}$, $f(k) = 0$ for all $k < 0$

Definition

The unilateral **Z-transform** of $f(k)$ is the function of the complex variable $z \in \mathbb{C}$ defined by

$$F(z) = \sum_{k=0}^{\infty} f(k) z^{-k}$$



Witold Hurewicz
(1904-1956)

Once $F(z)$ is computed using the series, it's extended to all $z \in \mathbb{C}$ for which $F(z)$ makes sense

Z-transforms convert difference equations into algebraic equations.

EXAMPLES OF Z-TRANSFORMS

- **Discrete impulse**

$$f(k) = \delta(k) \triangleq \begin{cases} 0 & \text{if } k \neq 0 \\ 1 & \text{if } k = 0 \end{cases} \quad \Rightarrow \quad \mathcal{Z}[\delta] = F(z) = 1$$

- **Discrete step**

$$f(k) = \mathbb{1}(k) \triangleq \begin{cases} 0 & \text{if } k < 0 \\ 1 & \text{if } k \geq 0 \end{cases} \quad \Rightarrow \quad \mathcal{Z}[\mathbb{1}] = F(z) = \frac{z}{z-1}$$

- **Geometric sequence**

$$f(k) = a^k \, \mathbb{1}(k) \quad \Rightarrow \quad \mathcal{Z}[f] = F(z) = \frac{z}{z-a}$$

PROPERTIES OF Z-TRANSFORMS

- **Linearity**

$$\mathcal{Z}[\alpha_1 f_1(k) + \alpha_2 f_2(k)] = \alpha_1 \mathcal{Z}[f_1(k)] + \alpha_2 \mathcal{Z}[f_2(k)]$$

Example: $f(k) = 3\delta(k) - \frac{5}{2^k}\,\mathbb{1}(t) \Rightarrow \mathcal{Z}[f] = 3 - \frac{5z}{z-\frac{1}{2}}$

- **Forward shift**[3]

$$\mathcal{Z}[f(k+1)\,\mathbb{1}(k)] = z\mathcal{Z}[f] - zf(0)$$

Example: $f(k) = a^{k+1}\,\mathbb{1}(k) \Rightarrow \mathcal{Z}[f] = z\frac{z}{z-a} - z = \frac{az}{z-a}$

[3] z is also called **forward shift operator**

PROPERTIES OF Z-TRANSFORMS

- **Backward shift** or **unit delay** [4]

$$\mathcal{Z}[f(k-1)\,\mathbb{1}(k)] = z^{-1}\mathcal{Z}[f]$$

Example: $f(k) = \mathbb{1}(k-1) \Rightarrow \mathcal{Z}[f] = \frac{z}{z(z-1)}$

- **Multiplication by k**

$$\mathcal{Z}[kf(k)] = -z\frac{d}{dz}\mathcal{Z}[f]$$

Example: $f(k) = k\,\mathbb{1}(k) \Rightarrow \mathcal{Z}[f] = \frac{z}{(z-1)^2}$

[4] z^{-1} is also called **backward shift operator**

DISCRETE-TIME TRANSFER FUNCTION

Apply forward-shift & linearity rules to $x(k+1) = Ax(k) + Bu(k)$, and linearity to $y(k) = Cx(k) + Du(k)$:

$$\begin{aligned} X(z) &= z(zI-A)^{-1}x_0 + (zI-A)^{-1}BU(z) \\ Y(z) &= \underbrace{zC(zI-A)^{-1}x_0}_{\text{z-transform of natural response}} + \underbrace{(C(zI-A)^{-1}B+D)U(z)}_{\text{z-transform of forced response}} \end{aligned}$$

Definition

The transfer function of the discrete-time linear system (A, B, C, D) is

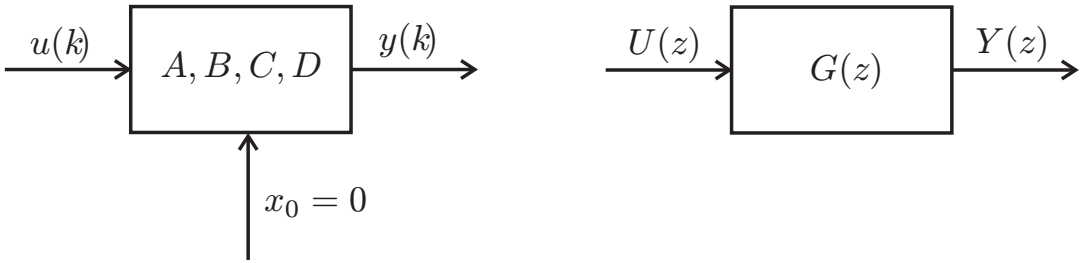
$$G(z) = C(zI-A)^{-1}B + D$$

that is the ratio between the Z-transform $Y(z)$ of the output and the Z-transform $U(z)$ of the input signals for the initial state $x_0 = 0$

MATLAB
sys=ss(A,B,C,D,Ts); G=tf(sys)

Python
sys=ctrl.ss(A,B,C,D,Ts) G=ctrl.tf(sys)

DISCRETE-TIME TRANSFER FUNCTION



Example: The linear system

$$\begin{cases} x(k+1) &= \begin{bmatrix} 0.5 & 1 \\ 0 & -0.5 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & -1 \end{bmatrix} x(k) \end{cases}$$

with sampling time $T_s = 0.1$ s has the transfer function

$$G(z) = \frac{-z + 1.5}{z^2 - 0.25}$$

The transfer function does not depend on the input $u(k)$, it is only a property of the linear system.

MATLAB

```
Ts=0.1;

A=[0.5 1;0 -0.5];
B=[0;1];
C=[1 -1];
sys=ss(A,B,C,0,Ts);
G=tf(sys)

Transfer function:
-z + 1.5
————————
z^2 - 0.25
```

Python

```
import numpy as np
Ts=0.1
A=np.array([[0.5, 1],[0, -0.5]])
B=np.array([[0],[1]])
C=np.array([[1, -1]])
sys=ctrl.ss(A,B,C,0,Ts)
G=ctrl.tf(sys)

Transfer function:
-z + 1.5
————————
z^2 - 0.25
```

DIFFERENCE EQUATIONS

- Consider the n^{th}-order difference equation forced by u

$$\begin{aligned} & a_n y(k-n) + a_{n-1} y(k-n+1) + \cdots + a_1 y(k-1) + y(k) \\ & = b_n u(k-n) + \cdots + b_1 u(k-1) \end{aligned}$$

- For zero initial conditions we get the transfer function

$$\begin{aligned} G(z) &= \frac{b_n z^{-n} + b_{n-1} z^{-n+1} + \cdots + b_1 z^{-1}}{a_n z^{-n} + a_{n-1} z^{-n+1} + \cdots + a_1 z^{-1} + 1} \\ &= \frac{b_1 z^{n-1} + \cdots + b_{n-1} z + b_n}{z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n} \end{aligned}$$

IMPULSE RESPONSE

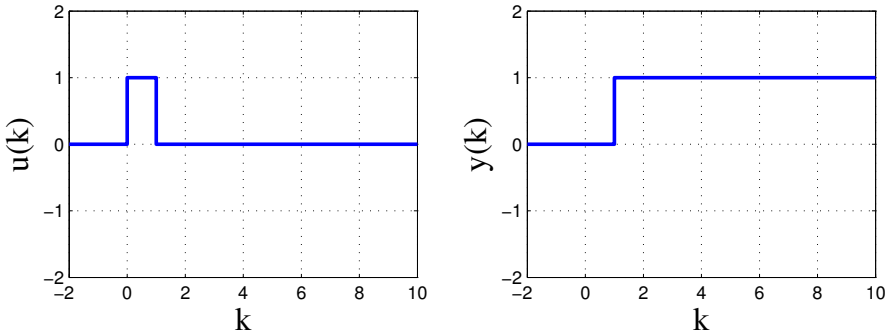
- Consider the impulsive input $u(k) = \delta(k), U(z) = 1$. The corresponding output $y(k)$ is called **impulse response**

- The Z-transform of $y(k)$ is $Y(z) = G(z) \cdot 1 = G(z)$

- Therefore the impulse response coincides with the **inverse Z-transform** $g(k)$ of the transfer function $G(z)$

Example (integrator:)

$$\begin{aligned} u(k) &= \delta(k) \\ y(k) &= \mathcal{Z}^{-1}\left[\frac{1}{z-1}\right] = \mathbb{1}(k-1) \end{aligned}$$



POLES, EIGENVALUES, MODES

- Linear discrete-time system

$$\begin{cases} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k)+Du(k) \end{cases} \qquad G(z) = C(zI-A)^{-1}B+D \triangleq \frac{N_G(z)}{D_G(z)}$$
$$x(0) = 0$$

- Use the adjugate matrix to represent the inverse of $zI - A$

$$C(zI-A)^{-1}B + D = \frac{C\,\mathrm{Adj}(zI-A)B}{\det(zI-A)} + D$$

- The denominator $D_G(z) = \det(zI - A)$!

> The poles of $G(z)$ coincide with the eigenvalues of A

- Well, not always ... There might be a zero/pole cancellation (we will see later)

STEADY-STATE SOLUTION AND DC GAIN

- Let A asymptotically stable ($|\lambda_i| < 1$). The natural response vanishes asymptotically
- Assume constant $u(k) \equiv u_r, \forall k \in \mathbb{N}$. What is the asymptotic value $x_r = \lim_{k\to\infty} x(k)$?

  Impose $x_r(k+1) = x_r(k) = Ax_r + Bu_r$ and get $x_r = (I-A)^{-1}Bu_r$

  The corresponding **steady-state** output $y_r = Cx_r + Du_r$ is

$$y_r = \underbrace{(C(I-A)^{-1}B + D)}_{\text{DC gain}} u_r$$

- Cf. **final value theorem** in complex analysis:

$$\begin{aligned} y_r &= \lim_{k\to+\infty} y(k) = \lim_{z\to 1}(z-1)Y(z) = \lim_{z\to 1}(z-1)G(z)U(z) \\ &= \lim_{z\to 1}(z-1)G(z)\frac{u_r z}{z-1} = G(1)u_r = (C(I-A)^{-1}B + D)u_r \end{aligned}$$

- $G(1)$ is called the **DC gain** of the system

EXAMPLE - STUDENT POPULATION DYNAMICS

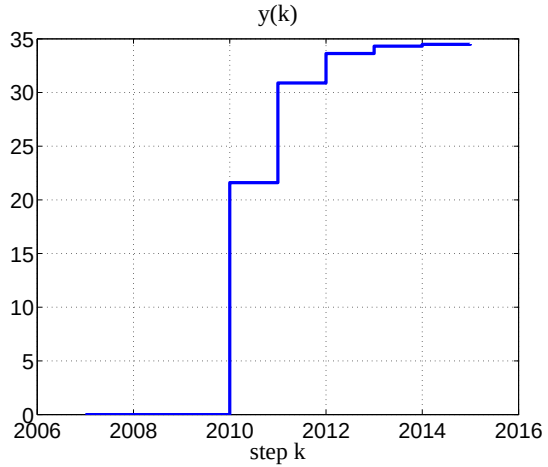
- Recall student population dynamics

$$\begin{cases} x(k+1) &= \begin{bmatrix} .2 & 0 & 0 \\ .6 & .15 & 0 \\ 0 & .8 & .08 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 0 & 0 & .9 \end{bmatrix} x(k) \end{cases}$$

- DC gain:

$$\begin{bmatrix} 0 & 0 & .9 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} .2 & 0 & 0 \\ .6 & .15 & 0 \\ 0 & .8 & .08 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \approx 0.69$$

- Transfer function: $G(z) = \frac{0.432}{z^3 - 0.43z^2 + 0.058z - 0.0024}, G(1) \approx 0.69$



MATLAB

```
A=[b1 0 0; a1 b2 0; 0 a2 b3];
B=[1;0;0];
C=[0 0 a3];
D=[0];
sys=ss(A,B,C,D,1);
dcgain(sys)

  0.6905
```

Python

```
A=[[b1, 0, 0],[a1, b2, 0],[0, a2, b3]]
B=[[1],[0],[0]]
C=[0, 0, a3]
D=[0]
sys=ctrl.ss(A,B,C,D,1)
ctrl.dcgain(sys)

   0.6905
```

- For $u(k) \equiv 50$ students enrolled steadily, $y(k) \to 0.6905 \cdot 50 \approx 34.5$ graduates

CLOSED-LOOP CONTROL

PROPORTIONAL INTEGRAL DERIVATIVE (PID) CONTROLLERS

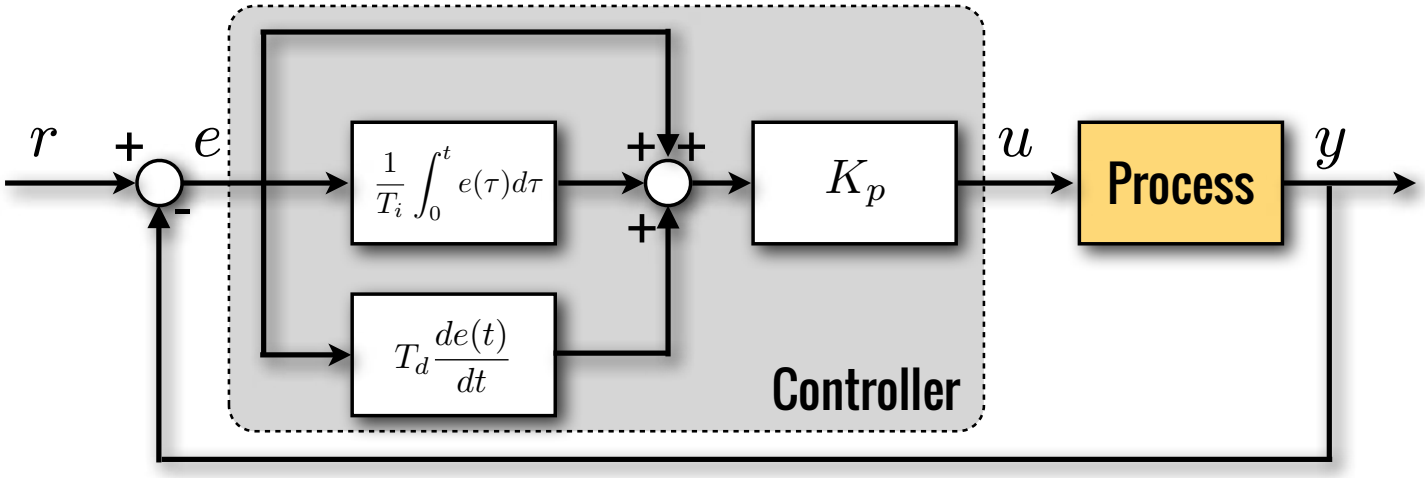
- **PID (proportional integrative derivative) controllers** are the most used controllers in industrial automation since the '30s

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right]$$

where $e(t) = r(t) - y(t)$ is the tracking error

- Initially constructed by analog electronic components, today they are implemented digitally
  - ad hoc digital devices
  - just few lines of C code included in the control unit

PID PARAMETERS



- K_p is the **controller gain**, determining the "aggressiveness" of the controller
- T_i is the **reset time**, determining the weight of the integral action. The integral action guarantees that in steady-state $y(t) = r(t)$
- T_d is the **derivative time**. The term $e(t) + T_d\frac{de(t)}{dt}$ provides a "prediction" of the tracking error at time $t + T_d$
- We call the controller P, PD, PI, or PID depending on the feedback terms included in the control law

STRUCTURE OF PID CONTROLLER

- In practice one implements the following version of the PID controller

$$u(t) = K_p \Big[\underbrace{br(t) - y(t)}_{\text{proportional action}} + \underbrace{\frac{1}{T_i}\int_0^t (r(\tau) - y(\tau))d\tau}_{\text{integral action}} + \underbrace{d(t)}_{\text{derivative action}} \Big]$$

$$d(t) + \frac{T_d}{N}\dot{d}(t) = -T_d\dot{y}(t)$$

- the reference signal $r(t)$ is not included in the derivative term ($r(t)$ may have abrupt changes)
- the proportional action $K_p(br(t) - y(t)$ only uses a fraction $b \leq 1$ of the reference signal $r(t)$
- the derivative term $d(t)$ is a filtered version of $\dot{y}(t)$

DIGITAL IMPLEMENTATION OF PID CONTROLLER

- In digital (=discrete-time) form with sampling time T_s, the PID controller takes the following form

$$u(k) = P(k) + I(k) + D(k)$$

$$P(k) = K_p(br(k) - y(k))$$

$$I(k+1) = I(k) + \frac{K_p T_s}{T_i}(r(k) - y(k))$$ forward differences

$$D(k) = \frac{T_d}{T_d + NT_s} D(k-1) - \frac{K_p T_d N}{T_d + NT_s}(y(k) - y(k-1))$$

backward differences

PID CONTROLLER: PROS AND CONS

- Very simple to implement, only 3 parameters to calibrate
- It only requires the measurement of the output signal $y(t)$
- The control law does not exploit the knowledge of the model of the process
- Achievable closed-loop performance is limited

STATE-FEEDBACK CONTROL

REACHABILITY ANALYSIS

- Consider the linear discrete-time system

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

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and initial condition $x(0) = x_0 \in \mathbb{R}^n$

- The solution is $x(k) = A^k x_0 + \sum_{j=0}^{k-1} A^j B u(k-1-j)$

Definition

The system $x(k+1) = Ax(k) + Bu(k)$ is **(completely) reachable** if $\forall x_1, x_2 \in \mathbb{R}^n$ there exist $k \in \mathbb{N}$ and $u(0), u(1), \ldots, u(k-1) \in \mathbb{R}^m$ such that

$$x_2 = A^k x_1 + \sum_{j=0}^{k-1} A^j B u(k-1-j)$$

- In simple words: a system is completely reachable if from any state x_1 we can reach any state x_2 at some time k, by applying a suitable input sequence

REACHABILITY

- Determine a sequence of n inputs transferring the state vector from x_1 to x_2 after n steps

$$\underbrace{x_2 - A^n x_1}_{X} = \underbrace{\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} u(n-1) \\ u(n-2) \\ \vdots \\ u(0) \end{bmatrix}}_{U}$$

- This is equivalent to solve with respect to U the linear system of equations

$$RU = X$$

- Matrix $R \in \mathbb{R}^{n \times nm}$ is called the **reachability matrix** of the system
- A solution U exists if and only if $X \in \text{Im}(R)$
  (Rouché-Capelli theorem: a solution exists $\Leftrightarrow \text{rank}([R\,X]) = \text{rank}(R)$)

REACHABILITY

Theorem

The system (A, B) is completely reachable $\Leftrightarrow \text{rank}(R) = n$

Proof:

($\Rightarrow$) Assume (A, B) reachable, choose $x_1 = 0$ and $x_2 = x$. Then $\exists k \geq 0$ such that

$$x = \sum_{j=0}^{k-1} A^j B u(k-1-j)$$

If $k \leq n$, then clearly $x \in \text{Im}(R)$. If $k > n$, by Cayley-Hamilton theorem we have again $x \in \text{Im}(R)$. Since x is arbitrary, $\text{Im}(R) = \mathbb{R}^n$, so $\text{rank}(R) = n$.

($\Leftarrow$) If $\text{rank}(R) = n$, then $\text{Im}(R) = \mathbb{R}^n$. Let $X = x_2 - A^n x_1$ and $U = [\, u(n-1)' \;\ldots\; u(1)'\; u(0)']'$. The system $X = RU$ can be solved with respect to $U, \forall X$, so any state x_1 can be transferred to x_2 in $k = n$ steps. Therefore, the system (A, B) is completely reachable.

MINIMUM-ENERGY CONTROL

- Let (A, B) reachable and consider steering the state from $x(0) = x_1$ into $x(k) = x_2, k > n$

$$\underbrace{x_2 - A^k x_1}_{X} = \underbrace{\begin{bmatrix} B & AB & \dots & A^{k-1}B \end{bmatrix}}_{R_k} \underbrace{\begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(0) \end{bmatrix}}_{U}$$

  ($R_k \in \mathbb{R}^{n \times km}$ is the reachability matrix for k steps)
- Since $\text{rank}(R_k) = \text{rank}(R) = n, \forall k > n$ (Cayley-Hamilton), we get $\text{rank}\, R_k = \text{rank}[R_k\ X] = n$
- Hence the system $X = R_k U$ admits solutions U

Problem

Determine the input sequence $\{u(j)\}_{j=0}^{k-1}$ that brings the state from $x(0) = x_1$ to $x(k) = x_2$ with minimum energy $\frac{1}{2}\sum_{j=0}^{k-1} \|u(j)\|^2 = \frac{1}{2}U'U$

MINIMUM-ENERGY CONTROL

- The problem is equivalent to finding the solution U of the system of equations

$$X = R_k U$$

  with minimum norm $\|U\|$
- We must solve the optimization problem

$$U^* = \text{arg min } \frac{1}{2}\|U\|^2 \quad \text{subject to} \quad X = R_k U$$

- Let's apply the method of Lagrange multipliers:

$$\mathcal{L}(U,\lambda) = \frac{1}{2}\|U\|^2 + \lambda'(X - R_k U)$$ Lagrangian function

$$\frac{\partial \mathcal{L}}{\partial U} = U - R_k'\lambda = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = X - R_k U = 0$$

$$\Rightarrow \quad U^* = \underbrace{R_k'(R_k R_k')^{-1}}_{R_k^{\#} \text{ = pseudoinverse}} \cdot X$$

MATLAB
```
U=pinv(Rk)*X
```

Python
```
from numpy.linalg import pinv
U=pinv(Rk)@X
```

- Note that $R_k R_k'$ is invertible because $\text{rank}(R_k) = \text{rank}(R) = n, \forall k \geq n$

CONTROLLABILITY

- If the system is completely reachable, we have seen that we can bring the state vector from any value $x(0) = x_1$ to any other value $x(n) = x_2$

- Let's focus on the subproblem of determining a finite sequence of inputs that brings the state to the final value $x(n) = 0$

Definition

A system $x(k+1) = Ax(k) + Bu(k)$ is **controllable** to the origin in k steps if $\forall x_0 \in \mathbb{R}^n$ there exists a sequence $u(0), u(1), \ldots, u(k-1) \in \mathbb{R}^m$ such that $0 = A^k x_0 + \sum_{j=0}^{k-1} A^j B u(k-1-j)$

- Controllability is a weaker condition than reachability

CONTROLLABILITY, STABILIZABILITY

- The linear system of equations

$$-A^n x_0 = \underbrace{\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}}_{R} \begin{bmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(0) \end{bmatrix}$$

admits a solution if and only if $A^n x_0 \in \mathrm{Im}(R), \forall x_0 \in \mathbb{R}^n$

Theorem

The system is controllable to the origin (in n steps) if and only if

$$\mathrm{Im}(A^n) \subseteq \mathrm{Im}(R)$$

Definition

A linear system $x(k+1) = Ax(k) + Bu(k)$ is called **stabilizable** if can be driven asymptotically to the origin

- Stabilizability is a weaker condition than controllability

CANONICAL REACHABILITY DECOMPOSITION

- **Goal**: Make a change of coordinates to separate reachable from unreachable states

- Let $\text{rank}(R) = n_c < n$ and consider the change of coordinates

$$T = \begin{bmatrix} w_{n_c+1} & \dots & w_n & v_1 & \dots & v_{n_c} \end{bmatrix}$$

  where $\{v_1, \dots, v_{n_c}\}$ is a basis of $\text{Im}(R)$, and $\{w_{n_c+1}, \dots, w_n\}$ is a completion to obtain a basis of $\mathbb{R}^n$ (i.e., a basis of $\ker(R')$, $R'w_i = 0$)

- As $\text{Im}(R)$ is A-invariant ($Ax \in \text{Im}(R), \forall x \in \text{Im}(R)$, follows from Cayley-Hamilton theorem), Av_i has no components along the basis vectors w_{n_c+1}, ..., w_n

- Since $T^{-1}Av_i$ are the new coordinates of Av_i, the first $n - n_c$ components of $T^{-1}Av_i$ are zero

CANONICAL REACHABILITY DECOMPOSITION

- The columns of B also have zero components along w_{n_c+1}, ..., w_n, because $\text{Im}(B) \subseteq \text{Im}(R)$

- In the new coordinates, the system has matrices $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$ and $\tilde{C} = CT$ in the **canonical reachability form** (a.k.a. **controllability staircase form**)

$$\tilde{A} = \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} 0 \\ B_c \end{bmatrix} \quad \tilde{C} = \begin{bmatrix} C_{uc} & C_c \end{bmatrix}$$

MATLAB
[At,Bt,Ct,Tinv]= ctrbf(A,B,C)

- Let $x = \begin{bmatrix} x_{uc} \\ x_c \end{bmatrix}$ be the coordinates of the state vector in the new coordinate system, $x_{uc} \in \mathbb{R}^{n-n_c}$, $x_c \in \mathbb{R}^{n_c}$

- We have that $x_{uc}(k) = A_{uc}^k x_{uc}(0)$, so $x_{uc}(k)$ does not depend on $u(k)$

REACHABILITY AND TRANSFER FUNCTION

PROPOSITION

The eigenvalues of A_{uc} are not poles of the transfer function $C(zI-A)^{-1}B+D$

Proof: Let T transform (A,B) to canonical reachability decomposition $(\tilde{A},\tilde{B})$. The transfer function is

$$\begin{aligned} G(z) &= C(zI-A)^{-1}B+D = \tilde{C}(zI-\tilde{A})^{-1}\tilde{B}+D \\ &= \begin{bmatrix} C_{uc} & C_c \end{bmatrix} \left(zI - \begin{bmatrix} A_{uc} & 0 \\ A_{21} & A_c \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ B_c \end{bmatrix} + D \\ &= \begin{bmatrix} C_{uc} & C_c \end{bmatrix} \begin{bmatrix} (zI-A_{uc})^{-1} & 0 \\ \star & (zI-A_c)^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ B_c \end{bmatrix} + D \\ &= C_c(zI-A_c)^{-1}B_c+D \end{aligned}$$

Clearly $G(z)$ does not depend on the eigenvalues of A_{uc}

Lack of reachability $\rightarrow$ zero/pole cancellations!

REACHABILITY AND TRANSFER FUNCTION

- Why are the eigenvalues of A_{uc} not appearing in the transfer function $G(z)$?
- Remember: $G(z)$ explains the forced response, i.e., the response for $x(0) = 0$
- Expressed in canonical decomposition, the system evolution is

$$\left\{\begin{array}{rcl} x_{uc}(k+1) &=& A_{uc}x_{uc}(k) \\ x_c(k+1) &=& A_c x_c(k) + B_c u(k) + A_{21}x_{uc}(k) \\ y(k) &=& C_{uc}x_{uc}(k) + C_c x_c(k) + Du(k) \end{array}\right.$$

- For $x_{uc}(0) = 0, x_c(0) = 0$, we get $x_{uc}(k) \equiv 0$ and

$$\left\{\begin{array}{rcl} x_c(k+1) &=& A_c x_c(k) + B_c u(k) \\ y(k) &=& C_c x_c(k) + Du(k) \\ x_c(0) &=& 0 \end{array}\right.$$

  so the forced response does not depend at all on A_{uc} !
- The input $u(k)$ only affects the output $y(k)$ through the reachable subsystem (A_c, B_c, C_c, D), not through the unreachable part A_{uc}

REACHABILITY ANALYSIS OF CONTINUOUS-TIME SYSTEMS

- Similar definitions of reachability, controllability, and stabilizability can be given for continuous-time systems

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

- No distinction between controllability and reachability in continuous-time (because no finite-time convergence of modal response exists)

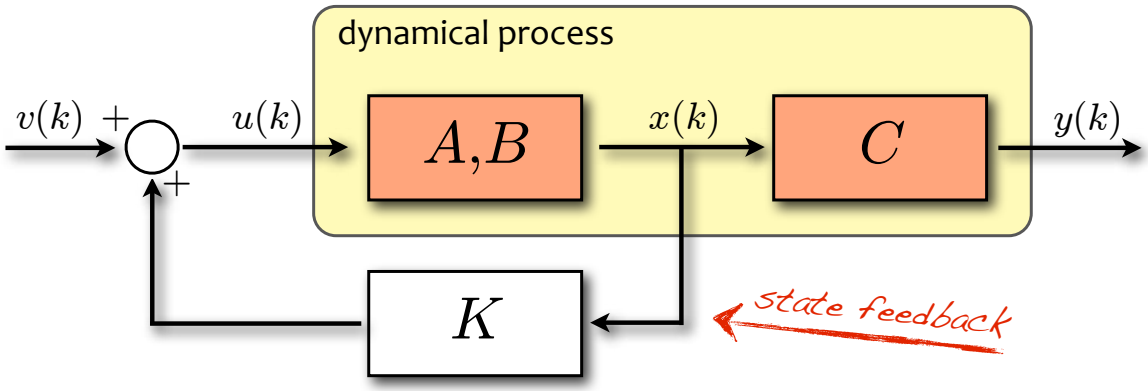
- Reachability matrix and canonical reachability decomposition are identical to discrete-time

- $\text{rank}\, R = n$ is also a necessary and sufficient condition for reachability

- A_{uc} asymptotically stable (all eigenvalues with negative real part) is also a necessary and sufficient condition for stabilizability

STABILIZATION BY STATE FEEDBACK

- **Main idea**: design a device that makes the process (A, B, C) asymptotically stable by manipulating the input u to the process



- If measurements of the state vector are available, we can set

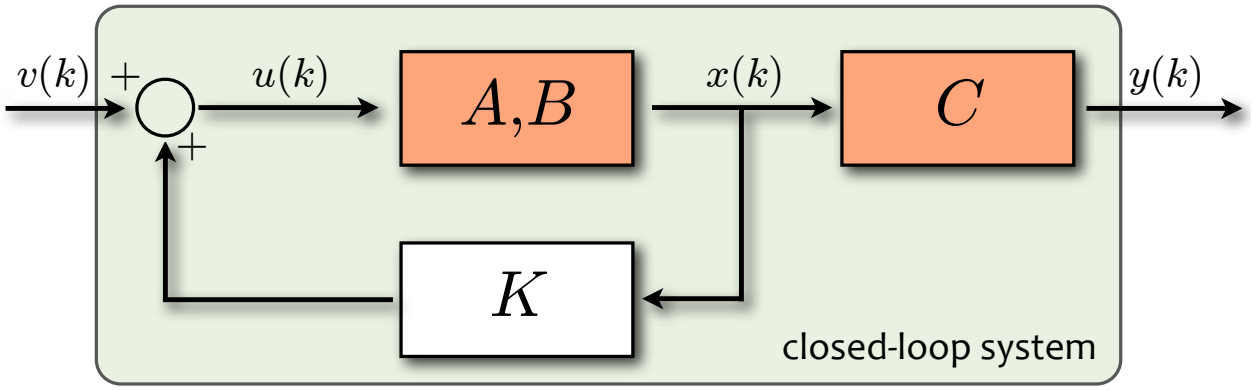
$$u(k) = k_1 x_1(k) + k_2 x_2(k) + \ldots + k_n x_n(k) + v(k)$$

- $v(k)$ is an exogenous signal exciting the closed-loop system

Problem

Find a feedback gain $K = [k_1 \; k_2 \; \ldots \; k_n]$ that makes the closed-loop system asymptotically stable.

STABILIZATION BY STATE FEEDBACK



- Let $u(k) = Kx(k) + v(k)$. The overall system is

$$\begin{aligned} x(k+1) &= (A+BK)x(k) + Bv(k) \\ y(k) &= (C+DK)x(k) + Dv(k) \end{aligned}$$

Theorem

(A, B) "reachable" ($\text{rank}\left[\begin{array}{cccc} B & AB & \dots & A^{n-1}B \end{array}\right] = n$) $\Rightarrow$ the eigenvalues of $(A+BK)$ can be decided **arbitrarily**.

EIGENVALUE ASSIGNMENT PROBLEM

Fact

(A, B) reachable $\Leftrightarrow$ (A, B) is algebraically equivalent to a pair $(\tilde{A}, \tilde{B})$ in **controllable canonical form**

$$\tilde{A} = \begin{bmatrix} 0 & & \\ \vdots & & I_{n-1} \\ 0 & & \\ -a_0 & -a_1 \quad \dots \quad & -a_{n-1} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

The transformation matrix T such that $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$ is

$$T = [B \; AB \; \dots A^{n-1}B] \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & 1 \\ a_2 & a_3 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1} & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

where $a_1, a_2, \dots, a_{n-1}$ are the coefficients of the characteristic polynomial

$$p_A(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \dots + a_1\lambda + a_0 = \det(\lambda I - A)$$

- Let (A, B) reachable and assume $m = 1$ (single input)
- Characteristic polynomials:

$$\begin{aligned} p_A(\lambda) &= \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0 \quad \text{(open-loop eigenvalues)} \\ p_d(\lambda) &= \lambda^n + d_{n-1}\lambda^{n-1} + \ldots + d_1\lambda + d_0 \quad \text{(desired closed-loop eigenvalues)} \end{aligned}$$

- Let (A, B) be in controllable canonical form

$$A = \begin{bmatrix} 0 & & \\ \vdots & & I_{n-1} \\ 0 & & \\ -a_0 & -a_1 \quad \ldots & -a_{n-1} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

- As $K = [k_1 \ \ldots \ k_n]$, we have

$$A + BK = \begin{bmatrix} 0 & & \\ \vdots & & I_{n-1} \\ 0 & & \\ -(a_0 - k_1) & -(a_1 - k_2) \quad \ldots & -(a_{n-1} - k_n) \end{bmatrix}$$

- The characteristic polynomial of $A + BK$ is therefore

$$\lambda^n + (a_{n-1} - k_n)\lambda^{n-1} + \ldots + (a_1 - k_2)\lambda + (a_0 - k_1)$$

- To match $p_d(\lambda)$ we impose

$$a_0 - k_1 = d_0 \ , \ a_1 - k_2 = d_1 \ , \ \ldots \ , \ a_{n-1} - k_n = d_{n-1}$$

Procedure

If (A, B) is in controllable canonical form, the feedback gain

$$K = \begin{bmatrix} a_0 - d_0 & a_1 - d_1 & \ldots & a_{n-1} - d_{n-1} \end{bmatrix}$$

makes $p_d(\lambda)$ the characteristic polynomial of $(A + BK)$

- If (A, B) is not in controllable canonical form we must set

$$\begin{aligned} \tilde{K} &= \begin{bmatrix} a_0 - d_0 & a_1 - d_1 & \dots & a_{n-1} - d_{n-1} \end{bmatrix} \\ K &= \tilde{K} T^{-1} \end{aligned}$$

← don't invert T, solve instead $T'K' = \tilde{K}'$ w.r.t. K' !

where

$$T = R \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & 1 \\ a_2 & a_3 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1} & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

- Explanation: a matrix M and $T^{-1}MT$ have the same eigenvalues

$$\begin{aligned} \det(\lambda I - T^{-1}MT) &= \det(T^{-1}T\lambda - T^{-1}MT) = \det(T^{-1}) \det(\lambda I - M) \\ &\quad \cdot \det(T) = \det(\lambda I - M) \end{aligned}$$

- Since $(\tilde{A} + \tilde{B}\tilde{K}) = T^{-1}AT + T^{-1}BKT = T^{-1}(A + BK)T$, it follows that $(\tilde{A} + \tilde{B}\tilde{K})$ and $(A + BK)$ have the same eigenvalues

ACKERMANN'S FORMULA

- Let (A, B) reachable and assume $m = 1$ (single input)
- Characteristic polynomials:

$$p_A(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_0 \quad \text{(open-loop eigenvalues)}$$
$$p_d(\lambda) = \lambda^n + d_{n-1}\lambda^{n-1} + \ldots + d_1\lambda + d_0 \quad \text{(desired closed-loop eigenvalues)}$$

- Let $p_d(A) = A^n + d_{n-1}A^{n-1} + \ldots + d_1A + d_0I$ ← (This is $n \times n$ matrix !)

Ackermann's formula

$$K = -[0 \ldots 0\,1][B\;AB\;\ldots\;A^{n-1}B]^{-1}p_d(A)$$

where $P = [\lambda_1 \lambda_2 \ldots \lambda_n]$ are the desired closed-loop poles

MATLAB
K=-acker(A,B,P);
K=-place(A,B,P);

Python
K=-ctrl.acker(A,B,P)
K=-ctrl.place(A,B,P)

- Numerically robust methods to solve the pole assignment problem exist

  (Tits, Yang, 1996)

ZEROS OF CLOSED-LOOP SYSTEM

Fact

In case of no feedthrough ($D = 0$) the zeros of the system are the same under state feedback

$$N_K(z) = N(z)$$

- Example for $x \in \mathbb{R}^3$: change the coordinates to canonical reachability form

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, K = \begin{bmatrix} k_3 & k_2 & k_1 \end{bmatrix}$$

$$\Rightarrow \quad \mathrm{Adj}(zI - A)B = \begin{bmatrix} z^2 + a_1 z + a_2 & z + a_1 & 1 \\ -a_3 & z(z + a_1) & z \\ -a_3 z & -a_2 z - a_3 & z^2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ z \\ z^2 \end{bmatrix}$$

- $\mathrm{Adj}(zI - A)B$ does not depend on the coefficients a_1, a_2, a_3
- Hence also $\mathrm{Adj}(zI - A - BK)B$ does not depend on $a_1 - k_1, a_2 - k_2, a_3 - k_3$

$$\Rightarrow \quad N(z) = C\,\mathrm{Adj}(zI - A)B = C\,\mathrm{Adj}(zI - A - BK)B = N_K(z),\ \forall K' \in \mathbb{R}^n$$

EXAMPLE - STUDENT POPULATION DYNAMICS

- The open-loop poles are $(0.8, 0.15, 0.2)$
- Say we want to place the closed-loop poles in $(0.1 \pm 0.2j, 0.1)$ by setting

$$u(k) = Kx(k) + Hr(k)$$

  where $r(k)$ is the desired reference signal

- First, design K by pole placement:

MATLAB
```
K=-place(A,B,[.1+.2*j,.1-.2*j,.1])
```

Python
```
K=-place(A,B,[.1+.2j,.1-.2j,.1])
```

- Then choose H such that the DC-gain from r to y is 1:

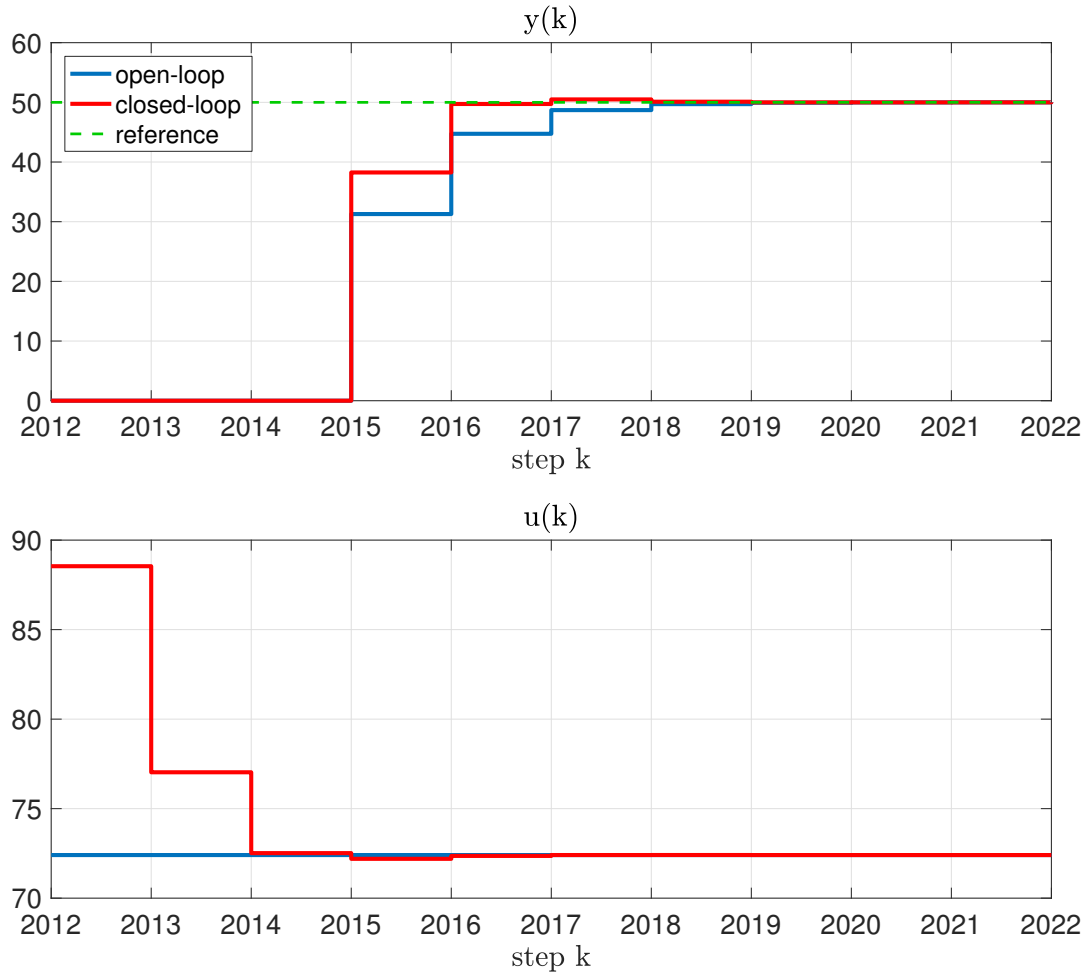
MATLAB
```
sys_cl=ss(A+B*K,B,C+D*K,D,1);
dc_cl=dcgain(sys_cl);
H=1/dc_cl;
```

Python
```
sys_cl=ctrl.ss(A+B@K,B,C+D@K,D,1)
dc_cl=ctrl.dcgain(sys_cl)
H=1/dc_cl
```

- We get $K = [-0.1300 \ -0.0698 \, 0.0017], H = 1.7708$

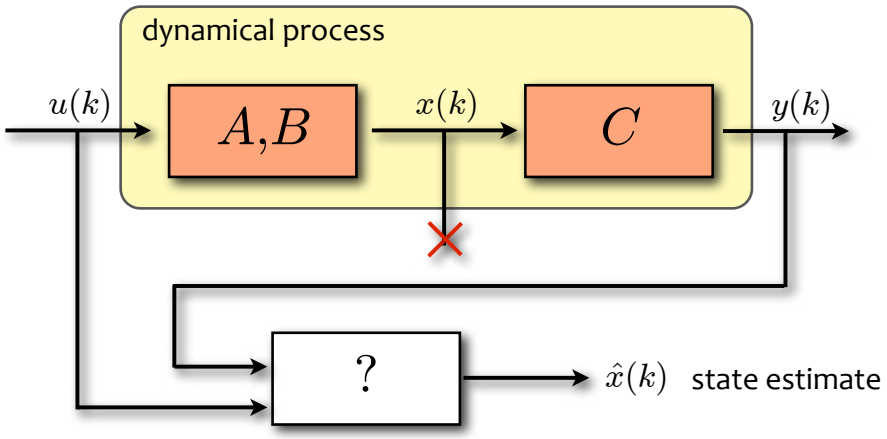
EXAMPLE - STUDENT POPULATION DYNAMICS

- Compare open-loop vs. closed-loop response



STATE ESTIMATION

OBSERVABILITY



- Implementing a state feedback controller $u(k) = Kx(k)$ requires the entire state vector $x(k)$
- **Problem:** often sensors only provide the measurements of output $y(k)$
- **Idea**: is it possible to estimate the state x by measuring only the output y and knowing the applied input u ?
- **Observability** analysis addresses this problem, telling us when and how the state estimation problem can be solved

OBSERVABILITY

- Consider

$$\begin{cases} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k)+Du(k) \end{cases}$$

with[5] $x \in \mathbb{R}^n, u \in \mathbb{R}, y \in \mathbb{R}$ and initial condition $x(0) = x_0 \in \mathbb{R}^n$

- The solution for the output is
$y(k, x_0, u(\cdot)) = CA^k x_0 + \sum_{j=0}^{k-1} CA^j Bu(k-1-j) + Du(k)$

Definition

The pair of states $x_1 \neq x_2 \in \mathbb{R}^n$ is called **indistinguishable** from the output $y(\cdot)$ if for any input sequence $u(\cdot)$

$$y(k, x_1, u(\cdot)) = y(k, x_2, u(\cdot)), \forall k \geq 0$$

A linear system is called **(completely) observable** if no pair of states are indistinguishable from the output

[5] Everything here can be easily generalized to multivariable systems $u \in \mathbb{R}^m, y \in \mathbb{R}^p$

OBSERVABILITY

- Consider the problem of reconstructing the initial condition x_0 from n output measurements, applying a known input sequence

$$\begin{aligned} y(0) &= Cx_0 + Du(0) \\ y(1) &= CAx_0 + CBu(0) + Du(1) \\ &\vdots \\ y(n-1) &= CA^{n-1}x_0 + \sum_{j=1}^{n-2} CA^j Bu(n-2-j) + Du(n-1) \end{aligned}$$

- Define

$$\Theta = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}}_{n \times n \text{ matrix}} \qquad Y = \underbrace{\begin{bmatrix} y(0) - Du(0) \\ y(1) - CBu(0) - Du(1) \\ \vdots \\ y(n-1) - \sum_{j=1}^{n-2} CA^j Bu(n-2-j) - Du(n-1) \end{bmatrix}}_{n\text{-th dimensional vector}}$$

OBSERVABILITY

- The initial state x_0 is determined by solving the linear system

$$Y = \Theta x_0$$

  The matrix $\Theta \in \mathbb{R}^{n \times n}$ is called the **observability matrix** of the system

- If we assume perfect knowledge of the output (i.e., no noise on output measurements), we can always solve the system $Y = \Theta x_0$. In particular:
  - There is only one solution if $\mathrm{rank}(\Theta) = n$
  - There are infinite solutions if $\mathrm{rank}(\Theta) < n$.
    In this case, all solutions are given by $x_0 + \ker(\Theta)$, where x_0 is any particular solution of the system (e.g., the true initial state)

- Knowing x_0, we know $x(k) = A^k x_0 + \sum_{i=0}^{k-1} A^i B u(k-1-i)$ for all $k \geq 0$

OBSERVABILITY

- The system of equations $\Theta x_0 = Y$ has a solution if and only if

$$\text{rank}(\Theta) = \text{rank}([\Theta\ Y]) \qquad \text{(Rouché-Capelli Theorem)}$$

- Because we have $\Theta \in \mathbb{R}^{n \times n}$, if $\text{rank}(\Theta) = n \Rightarrow \text{rank}([\Theta\ Y]) = n$ for each Y

- The solution is unique if and only if $\text{rank}(\Theta) = n$

- The input $u(k)$ only influences Y, not Θ

- Then, for linear systems the observability property only depends on A and C

OBSERVABILITY

Theorem

A linear system is observable if and only if $\text{rank}(\Theta) = n$

- As the observability property of a system depends only on matrices A and C, we call a pair (A, C) **observable** if

$$\text{rank}\left(\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n$$

- It can be proved that $\ker(\Theta)$ is the set of states $x \in \mathbb{R}^n$ that are indistinguishable from the origin $x = 0$

$$y(k, x, u(\cdot)) = y(k, 0, u(\cdot)),\ \forall k \geq 0$$

for any input sequence $u(\cdot)$

- Hence, since $\ker(\Theta) = \{0\}$ if and only if $\text{rank}(\Theta) = n$, a system is observable if and only if there are no states that are indistinguishable from $x = 0$

RECONSTRUCTABILITY

- Under observability assumptions, we just saw that it is possible to determine the initial condition x_0 from n input/output measurements

$$x(0) = \Theta^{-1} Y$$

- To close the control loop at time k it is enough to know the current $x(k)$

- If the initial condition $x(0)$ is known, it is possible to calculate $x(k)$ as

$$x(k) = A^k \Theta^{-1} Y + \sum_{i=0}^{k-1} A^i B u(k-1-i)$$

- **Question:** Can we determine the current state $x(k)$ even if the system is not completely observable?

RECONSTRUCTABILITY

Definition

A linear system $x(k+1) = Ax(k)+Bu(k)$ is called **reconstructable** in k steps if, for each initial condition x_0, $x(k)$ is uniquely determined by $\{u(j), y(j)\}_{j=0}^{k-1}$

The solutions of the system

$$Y_k \triangleq \begin{bmatrix} y(0) - Du(0) \\ y(1) - CBu(0) - Du(1) \\ \vdots \\ y(k-1) - \sum_{j=1}^{k-2} CA^j Bu(k-2-j) + Du(k-1) \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix}}_{\Theta_k} x$$

are given by $x = x_0 + \ker(\Theta_k)$, where x_0 is the "true" (unknown) initial state

RECONSTRUCTABILITY

- Let x_0 be the initial (unknown) "true" state, and $x = x_0 + \bar{x}$ be a generic initial state, where $\bar{x} \in \ker(\Theta_k)$. An estimation $\hat{x}(k)$ of the current state $x(k)$ is

$$\hat{x}(k) = A^k x_0 + A^k \bar{x} + \sum_{j=1}^{k-1} A^j B u(k-1-j)$$

- $\hat{x}(k)$ coincides with $x(k)$ if and only if $\bar{x} \in \ker(A^k)$. Because this must hold for any $\bar{x} \in \ker(\Theta_k)$, we have the following

Lemma

A system is reconstructable in k steps if and only if $\ker(\Theta_k) \subseteq \ker(A^k)$

Definition

A system is **detectable** if it is reconstructable asymptotically for $k \to +\infty$

CANONICAL OBSERVABILITY DECOMPOSITION

Goal: Change coordinates to separate observable and unobservable states

- Let $\dim(\ker(\Theta)) = n - n_o \geq 1$ and consider the change of coordinates

$$T = \begin{bmatrix} v_{n_o+1} & \dots & v_n & w_1 & \dots & w_{n_o} \end{bmatrix}$$

where $\{v_{n_o+1}, \dots, v_n\}$ is a basis of $\ker(\Theta)$, and $\{w_1, \dots, w_{n_o}\}$ is a completion to obtain a basis of $\mathbb{R}^n$

- By Cayley-Hamilton theorem, $\ker(\Theta)$ is A-invariant ($Ax \in \ker(\Theta), \forall x \in \Theta$), and hence Av_i has no components along the basis vector $w_1, \dots, w_{n_o}$, $\forall i = n_o + 1, \dots, n$
- Note also that $Cv_i = 0$, because $\Theta v_i = 0, \forall i = n_o + 1, \dots, n$
- In the new coordinates the system has matrices $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$ and $\tilde{C} = CT$ in the **canonical observability form**

$$\tilde{A} = \begin{bmatrix} A_{uo} & A_{12} \\ 0 & A_o \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} B_{uo} \\ B_o \end{bmatrix} \quad \tilde{C} = \begin{bmatrix} 0 & C_o \end{bmatrix}$$

MATLAB
[At,Bt,Ct,Tinv]= obsvf(A,B,C)

OBSERVABILITY AND TRANSFER FUNCTION

PROPOSITION

The eigenvalues of A_{uo} are not poles of the transfer function $C(zI-A)^{-1}B+D$

Proof: Consider a matrix T changing the state coordinates to canonical observability decomposition of (A,C). The transfer function is

$$\begin{aligned} G(z) &= C(zI-A)^{-1}B+D = \tilde{C}(zI-\tilde{A})^{-1}\tilde{B}+D = \\ &\quad \begin{bmatrix} 0 & C_o \end{bmatrix} \left(zI - \begin{bmatrix} A_{uo} & A_{12} \\ 0 & A_o \end{bmatrix} \right)^{-1} \begin{bmatrix} B_{uo} \\ B_o \end{bmatrix} + D \\ &= \begin{bmatrix} 0 & C_o \end{bmatrix} \begin{bmatrix} (zI-A_{uo})^{-1} & \star \\ 0 & (zI-A_o)^{-1} \end{bmatrix} \begin{bmatrix} B_{no} \\ B_o \end{bmatrix} + D \\ &= C_o(zI-A_o)^{-1}B_o + D \end{aligned}$$

Clearly $G(z)$ does not depend on the eigenvalues of A_{uo}

Lack of observability $\rightarrow$ zero/pole cancellations!

OBSERVABILITY AND TRANSFER FUNCTION

- Why are the eigenvalues of A_{uo} not appearing in the transfer function $G(z)$?

- Expressed in canonical decomposition, the system evolution is

$$\left\{\begin{array}{rcl} x_{uo}(k+1) &=& A_{uo}x_{uo}(k) + A_{12}x_o(k) + B_{uo}u(k) \\ x_o(k+1) &=& A_o x_o(k) + B_o u(k) \\ y(k) &=& C_o x_o(k) + Du(k) \end{array}\right.$$

- The evolution of $x_o(k)$ is not affected by the unobservable states $x_{uo}(k)$

$$x_o(k) = A_o^k x_o(0) + \sum_{i=0}^{k-1} A_o^i B_o u(k-1-i)$$

so the output $y(k) = C_o x_o(k) + Du(k)$ does not depend at all on A_{uo} !

CANONICAL OBSERVABILITY DECOMPOSITION

PROPOSITION

$A_o \in \mathbb{R}^{n_o \times n_o}$ and $C_o \in \mathbb{R}^{p \times n_o}$ are a completely observable pair

Proof:

- We have that

$$n_o = \operatorname{rank}\Theta = \operatorname{rank}\Theta T = \operatorname{rank}\begin{bmatrix} CT \\ CTT^{-1}AT \\ \vdots \\ CTT^{-1}A^{n-1}T \end{bmatrix} = \operatorname{rank}\begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \\ \vdots \\ \tilde{C}\tilde{A}^{n-1} \end{bmatrix}$$

$$= \operatorname{rank}\begin{bmatrix} 0 & C_o \\ 0 & \\ \vdots & \vdots \\ 0 & C_oA_o^{n_o-1} \\ 0 & C_oA_o^{n_o} \\ \vdots & \vdots \\ 0 & C_oA_o^{n-1} \end{bmatrix} = \operatorname{rank}\begin{bmatrix} C_o \\ C_oA_o \\ \vdots \\ C_oA_o^{n_o-1} \end{bmatrix}$$

- The last equality follows by Cayley-Hamilton theorem (the last $n - n_o$ rows $[0\, C_0A_o^i]$ are a linear combination of the first n_o rows). Hence, (A_o, C_o) is completely observable

DUALITY

- Given a linear system (A, B, C, D), with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $y \in \mathbb{R}^p$, we call **dual system** the system

$$\begin{cases} \tilde{x}(k+1) &= A'\tilde{x}(k) + C'\tilde{u}(k) \\ \tilde{y}(k) &= B'\tilde{x}(k) + D'\tilde{u}(k) \end{cases}$$

  where $\tilde{x} \in \mathbb{R}^n$, $\tilde{u} \in \mathbb{R}^p$ and $\tilde{y} \in \mathbb{R}^m$

- The reachability [observability] matrix of the dual system is equal to the transpose of the observability [reachability] matrix of the original system

$$\tilde{R} = \begin{bmatrix} C' & A'C' & \dots & (A')^{n-1}C' \end{bmatrix} = \Theta'$$

$$\tilde{\Theta} = \begin{bmatrix} B' \\ B'A' \\ \vdots \\ B'(A')^{n-1} \end{bmatrix} = R'$$

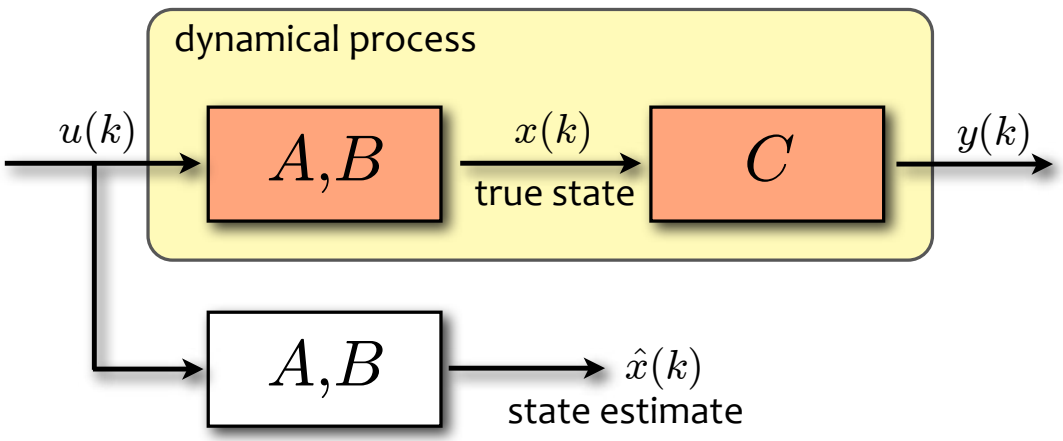
- The system (A, B, C, D) is reachable [observable] if and only if its dual system (A', C', B', D') is observable [reachable]

STATE ESTIMATION

State estimation problem

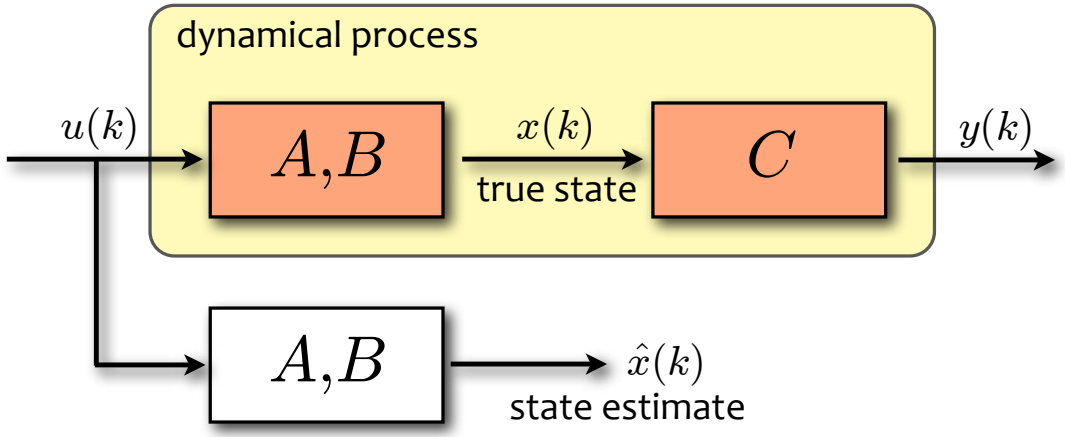
At each time k construct an estimate $\hat{x}(k)$ of the state $x(k)$, by only measuring the output $y(k)$ and input $u(k)$.

- **Open-loop observer**: Build an artificial copy of the system, fed in parallel by with the same input signal $u(k)$



- The "copy" is a numerical simulator $\hat{x}(k+1) = A\hat{x}(k) + Bu(k)$ reproducing the behavior of the real system

OPEN-LOOP OBSERVER



- The dynamics of the real system and of the numerical copy are

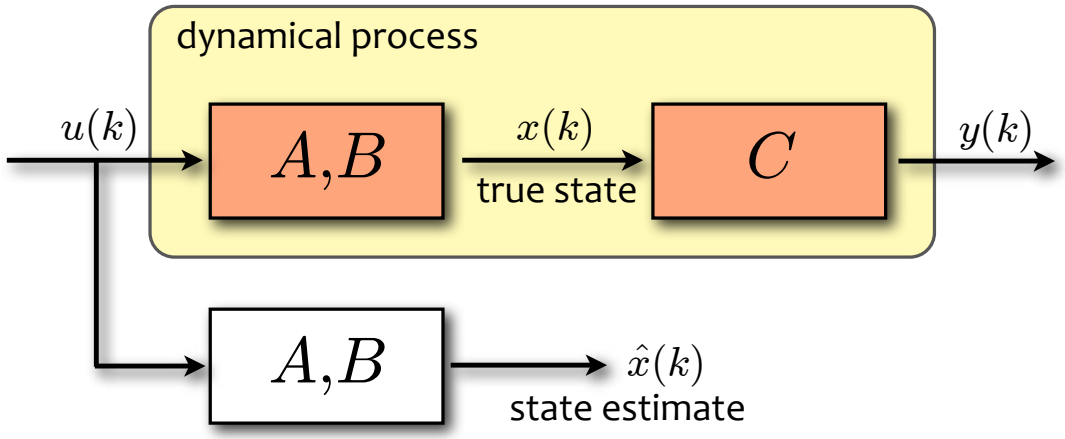
$$\begin{aligned} x(k+1) &= Ax(k)+Bu(k) && \text{True process} \\ \hat{x}(k+1) &= A\hat{x}(k)+Bu(k) && \text{Numerical copy} \end{aligned}$$

- The dynamics of the **estimation error** $\tilde{x}(k) = x(k) - \hat{x}(k)$ are

$$\tilde{x}(k+1) = Ax(k) + Bu(k) - A\hat{x}(k) - Bu(k) = A\tilde{x}(k)$$

and then $\tilde{x}(k) = A^k(x(0) - \hat{x}(0))$

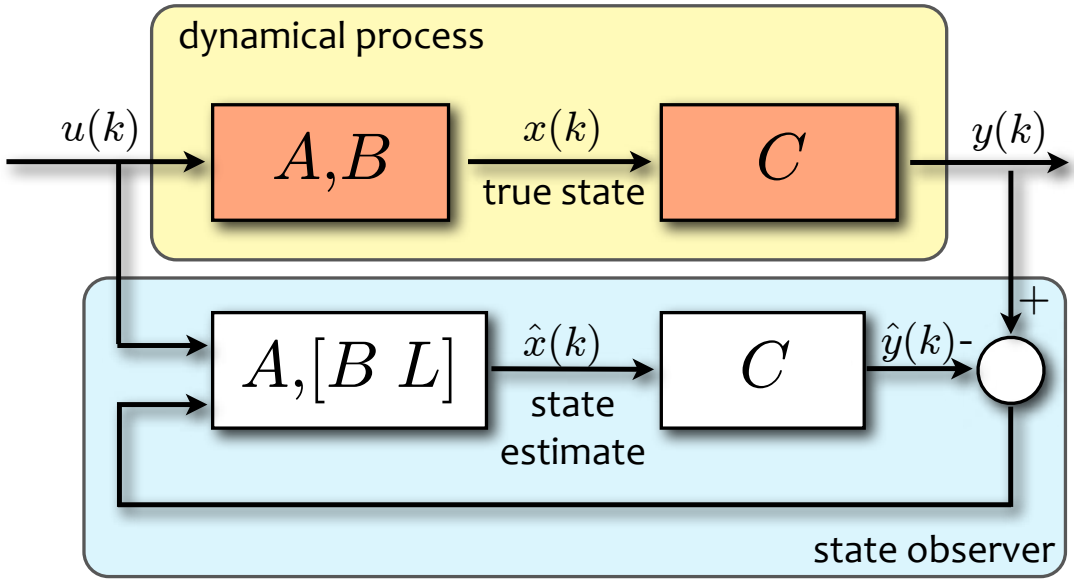
OPEN-LOOP OBSERVER



The estimation error is $\tilde{x}(k) = A^k(x(0) - \hat{x}(0))$. This is not ideal, because

- The dynamics of the estimation error are fixed by the eigenvalues of A and cannot be modified
- The estimation error vanishes asymptotically if and only if A is asymptotically stable
- Note that we are not exploiting $y(k)$ to compute the state estimate $\hat{x}(k)$!

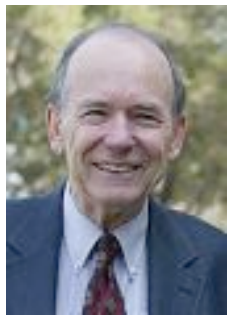
LUENBERGER OBSERVER



- **Luenberger observer**: Correct the estimation equation with a feedback from the estimation error $y(k) - \hat{y}(k)$

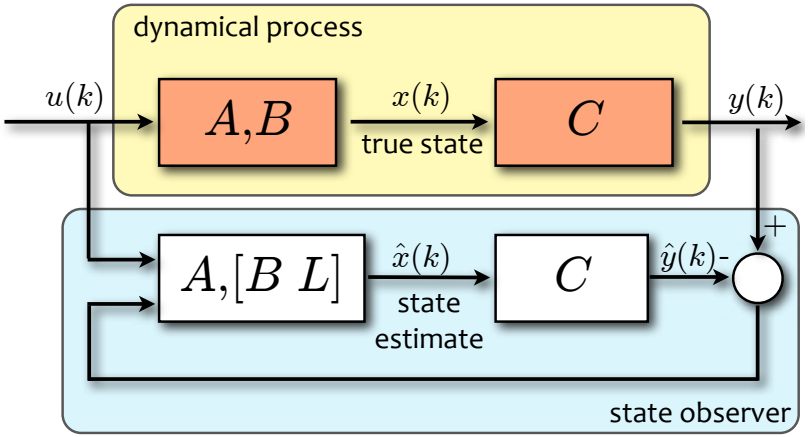
$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + \underbrace{L(y(k) - C\hat{x}(k))}_{\text{feedback on estimation error}}$$

where $L \in \mathbb{R}^{n \times p}$ is the **observer gain**



David G. Luenberger
(1937–)

LUENBERGER OBSERVER



- The dynamics of the state estimation error $\tilde{x}(k) = x(k) - \hat{x}(k)$ is

$$\begin{aligned} \tilde{x}(k+1) &= Ax(k) + Bu(k) - A\hat{x}(k) - Bu(k) - L[y(k) - C\hat{x}(k)] \\ &= (A - LC)\tilde{x}(k) \end{aligned}$$

  and then $\tilde{x}(k) = (A - LC)^k(x(0) - \hat{x}(0))$

- Same idea for continuous-time systems $\dot{x}(t) = Ax(t) + Bu(t)$

$$\frac{d\hat{x}(t)}{dt} = A\hat{x}(t) + Bu(t) + L[y(t) - C\hat{x}(t) - Du(t)]$$

  The dynamics of the state estimation error are $\frac{d\tilde{x}(t)}{dt} = (A - LC)\tilde{x}(t)$

EIGENVALUE ASSIGNMENT OF STATE OBSERVER

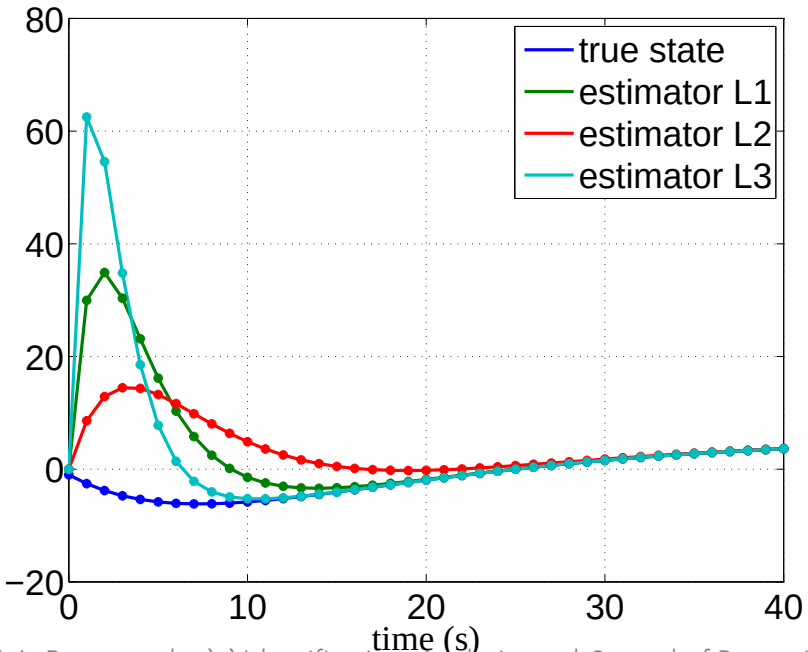
Theorem

If the pair (A, C) is "observable" (= (A', C') "reachable"), then the eigenvalues of $(A - LC)$ can be placed arbitrarily.

MATLAB
L=acker(A',C',P)';
L=place(A',C',P)';

Python
L=ctrl.acker(A.T,C.T,p).T
L=ctrl.place(A.T,C.T,p).T

where $P = [\lambda_1 \lambda_2 \dots \lambda_n]$ = desired observer eigenvalues



response from initial conditions $x(0) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \hat{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ for $u(k) \equiv 0.1$ for different choices of the observer poles

DYNAMIC COMPENSATORS

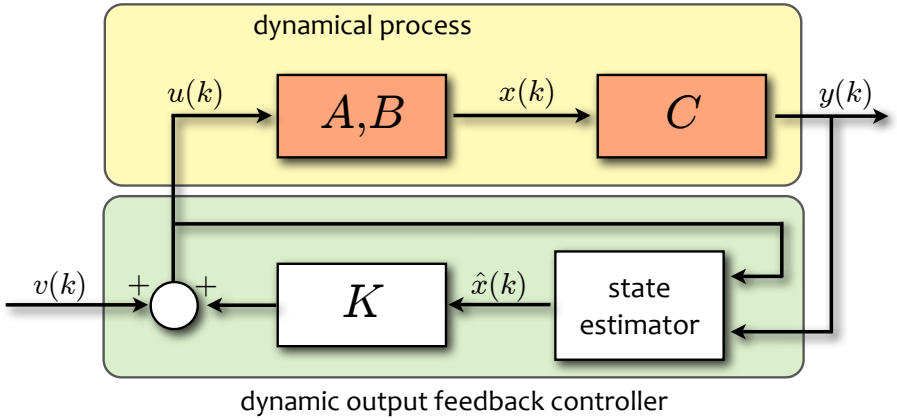
POTENTIAL ISSUES IN STATE FEEDBACK CONTROL

- Measuring the entire state vector may be too expensive (many sensors)

- It may be even impossible (high temperature, high pressure, inaccessible environment)

Can we use the estimate $\hat{x}(k)$ instead of $x(k)$ to close the loop ?

DYNAMIC COMPENSATOR



- Assume the open-loop system is completely observable and reachable
- Construct the linear state observer

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k))$$

- Set $u(k) = K\hat{x}(k) + v(k)$
- The dynamics of the error estimate $\tilde{x}(k) = x(k) - \hat{x}(k)$ is

$$\tilde{x}(k+1) = Ax(k)+Bu(k)-A\hat{x}(k)-Bu(k)+L(Cx(k)-C\hat{x}(k)) = (A-LC)\tilde{x}(k)$$

The error estimate does not depend on the feedback gain K !

CLOSED-LOOP DYNAMICS

- Let's combine the dynamics of the system, observer, and feedback gain

$$\begin{cases} x(k+1) &= Ax(k)+Bu(k) \\ \hat{x}(k+1) &= A\hat{x}(k)+Bu(k)+L(y(k)-C\hat{x}(k)) \\ u(k) &= K\hat{x}(k)+v(k) \\ y(k) &= Cx(k) \end{cases}$$

- Take $x(k), \tilde{x}(k)$ as state components of the closed-loop system

$$\begin{bmatrix} x(k) \\ \tilde{x}(k) \end{bmatrix} = \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k) \end{bmatrix}$$

(it is indeed a change of coordinates)

- The closed-loop dynamics is

$$\begin{cases} \begin{bmatrix} x(k+1) \\ \tilde{x}(k+1) \end{bmatrix} = \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} x(k) \\ \tilde{x}(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v(k) \\ y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \tilde{x}(k) \end{bmatrix} \end{cases}$$

CLOSED-LOOP DYNAMICS

- The transfer function from $v(k)$ to $y(k)$ is

$$\begin{aligned} G(z) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} zI-A-BK & BK \\ 0 & zI-A+LC \end{bmatrix}^{-1} \begin{bmatrix} B \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} (zI-A-BK)^{-1} & \star \\ 0 & (zI-A+LC)^{-1} \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix} \\ &= C(zI-A-BK)^{-1}B = \frac{N(z)}{D_K(z)} \end{aligned}$$

- Even if we substituted $x(k)$ with $\hat{x}(k)$, the input-output behavior of the closed-loop system didn't change !

The closed-loop poles can be assigned arbitrarily using **dynamic** output feedback, as in the state feedback case

The closed-loop transfer function does not depend on the observer gain L

SEPARATION PRINCIPLE

Separation principle

The design of the control gain K and of the observer gain L can be done independently

- Watch out ! $G(z) = C(zI - A - BK)^{-1}B$ only represents the I/O (=input/output) behavior of the closed-loop system

- The complete set of poles of the closed-loop system are given by

$$\det(zI - \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix}) = \det(zI-A-BK)\det(zI-A+LC) = D_K(z)D_L(z)$$

- A zero/pole cancellation of the observer poles has occurred:

$$G(z) = \begin{bmatrix} C & 0 \end{bmatrix} (zI - \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix})^{-1} \begin{bmatrix} B \\ 0 \end{bmatrix} = \frac{N(z)D_L(z)}{D_K(z)D_L(z)}$$

TRANSIENT EFFECTS OF THE ESTIMATOR GAIN

- L has an effect on the natural response of the system !
- To see this, consider the effect of a nonzero initial condition $\begin{bmatrix} x(0) \\ \tilde{x}(0) \end{bmatrix}$ for $v(k) \equiv 0$

$$
\begin{aligned}
y(0) &= Cx(0) \\
y(1) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} x(0) \\ \tilde{x}(0) \end{bmatrix} \\
&= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} (A+BK)x(0)-BK\tilde{x}(0) \\ (A-LC)\tilde{x}(0) \end{bmatrix} = C(A+BK)x(0) - CBK\tilde{x}(0) \\
y(2) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} A+BK & -BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} x(1) \\ \tilde{x}(1) \end{bmatrix} \\
&= C(A+BK)x(1) - CBK\tilde{x}(1) \\
&= C(A+BK)^2x(0) - C(A+BK)BK\tilde{x}(0) - CBK(A-\mathbf{L}C)\tilde{x}(0)
\end{aligned}
$$

- If $\tilde{x}(0) \neq 0$, L has an effect during the transient !

CHOOSING THE ESTIMATOR GAIN

- Intuitively, if $\hat{x}(k)$ is a poor estimate of $x(k)$ then the control action will also be poor

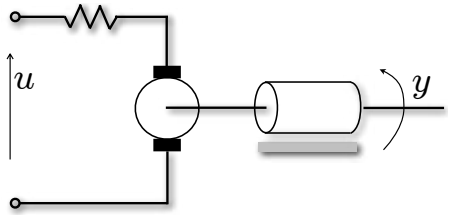
Rule of thumb: place the observer poles ≈ 10 times faster than the controller poles

- Optimal methods exist to choose the observer poles (Kalman filter)

- Fact: The choice of L is very important for determining the sensitivity of the closed-loop system with respect to input and output noise

EXAMPLE: CONTROL OF A DC MOTOR

$$\frac{d^3y}{dt} + \beta\frac{d^2y}{dt} + \alpha\frac{dy}{dt} = Ku$$



MATLAB

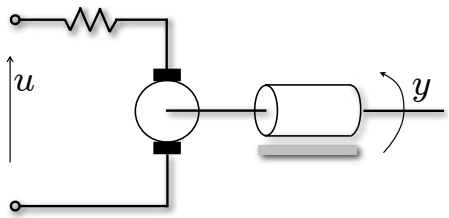
```
K=1; beta=.3; alpha=1;
G=tf(K,[1 beta alpha 0]);

ts=0.5; % sampling time
Gd=c2d(G,ts);
sysd=ss(Gd);
[A,B,C,D]=ssdata(sysd);

% Controller
polesK=[-1,-0.5+0.6*j,-0.5-0.6*j];
polesKd=exp(ts*polesK);
K=-place(A,B,polesKd);

% Observer
polesL=[-10, -9, -8];
polesLd=exp(ts*polesL);
L=place(A',C',polesLd)';
```

MATLAB

```
% Closed-loop system, state=[x;xhat]

bigA=[A,B*K;L*C,A+B*K-L*C];
bigB=[B;B];
bigC=[C,zeros(1,3)];
bigD=0;
clsys=ss(bigA,bigB,bigC,bigD,ts);

x0=[1 1 1]';    % Initial state
xhat0=[0 0 0]'; % Initial estimate
T=20;
initial(clsys, [x0;xhat0],T);
pause

t=(0:ts:T)';
v=ones(size(t));
lsim(clsys,v);
```

EXAMPLE: CONTROL OF A DC MOTOR

$$\frac{d^3y}{dt} + \beta\frac{d^2y}{dt} + \alpha\frac{dy}{dt} = Ku$$



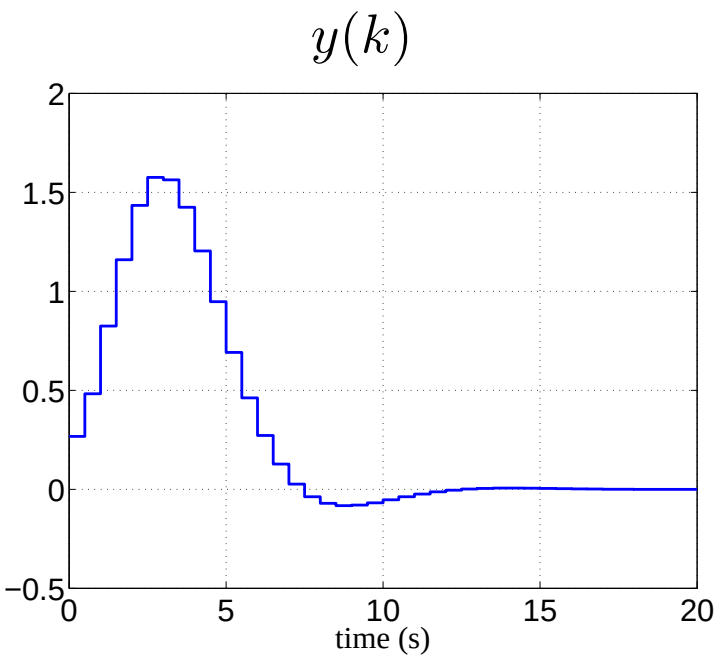
Python

```
import numpy as np
K,beta,alpha = 1,.3,1
G=ctrl.tf(K,[1,beta,alpha,0])

ts=0.5 # sampling time
Gd=ctrl.c2d(G,ts)
sysd=ctrl.ss(Gd)
A,B,C,D=ctrl.ssdata(sysd)

# Controller
polesK=np.array([-1,-0.5+0.6j,-0.5-0.6j])
polesKd=np.exp(ts*polesK)
K=-ctrl.place(A,B,polesKd)

# Observer
polesL=np.array([-10, -9, -8])
polesLd=np.exp(ts*polesL)
L=ctrl.place(A.T,C.T,polesLd).T
```

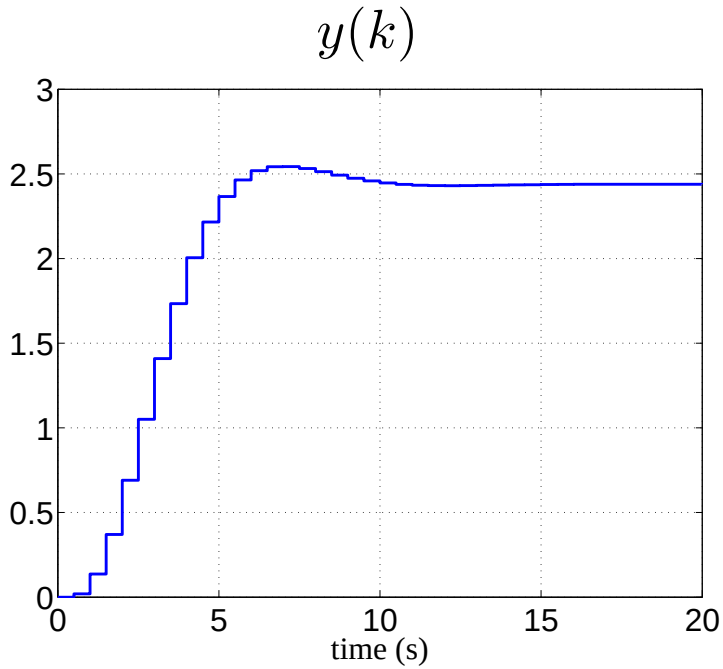
Python

```
# Closed-loop system, state=[x;xhat]

bigA=np.vstack((np.hstack((A,B@K)),
        np.hstack((L@C,A+B@K-L@C))))
bigB=np.vstack((B,B))
bigC=np.hstack((C,np.zeros((1,3))))
bigD=0
clsys=ctrl.ss(bigA,bigB,bigC,bigD,ts)

x0=np.array([[1],[1],[1]]) # Initial state
xhat0=np.zeros((3,1)) # Initial estimate
t=np.arange(0,20+ts,ts)
_,y=ctrl.initial_response(clsys,t,np.vstack((x0,xhat0)))

v=np.ones(t.size)
_,yf,xf=ctrl.forced_response(clsys,t,v)
```

EXAMPLE: CONTROL OF A DC MOTOR



$$x(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \hat{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, v(k) \equiv 0$$



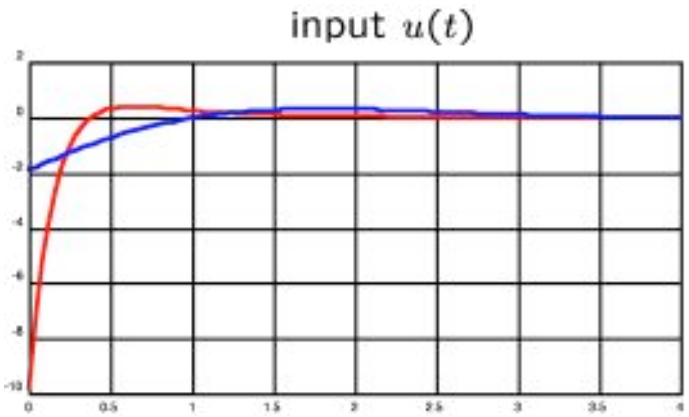
$$x(0) = \hat{x}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, v(k) \equiv 1$$

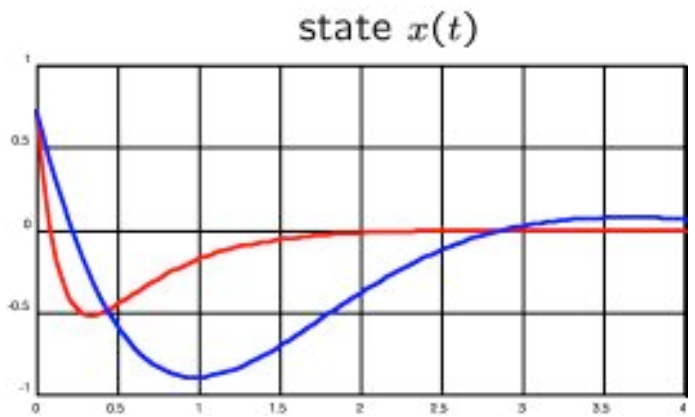
LINEAR QUADRATIC REGULATION

LINEAR QUADRATIC REGULATION (LQR)

- State-feedback control via pole placement requires one to assign the closed-loop poles
- Any way to place closed-loop poles automatically and optimally ?
- The main control objectives are
  1. Make the state $x(k)$ "small" (to converge to the origin)
  2. Use "small" input signals $u(k)$ (to minimize actuators' effort)

These are conflicting goals !





- LQR is a technique to place automatically and optimally the closed-loop poles

FINITE-TIME OPTIMAL CONTROL

- Linear system $x(k+1) = Ax(k) + Bu(k)$ with initial condition $x(0)$

- We look for the optimal sequence of inputs

$$U = \{u(0),\ u(1),\ \ldots,\ u(N-1)\}$$

driving $x(k)$ towards the origin while minimizing the performance index

$$J(x(0), U) = x'(N)Q_N x(N) + \sum_{k=0}^{N-1} x'(k)Qx(k) + u'(k)Ru(k)$$ quadratic cost

where $Q = Q' \succeq 0, R = R' \succ 0, Q_N = Q'_N \succeq 0$[6]

[6] For a matrix $Q \in \mathbb{R}^{n\times n}$, $Q \succ 0$ means that Q is a **positive definite** matrix, i.e., $x'Qx > 0$ for all $x \neq 0, x \in \mathbb{R}^n$. $Q \succeq 0$ means **positive semidefinite**, $x'Qx \geq 0, \forall x \in \mathbb{R}^n$.

FINITE-TIME OPTIMAL CONTROL

- Example: Q diagonal $Q = \mathrm{diag}(q_1, \ldots, q_n)$, single input, $Q_N = 0$

$$J(x(0), U) = \sum_{k=0}^{N-1} \left(\sum_{i=1}^{n} q_i x_i^2(k) \right) + Ru^2(k)$$

- Consider again the general **linear quadratic (LQ)** problem

$$J(x(0), U) = x'(N)Q_N x(N) + \sum_{k=0}^{N-1} x'(k)Qx(k) + u'(k)Ru(k)$$

  - N is called the time horizon over which we optimize performance
  - The first term $x'Qx$ penalizes the deviation of x from the desired target $x = 0$
  - The second term $u'Ru$ penalizes actuator authority
  - The third term $x'(N)Q_N x(N)$ penalizes how much the final state $x(N)$ deviates from the target $x = 0$

- Q, R, Q_N are the tuning parameters of optimal control design (cf. the parameters of the PID controller K_p, T_i, T_d)

MINIMUM-ENERGY CONTROLLABILITY

- Consider again the problem of controllability of the state to zero with minimum energy input

$$\begin{aligned} \min_U \quad & \left\| \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} \right\|_2^2 \\ \text{s.t.} \quad & x(N) = 0 \end{aligned}$$

- The minimum-energy control problem can be seen as a particular case of the LQ optimal control problem by setting

$$R = I,\ Q = 0,\ Q_N = \infty \cdot I$$

SOLUTION TO LQ OPTIMAL CONTROL PROBLEM

- By substituting $x(k) = A^k x(0) + \sum_{i=0}^{k-1} A^i B u(k-1-i)$ in

$$J(x(0), U) = \sum_{k=0}^{N-1} x'(k)Qx(k) + u'(k)Ru(k) + x'(N)Q_N x(N)$$

  we obtain

$$J(x(0), U) = \frac{1}{2}U'HU + x(0)'FU + \frac{1}{2}x(0)'Yx(0)$$

  where $H = H' \succ 0$ is a positive definite matrix

- The optimizer U^* is obtained by zeroing the gradient

$$\begin{aligned} 0 &= \nabla_U J(x(0), U) = HU + F'x(0) \\ \longrightarrow \; U^* &= \begin{bmatrix} u^*(0) \\ u^*(1) \\ \vdots \\ u^*(N-1) \end{bmatrix} = -H^{-1}F'x(0) \end{aligned}$$

[LQ PROBLEM MATRIX COMPUTATION]

$$J(x(0),U) = x'(0)Qx(0) + \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N-1) \\ x(N) \end{bmatrix}' \underbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & Q_N \end{bmatrix}}_{\bar{Q}} \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N-1) \\ x(N) \end{bmatrix} +$$

$$\begin{bmatrix} u'(0) & u'(1) & \dots & u'(N-1) \end{bmatrix} \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix}$$

$$\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix} = \overbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}^{\bar{S}} \begin{bmatrix} u(0) \\ u(1) \\ \dots \\ u(N-1) \end{bmatrix} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{N}} x(0)$$

$$J(x(0),U) = x'(0)Qx(0) + (\bar{S}U + \bar{N}x(0))'\bar{Q}(\bar{S}U + \bar{N}x(0)) + U'\bar{R}U$$

$$= \frac{1}{2}U' \underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H} U + x'(0) \underbrace{2\bar{N}'\bar{Q}\bar{S}}_{F} U + \frac{1}{2}x'(0) \underbrace{2(Q + \bar{N}'\bar{Q}\bar{N})}_{Y} x(0)$$

SOLUTION TO LQ OPTIMAL CONTROL PROBLEM

- The solution

$$U^* = \begin{bmatrix} u^*(0) \\ u^*(1) \\ \vdots \\ u^*(N-1) \end{bmatrix} = -H^{-1}F'x(0)$$

  is an open-loop one: $u^*(k) = f_k(x(0)), k = 0, 1, \ldots, N-1$

- Moreover the dimensions of the H and F matrices is proportional to the time horizon N

- We use optimality principles next to find a better solution (computationally more efficient, and more elegant)

DYNAMIC PROGRAMMING

- Consider the following basic fact in optimization

$$V_0 \triangleq \min_{z,y} f(z,y) = \min_z \{ \underbrace{\min_y f(z,y)}_{\text{this is a function of } z} \}$$

- In case f is separable in the sum of two functions

$$f(z,y) \triangleq f_0(z) + f_1(z,y)$$

we get $\min_y f(z,y) = f_0(z) + \min_y f_1(z,y)$

- Therefore we can compute V_0 in two steps:

$$\begin{aligned} V_1(z) &= \min_y f_1(z,y) \\ V_0 &= \min_z \{f_0(z) + V_1(z)\} \end{aligned}$$

- We apply the above reasoning to $f = J(x(0), U)$, $z = [u'(0) \ \dots \ u(k_1 - 1)']'$, $y = [u'(k_1) \ \dots \ u(N-1)']'$

DYNAMIC PROGRAMMING

- At a generic instant k_1 and state $x(k_1) = \bar{x}(z)$ consider the optimal **cost-to-go**

$$V_{k_1}(\bar{x}(z)) = \min_{u(k_1),\ldots,u(N-1)} \left\{ \sum_{k=k_1}^{N-1} x'(k)Qx(k) + u'(k)Ru(k) + x'(N)Q_N x(N) \right\}$$

Principle of dynamic programming

$$V_0(x(0)) = \min_{U \triangleq \{u(0),\ldots,u(N-1)\}} J(x(0), U)$$

$$= \min_{u(0),\ldots,u(k_1-1)} \left\{ \sum_{k=0}^{k_1-1} x'(k)Qx(k) + u'(k)Ru(k) + V_{k_1}(x(k_1)) \right\}$$

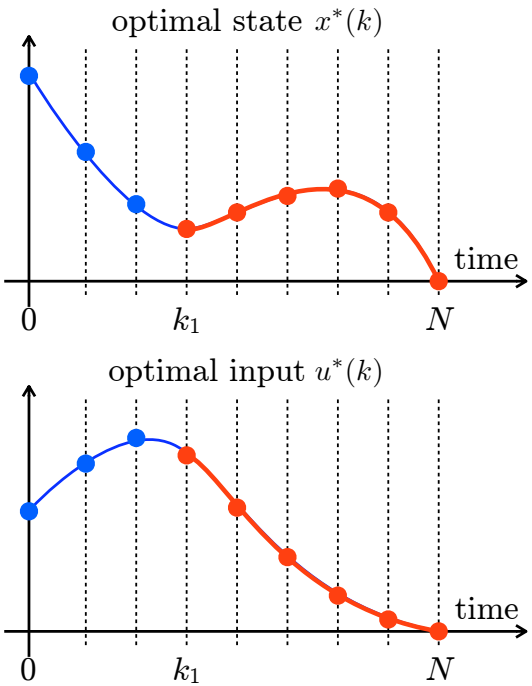
- Solving over $[0, k_1]$ with terminal weight equal to the optimal cost-to-go from k_1 to N at $x(k_1)$ is the same as solving over $[0, T]$

BELLMAN'S PRINCIPLE OF OPTIMALITY

Bellman's principle

Given the optimal sequence $U^* = [u^*(0), \ldots, u^*(N-1)]$ (and the corresponding optimal trajectory $x^*(k)$), the subsequence $[u^*(k_1), \ldots, u^*(N-1)]$ is optimal for the problem on the horizon $[k_1, N]$, starting from the optimal state $x^*(k_1)$

Richard Bellman
(1920-1984)



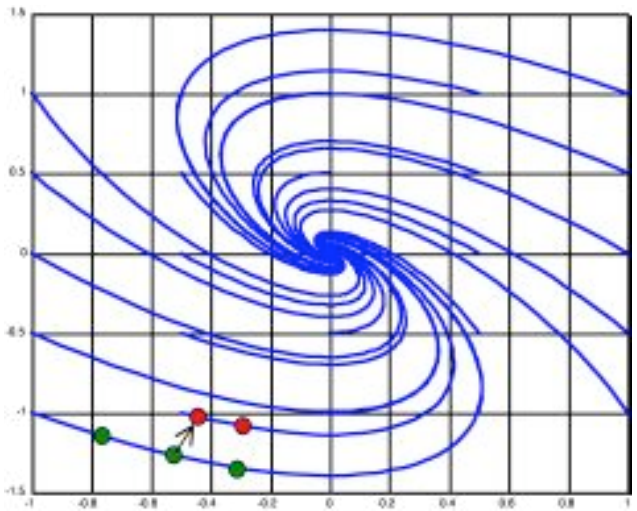
- Given the state $x^*(k_1)$, the optimal input trajectory u^* on the remaining interval $[k_1, N]$ only depends on $x^*(k_1)$
- Then each optimal move $u^*(k)$ of the optimal trajectory on $[0, N]$ only depends on $x^*(k)$
- The optimal control policy can be always expressed in state feedback form $u^*(k) = u_k^*(x^*(k))$!

BELLMAN'S PRINCIPLE OF OPTIMALITY

- The principle also applies to nonlinear systems and/or non-quadratic cost functions: the optimal control law can be always written in state-feedback form

$$u^*(k) = f_k(x^*(k)), \quad \forall k = 0, \ldots, N-1$$

optimal state trajectories x^*



- Compared to the open-loop solution $\{u^*(0), \ldots, u^*(N-1)\} = f(x(0))$ the feedback form $u^*(k) = f_k(x^*(k))$ has the big advantage of being more robust with respect to perturbations: at each time k we apply the best move on the remaining period $[k, N]$

RICCATI ITERATIONS

By applying the dynamic programming principle, we can compute the optimal inputs $u^*(k)$ recursively as a function of $x^*(k)$ (**Riccati iterations**):

1. Initialization: $P(N) = Q_N$
2. For $k = N, \dots, 1$, compute recursively the following matrix

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

3. Define

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

The optimal input is

$$u^*(k) = K(k)x^*(k)$$

Jacopo Francesco Riccati
(1676–1754)

The optimal input policy $u^*(k)$ is a (linear time-varying) state feedback !

LINEAR QUADRATIC REGULATION

- Consider the infinite-horizon optimal control problem

$$V^{\infty}(x(0)) = \min_{u(0),u(1),\dots} \sum_{k=0}^{\infty} x'(k)Qx(k) + u'(k)Ru(k)$$

Result

Let (A, B) be a stabilizable pair, $R \succ 0, Q \succeq 0$. There exists a unique solution P_∞ of the **algebraic Riccati equation (ARE)**

$$P_\infty = A'P_\infty A + Q - A'P_\infty B(B'P_\infty B + R)^{-1}B'P_\infty A$$

such that the optimal cost is $V^{\infty}(x(0)) = x'(0)P_\infty x(0)$ and the optimal control law is the constant linear state feedback $u(k) = K_{\rm LQR}x(k)$ with

$$K_{\rm LQR} = -(R + B'P_\infty B)^{-1}B'P_\infty A.$$

MATLAB

P_∞ = dare(A,B,Q,R)

[-K_∞,P_∞] = dlqr(A,B,Q,R)

[-K_∞,P_∞,E] = lqr(sysd,Q,R)

Python

P_∞,E,Km=ctrl.dare(A,B,Q,R)

K_∞=-Km

E= closed-loop poles

= eigenvalues of $(A + BK_{\rm LQR})$

LINEAR QUADRATIC REGULATION

- Go back to Riccati iterations: starting from $P(\infty) = P_\infty$ and going backwards we get $P(j) = P_\infty, \forall j \geq 0$
- Accordingly, we get

$$K(j) = -(R + B'P_\infty B)^{-1} B'P_\infty A \triangleq K_{\rm LQR}, \ \ \forall j = 0, 1, \ldots$$

- The LQR control law is linear and time-invariant

- (A, B) stabilizable implies closed-loop asymptotic stability, $\forall R, Q \succ 0$[7]
- LQR is an automatic and optimal way of placing poles !
- A similar result holds for continuous-time linear systems

[7] $R \succ 0, Q \succeq 0$ with $Q = F'F, F \in \mathbb{R}^{n \times n_q}, n_q = \operatorname{rank} Q$, and (A, F) detectable also ensures closed-loop asymptotic stability. Matrix F can be obtained for example by the LDL^T decomposition $Q = [L_1 \ L_2] \left[\begin{smallmatrix} D_1 & 0 \\ 0 & 0 \end{smallmatrix}\right] [L_1 \ L_2]', F = D_1^{\frac{1}{2}} L_1'$, with $D_1 \in \mathbb{R}^{n_q \times n_q}$.

LQR WITH OUTPUT WEIGHTING

- We often want to regulate only $y(k) = Cx(k)$ to zero, so define

$$V^{\infty}(x(0)) = \min_{u(0),u(1),\dots} \sum_{k=0}^{\infty} y'(k)Q_y y(k) + u'(k)Ru(k)$$

- The problem is again an LQR problem with equivalent state weight $Q = C'Q_yC$

MATLAB
[-K_∞,P_∞,E] = dlqry(sysd,Qy,R)

Python
P_∞,E,-K_∞ = ctrl.dare(A,B,C.T@Qy@C),R)

Corollary

Let (A, B) stabilizable, (A, C) detectable, $R > 0, Q_y > 0$. The LQR control law $u(k) = K_{\text{LQR}}x(k)$ the asymptotically stabilizes the closed-loop system

$$\lim_{t\to\infty} x(t) = 0, \quad \lim_{t\to\infty} u(t) = 0$$

Intuitively: the minimum cost $x'(0)P_\infty x(0)$ is finite $\Rightarrow y(k) \to 0$ and $u(k) \to 0$.

$y(k) \to 0$ implies that the observable part of the state $\to 0$. As $u(k) \to 0$, the unobservable states remain undriven and go to zero spontaneously (=detectability condition)

LQR EXAMPLE

- Two-dimensional single input single output (SISO) dynamical system (double integrator)

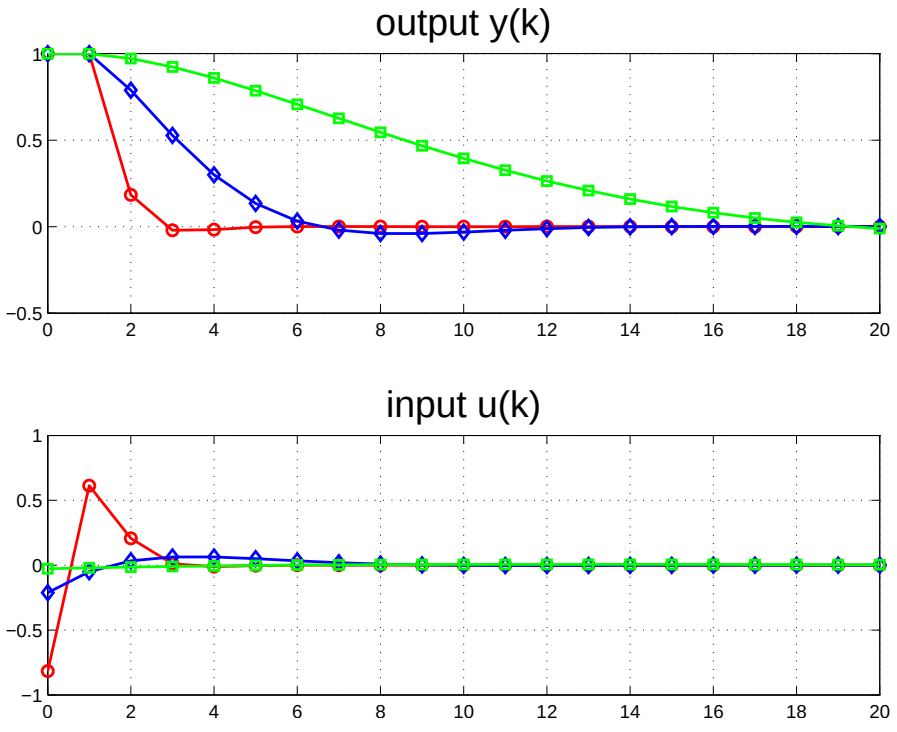
$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{aligned}$$

- LQR (infinite horizon) controller defined on the performance index

$$V^{\infty}(x(0)) = \min_{u(0),u(1),\ldots} \sum_{k=0}^{\infty} \frac{1}{\rho} y^2(k) + u^2(k), \quad \rho > 0$$

- Weights: $Q = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] \cdot \frac{1}{\rho} \cdot \left[\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right] = \left[\begin{smallmatrix} \frac{1}{\rho} & 0 \\ 0 & 0 \end{smallmatrix}\right], R = 1$
- Note that only the ratio $Q_{11}/R = \frac{1}{\rho}$ matters, as scaling the cost function does not change the optimal control law

LQR EXAMPLE



$\rho = 0.1$ (red line)

$K = [-0.8166 \quad -1.7499]$

$\rho = 10$ (blue line)

$K = [-0.2114 \quad -0.7645]$

$\rho = 1000$ (green line)

$K = [-0.0279 \quad -0.2505]$

Initial state: $x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$$V^{\infty}(x(0)) = \min_{u(0),u(1),\ldots} \sum_{k=0}^{\infty} \frac{1}{\rho} y^2(k) + u^2(k)$$

KALMAN FILTERING

KALMAN FILTERING - INTRODUCTION

- **Problem:** assign observer poles in an optimal way, that is to minimize the state estimation error $\tilde{x} = x - \hat{x}$
- Information comes in two ways: from sensors measurements (a posteriori) and from the model of the system (a priori)
- We need to mix the two information sources optimally, given a probabilistic description of their reliability (sensor precision, model accuracy)

Rudolf E. Kalman*
(1930–2016)

The **Kalman filter** solves this problem, and is now the most used state observer in most engineering fields (and beyond)

* R.E. Kalman receiving the Medal of Science from the President of the USA on October 7, 2009

PROCESS MODEL

- The process is modeled as the **linear time-varying system with noise**

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + G(k)\xi(k) \\ y(k) &= C(k)x(k) + D(k)u(k) + \zeta(k) \end{aligned} \qquad x(0) = x_0$$

- $\xi(k) \in \mathbb{R}^q$ = **process noise**, $E[\xi(k)] = 0$ (zero mean), $E[\xi(k)\xi'(j)] = 0, \forall k \neq j$, (white noise), $E[\xi(k)\xi'(k)] = Q(k) \succeq 0$ (covariance matrix)

- $\zeta(k) \in \mathbb{R}^p$ = **measurement noise**, $E[\zeta(k)] = 0$, $E[\zeta(k)\zeta'(j)] = 0 \, \forall k \neq j$, $E[\zeta(k)\zeta'(k)] = R(k) \succ 0$

- $x_0 \in \mathbb{R}^n$ is a random vector, $E[x_0] = \bar{x}_0$, $P_0 = E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)']$, $P_0 \succeq 0$

- Vectors $\xi(k), \zeta(k), x_0$ are uncorrelated: $E[\xi(k)\zeta_i(j)] = 0$, $E[\xi(k)x_0'] = 0$, $E[\zeta_i(k)x_0'] = 0, \forall k, j \in \mathbb{Z}, \forall i = 1, \ldots, p$

- Probability distributions: we often assume **normal** (=**Gaussian**) distributions $\xi(k) \sim \mathcal{N}(0, Q(k)), \zeta(k) \sim \mathcal{N}(0, R(k)), x_0 \sim \mathcal{N}(\bar{x}_0, P_0)$

KALMAN FILTER

In formulating the Kalman filter we use the following notation:

$\hat{x}(k\|k-1)$	state estimate at time k based on data up to time $k-1$
$\hat{x}(0\|-1)=\bar{x}_0$	initial state estimate
$\tilde{x}(k\|k-1)=x(k)-\hat{x}(k\|k-1)$	state estimation error
$P(k\|k-1)=E\left[\tilde{x}(k\|k-1)\tilde{x}(k\|k-1)'\right]$	covariance of state estimation error
$\hat{x}(k\|k)$	state estimate at time k based on data up to time k
$\tilde{x}(k\|k)=x(k)-\hat{x}(k\|k)$	state estimation error
$P(k\|k)=E\left[\tilde{x}(k\|k)\tilde{x}(k\|k)'\right]$	covariance of state estimation error
$\hat{x}(k+1\|k)$	state prediction at time $k+1$ based on data up to time k

KALMAN FILTER

- The Kalman filter provides the optimal estimate $\hat{x}(k|k)$ of $x(k)$ given the measurements up to time k
- Optimality means that the trace of the variance $P(k+1|k)$ is minimized
- The filter is based on two steps:

1. measurement update based on the most recent $y(k)$

$$\begin{aligned} M(k) &= P(k|k-1)C(k)'[C(k)P(k|k-1)C(k)'+R(k)]^{-1} \\ \hat{x}(k|k) &= \hat{x}(k|k-1)+M(k)\left(y(k)-C(k)\hat{x}(k|k-1)-D(k)u(k)\right) \\ P(k|k) &= (I-M(k)C(k))P(k|k-1) \end{aligned}$$

with initial conditions $\hat{x}(0|-1)=\bar{x}_0, P(0|-1)=P_0$

2. time update based on the model of the system

$$\begin{aligned} \hat{x}(k+1|k) &= A(k)\hat{x}(k|k)+B(k)u(k) \\ P(k+1|k) &= A(k)P(k|k)A(k)'+G(k)Q(k)G(k)' \end{aligned}$$

STATIONARY KALMAN FILTER

- Assume A, C, G, Q, R are constant (time-invariant case)
- Under suitable assumptions[8], $P(k|k-1), M(k)$ converge to constant matrices

$$\begin{aligned} P_\infty &= AP_\infty A' + GQG' - AP_\infty C' \left[CP_\infty C' + R\right]^{-1} CP_\infty A' \\ M_\infty &= P_\infty C'(CP_\infty C' + R)^{-1} \end{aligned}$$

- By setting $L_\infty = AM_\infty$ the dynamics of the prediction $\hat{x}(k|k-1)$ becomes the Luenberger observer

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + B(k)u(k) + L_\infty(y(k) - C\hat{x}(k|k-1) - D(k)u(k))$$

with all the eigenvalues of $(A - L_\infty C)$ inside the unit circle

MATLAB

[~,L,P_∞,M]=kalman(sys,Q,R)

Python

P_∞,_,Lt=ctrl.dare(A.T,C.T,G.T@Q@G,R)
L=Lt.T; M=np.linalg.lstsq(A,L)[0]

[8] (A, C) observable, and (A, GB_q) stabilizable, where B_q is such that $Q = B_q B_q'$, cf. conditions for asymptotic stability of LQR

TUNING KALMAN FILTERS

- It is usually hard to quantify exactly the correct values of Q and R for a given process
- The diagonal terms of R are related to how noisy are output sensors
- Q is harder to relate to physical noise, it mainly relates to how rough is the (A, B) model
- After all, Q and R are the tuning knobs of the observer (similar to LQR)
- The "larger" is R with respect to Q the "slower" is the observer to converge (L, M will be small)
- On the contrary, the "smaller" is R than Q, the more precise are considered the measurments, and the "faster" observer will be to converge

EXTENDED KALMAN FILTER

- The Kalman filter can be extended to nonlinear systems

$$\begin{aligned} x(k+1) &= f(x(k), u(k), \xi(k)) \\ y(k) &= g(x(k), u(k)) + \zeta(k) \end{aligned}$$

1. Measurement update:

$$\begin{aligned} C(k) &= \frac{\partial g}{\partial x}(\hat{x}(k|k-1), u(k)) \\ M(k) &= P(k|k-1)C(k)'[C(k)P(k|k-1)C(k)' + R(k)]^{-1} \\ \hat{x}(k|k) &= \hat{x}(k|k-1) + M(k)\,(y(k) - g(\hat{x}(k|k-1), u(k))) \\ P(k|k) &= (I - M(k)C(k))P(k|k-1) \end{aligned}$$

2. Time update:

$$\hat{x}(k+1|k) = f(\hat{x}(k|k), u(k), E[\xi(k)]),\ \hat{x}(0|-1) = \hat{x}_0$$

$$A(k) = \frac{\partial f}{\partial x}(\hat{x}(k|k), u(k), E[\xi(k)]),\ G(k) = \frac{\partial f}{\partial \xi}(\hat{x}(k|k), u(k), E[\xi(k)])$$

$$P(k+1|k) = A(k)P(k|k)A(k)' + G(k)Q(k)G(k)',\ P(0|-1) = P_0$$

- The EKF is in general not optimal and may even diverge, due to linearization. But is the de-facto standard in nonlinear state estimation

LQG CONTROL

- **Linear Quadratic Gaussian (LQG)** control combines an LQR control law and a stationary Kalman predictor/filter
- Consider the stochastic dynamical system

$$x(k+1) = Ax(k) + Bu(k) + \xi(k),\ \xi(k) \sim \mathcal{N}(0, Q_{KF})$$
$$y(k) = Cx(k) + \zeta(k),\ \zeta(k) \sim \mathcal{N}(0, R_{KF})$$

with initial condition $x(0) = x_0, x_0 \sim \mathcal{N}(\bar{x}_0, P_0), P_0, Q_{KF} \succeq 0, R_{KF} \succ 0$, and ζ and ξ are independent and white noise terms.

- The objective is to minimize the cost function

$$J(x(0), U) = \lim_{T\to\infty} \frac{1}{T} E\left[\sum_{k=0}^{T} x'(k)Q_{LQ}x(k) + u'(k)R_{LQ}u(k)\right]$$

when the state x is not measurable

LQG CONTROL

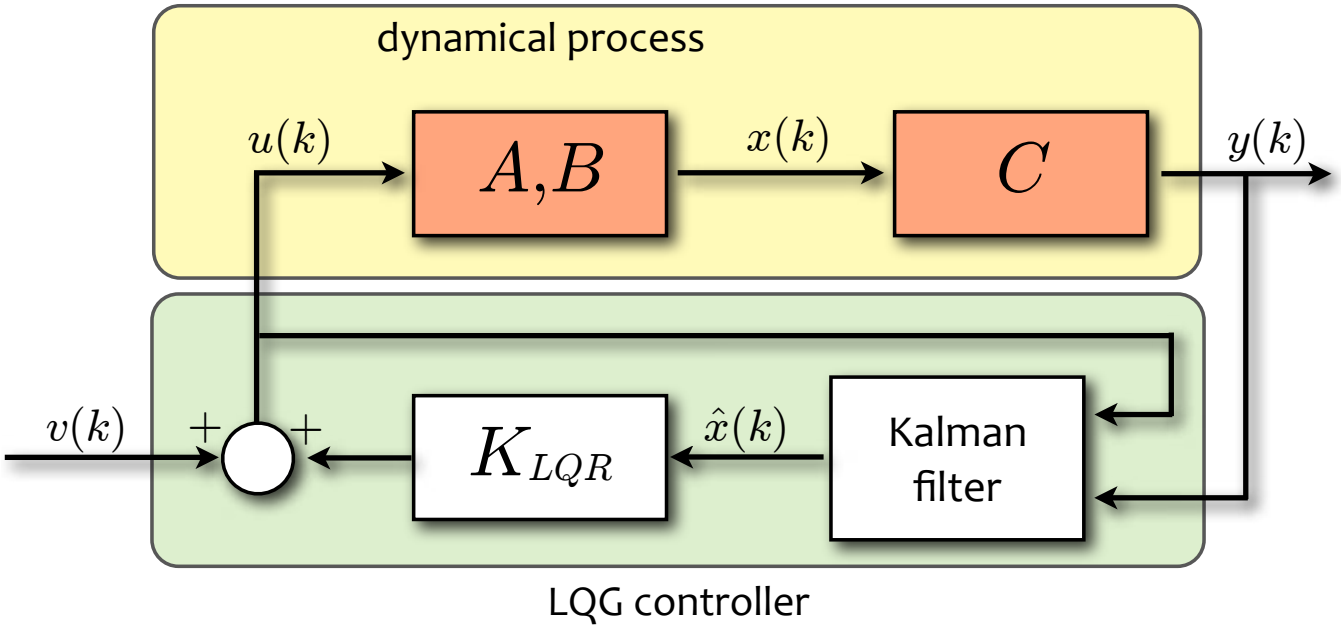
If we assume that all the assumptions for LQR control and Kalman predictor/filter hold, i.e.

- the pair (A, B) is reachable and the pair (A, C_q) with C_q such that $Q_{LQ} = C_q C_q'$ is observable (here Q is the weight matrix of the LQ controller)
- the pair (A, B_q), with B_q s.t. $Q_{KF} = B_q B_q'$, is stabilizable, and the pair (A, C) is observable (here Q is the covariance matrix of the Kalman predictor/filter)

Then, apply the following procedure:

1. Determine the optimal stationary Kalman predictor/filter, neglecting the fact that the control variable u is generated through a closed-loop control scheme, and find the optimal gain L_{KF}
2. Determine the optimal LQR strategy assuming the state accessible, and find the optimal gain K_{LQR}

LQG CONTROL



Analogously to the case of output feedback control using a Luenberger observer, it is possible to show that the extended state $[x' \ \tilde{x}']'$ has eigenvalues equal to the eigenvalues of $(A + BK_{LQR})$ plus those of $(A - L_{KF}C)$ ($2n$ in total)

SYSTEM IDENTIFICATION

MODEL IDENTIFICATION

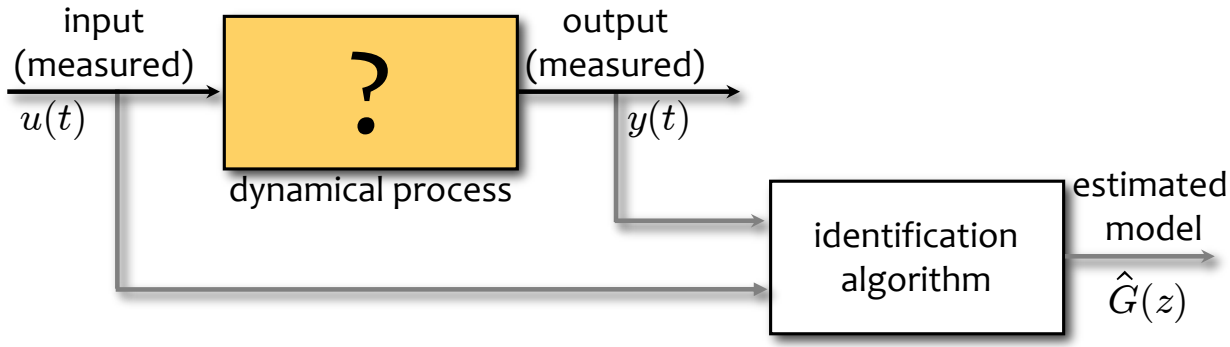
- Designing a control system requires a **dynamical model** of the process
- Often a dynamical model can be difficult to obtain due to the complexity of the process, whose dynamics may be even (partially or completely) unknown
- Even if we have a mathematical model, sometimes this is too complex to base a controller design on it (large state dimensions, nonlinearities, etc.)

System identification is a procedure to build a mathematical model of the dynamics of a system from **measured data**

Lecture based on

[1] L. Ljung, "System Identification," Control Systems Handbook (W. Levine ed.), CRC Press, pp. 1033–1054, 1995
[2] L. Ljung, "System Identification: Theory for the User," Prentice Hall, 1987

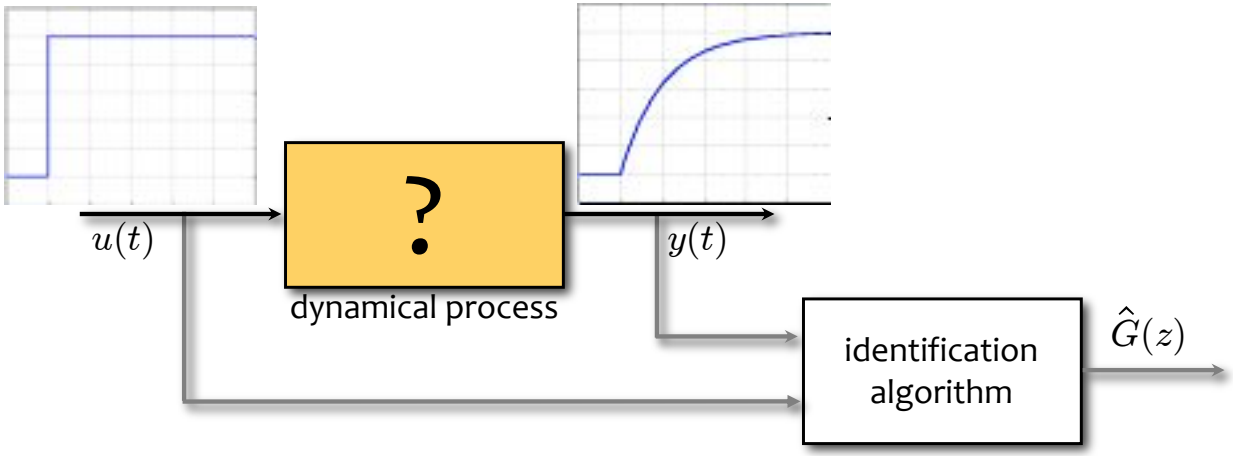
MODEL IDENTIFICATION



Different types of identification:

- **White box**: model structure based on first principles (e.g., Newton's law), model parameters estimated from measured data
- **Grey box**: model structure partially known from first principles, the rest is reconstructed from data
- **Black box**: model structure and its parameters completely unknown, they are only estimated from I/O data

STEP-RESPONSE IDENTIFICATION



- Excite the process with a step $u(t) = \mathbb{1}(t)$, record output response $y(t)$
- Observe the shape of $y(t)$ and reconstruct $G(z)$
  (1st-order response ? 2nd-order undamped response ? Any delay ? ...)
- Mostly used in process control: excitation experiment is easily done, superposition of effects can be used in the multivariable case to identify each entry $G_{ij}(z)$ of the **transfer matrix** $G(z)$, one at the time

BLACK-BOX IDENTIFICATION VIA LINEAR REGRESSION

- Consider the black-box **ARX (AutoRegressive eXogenous)** model

$$y(k)+a_1y(k-1)+...+a_{n_a}y(k-n_a) = b_1u(k-n_k)+...+b_{n_b}u(k-n_k-n_b+1)+e(k)$$

where $e(k)$ is zero-mean white noise and $y(k), u(k), e(k) \in \mathbb{R}$

- We can predict the next output value given previous observations

$$y(k) = -a_1y(k-1)-...-a_{n_a}y(k-n_a)+b_1u(k-n_k)+...+b_{n_b}u(k-n_k-n_b+1)+e(k)$$

- In more compact form

$$y(k) = \varphi'(k)\theta + e(k)$$

$$\theta = \begin{bmatrix} a_1 & \dots & a_{n_a} & b_1 & \dots & b_{n_b} \end{bmatrix}'$$ unknown parameter vector

$$\varphi(k) = \begin{bmatrix} -y(k-1) & \dots -y(k-n_a) & u(k-n_k) & \dots & u(k-n_k-n_b+1) \end{bmatrix}'$$

regressor

BLACK-BOX IDENTIFICATION VIA LINEAR REGRESSION

- Let $\hat{y}(k|\theta) = \varphi'(k)\theta$ = output prediction, which is an estimate of $y(k)$ based on the parameter vector θ and past data ($e(k) = 0$ is the best we can assume)

- We don't know θ, but we have collected a set Z^N of measured data

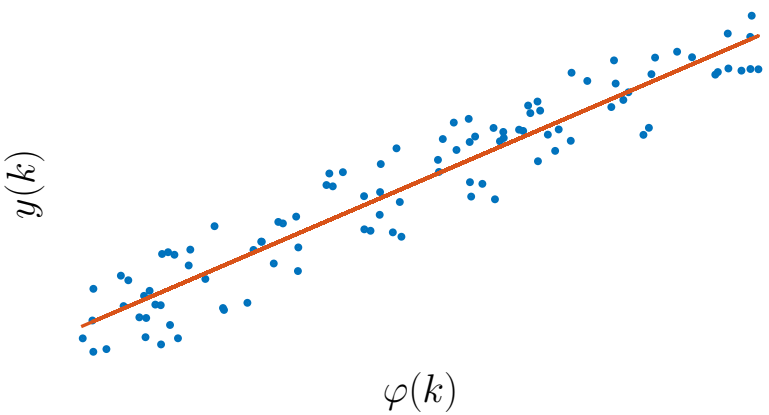
$$Z^N = \{u(-n), y(-n), ..., u(N-1), y(N-1)\}\,, \;\; n = \max\{n_a, n_b + n_k - 1\}$$

- We solve a **least-squares** problem to estimate the vector θ^* that best makes $\hat{y}(k|\theta)$ fit $y(k)$

$$\theta^* = \arg\min_{\theta} \left\{ V(\theta, Z^N) \right\}$$

with

$$\begin{aligned} V(\theta, Z^N) &= \frac{1}{N} \sum_{k=0}^{N-1} \left(y(k) - \hat{y}(k|\theta)\right)^2 \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(y(k) - \varphi'(k)\theta\right)^2 \end{aligned}$$



BLACK-BOX IDENTIFICATION VIA LINEAR REGRESSION

- $V(\theta, Z^N)$ is a quadratic function of θ. We find the minimum by zeroing the derivative of V

$$0 = \frac{d}{d\theta} V_N(\theta, Z^N) = -\frac{2}{N} \sum_{k=0}^{N-1} \varphi(k) \left(y(k) - \varphi'(k)\theta\right)$$

or

$$\sum_{k=0}^{N-1} \varphi(k) y(k) = \sum_{k=1}^{N} \varphi(k)\varphi'(k)\theta$$

- The best parameter vector we can choose is therefore

$$\theta^* = \left[\sum_{k=0}^{N-1} \varphi(k)\varphi'(k)\right]^{-1} \sum_{k=0}^{N-1} \varphi(k) y(k)$$

MATLAB
θ^*=arx(Z^N,[n_a n_b n_k])

RECURSIVE LINEAR REGRESSION

- Drawback of (batch) linear regression: if a new data pair $u(N), y(N)$ is acquired the new matrix $\left[\sum_{k=0}^{N} \varphi(k)\varphi'(k)\right]^{-1}$ is required to compute the new optimal parameter vector θ^*

- Computations become more and more expensive as N keeps growing

- Given the best estimate $\theta^*(k-1)$ obtained using $k-1$ data points

$$\theta^*(k-1) = P(k-1)\sum_{j=0}^{k-1}\varphi(j)y(j), \quad P(k-1) = \left[\sum_{j=0}^{k-1}\varphi(j)\varphi'(j)\right]^{-1}$$

  we would like to get $\theta^*(k)$ without solving the regression problem from scratch

RECURSIVE LINEAR REGRESSION

- Since

$$P^{-1}(k) = P^{-1}(k-1) + \varphi(k)\varphi'(k), \qquad P^{-1}(k-1)\theta^*(k-1) = \sum_{j=0}^{k-1} \varphi(j)y(j)$$

we get

$$\begin{aligned}
\theta^*(k) &= P(k)\sum_{j=0}^{k} \varphi(j)y(j) = P(k)\left(\sum_{j=0}^{k-1} \varphi(j)y(j) + \varphi(k)y(k)\right) \\
&= P(k)\left(P^{-1}(k-1)\theta^*(k-1) + \varphi(k)y(k)\right) \\
&= P(k)\left((P^{-1}(k) - \varphi(k)\varphi'(k))\theta^*(k-1) + \varphi(k)y(k)\right)
\end{aligned}$$

and therefore

$$\theta^*(k) = \theta^*(k-1) + P(k)\varphi(k)\underbrace{(y(k) - \theta^*(k-1)'\varphi(k))}_{\text{estimation error}}$$

RECURSIVE LINEAR REGRESSION

- Since

$$P(k) = \left[\sum_{j=0}^{k} \varphi(j)\varphi'(j)\right]^{-1} = \left[P^{-1}(k-1) + \varphi(k)\varphi'(k)\right]^{-1}$$

we can apply the **Matrix Inversion Lemma** to update $P(k)$ recursively and get

$$P(k) = P(k-1) - \frac{P(k-1)\varphi(k)\varphi'(k)P(k-1)}{1+\varphi'(k)P(k-1)\varphi(k)}$$

- Let $m = \dim\theta = n_a + n_b$. Matrix $\sum_{j=0}^{k-1} \varphi(j)\varphi'(j)$ is not invertible for $k < m$
- We can start the recursions after m steps with $P(m-1) = \left[\sum_{j=0}^{m-1} \varphi(j)\varphi'(j)\right]^{-1}$

(if the inverse exists) and $\theta^*(m-1) = P(m-1)^{-1} \sum_{j=0}^{m-1} \varphi(j)y(j)$

RECURSIVE LINEAR REGRESSION

- Alternatively, we can initialize $P(-1) = \rho^2 I, \rho > 0$, and $\theta^*(-1) = \bar{\theta}$
- Interpretation: we are adding a **regularization term** on θ

$$\min \frac{1}{\rho^2}\|\theta - \bar{\theta}\|_2^2 + \sum_{j=0}^{k} \big(y(j) - \varphi'(j)\theta\big)^2 = \frac{1}{\rho^2}\sum_{i=1}^{m}(\theta_i - \bar{\theta}_i)^2 + \sum_{j=0}^{k} \big(y(j) - \varphi'(j)\theta\big)^2$$

$$= \sum_{i=1}^{m}\left(\frac{1}{\rho}\bar{\theta}_i - \frac{1}{\rho}e_i'\theta\right)^2 + \sum_{j=0}^{k} \big(y(j) - \varphi'(j)\theta\big)^2 = \sum_{j=-m}^{k} \big(y(j) - \varphi'(j)\theta\big)^2$$

where we replaced $y(-j) = \frac{1}{\rho}\bar{\theta}_i, \varphi(-j) = \frac{1}{\rho}e_j, e_i$ = i-th column $I, j = 1, \ldots, m$

- Therefore $P(-1) = \left[\sum_{j=-m}^{-1} \varphi(j)\varphi'(j)\right]^{-1} = \left[\sum_{j=-m}^{-1} \frac{1}{\rho^2} e_{-j}e_{-j}'\right]^{-1} = \rho^2 I$

and $\theta^*(-1) = P(-1)\sum_{j=-m}^{-1} \varphi(j)y(j) = \rho^2 I \sum_{i=1}^{m} \frac{1}{\rho^2}\bar{\theta}_i e_i = \bar{\theta}$

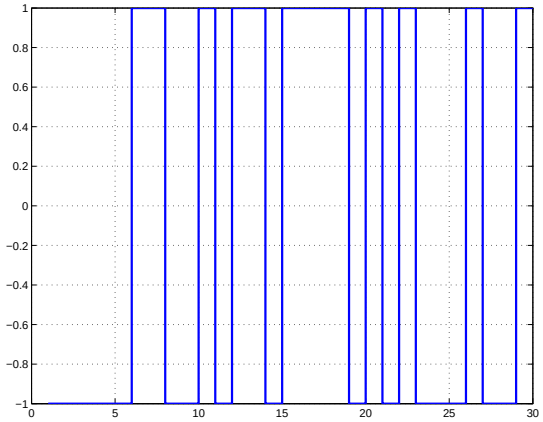
BLACK-BOX IDENTIFICATION: THE GENERAL PROCEDURE

- **Design of experiment**: What kind of input excitation $u(k)$ to apply ?
- **Model structure**: Which class of models do I choose to fit my data ?
- **Fit criterion** between data and model: How do I best choose the model within that class (=the parameter vector) ?
- **Validation criterion**: Is the model that I have identified good enough to reproduce the dynamics of the process ?

DESIGN OF EXPERIMENT

- Collecting data is a very crucial (and most expensive) step
- Some theory is available, as well as some practical rules
- The data set Z^N should be as informative as possible to fully identify the model[9]
- **Pseudo-random binary signals** (PRBS) randomly switching between ± 1 are a good choice

MATLAB
```
» u=idinput(N,'PRBS');
```



[9] Sinusoidal signals $u(t) = \sin(\omega t)$ are **not** good, as only $G(j\omega)$ would be captured. The input signal must at least contain as many different frequencies as the order of the chosen structure of linear models. Step responses are not ideal but ok: $|\mathcal{F}[\mathbb{1}(t)]| = \frac{1}{\omega}$ (the Fourier transform of the continuous-time signal $\mathbb{1}(t)$) has in infinite number of frequencies, although decreasing in amplitude.

MODEL STRUCTURE

- A linear system with additive disturbance $v(k)$ can be described as

$$y(k) = G(z)u(k) + v(k)$$

where $G(z)$ is a transfer function ($z^{-m}x(k) = x(k-m)$)

$$G(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_a} z^{-n_a}}$$

- Alternatively, it can be described by perturbing the difference equation:

$$\begin{aligned} y(k) = &- a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{n_a} y(k-n_a) \\ &+ b_0 u(k) + b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) + w(k) \end{aligned}$$

- The two models are equivalent if we set
$w(k) = v(k) + a_1 v(k-1) + a_2 v(k-2) + \ldots + a_{n_a} v(k-n_a)$

MODEL STRUCTURE

- The disturbance $v(k)$ is not necessarily white noise, but can be **colored noise**

$$v(k) = H(z)e(k)$$

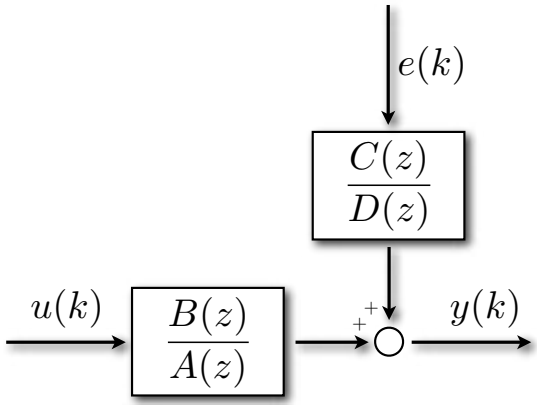
where $e(k)$ is white noise and $H(z)$ is another transfer function

$$H(z) = \frac{C(z)}{D(z)} = \frac{1 + c_1 z^{-1} + \ldots + c_{n_c} z^{-n_c}}{1 + d_1 z^{-1} + d_2 z^{-2} + \ldots + d_{n_d} z^{-n_d}}$$

and $H(z) = 0$ for $z^{-1} = 0$

- The overall model is called **Box-Jenkins (BJ)** model

$$y(k) = \frac{B(z)}{A(z)} u(k) + \frac{C(z)}{D(z)} e(k)$$



MODEL STRUCTURE - SPECIAL CASES

- **Output Error (OE)** model: $v(k)$ is white noise ($C(z) = D(z) = 1$)

$$y(k) = \frac{B(z)}{A(z)}u(k) + e(k)$$

- **Auto-Regressive Moving-Average with eXogenous variable (ARMAX)** model: $G(z)$ and $H(z)$ have the same denominator ($A(z) = D(z)$)

$$A(z)y(k) = B(z)u(k) + C(z)e(k)$$

- ARX models are a particular case of ARMAX models ($C(z) = 1$)

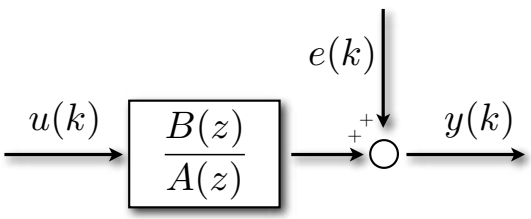
$$A(z)y(k) = B(z)u(k) + e(k)$$

- ARX and ARMAX models are the most used in practice

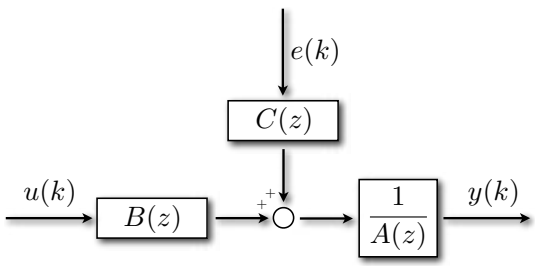
MODEL STRUCTURE

- Differently from BJ models, in ARMAX models $v(k)$ and $u(k)$ are filtered by the same dynamics $\frac{1}{D(z)}$
- This is justified if the source of disturbance enters early in the process, together with the input

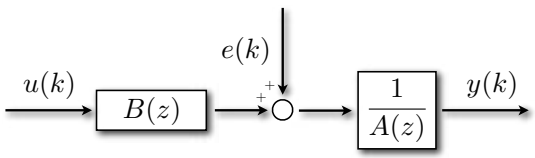
  Example: in airplanes, the disturbances from wind blasts create the same kind of forces on the airplane as the deflections of the control surfaces
- ARX models are the simplest to compute numerically



OE model



ARMAX model



ARX model

FIT CRITERION

- Let us consider BJ models, which is the most general structure

- Let θ collect all the parameters in the transfer functions $G(z)$ and $H(z)$ to be estimated from data

$$\begin{aligned} y(k) &= G(z,\theta)u(k) + H(z,\theta)e(k) \\ H^{-1}(z,\theta)y(k) &= H^{-1}(z,\theta)G(z,\theta)u(k) + e(k) \\ y(k) + H^{-1}(z,\theta)y(k) &= y(k) + H^{-1}(z,\theta)G(z,\theta)u(k) + e(k) \end{aligned}$$

- Finally, we get

$$y(k) = \left(1 - H^{-1}(z,\theta)\right) y(k) + H^{-1}(z,\theta)G(z,\theta)u(k) + e(k)$$

- Note that $1 - H^{-1}(z,\theta) = h_1 z^{-1} + h_2 z^{-2} + \ldots$ for some coefficients $\{h_i\}_{i=1}^{\infty}$

FIT CRITERION

- For $e(k) = 0$ (=the best estimate of $e(k)$ we can make), the one-step ahead prediction of $y(k)$ based on previous measurements is

$$\hat{y}(k|\theta) = \left(1 - H^{-1}(z,\theta)\right) y(k) + H^{-1}(z,\theta) G(z,\theta) u(k)$$

- Assuming we have enough data ($N \geq \max(n_a, n_b, n_c, n_d)$), we compute the **residual**

$$\epsilon(k|\theta) = y(k) - \hat{y}(k|\theta)$$

- The most used fit criterion is

$$V(\theta, Z^N) = \frac{1}{N} \sum_{k=0}^{N-1} \epsilon^2(k|\theta)$$

- The optimal vector θ^* is determined by solving the optimization problem

$$\theta^* = \arg\min_{\theta} V(\theta, Z^N)$$

COMPLEXITY

- Ideally ϵ should depend linearly on θ, so we can get the explicit solution of a least-squares problem. This only happens for ARX models
- Besides choosing the model structure (ARX, ARMAX, etc.) we also need to decide the **order** of the model, i.e., the number of free parameters to optimize
- A small number of parameters could make the model too simple, and not able to explain the data
- A large number of parameters could make the model more complex than we need and **overfit** the data in Z^N, resulting in poor predictions on new data
- How to choose the right model complexity?

MODEL VALIDATION

- Usually to avoid being fooled by overfitting the data set we split Z^N in two subsets: **estimation data** Z_{est} and **validation data** Z_{val}:
  - Z_{est} is used to compute the optimal parameter vector θ^*
  - Z_{val} is used to see how the estimated model behaves on fresh data
- A validation criterion is to look at one-step prediction errors

$$V(\theta^*, Z_{val}) = \frac{1}{N}\sum_{k=0}^{N-1}(y(k) - \hat{y}(k|\theta^*))^2$$

nothing to optimize here, just substitute θ^*, Z_{val} and evaluate

- Another validation criterion is to simulate the model completely in "open-loop"

$$y_{\rm sim}(k,\theta^*) = G(z,\theta^*)u(k)$$

and to look at

$$\frac{1}{N}\sum_{k=0}^{N-1}(y(k) - y_{\rm sim}(k,\theta^*))^2$$

(or just observe how much the plots of $y(k)$ and $y_{\rm sim}(k,\theta^*)$ differ)

MODEL VALIDATION - RESIDUAL ANALYSIS

- Ideally the prediction error (or **prediction residual**) $\epsilon(k|\theta)$ should be white noise and uncorrelated with $u(k)$

- To test whiteness of $\epsilon(k|\theta)$ we compute the **auto-correlation** function

$$R_\epsilon(\tau) = \frac{1}{N}\sum_{k=0}^{N-\tau-1} \epsilon(k+\tau|\theta)\epsilon(k|\theta)$$

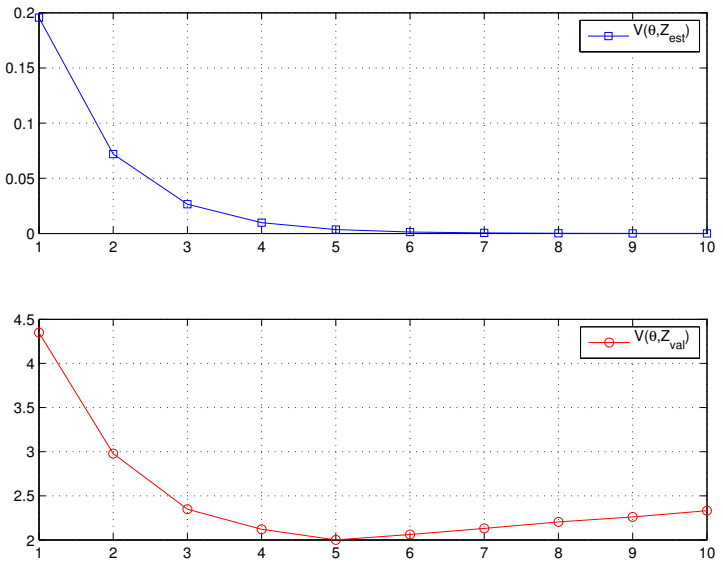
- To test correlation between $\epsilon(k|\theta)$ and $u(k)$ we compute the sample covariance

$$R_{\epsilon u}(\tau) = \frac{1}{N}\sum_{k=\tau-1}^{N-1} \epsilon(k|\theta)u(k-\tau)$$

- Both $R_\epsilon(\tau)$ and $R_{\epsilon u}(\tau)$ should be small

MODEL SELECTION

- Which model structure to choose (ARX, ARMAX, OE, etc.)? Which model orders n_a, n_b, n_k, etc. ?
- **Cross-validation** is the procedure that compares the quality of fit of different models, by validating them on a data set where neither of them was estimated
- Let $\theta_1^*, ..., \theta_s^*$ a set of optimal parameters for different model structures
- The best model θ_i^* is the one for which $V(\theta_i^*, Z_{val})$ is smallest
- Often $V_i(\theta^*, Z_{est})$ decreases as the model complexity increases, while $V_i(\theta^*, Z_{val})$ starts increasing when the model complexity becomes excessive (=overfit of estimation data)



MODEL SELECTION

- If fresh validation data are not available (=no cross-validation), we can use the same performance figures, but in addition penalize overfit (we want a good balance between simplicity and accuracy)
- Let d_i = number of elements of θ_i^* (=model complexity)
- We look for the model that minimizes one of the following figures:
  - Akaike's Information theoretic Criterion (AIC):

$$\left(1+\frac{2d_i}{N}\right)\frac{1}{N}\sum_{k=0}^{N-1}\epsilon^2(k|\theta_i^*)$$

  - Akaike's Final Prediction Error (FPE):

$$\left(\frac{1+\frac{d_i}{N}}{1-\frac{d_i}{N}}\right)\frac{1}{N}\sum_{k=0}^{N-1}\epsilon^2(k|\theta_i^*)$$

  - Rissanen's Minimum Description Length (MDL):

$$\frac{1}{N}\sum_{k=0}^{N-1}\epsilon^2(k|\theta_i^*)+\frac{d_i\cdot\ln N}{N}$$

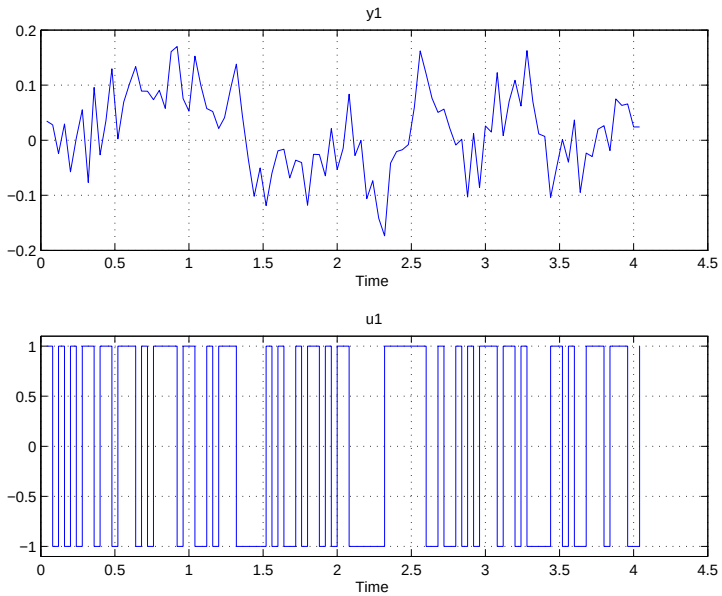
IDENTIFICATION EXAMPLE

- Assume that the real (unknown) process is

$$G(z) = \frac{0.03726z^{-1} - 0.09676z^{-2} + 0.08355z^{-3} - 0.024z^{-4}}{1 - 3.464z^{-1} + 4.493z^{-2} - 2.586z^{-3} + 0.5577z^{-4}}$$

with sample time $T = 0.04$ s

- Input excitation: PRBS sequence
- We have 200 samples. The first 100 samples are used for estimation of θ^*, the rest for validation



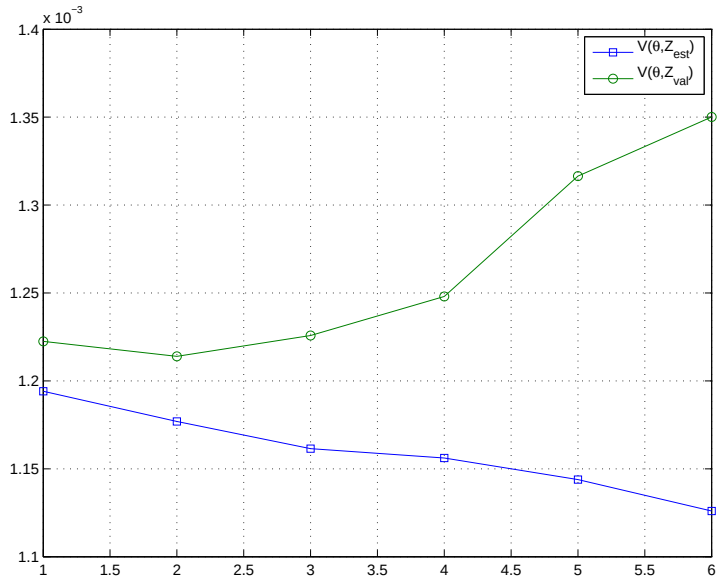
IDENTIFICATION EXAMPLE (CONT'D)

We try five different ARX model structures ARX(n_a,n_b,n_k):

$$A(z)y(t) = B(z)u(t) + e(t)$$

$$\frac{1}{N_{\rm val}} \sum_{k=0}^{N_{\rm val}-1} \epsilon^2(k|\theta_i^*), \quad \frac{1}{N_{\rm est}} \sum_{k=0}^{N_{\rm est}-1} \epsilon^2(k|\theta_i^*)$$

- $i = 1$: ARX(1,1,1)
- $i = 2$: ARX(2,2,1)
- $i = 3$: ARX(3,3,1)
- $i = 4$: ARX(4,4,1)
- $i = 5$: ARX(5,5,1)
- $i = 6$: ARX(6,6,1)

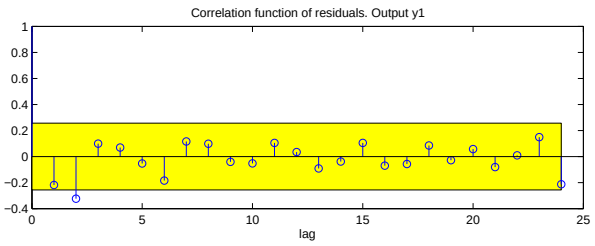


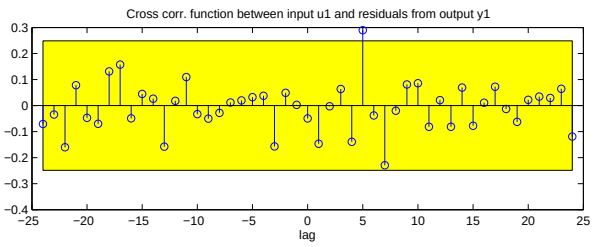
model structure index i

The best model structures are $i = 2, 3, 4$

IDENTIFICATION EXAMPLE (CONT'D)

Residual analysis:

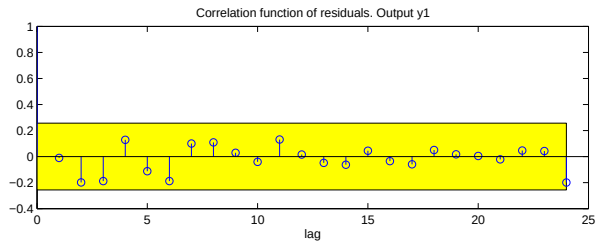


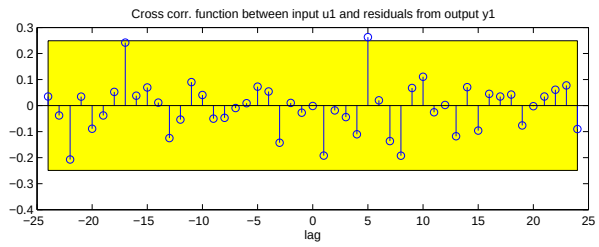


ARX(2,2,1)

$$A(z) = 1 - 0.2153z^{-1} - 0.5624z^{-2}$$

$$B(z) = 0.04041z^{-1} + 0.02456z^{-2}$$

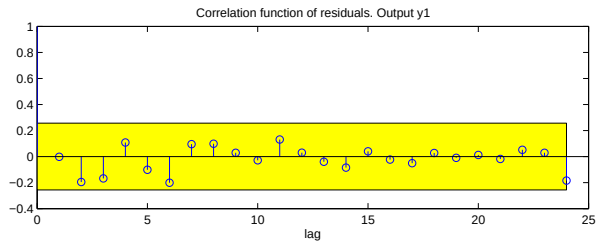


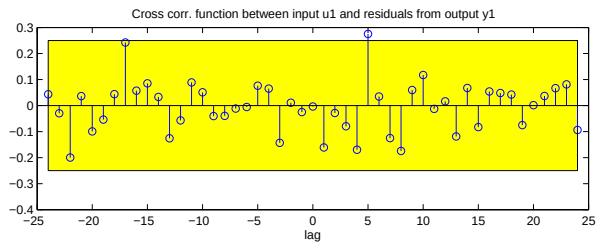


ARX(3,3,1)

$$A(z) = 1 + 0.1228z^{-1} - 0.3396z^{-2} - 0.4444z^{-3}$$

$$B(z) = 0.04014z^{-1} + 0.037z^{-2} + 0.02247z^{-3}$$





ARX(4,4,1)

$$A(z) = 1 + 0.1451z^{-1} - 0.319z^{-2} - 0.4258z^{-3} - 0.03208z^{-4}$$

$$B(z) = 0.03912z^{-1} + 0.03826z^{-2} + 0.02476z^{-3} + 0.004177z^{-4}$$

not much different from ARX(3,3,1)

CONCLUDING REMARKS ON SYSTEM IDENTIFICATION

- System identification and control design are **complementary**: no controller without a model, but identified model only useful for control and/or estimation
- If model parameters change on-line, one can use **adaptive control**, by identifying the model and changing the controller accordingly in real-time (**caveat: closed-loop stability may be an issue** )
- If linear model structures are not able to capture the model well, one should use **nonlinear models**, like **artificial neural networks**, **piecewise affine functions**, and other general function approximation methods available in **machine learning** (Schoukens, Ljung, 2019)
- In general, the more a-priori knowledge of the process we can exploit (e.g., from physical principles), the better. Sometimes black-box identification fails because its very difficult to guess the right model structure